



FANSHAWE COLLEGE

Electronics and Embedded Systems Development ELNC-6008 Practical Project Report

KEYLUMINATE

Student Names	Student Numbers
Nisha Desai	1086134
Manan Patel	1202955
Mohd. Saif Malam	1162507
Viral Gajera	1196217

Faculty Advisor:	Xio Ming Guo
Date Submitted:	20/04/2025

Executive Summary

This report presents the development, execution, and analysis of the Piano LED Assistant project by Team Key-Luminate. The system utilizes the ESP32-S3 microcontroller to control a WS2812 LED strip, lighting up in synchronization with piano notes to assist learners. Additional features include an LCD display for current/next note, push buttons for play/pause, song switching, and tempo adjustment. The report outlines design objectives, technical challenges, deliverables from each member, evaluation against project goals, and recommendations for future improvements.

Table of Contents

Executive Summary	ii
Table of Contents	iii
List of figures.....	v
List of Tables	vi
Acknowledgements	Error! Bookmark not defined.
1. Document Content and Organization.....	1
1.1 Introduction	1
1.2 Background	1
1.3 Purpose	1
1.4 Scope of Deliverables	1
1.5 Proposed Improvement	1
1.6 Proposed Potential Benefits	1
1.7 Preferred Embodiment	2
1.8 Operational Overview	2
1.9 Technical Problems	2
1.10 Justification of Design Decisions	2
1.11 Presentation of Results/Deliverables.....	3
1.12 Analysis of Results/Deliverables.....	3
1.13 Assessment of Results/Deliverables	4
1.14 Conclusions.....	5
1.15 Recommendations.....	5
References	6

2.	Declaration of Originality	7
	Appendix I: System Description	8
	Appendix II: Final Project	9
	Appendix Code	11

List of figures

Figure 1: Schematic Diagram.....	8
Figure 2:Project Hardware	9
Figure 3: LED strip mounted on Piano	10

List of Tables

Table 1. This is a sample table with recommended formatting**Error! Bookmark not defined.**

Table 2. This is a sample table with less recommended formatting **Error! Bookmark not defined.**

1. Document Content and Organization

1.1 Introduction

- Modern education tools demand interactive and engaging technologies to support learning. This project introduces the Piano LED Assistant, a solution designed to guide piano learners with real-time visual cues. By bridging hardware control and user interface, we developed a system that simplifies piano learning through direct, interactive feedback.

1.2 Background

- Many piano learning aids are either too basic or inaccessible due to cost. Sheet music can be difficult for beginners, and MIDI systems are typically expensive and hard to customize. Our project addresses this gap by offering a hands-on, programmable, and scalable visual guide system using addressable LEDs controlled by a microcontroller. This offers an affordable, interactive alternative to more expensive, less intuitive systems.

1.3 Purpose

- The purpose of this project is to design a device that supports new learners by making piano practice more visual and intuitive. The system highlights keys in real time as a song plays, giving users a clear guide on what to play next. This addresses the primary design deficiency in existing products—lack of real-time visual support for beginners.

1.4 Scope of Deliverables

- This project includes the following deliverables:
- A functioning prototype using ESP32-S3
- WS2812 LED strip fully mapped to 88 piano keys
- LCD display for showing current and next note
- Physical buttons for user control (Play/Pause, Song Change, Tempo)
- Pre-programmed songs: Happy Birthday, Twinkle-Twinkle, Jingle Bells, La Vie En Rose, and Für Elise
- Well-documented source code (Arduino & ESP-IDF)
- Project report and final team presentation

1.5 Proposed Improvement

- We improved the base design by implementing tempo control, expanding song options, and enabling note display on an LCD. Delay logic was optimized to maintain synchronization, and an instant song-change feature was added to enhance usability. These improvements made the system more responsive, intuitive, and enjoyable to use.

1.6 Proposed Potential Benefits

- Our solution provides several advantages:
- Encourages musical learning through visual feedback
- Engages learners with physical interaction

- Reduces learning curve for beginners
- Budget-friendly compared to commercial systems
- Easily upgradable for more songs or features

1.7 Preferred Embodiment

- The final implementation consists of an ESP32-S3-WROOM microcontroller controlling an 88-key WS2812B LED strip. A 16x2 LCD is used to display real-time note updates [1]. User interaction is handled through three buttons, each assigned a unique GPIO for play/pause, song switching, and tempo control. This configuration balances functionality with affordability and simplicity.

