Electronics and Embedded Systems Development

ELNC-6012
Practical Project

TEST REPORT #1

Test Date:	24 FEB 2025
Conducted By:	*Nisha Desai* *Mohd. Saif Malam*
Test Location:	B1020
Document:	This is page 1 of 22

Test Particulars:	
Team Name:	Key Luminate
Team Identifier:	ELNC-6012 – 2025 – Team 1
Test Title:	Working phase of LED strip and Pushbuttons
Test Revision:	1.0
WBS Identifier:	5.1.2

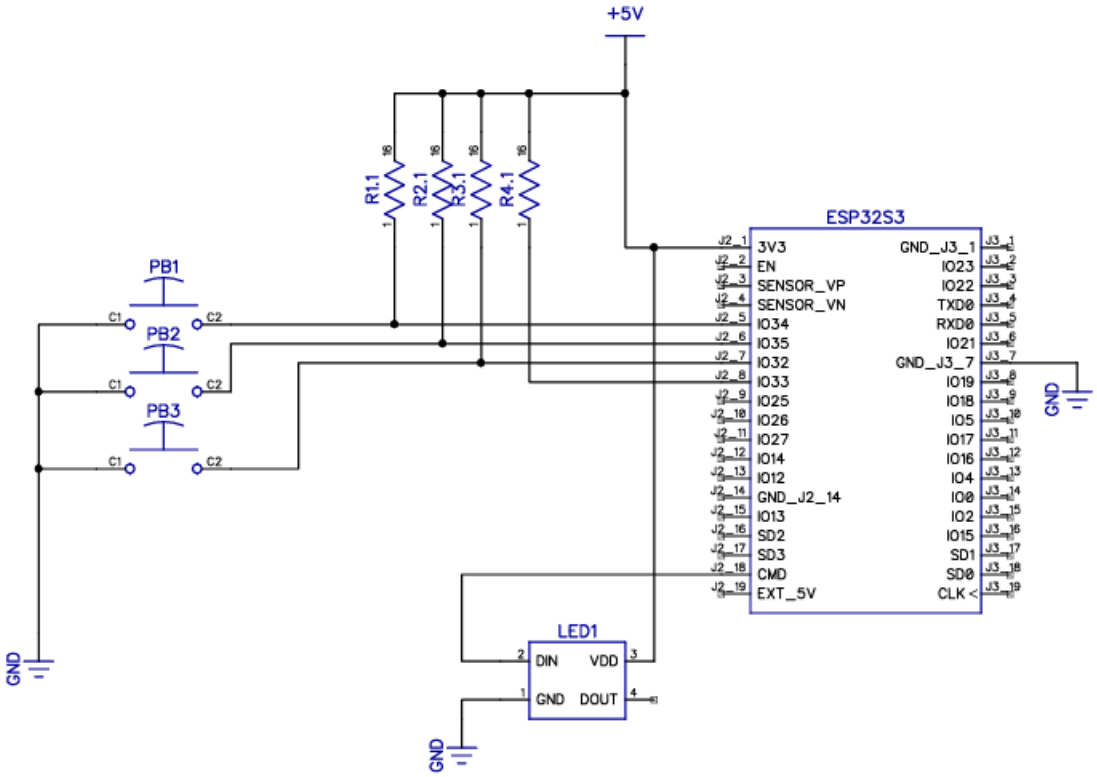
Purpose:

This testing ensures that the ESP32-based LED piano assistant responds accurately to button inputs for **playback, song selection, and tempo changes**. It verifies that each song is played in its designated **color (Red, Green, or Blue)** and that tempo adjustments trigger the correct **LED flashes**. The test also checks for smooth transitions between songs and tempo levels without flickering or delays.

Test Methodology:

This test should confirm the desired operation of the **ESP32-based LED piano assistant** by ensuring that **button inputs correctly control playback, song selection, and tempo changes**. When **Button 1 is pressed**, the system should start playing the selected song. Pressing **Button 2** should instantly change the song and flash all LEDs, while **Button 3** should adjust the tempo and flash LEDs in a corresponding color. Each song should play with its assigned **LED color** (Red for Happy Birthday, Green for Twinkle Twinkle, Blue for Jingle Bells). When the tempo is changed, the LED strip should flash all LEDs indicating a change in speed. The system should handle **button debouncing properly**, preventing accidental multiple triggers. LED transitions should be smooth, with **no flickering or unexpected behavior**. The test should also verify that song changes and tempo adjustments **happen instantly without delays**.

