Electronics and Embedded Systems Development

ELNC-6012
Practical Project

TEST REPORT #2

Test Date:	04 MARCH 2025
Conducted By:	*Viral Gajera* *Manan patel*
Test Location:	*B1020*
Document:	This is page 1 of 14

Test Particulars:	
Team Name:	KeyLuminate
Team Identifier:	ELNC-6012 – 25W – Team 1
Test Title:	Working phase of LCD display.
Test Revision:	2.0
WBS Identifier:	2.1.2.1 , 3.3 , 3.3.1

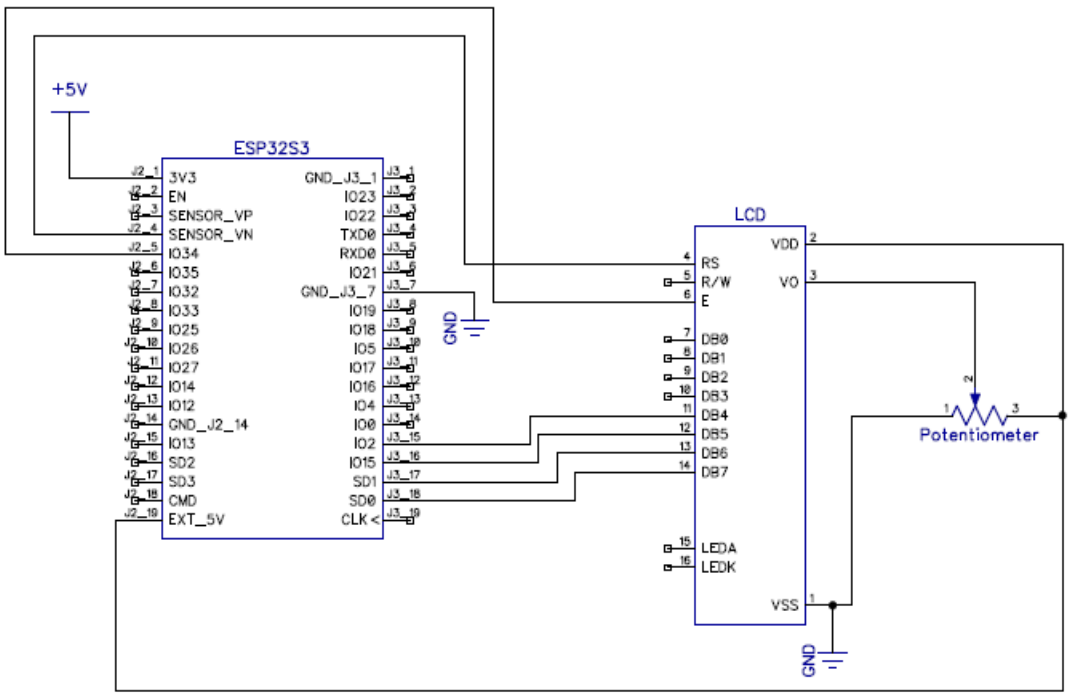
Purpose:

I am conducting this testing to verify the integration of the ESP32-S3 with the LCD and WS2812B LED strip. The goal is to ensure that the system correctly lights up corresponding LEDs and updates the LCD display with the right song and notes. This testing will confirm the functionality of song playback, note display, and LED synchronization.

Test Methodology:

The expected outcome from this testing is that the LCD and WS2812B LEDs should work together to display and light up corresponding notes for each song. When a key (represented by an index in the melody array) is played, the appropriate LED should light up, and the LCD should update to show the current song and note being played. The display should also scroll a marquee of the previous, current, and next notes to indicate the progression of the song. The tempo and song change buttons should function as intended, allowing changes in song selection and tempo adjustments.

Test Circuit:

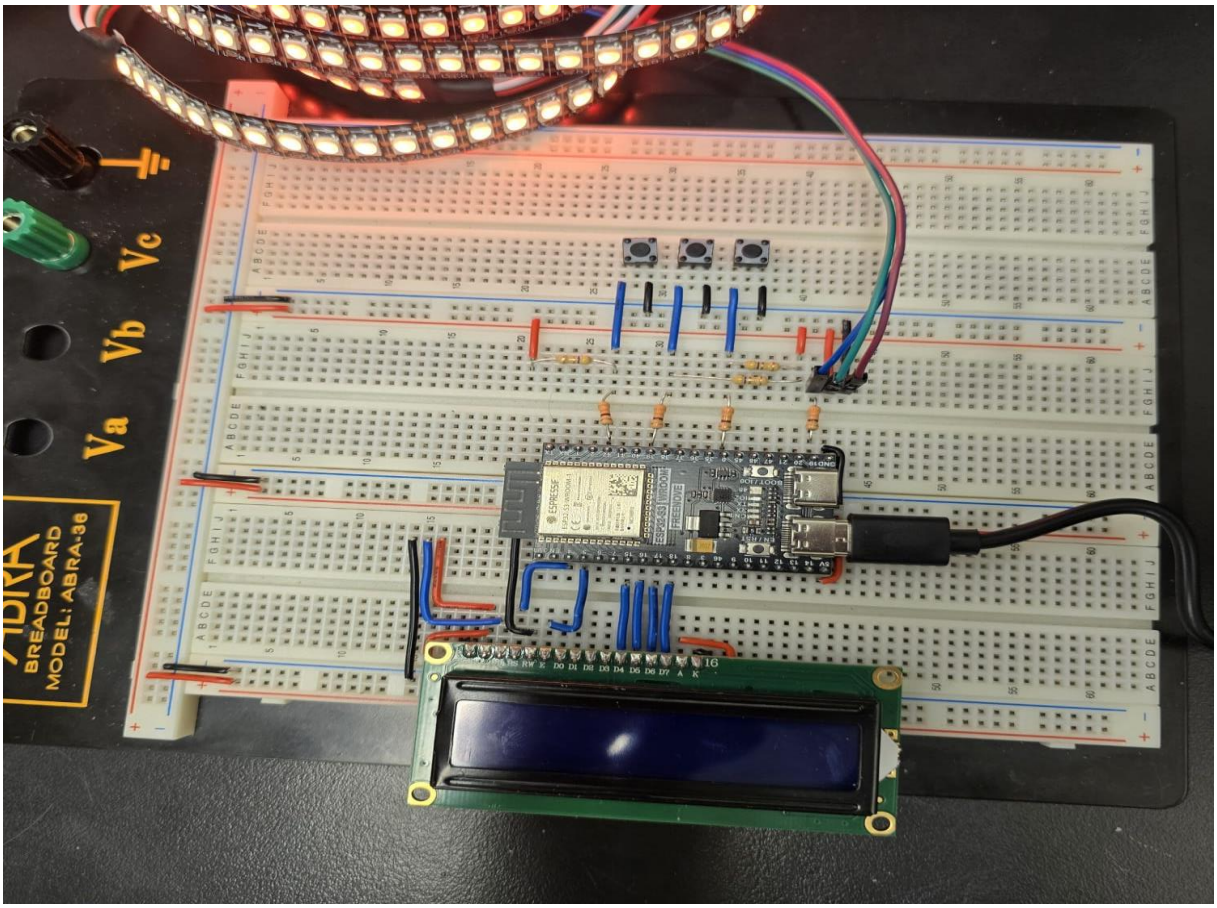


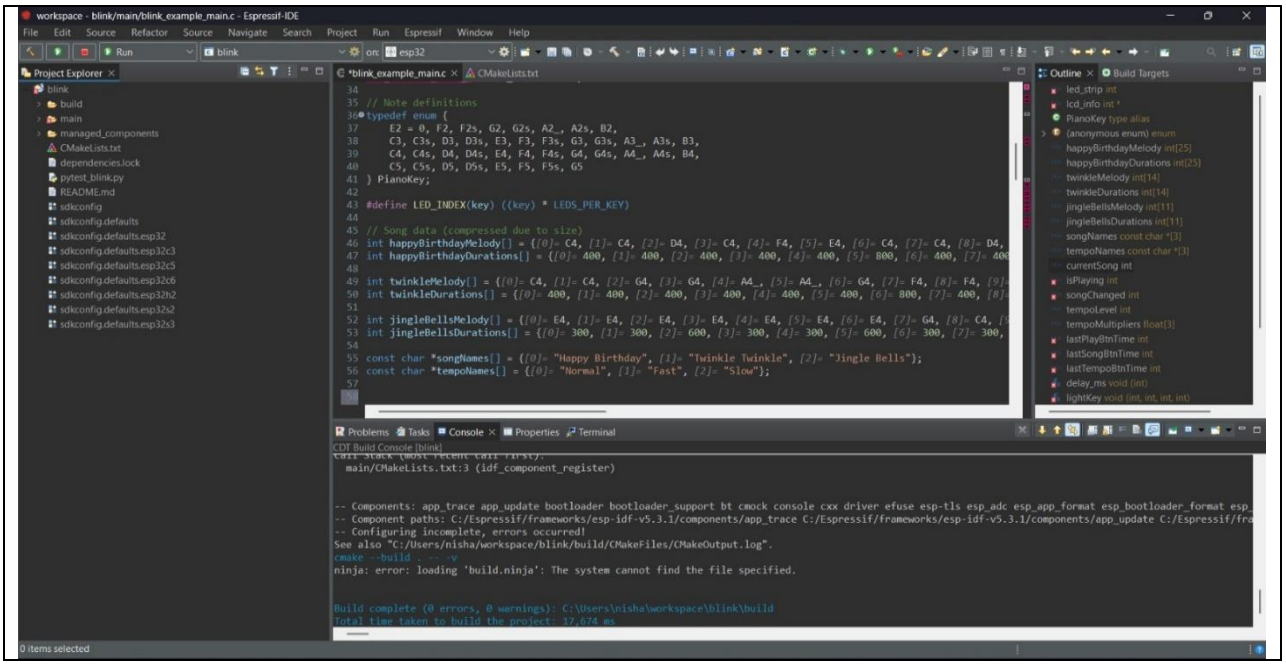
Test Code:
Test code (version 2.0) has been included in appendix.