1.8 Operational Overview

- The system begins in idle mode. Pressing the play/pause button (GPIO5) starts the LED sequence that lights up piano keys corresponding to the notes in the selected song. Pressing the song change button (GPIO6) immediately switches to the next song and restarts the LED sequence. The tempo button (GPIO7 or GPIO40) cycles between three playback speeds (0.5x, 1x, and 1.5x), indicated by a brief purple flash on all LEDs. Meanwhile, the LCD displays the current and upcoming note for visual reference.

1.9 Technical Problems

- Like most hardware projects, we ran into a few technical challenges along the way. One of the first issues was with the LED strip flickering [2]. We found that the LEDs weren't getting consistent power, especially when a lot of them were turned on at once. We solved this by using a better power supply and adding a capacitor to smooth out the voltage.
- Another issue was button bouncing, where a single press sometimes registered multiple times. This made actions like switching songs or changing tempo unreliable. We fixed it by adding debounce logic in the code. [3]
- We also had trouble with the LCD screen at first—the contrast settings made it hard to read. After adjusting the potentiometer and testing pin connections, the display worked clearly.
- Finally, managing timing and synchronization between the LEDs and songs became harder as more features were added. We adjusted our code to use non-blocking delays, which helped maintain smooth operation.

1.10 Justification of Design Decisions

- We made our design decisions based on what would be simple, affordable, and useful for someone trying to learn piano. We chose the ESP32-S3-WROOM because it gave us plenty of GPIO pins and had more than enough power to run the LED animations and LCD screen at the same time [3]. It also has Wi-Fi and Bluetooth built-in, which gives us room to add more features later if needed.
- For the LEDs, we went with WS2812 addressable LEDs because they're bright, easy to control, and let us map one LED to each piano key using just a single data line [2]. That made wiring much easier and reduced the number of connections we had to worry about.
- We decided to use three simple push buttons instead of a touchscreen or app because we wanted the system to be easy to use for anyone—especially beginners. The buttons control

play/pause, switch songs, and change the tempo. They're basic, but they work reliably and were easy to program.

- The LCD screen was added to show which note is playing and what's coming next [4]. This helped users follow along more easily and made the experience more interactive. Each song also has a unique LED color, which makes it easier to remember and keeps things fun.
- Instead of using MIDI files or complex inputs, we kept it simple by hardcoding each song in the program. This kept the system fast and smooth, without delays or crashes [6].
- Overall, we focused on making something practical, easy to build, and helpful for learners. The choices we made kept costs low, reduced complexity, and gave us a lot of flexibility for future improvements.

1.11 Presentation of Results/Deliverables

- The deliverables of the Piano LED Assistant project were the result of clearly assigned tasks and effective collaboration among all team members. Everyone contributed specific components to ensure the successful development, testing, and presentation of the final system:
- *Manan Patel* and *Viral Gajera* were responsible for integrating all hardware components, including the ESP32, WS2812 LED strip, and 16x2 LCD display. They also implemented the real-time LCD note feedback feature and structured the final technical report. Both members jointly led the final demonstration and presentation of the working prototype.
- *Nisha Desai* focused on the software aspects of song management. She was responsible for coding the note sequences, mapping them accurately to corresponding LED positions, and designing the physical layout for efficient component placement. Additionally, she documented the user-facing operation of the system to ensure clarity and ease of use.
- *Mohd. Saif Shehzad Malam* contributed by translating the Arduino-based prototype logic into the ESP-IDF framework. He optimized the control algorithms for the buttons (play/pause, song switch, and tempo), ensuring smooth and reliable interaction. Saif also designed the mechanical casing that housed the electronics and LED strip, ensuring both aesthetics and protection.
- Each team member's work aligned with the overall project objectives and contributed directly to the functional and presentational quality of the final deliverable.