Test Circuit:

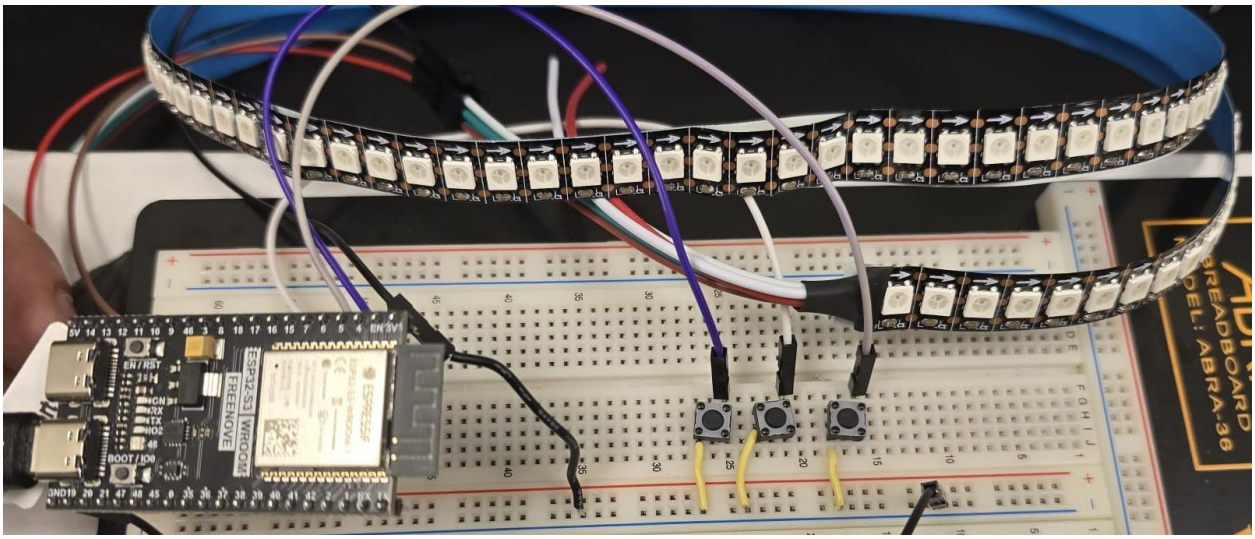


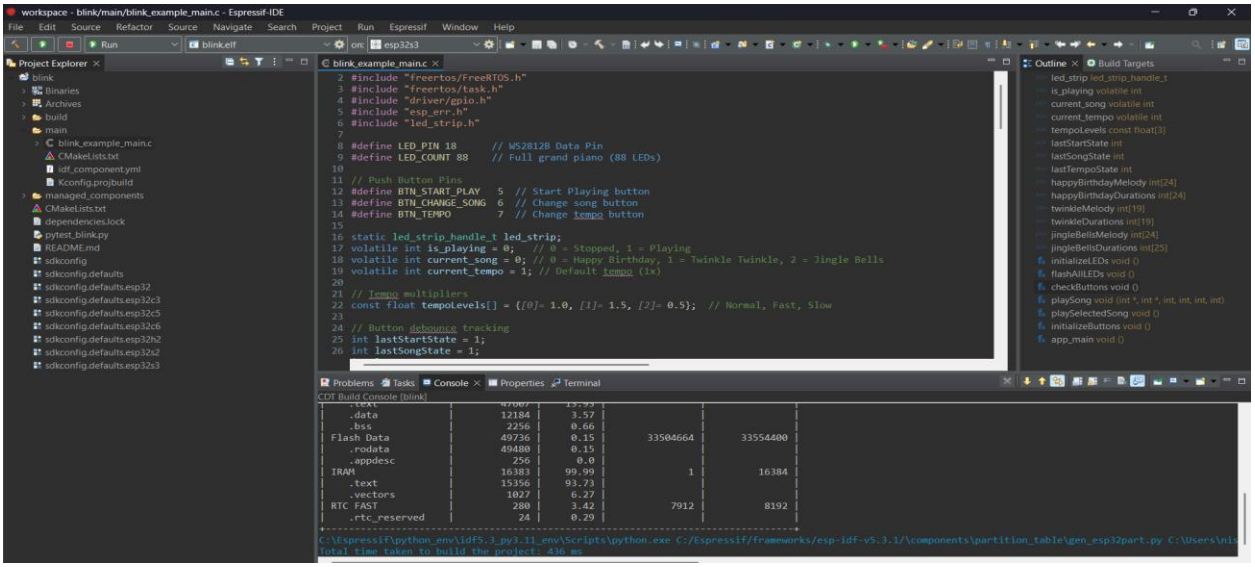
Test Code:
Test code (version 1.0, 1.1, 1.2, and final) has been included in appendix.

Pass/Fail Criteria:
Test Criterion #1
Pressing Button 1 (Start Playback) should initiate song playback without any interruptions or flickering of LEDs.
Test Criterion #2
Pressing Button 2 (Change Song) should instantly switch to the next song in the sequence and flash all LEDs in White before the new song begins.
Test Criterion #3
Pressing **Button 3 (Change Tempo)** should cycle through the three tempo levels (Normal, Fast, Slow) and flash all LEDs at once & play sequentially based on the selected speed.
Test Criterion #4
Each song should play with its **assigned LED color** (Happy Birthday - **Red**, Twinkle Twinkle - **Green**, Jingle Bells - **Blue**) throughout playback.
Test Criterion #5
When playback stops, all LEDs should turn off until a song is restarted.

Apparatus:
1. ESP32 - S3 – WROOM Development board Model: ESP32 - S3 – WROOM – 1
2. WS2812B LED strip Model #: WS2812B 5V Addressable LED Strip
3. Pushbuttons
4. Breadboard
5. Jumper Wires

Procedure:	
Step 1:	Breadboard connected to **laptop via USB cable** and **5V power supply** as shown in the circuit diagram.
Step 2:	Run test code by opening the **ESP-IDF terminal** and executing:
Step 3:	Attempt to initiate **song playback** by depressing **PUSH button #1 (GPIO 5)**. The **first song should start playing**, and LEDs should light up sequentially based on the melody.
Step 4:	Attempt to **change the song** by depressing **PUSH button #2 (GPIO 6)**.
Step 5:	Attempt to **change the tempo** by depressing **PUSH button #3 (GPIO 7)**.
Step 6:	Monitor ESP32 serial output for correct button responses and system logs, confirm that each button press registers correctly and that the LED transitions occur without flickering or delay.





Results:
Test Criterion #1: Pressing Button 1 (Start Playback) should initiate song playback without any interruptions or flickering of LEDs. Outcome: Upon pressing Button 1 (GPIO 5), the selected song started playing instantly. LEDs lit up sequentially without flickering, and playback remained smooth throughout.
Test Criterion #2: Pressing Button 2 (Change Song) should instantly switch to the next song in the sequence and flash all LEDs in White before the new song begins. Outcome: Upon pressing Button 2 (GPIO 6), the current song immediately stopped, all LEDs flashed White for 500ms, and the next song in the sequence started playing instantly.
Test Criterion #3: Pressing Button 3 (Change Tempo) should cycle through the three tempo levels (Normal, Fast, Slow) and flash all LEDs & start lighting up based on the selected speed. Outcome: Upon pressing Button 3 (GPIO 7), the tempo cycled correctly between Normal (1x), Fast (1.5x), and Slow (0.5x). LEDs flashed at once for 500ms before continuing playback at the new tempo.
Test Criterion #4: Each song should play with its assigned LED color (Happy Birthday - Blue, Twinkle Twinkle - Green, Jingle Bells - Red) throughout playback. Outcome: Each song played with the correctly assigned LED color throughout playback: • Happy Birthday → Blue (0, 0, 255) • Twinkle Twinkle → Green (0, 255, 0) • Jingle Bells → Red (255, 0, 0)
Test Criterion #5: When playback stops, all LEDs should turn OFF until a song is restarted. Outcome: When playback was stopped or power was cycled, all LEDs turned OFF immediately and remained off until Button 1 was pressed again to start playback.

Analysis of Results:
1 Button 1 (Start Playback) worked as expected, starting the song instantly with smooth LED transitions and no flickering. 2 Button 2 (Change Song) switched songs immediately, flashing all LEDs white for 500ms before playing the next song. 3 Button 3 (Change Tempo) correctly cycled through speeds, flashing the LEDs altogether to indicate Normal, Fast, or Slow tempo. 4 Each song played with its assigned LED color (Red, Green, or Blue), confirming proper LED mapping and smooth transitions. 5 Playback stopped, or power cycled turned all LEDs off, resetting the system correctly. **Final Analysis:** The ESP32 system worked as expected, with accurate button response, smooth LED transitions, and no unexpected issues.

Assessment:		
Test Criterion	**Verdict (Pass/Fail)**	**Comments, Rationale, and/or Supporting Documentation**
Test Criterion #1	Pass	Song started instantly; LEDs worked fine.
Test Criterion #2	Pass	Song changed immediately; LEDs flashed White.
Test Criterion #3	Pass	Tempo changed correctly, LEDs flashed on speed change.
Test Criterion #4	Pass	Songs played with correct colors: Red, Green, or Blue.
Test Criterion #5	Pass	All LEDs turned OFF when playback stopped or restarted.

Conclusion:

The test successfully demonstrated the desired functionality of the ESP32-based LED piano assistant, with all buttons responding correctly and LEDs behaving as expected. Songs played with the correct colors, tempo changes worked smoothly, and playback started and stopped as intended. There were no unexpected issues, and the system performed reliably without flickering, lag, or crashes. Testing is complete for this part of the project.

Future Actions:

Manan and Viral will begin working on the LCD interface to display song titles, tempo settings, and playback status. Additionally, they will focus on hardware integration to seamlessly connect the system to the prebuilt prototype, ensuring all components function together as intended.