1.12 Analysis of Results/Deliverables

- Based on testing, the LED strip successfully illuminated corresponding keys with <500ms delay accuracy. The tempo adjustment worked as intended, and users could switch songs instantly without resets. The LCD correctly displayed current and upcoming notes, helping users anticipate transitions. Each deliverable directly aligned with project objectives and enhanced user experience.

1.13 Assessment of Results/Deliverables

- All project goals outlined in the initial scope were successfully achieved. In addition, several enhancements were implemented that extended the system's capabilities beyond the original plan. These included the integration of tempo control, distinct LED colors for each song, and immediate song switching functionality.
- System testing confirmed that each module—LED output, button controls, LCD note display, and song sequencing—performed reliably under multiple usage scenarios. The system demonstrated consistent responsiveness and stability. These improvements contributed to a more refined and user-friendly prototype, enhancing the overall value and functionality of the final deliverable.

1.14 Conclusions

- The Piano LED Assistant successfully met its primary objectives by providing a low-cost, intuitive, and interactive tool to assist beginner piano learners. The project demonstrated the effective integration of embedded hardware, software control, and user-focused design principles, resulting in a functional and educational prototype.
- The system's ability to visually guide learners through songs using synchronized LED cues and real-time LCD feedback significantly improves the learning experience. The collaborative efforts of all team members contributed to addressing technical challenges, refining the design, and delivering the final product within the required timeframe. The outcome is a robust prototype that holds strong potential for further development and real-world application in music education.

1.15 Recommendations





- Based on the outcomes and testing of the Piano LED Assistant, several recommendations are proposed to enhance future versions of the system:
- Bluetooth or Wi-Fi Integration: Implementing wireless connectivity would allow for mobile app control, remote song uploads, and firmware updates, increasing user convenience and flexibility.
- MIDI File Support: Incorporating MIDI file parsing would enable dynamic song input, allowing users to learn any song without hardcoding sequences.
- Speaker Output: Adding basic sound playback synchronized with LED cues could improve the audio-visual learning experience and eliminate the need for a separate piano during demonstrations.
- Rechargeable Battery Integration: A portable, battery-powered version would increase usability for learners who practice on different keyboards or in shared spaces.
- User Testing and Feedback Collection: Conducting usability testing with beginner musicians or music educators would provide insights into further user experience improvements and feature priorities.
- These enhancements would expand the system's functionality, accessibility, and overall value, supporting its use in both personal and instructional environments.

References

- [1] E. Systems, "ESP32-S3 Series Datasheet," [Online]. Available:
https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf.
- [2] Adafruit, "NeoPixel Uberguide," [Online]. Available: <https://learn.adafruit.com/adafruit-neopixel-uberguide>.
- [3] Arduino, "LiquidCrystal Library," [Online]. Available:
<https://www.arduino.cc/en/Reference/LiquidCrystal>.
- [4] E. Systems, "ESP-IDF Programming Guide," [Online]. Available:
<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>.
- [5] W. James, "MIDI Note Numbers to Frequencies,," [Online]. Available:
<https://newt.phys.unsw.edu.au/jw/notes.html>.
- [6] E. Systems, "ESP32-S3-WROOM-1/1U Datasheet,," [Online]. Available:
https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_datasheet_en.pdf.
- [7] P. F. S. Marek Babiuch, "Using the ESP32 Microcontroller for Data Processing," VSB - Technical University of Ostrava, Ostrava, Czech Republic, 2019.

2. Declaration of Originality

We, the undersigned, do hereby declare that this report is entirely original and was authored by the below. All included images, objects, tables, and/or other figures are entirely original and/or where/when appropriate, all citations and/or references have been suitably detailed.

Student Name	Signature	Date (DD MMM YYYY)
Nisha Desai		20-04-2025
Manan Patel		20-04-2025
Mohd. Saif Malam		20-04-2025
Viral Gajera		20-04-2025

Appendix I: System Description

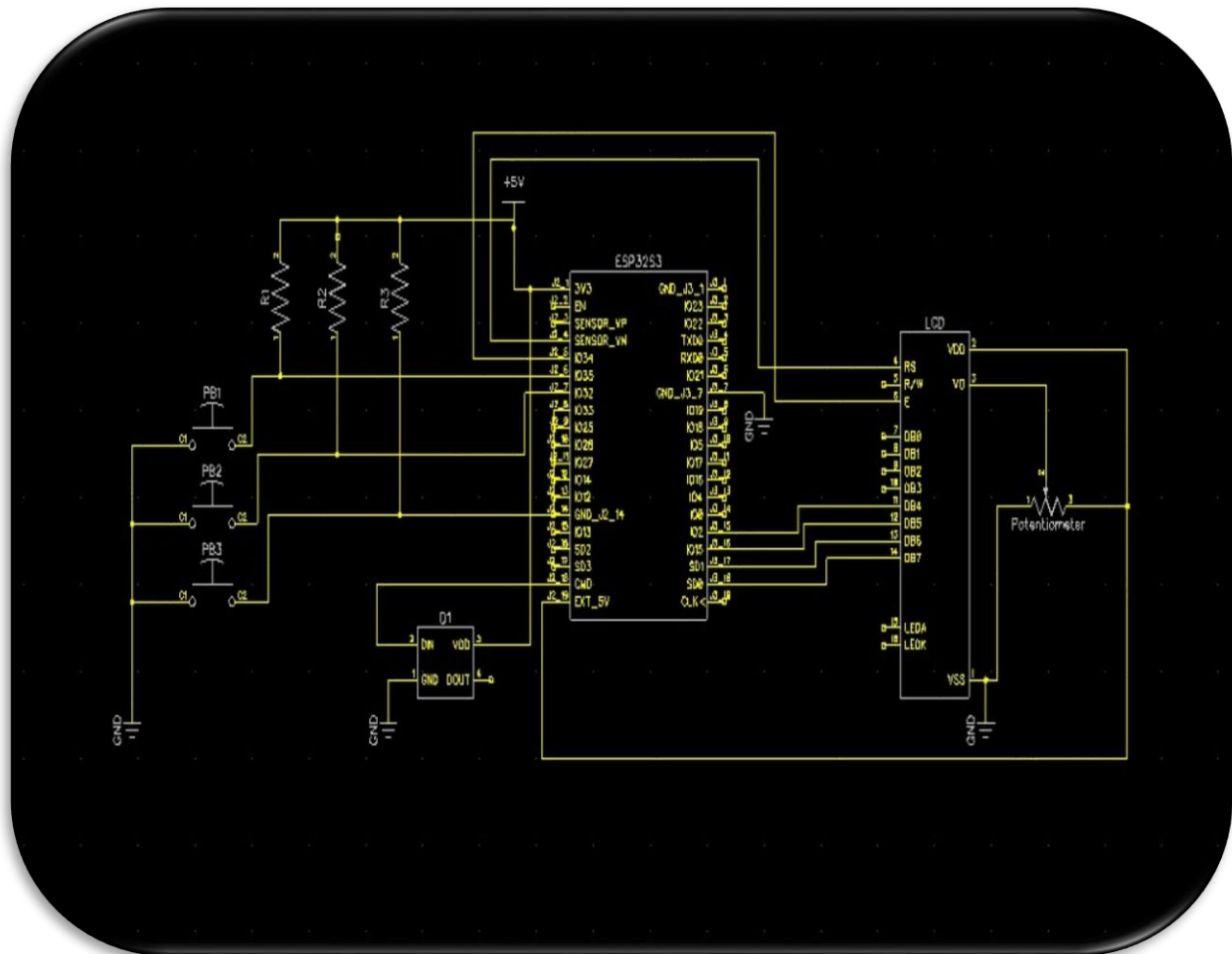


Figure 1: Schematic Diagram

Appendix II: Final Project

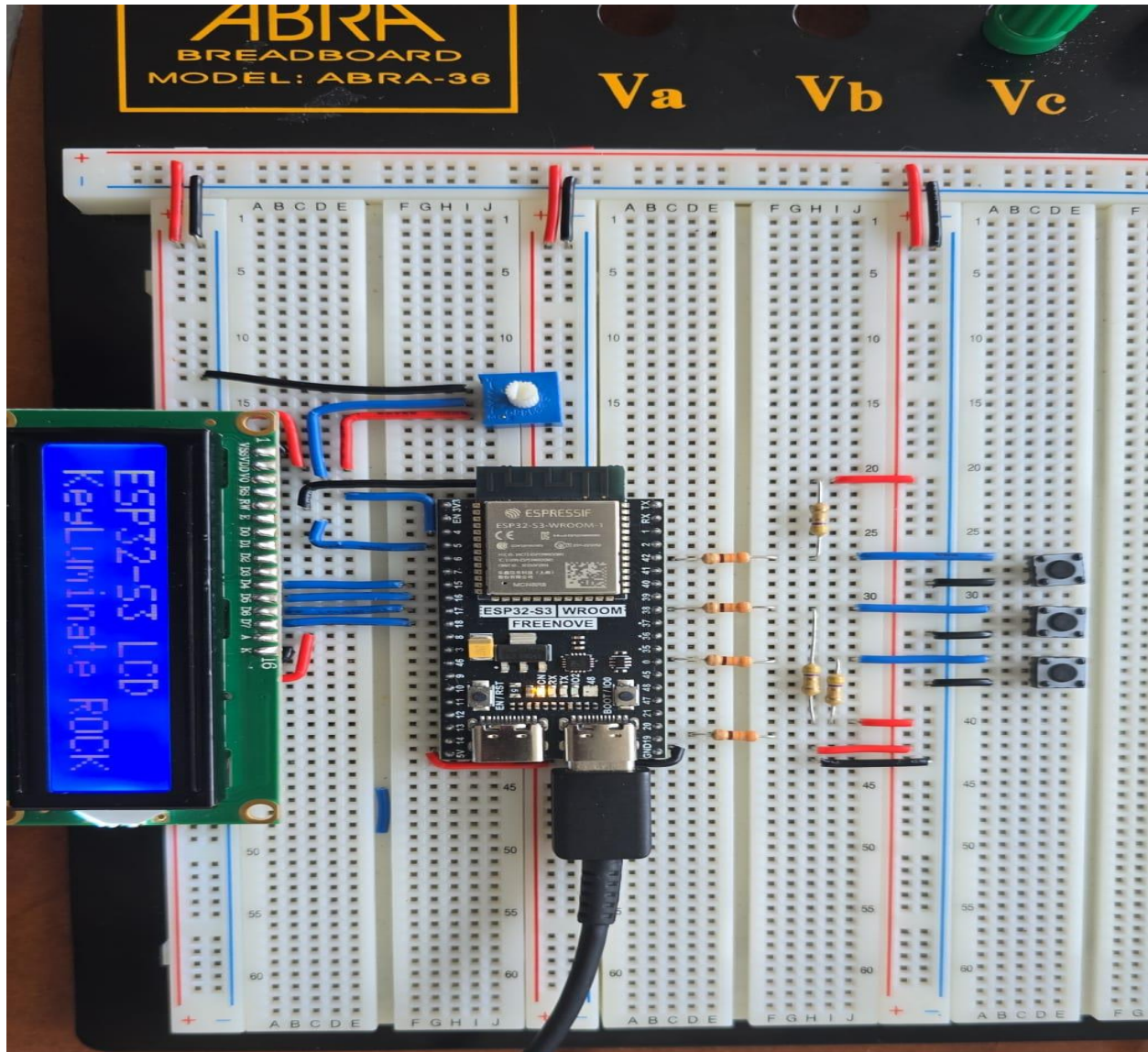


Figure 2: Project Hardware

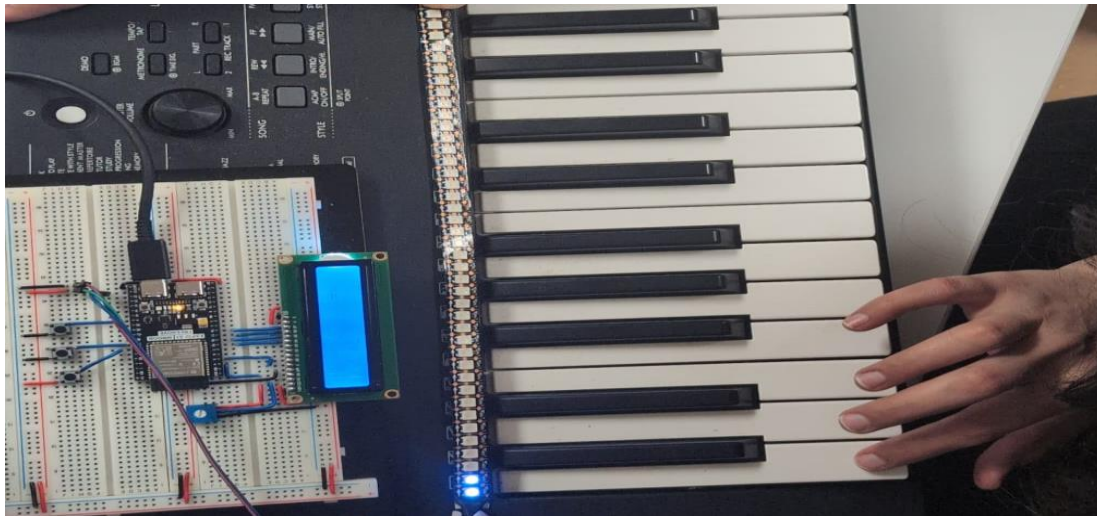


Figure 3: LED strip mounted on Piano

Appendix III Code

/Use of AI / Cognitive Assistance Software is not allowed in any evaluation, assessment or exercise./

/*=====

File Name: KEYLUMINATE.c

Author: Key-Luminate

Date: 18/03/2025

Modified: None

Fanshawe College, 2025

Description: This program runs on an ESP32 using ESP-IDF and controls a piano
LED guide system. It manages button inputs, updates an LCD screen,
and uses an LED strip to light up notes of songs for user interaction.

=====*/

// Libraries =====

#include <stdio.h>

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "freertos/timers.h"

#include "driver/gpio.h"

#include "esp_timer.h"

#include "sdkconfig.h"

#include "string.h"

#include "led_strip.h"

#include "hd44780.h"

#include "i2c-lcd1602.h"

// Constants =====

#define NUM_KEYS 40

#define LEDS_PER_KEY 2

#define TOTAL_LEDS (NUM_KEYS * LEDS_PER_KEY)