Appendix A – Test Code Version 1.0

Test code 1.0

```
/*==========================================================================
   File Name:    Piano_LED_ESP32.c
   Author:       Nisha Desai
   Date:         07/03/2025
   Modified:     None

   Description:  This program controls a WS2812 LED strip using an ESP32-S3
                 microcontroller. The system allows users to blink all LEDs
                 in RED, GREEN, and BLUE using Start and Stop buttons.
==========================================================================*/

// Libraries ================================================================
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_err.h"
#include "led_strip.h"

// Constants ================================================================
#define LED_PIN        18  // WS2812B Data Pin
#define LED_COUNT      80  // Number of LEDs in the strip
#define DELAY_MS       500 // Delay between color changes (ms)

// Push Button Pins
#define BTN_START      5   // Start Blinking
#define BTN_STOP       6   // Stop Blinking

// Global Variables =========================================================
static led_strip_handle_t led_strip;
volatile int is_running = 0;  // 1 = Blinking, 0 = Stopped

/*>>> button_isr_handler: ==================================================
Author:       Nisha Desai
Date:         07/03/2025
Description:  Interrupt handler for start and stop buttons.
Input:        void* arg - Button pin number
Returns:      None
==========================================================================*/
void IRAM_ATTR button_isr_handler(void* arg) {
   int button = (int) arg;
   if (button == BTN_START) {
      is_running = 1;
   } else if (button == BTN_STOP) {
```

```
        is_running = 0;
    }
}

/*>>> initializeLEDs: ============================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Configures the WS2812 LED strip and clears initial state.
Input:       None
Returns:     None
================================================================================*/
void initializeLEDs() {
    led_strip_config_t strip_config = {
        .strip_gpio_num = LED_PIN,
        .max_leds = LED_COUNT,
        .led_pixel_format = LED_PIXEL_FORMAT_GRB,
        .led_model = LED_MODEL_WS2812,
    };
    led_strip_rmt_config_t rmt_config = {
        .clk_src = RMT_CLK_SRC_DEFAULT,
        .resolution_hz = 10 * 1000 * 1000,
        .flags.with_dma = false,
    };
    ESP_ERROR_CHECK(led_strip_new_rmt_device(&strip_config, &rmt_config, &led_strip));
    ESP_ERROR_CHECK(led_strip_clear(led_strip));
}

/*>>> setAllLEDs: ================================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Sets all LEDs to a specified color.
Input:       uint8_t red, uint8_t green, uint8_t blue
Returns:     None
================================================================================*/
void setAllLEDs(uint8_t red, uint8_t green, uint8_t blue) {
    for (int i = 0; i < LED_COUNT; i++) {
        led_strip_set_pixel(led_strip, i, red, green, blue);
    }
    led_strip_refresh(led_strip);
}

/*>>> blinkAllLEDs: ==============================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Continuously blinks all LEDs in RED, GREEN, and BLUE when started.
Input:       None
Returns:     None
```

```
================================================================================*/
void blinkAllLEDs() {
   while (1) {
      if (is_running) {
         setAllLEDs(255, 0, 0); // RED
         vTaskDelay(pdMS_TO_TICKS(DELAY_MS));
         setAllLEDs(0, 255, 0); // GREEN
         vTaskDelay(pdMS_TO_TICKS(DELAY_MS));
         setAllLEDs(0, 0, 255); // BLUE
         vTaskDelay(pdMS_TO_TICKS(DELAY_MS));
      } else {
         ESP_ERROR_CHECK(led_strip_clear(led_strip));
         vTaskDelay(pdMS_TO_TICKS(100));
      }
   }
}
/*>>> initializeButtons: ================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Configures push buttons and enables interrupts.
Input:       None
Returns:     None
================================================================================*/
void initializeButtons() {
   gpio_config_t btn_config = {
      .pin_bit_mask = (1ULL << BTN_START) | (1ULL << BTN_STOP),
      .mode = GPIO_MODE_INPUT,
      .pull_up_en = GPIO_PULLUP_ENABLE,
      .intr_type = GPIO_INTR_NEGEDGE
   };
   gpio_config(&btn_config);
   gpio_install_isr_service(0);
   gpio_isr_handler_add(BTN_START, button_isr_handler, (void*) BTN_START);
   gpio_isr_handler_add(BTN_STOP, button_isr_handler, (void*) BTN_STOP);
}
/*>>> app_main: ========================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Main function to initialize system and start LED blinking.
Input:       None
Returns:     None
================================================================================*/
void app_main() {
   initializeLEDs();
   initializeButtons();
   blinkAllLEDs();
}
```

Appendix B – Test Code Version 1.1

```
/*==========================================================================
  File Name:   Piano_LED_ESP32.c
  Author:      Nisha Desai
  Date:        07/03/2025
  Modified:    None

  Description:  This program controls a WS2812 LED strip using an ESP32-S3
          microcontroller. The system allows users to play and pause
          an LED-based 'Happy Birthday' song representation. The LEDs
          light up in a sequence corresponding to musical notes.
==========================================================================*/

// Libraries ===============================================================
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_err.h"
#include "led_strip.h"

// Constants ===============================================================
#define LED_PIN       18  // WS2812B Data Pin
#define LED_COUNT     12  // Number of LEDs in the strip
#define NOTE_DELAY    100 // Delay between notes (ms)

// Push Button Pins
#define BTN_PLAY_PAUSE  5  // Play/Pause Button
#define BTN_HAPPY_BDAY  6  // Happy Birthday Song Button

// Global Variables ========================================================
static led_strip_handle_t led_strip;
volatile int is_paused = 1;  // 1 = Paused, 0 = Playing
volatile int play_happy_birthday = 0;  // 1 = Play, 0 = Stop

// Happy Birthday melody (LED index)
const int melody[] = {0, 0, 2, 0, 5, 4, 0, 0, 2, 0, 7, 5, 0, 0, 9, 5, 4, 2, 11, 11, 9, 5, 7, 5};

// Note durations (ms)
const int noteDurations[] = {500, 500, 500, 500, 500, 1000, 500, 500, 500, 500, 500, 1000,
                             500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 1000};

/*>>> button_isr_handler: ================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Interrupt handler for push buttons.
```

```
Input:      void* arg - Button pin number
Returns:    None
==========================================================================*/
void IRAM_ATTR button_isr_handler(void* arg) {
   int button = (int) arg;
   if (button == BTN_PLAY_PAUSE) {
      is_paused = !is_paused;  // Toggle play/pause state
   } else if (button == BTN_HAPPY_BDAY) {
      play_happy_birthday = 1;  // Start Happy Birthday song
      is_paused = 0;  // Ensure auto-play
   }
}

/*>>> initializeLEDs: ======================================================
Author:     Nisha Desai
Date:       07/03/2025
Description:  Configures the WS2812 LED strip and clears initial state.
Input:      None
Returns:    None
==========================================================================*/
void initializeLEDs() {
   led_strip_config_t strip_config = {
      .strip_gpio_num = LED_PIN,
      .max_leds = LED_COUNT,
      .led_pixel_format = LED_PIXEL_FORMAT_GRB,
      .led_model = LED_MODEL_WS2812,
   };
   led_strip_rmt_config_t rmt_config = {
      .clk_src = RMT_CLK_SRC_DEFAULT,
      .resolution_hz = 10 * 1000 * 1000,  // 10MHz resolution
      .flags.with_dma = false,
   };
   ESP_ERROR_CHECK(led_strip_new_rmt_device(&strip_config, &rmt_config, &led_strip));
   ESP_ERROR_CHECK(led_strip_clear(led_strip));
}

/*>>> playNote: ============================================================
Author:     Nisha Desai
Date:       07/03/2025
Description:  Turns on a specific LED in yellow and turns it off after duration.
Input:      int index - LED index, int duration - duration to hold LED on
Returns:    None
==========================================================================*/
void playNote(int index, int duration) {
   if (index >= 0 && index < LED_COUNT) {
      led_strip_set_pixel(led_strip, index, 255, 255, 0);  // Yellow color
      led_strip_refresh(led_strip);
```

```
    vTaskDelay(pdMS_TO_TICKS(duration));  // Hold note duration
    led_strip_set_pixel(led_strip, index, 0, 0, 0);  // Turn off
    led_strip_refresh(led_strip);
    vTaskDelay(pdMS_TO_TICKS(NOTE_DELAY));  // Small delay between notes
  }
}

/*>>> playHappyBirthday: ================================================
Author:     Nisha Desai
Date:       07/03/2025
Description:  Plays 'Happy Birthday' melody with LED effects.
Input:      None
Returns:    None
========================================================================*/
void playHappyBirthday() {
  while (play_happy_birthday) {
    for (int i = 0; i < sizeof(melody) / sizeof(melody[0]); i++) {
      if (is_paused) return;  // Stop playing if paused
      playNote(melody[i], noteDurations[i]);
    }
    vTaskDelay(pdMS_TO_TICKS(2000));  // Pause before repeating
  }
}

/*>>> playSelectedSong: ==================================================
Author:     Nisha Desai
Date:       07/03/2025
Description:  Continuously checks and plays the selected song.
Input:      None
Returns:    None
========================================================================*/
void playSelectedSong() {
  while (1) {
    if (!is_paused && play_happy_birthday) {
      playHappyBirthday();
    }
    vTaskDelay(pdMS_TO_TICKS(100));  // Check every 100ms
  }
}

/*>>> initializeButtons: =================================================
Author:     Nisha Desai
Date:       07/03/2025
Description:  Configures GPIO push buttons and enables interrupts.
Input:      None
Returns:    None
========================================================================*/
```

```
void initializeButtons() {
   gpio_config_t btn_config = {
      .pin_bit_mask = (1ULL << BTN_PLAY_PAUSE) | (1ULL << BTN_HAPPY_BDAY),
      .mode = GPIO_MODE_INPUT,
      .pull_up_en = GPIO_PULLUP_ENABLE,
      .pull_down_en = GPIO_PULLDOWN_DISABLE,
      .intr_type = GPIO_INTR_NEGEDGE  // Detect button press (falling edge)
   };
   gpio_config(&btn_config);
   gpio_install_isr_service(0);
   gpio_isr_handler_add(BTN_PLAY_PAUSE, button_isr_handler, (void*) BTN_PLAY_PAUSE);
   gpio_isr_handler_add(BTN_HAPPY_BDAY, button_isr_handler, (void*) BTN_HAPPY_BDAY);
}

/*>>> app_main: ================================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Main function to initialize system and start LED-based song playback.
Input:       None
Returns:     None
==============================================================================*/
void app_main() {
   initializeLEDs();
   initializeButtons();
   playSelectedSong();
}
```