#define LED_PIN 19

#define PLAY_BUTTON_PIN 0

#define SONG_BUTTON_PIN 39

#define TEMPO_BUTTON_PIN 40

#define DEBOUNCE_DELAY_MS 200

// LED Strip and LCD Handle =====

```
led_strip_handle_t led_strip;
i2c_lcd1602_info_t * lcd_info = NULL;

// Enumeration =====
typedef enum {
    E2 = 0, F2, F2s, G2, G2s, A2_, A2s, B2,
    C3, C3s, D3, D3s, E3, F3, F3s, G3, G3s, A3_, A3s, B3,
    C4, C4s, D4, D4s, E4, F4, F4s, G4, G4s, A4_, A4s, B4,
    C5, C5s, D5, D5s, E5, F5, F5s, G5
} PianoKey;

#define LED_INDEX(key) ((key) * LEDS_PER_KEY)

// Song Data =====
int happyBirthdayMelody[] = {C4, C4, D4, C4, F4, E4, C4, C4, D4, C4, G4, F4, C4, C4, C5, A4_, F4, E4, D4,
A4s, A4s, A4_, F4, G4, F4};
int happyBirthdayDurations[] = {400, 400, 400, 400, 400, 800, 400, 400, 400, 400, 400, 800, 400, 400,
400, 400, 400, 400, 800, 400, 400, 400, 400, 400, 800};

int twinkleMelody[] = {C4, C4, G4, G4, A4_, A4_, G4, F4, F4, E4, E4, D4, D4, C4};
int twinkleDurations[] = {400, 400, 400, 400, 400, 400, 800, 400, 400, 400, 400, 400, 400, 800};

int jingleBellsMelody[] = {E4, E4, E4, E4, E4, E4, E4, G4, C4, D4, E4};
int jingleBellsDurations[] = {300, 300, 600, 300, 300, 600, 300, 300, 300, 300, 800};

const char *songNames[] = {"Happy Birthday", "Twinkle Twinkle", "Jingle Bells"};
const char *tempoNames[] = {"Normal", "Fast", "Slow"};

// Global Variables =====
int currentSong = 0;
bool isPlaying = false;
bool songChanged = false;
int tempoLevel = 0;
float tempoMultipliers[] = {1.0, 1.5, 0.5};

uint64_t lastPlayBtnTime = 0;
uint64_t lastSongBtnTime = 0;
uint64_t lastTempoBtnTime = 0;

// Functions =====
```

```
/*>>> delay_ms: =====
Author:   Key-Luminate
Date:    18/03/2025
Modified: None
Desc:    Delay helper using FreeRTOS delay
Input:   int ms - delay time in milliseconds
Returns: None
=====*/

static void delay_ms(int ms) {
    vTaskDelay(ms / portTICK_PERIOD_MS);
}

/*>>> lightKey: =====
Author:   Key-Luminate
Date:    18/03/2025
Modified: None
Desc:    Light up LED for a specific piano key.
Input:   keyIndex (int), r/g/b (color values)
Returns: None
=====*/

static void lightKey(int keyIndex, uint32_t r, uint32_t g, uint32_t b) {
    if (keyIndex < 0 || keyIndex >= NUM_KEYS) return;
    for (int j = 0; j < LEDS_PER_KEY; j++) {
        led_strip_set_pixel(led_strip, LED_INDEX(keyIndex) + j, r, g, b);
    }
    led_strip_refresh(led_strip);
}

/*>>> clearKey: =====
Author:   Key-Luminate
Date:    18/03/2025
Modified: None
Desc:    Turn off LED for a specific piano key.
Input:   keyIndex (int)
Returns: None
=====*/

static void clearKey(int keyIndex) {
    if (keyIndex < 0 || keyIndex >= NUM_KEYS) return;
    for (int j = 0; j < LEDS_PER_KEY; j++) {
        led_strip_set_pixel(led_strip, LED_INDEX(keyIndex) + j, 0, 0, 0);
    }
}
```

```
led_strip_refresh(led_strip);
}

/*>>> flashAll: =====
Author:   Key-Luminate
Date:    18/03/2025
Modified: None
Desc:    Flash all LEDs for a moment with a given color.
Input:   r/g/b (color), duration in ms
Returns: None
=====*/
static void flashAll(uint8_t r, uint8_t g, uint8_t b, int duration_ms) {
    for (int i = 0; i < TOTAL_LEDS; i++) {
        led_strip_set_pixel(led_strip, i, r, g, b);
    }
    led_strip_refresh(led_strip);
    delay_ms(duration_ms);
    led_strip_clear(led_strip);
    led_strip_refresh(led_strip);
}

/*>>> updateLCD: =====
Author:   Key-Luminate
Date:    18/03/2025
Modified: None
Desc:    Update LCD with current song and play/tempo status.
Input:   None
Returns: None
=====*/
static void updateLCD() {
    char line1[17];
    char line2[17];
    snprintf(line1, sizeof(line1), "%s", songNames[currentSong]);
    snprintf(line2, sizeof(line2), "%s %s", isPlaying ? "Playing" : "Paused", tempoNames[tempoLevel]);

    i2c_lcd1602_clear(lcd_info);