Appendix C – Test Code Version 1.2

```
/*=========================================================================
   File Name:    Piano_LED_ESP32.c
   Author:       Nisha Desai
   Date:         07/03/2025
   Modified:     None

   Description:  This program controls a WS2812 LED strip using an ESP32-S3
                 microcontroller. The system allows users to play LED-based
                 'Happy Birthday' and 'Twinkle Twinkle Little Star' songs.
                 The LEDs light up in a sequence corresponding to musical notes.
=========================================================================*/

// Libraries ===============================================================
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_err.h"
#include "led_strip.h"

// Constants ===============================================================
#define LED_PIN        18  // WS2812B Data Pin
#define LED_COUNT      12  // Number of LEDs in the strip
#define NOTE_DELAY     100 // Delay between notes (ms)

// Push Button Pins
#define BTN_HAPPY_BDAY  5  // Happy Birthday Song Button
#define BTN_TWINKLE     6  // Twinkle Twinkle Song Button

// Global Variables ========================================================
static led_strip_handle_t led_strip;
volatile int selected_song = 0;  // 1 = Happy Birthday, 2 = Twinkle Twinkle

// Happy Birthday melody (LED index)
const int happyBirthdayMelody[] = {0, 0, 2, 0, 5, 4, 0, 0, 2, 0, 7, 5, 0, 0, 9, 5, 4, 2, 11, 11, 9, 5, 7, 5};
const int happyBirthdayDurations[] = {500, 500, 500, 500, 500, 1000, 500, 500, 500, 500, 500, 1000,
                                      500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 1000};

// Twinkle Twinkle melody (LED index)
const int twinkleMelody[] = {0, 0, 4, 4, 5, 5, 4, 2, 2, 1, 1, 0};
const int twinkleDurations[] = {500, 500, 500, 500, 500, 1000, 500, 500, 500, 500, 500, 1000};

/*>>> button_isr_handler: ================================================
Author:       Nisha Desai
Date:         07/03/2025
```

```
Description:  Interrupt handler for push buttons.
Input:       void* arg - Button pin number
Returns:     None
==========================================================================*/
void IRAM_ATTR button_isr_handler(void* arg) {
   int button = (int) arg;
   if (button == BTN_HAPPY_BDAY) {
      selected_song = 1;  // Play Happy Birthday
   } else if (button == BTN_TWINKLE) {
      selected_song = 2;  // Play Twinkle Twinkle
   }
}

/*>>> initializeLEDs: ======================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Configures the WS2812 LED strip and clears initial state.
Input:       None
Returns:     None
==========================================================================*/
void initializeLEDs() {
   led_strip_config_t strip_config = {
      .strip_gpio_num = LED_PIN,
      .max_leds = LED_COUNT,
      .led_pixel_format = LED_PIXEL_FORMAT_GRB,
      .led_model = LED_MODEL_WS2812,
   };
   led_strip_rmt_config_t rmt_config = {
      .clk_src = RMT_CLK_SRC_DEFAULT,
      .resolution_hz = 10 * 1000 * 1000,  // 10MHz resolution
      .flags.with_dma = false,
   };
   ESP_ERROR_CHECK(led_strip_new_rmt_device(&strip_config, &rmt_config, &led_strip));
   ESP_ERROR_CHECK(led_strip_clear(led_strip));
}

/*>>> playNote: ============================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Turns on a specific LED with a specified color and turns it off after duration.
Input:       int index - LED index, int duration - duration to hold LED on, int red, int green, int blue
Returns:     None
==========================================================================*/
void playNote(int index, int duration, int red, int green, int blue) {
   if (index >= 0 && index < LED_COUNT) {
      led_strip_set_pixel(led_strip, index, red, green, blue);
      led_strip_refresh(led_strip);
```

```
      vTaskDelay(pdMS_TO_TICKS(duration));
      led_strip_set_pixel(led_strip, index, 0, 0, 0);
      led_strip_refresh(led_strip);
      vTaskDelay(pdMS_TO_TICKS(NOTE_DELAY));
   }
}

/*>>> playSelectedSong: ==========================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Plays the selected song using LED effects.
Input:       None
Returns:     None
================================================================================*/
void playSelectedSong() {
   while (1) {
      if (selected_song == 1) {
         for (int i = 0; i < sizeof(happyBirthdayMelody) / sizeof(happyBirthdayMelody[0]); i++) {
            if (selected_song != 1) return;
            playNote(happyBirthdayMelody[i], happyBirthdayDurations[i], 255, 0, 0);
         }
      } else if (selected_song == 2) {
         for (int i = 0; i < sizeof(twinkleMelody) / sizeof(twinkleMelody[0]); i++) {
            if (selected_song != 2) return;
            playNote(twinkleMelody[i], twinkleDurations[i], 0, 255, 0);
         }
      }
      vTaskDelay(pdMS_TO_TICKS(100));
   }
}
/*>>> app_main: ==================================================================
Author:      Nisha Desai
Date:        07/03/2025
Description:  Main function to initialize system and start LED-based song playback.
Input:       None
Returns:     None
================================================================================*/
void app_main() {
   initializeLEDs();
   gpio_config_t btn_config = {
      .pin_bit_mask = (1ULL << BTN_HAPPY_BDAY) | (1ULL << BTN_TWINKLE),
      .mode = GPIO_MODE_INPUT,
      .pull_up_en = GPIO_PULLUP_ENABLE,
      .pull_down_en = GPIO_PULLDOWN_DISABLE,
      .intr_type = GPIO_INTR_NEGEDGE
   };
   gpio_config(&btn_config);
```

```
   gpio_install_isr_service(0);
   gpio_isr_handler_add(BTN_HAPPY_BDAY, button_isr_handler, (void*) BTN_HAPPY_BDAY);
   gpio_isr_handler_add(BTN_TWINKLE, button_isr_handler, (void*) BTN_TWINKLE);
   playSelectedSong();
}
```

Appendix D – Test Code Version Final

```
/*===========================================================================
   File Name: Piano_LED_ESP32.c
   Author: Nisha Desai
   Date: 07/03/2025
   Modified: 08/03/2025

   Description:  This program implements an ESP32-S3-based LED Piano Assistant. It uses an 88-LED WS2812 strip
   to represent a full grand piano. The system allows users to play three pre-programmed songs:
   'Happy Birthday', 'Twinkle Twinkle Little Star', and 'Jingle Bells'. The tempo of the songs
   can be adjusted with a button, and LED colors are mapped to different songs.
===========================================================================*/

// Libraries ================================================================
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_err.h"
#include "led_strip.h"

// Constants ================================================================
#define LED_PIN 18  // WS2812B Data Pin
#define LED_COUNT 88  // Full grand piano (88 LEDs)

// Push Button Pins
#define PLAY_BUTTON 5  // Play/Pause button
#define SONG_BUTTON 6  // Change Song button
#define TEMPO_BUTTON 7  // Change Tempo button

// Global Variables =========================================================
static led_strip_handle_t led_strip;
volatile bool isPlaying = false;
volatile int currentSong = 0;  // 0 = Happy Birthday, 1 = Twinkle Twinkle, 2 = Jingle Bells
volatile int currentTempo = 0; // 0 = Slow, 1 = Medium, 2 = Fast
volatile bool songChanged = false;
const float tempoLevels[] = {0.5, 1.0, 1.5};
```

```
// Song Data ===================================================================
const int happyBirthdayMelody[] = {0, 0, 2, 0, 5, 4, 0, 0, 2, 0, 7, 5, 0, 0, 9, 5, 4, 2, 11, 11, 9, 5, 7, 5};
const int twinkleMelody[] = {0, 0, 7, 7, 9, 9, 7, 5, 5, 4, 4, 2, 7, 7, 5, 5, 4, 4, 2};
const int jingleBellsMelody[] = {4, 4, 4, 4, 4, 4, 4, 2, 4, 7, 0, 1, 4, 4, 4, 4, 4, 4, 4, 2, 4, 0, 7, 5};

/*>>> initializeLEDs: ========================================================
   Author:      Nisha Desai
   Date:        07/03/2025
   Modified:    08/03/2025
   Description:  Configures the WS2812 LED strip and clears initial state.
   Input:       None
   Returns:     None
==============================================================================*/
void initializeLEDs() {
   led_strip_config_t strip_config = {
      .strip_gpio_num = LED_PIN,
      .max_leds = LED_COUNT,
      .led_pixel_format = LED_PIXEL_FORMAT_GRB,
      .led_model = LED_MODEL_WS2812,
   };
   led_strip_rmt_config_t rmt_config = {
      .clk_src = RMT_CLK_SRC_DEFAULT,
      .resolution_hz = 10 * 1000 * 1000,
      .flags.with_dma = false,
   };
   ESP_ERROR_CHECK(led_strip_new_rmt_device(&strip_config, &rmt_config, &led_strip));
   ESP_ERROR_CHECK(led_strip_clear(led_strip));
}

/*>>> flashAllLEDs: ==========================================================
   Author:      Nisha Desai
   Date:        07/03/2025
      Description:  Flashes all LEDs in a given color for visual feedback.
   Input:       int red, int green, int blue - RGB color values
   Returns:     None
==============================================================================*/
void flashAllLEDs(int red, int green, int blue) {
   for (int i = 0; i < LED_COUNT; i++) {
      led_strip_set_pixel(led_strip, i, red, green, blue);
   }
   led_strip_refresh(led_strip);
   vTaskDelay(pdMS_TO_TICKS(500));
   ESP_ERROR_CHECK(led_strip_clear(led_strip));
}

/*>>> playButtonHandler: ======================================================
   Author:      Nisha Desai
```

```
   Date:       07/03/2025
   Description:  ISR for the Play/Pause button.
   Input:      void* arg - Button pin number
   Returns:    None
==========================================================================*/
void IRAM_ATTR playButtonHandler(void* arg) {
   isPlaying = !isPlaying;
}

/*>>> songButtonHandler: ==================================================
   Author:     Nisha Desai
   Date:       07/03/2025