    i2c_lcd1602_write_string(lcd_info, line1);
    i2c_lcd1602_move_cursor(lcd_info, 0, 1);
    i2c_lcd1602_write_string(lcd_info, line2);
}
```

```
/*>>> checkButtons: =====
Author:   Key-Luminate
Date:    18/03/2025
Modified: None
Desc:    Handle button press events with debounce logic.
Input:   None
Returns: None
=====*/

static void checkButtons() {
    uint64_t now = esp_timer_get_time() / 1000;

    if (!gpio_get_level(PLAY_BUTTON_PIN) && now - lastPlayBtnTime > DEBOUNCE_DELAY_MS) {
        isPlaying = !isPlaying;
        lastPlayBtnTime = now;
        updateLCD();
    }

    if (!gpio_get_level(SONG_BUTTON_PIN) && now - lastSongBtnTime > DEBOUNCE_DELAY_MS) {
        currentSong = (currentSong + 1) % 3;
        lastSongBtnTime = now;
        songChanged = true;
        updateLCD();
        flashAll(128, 0, 128, 300);
        isPlaying = true;
    }

    if (!gpio_get_level(TEMPO_BUTTON_PIN) && now - lastTempoBtnTime > DEBOUNCE_DELAY_MS) {
        tempoLevel = (tempoLevel + 1) % 3;
        lastTempoBtnTime = now;
        updateLCD();
        flashAll(200, 0, 200, 200);
    }
}

/*>>> playSong: =====
Author:   Key-Luminate
Date:    18/03/2025
Modified: None
Desc:    Play the selected song's melody with LED light effects.
Input:   melody array, durations array, length, r/g/b values
Returns: None
```

```
=====*/
static void playSong(const int *melody, const int *durations, int length, uint8_t r, uint8_t g, uint8_t b) {
    for (int i = 0; i < length; i++) {
        checkButtons();
        if (!isPlaying || songChanged) {
            songChanged = false;
            return;
        }

        int note = melody[i];
        int duration = durations[i];

        lightKey(note, r, g, b);
        delay_ms(duration / tempoMultipliers[tempoLevel]);
        clearKey(note);
        delay_ms(100 / tempoMultipliers[tempoLevel]);
    }
    isPlaying = false;
}
```

```
/*>>> app_main: =====
Author:   Key-Luminate
Date:    18/03/2025
Modified: None
Desc:    Entry point for ESP-IDF application. Initializes hardware and
         manages playback loop.
Input:   None
Returns: None
```

```
=====*/
void app_main(void) {
    // GPIO Configuration
    gpio_set_direction(PLAY_BUTTON_PIN, GPIO_MODE_INPUT);
    gpio_set_pull_mode(PLAY_BUTTON_PIN, GPIO_PULLUP_ONLY);
    gpio_set_direction(SONG_BUTTON_PIN, GPIO_MODE_INPUT);
    gpio_set_pull_mode(SONG_BUTTON_PIN, GPIO_PULLUP_ONLY);
    gpio_set_direction(TEMPO_BUTTON_PIN, GPIO_MODE_INPUT);
    gpio_set_pull_mode(TEMPO_BUTTON_PIN, GPIO_PULLUP_ONLY);

    // LED Strip Initialization
    led_strip_config_t strip_config = {
        .strip_gpio_num = LED_PIN,
```

```
.max_leds = TOTAL_LEDS,
.led_pixel_format = LED_PIXEL_FORMAT_GRB,
.led_model = LED_MODEL_WS2812,
.flags = 0,
};
led_strip_rmt_config_t rmt_config = {
    .clk_src = RMT_CLK_SRC_DEFAULT,
    .resolution_hz = 10 * 1000 * 1000,
    .flags = 0,
};
led_strip_new_rmt_device(&strip_config, &rmt_config, &led_strip);
led_strip_clear(led_strip);

// LCD Initialization
lcd_info = i2c_lcd1602_malloc();
i2c_lcd1602_init(lcd_info, 0x27, I2C_NUM_0, 16, 2);
i2c_lcd1602_reset(lcd_info);
i2c_lcd1602_clear(lcd_info);
i2c_lcd1602_write_string(lcd_info, "Key-Luminate");
delay_ms(1000);
i2c_lcd1602_clear(lcd_info);

// Main Loop
while (1) {
    checkButtons();
    if (currentSong == 0)
        playSong(happyBirthdayMelody, happyBirthdayDurations,
sizeof(happyBirthdayMelody)/sizeof(int), 255, 150, 0);
    else if (currentSong == 1)
        playSong(twinkleMelody, twinkleDurations, sizeof(twinkleMelody)/sizeof(int), 0, 255, 200);
    else if (currentSong == 2)
        playSong(jingleBellsMelody, jingleBellsDurations, sizeof(jingleBellsMelody)/sizeof(int), 255, 0, 0);

    delay_ms(100);
}
} // eo app_main::
```