   Description:  ISR for changing the song. Cycles through the available songs.
   Input:      void* arg - Button pin number
   Returns:    None
==========================================================================*/
void IRAM_ATTR songButtonHandler(void* arg) {
   currentSong = (currentSong + 1) % 3;
   flashAllLEDs(255, 255, 255);  // White flash
   songChanged = true;
   isPlaying = true;
}

/*>>> tempoButtonHandler: =================================================
   Author:     Nisha Desai
   Date:       07/03/2025
   Description:  ISR for changing the playback tempo.
   Input:      void* arg - Button pin number
   Returns:    None
==========================================================================*/
void IRAM_ATTR tempoButtonHandler(void* arg) {
   currentTempo = (currentTempo + 1) % 3;
   flashAllLEDs(255, 255, 255);  // White flash
}

/*>>> playSong: ===========================================================
   Author:     Nisha Desai
   Date:       07/03/2025
   Description:  Plays the given melody with specified LED colors and tempo.
   Input:      const int* melody - Array of LED indices for melody
               const int* durations - Array of note durations
               int length - Number of notes in the melody
               int red, int green, int blue - RGB color values for LEDs
   Returns:    None
==========================================================================*/
void playSong(const int *melody, const int *durations, int length, int red, int green, int blue) {
   for (int i = 0; i < length; i++) {
```

```
    if (!isPlaying || songChanged) {
      songChanged = false;
      return;
    }
    led_strip_set_pixel(led_strip, melody[i], red, green, blue);
    led_strip_refresh(led_strip);
    vTaskDelay(pdMS_TO_TICKS(durations[i] * tempoLevels[currentTempo]));
    led_strip_set_pixel(led_strip, melody[i], 0, 0, 0);
    led_strip_refresh(led_strip);
  }
}

/*>>> playSelectedSong: ====================================================
  Author:      Nisha Desai
  Date:        07/03/2025
  Description:  Main loop to continuously play selected song based on user input.
  Input:       None
  Returns:     None
==========================================================================*/
void playSelectedSong() {
  while (1) {
    if (isPlaying) {
      if (currentSong == 0) {
        playSong(happyBirthdayMelody, happyBirthdayDurations, 24, 255, 0, 0);  // Red
      } else if (currentSong == 1) {
        playSong(twinkleMelody, twinkleDurations, 19, 0, 255, 0);  // Green
      } else {
        playSong(jingleBellsMelody, jingleBellsDurations, 25, 0, 0, 255);  // Blue
      }
    } else {
      ESP_ERROR_CHECK(led_strip_clear(led_strip));
    }
    vTaskDelay(pdMS_TO_TICKS(100));
  }
}

/*>>> initializeButtons: ===================================================
  Author:      Nisha Desai
  Date:        07/03/2025
  Description:  Configures GPIO push buttons and enables interrupts.
  Input:       None
  Returns:     None
==========================================================================*/
void initializeButtons() {
  gpio_config_t btn_config = {
    .pin_bit_mask = (1ULL << PLAY_BUTTON) | (1ULL << SONG_BUTTON) | (1ULL <<
TEMPO_BUTTON),
```

```
      .mode = GPIO_MODE_INPUT,
      .pull_up_en = GPIO_PULLUP_ENABLE,
      .intr_type = GPIO_INTR_POSEDGE
   };
   gpio_config(&btn_config);
   gpio_install_isr_service(0);
   gpio_isr_handler_add(PLAY_BUTTON, playButtonHandler, NULL);
   gpio_isr_handler_add(SONG_BUTTON, songButtonHandler, NULL);
   gpio_isr_handler_add(TEMPO_BUTTON, tempoButtonHandler, NULL);
}

/*>>> app_main: ================================================================
   Author:      Nisha Desai
   Date:        07/03/2025
   Description:  Main function to initialize the system and start the loop.
   Input:       None
   Returns:     None
==============================================================================*/
void app_main() {
   initializeLEDs();
   initializeButtons();
   playSelectedSong();
}
```