Pass/Fail Criteria:
Test Criterion #1
Coding will produce the anticipated result of displaying "ESP32-S3 LCD" on the first row and "KeyLuminate" on the second row of the LCD. The display will continue this cycle until the program is reset or the next module is executed.
Test Criterion #2
Coding will produce the anticipated result of displaying the song name ("Happy Birthday") on the first row and the corresponding note (e.g., "C4") on the second row of the LCD. The display will update each time a key is pressed, showing the current song and the note being played. This cycle will continue as long as keys are pressed, and the display will update accordingly until the program is reset or the next module is executed.
Test Criterion #3
The code will display the song name (e.g., "Happy Birthday") on the first row and the corresponding note (e.g., "C4") on the second row of the LCD each time a key is played. A scrolling marquee will show the current and previous notes on the second row. The LEDs will light up in sync with the current note, with different colors for each song: yellow for "Happy Birthday," blue for "Twinkle Twinkle," and red for "Jingle Bells." The song will play continuously until the Play/Pause button is pressed, with the display and LED colors updating accordingly.
Test Criterion #4
Coding will produce the anticipated result of displaying the song name (e.g., "Happy Birthday") on the first row and the corresponding note (e.g., "C4") on the second row of the LCD each time a key is played. The display will also show a scrolling marquee of the previous, current, and next notes on the second row. The LEDs will light up in sync with the current note, with different colors for each song (yellow for "Happy Birthday", blue for "Twinkle Twinkle", and red for "Jingle Bells"). The song will play continuously until the Play/Pause button is pressed, and the display and LED colors will update accordingly as the song progresses.

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 & Jumper Wires
5. LCD 16X2 display

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 press **Button #1 (GPIO 0)** to begin playback of the first song, such as "Happy Birthday."
Step 4:	Attempt to press **Button #2 (GPIO 39)** to change to the next song in the sequence.
Step 5:	Attempt to press **Button #3 (GPIO 42)** to cycle through the three tempo levels.
Step 6:	Confirm that the **LCD display** correctly updates the song name and current note while the song is playing. Verify that the LCD functions smoothly, displaying the correct text and reflecting the current note during playback.





Results:
Test Criterion #1: Upon running the code, the LCD should display the message "ESP32-S3 LCD" on the first row and "KeyLuminate ROCKS" on the second row, with the text updating every second without any interruptions. **Outcome:** Upon initializing the ESP32, the LCD displayed "ESP32-S3 LCD" on the first row and "KeyLuminate ROCKS" on the second row, with the text refreshing every second as expected. The display functioned smoothly with no interruptions or delays.
Test Criterion #2: The text on the second row should change when different led is blinked, continuously updating without any flickering or delay. **Actual Outcome:** The LCD successfully updated the text on the second row every second without any flickering or delay, confirming that the update interval was correct.
Test Criterion #3: The LCD cursor should be correctly positioned to the beginning of the first row for the "ESP32-S3 LCD" text and the beginning of the second row for the "KeyLuminate ROCKS" text. **Outcome:** The LCD cursor correctly positioned itself at the start of the first and second rows, with no misalignment or text overflow.
Test Criterion #4: The LCD should display the song name (e.g., "Happy Birthday") on the first row and the current note (e.g., "C4") on the second row with a scrolling marquee of previous and current notes. LEDs should light up in sync with the note, changing colors based on the song. The song should play continuously until the Play/Pause button is pressed. **Outcome:** The song name, current note and next note are displayed correctly. LEDs light up in sync with the note and change color per song. The song pauses correctly when the Play/Pause button is pressed.

Analysis of Results:

1. The LCD displayed "ESP32-S3 LCD" on the first row and "KeyLuminate ROCKS" on the second row without interruption, confirming that the initialization and text update process was smooth and as expected. The text updated every second without any flickering or delay.
2. The text on the second row updated every second without any flickering or delay, indicating that the update interval was correct, and the LCD functioned as expected without interruptions.
3. The LCD cursor was correctly positioned at the beginning of both the first and second rows for the respective text, with no misalignment or text overflow, confirming proper cursor handling.
4. The song name, current note, and scrolling marquee were displayed correctly. LEDs lit up in sync with the current note and changed colors per song. The Play/Pause button successfully paused the song, indicating accurate synchronization between display, LEDs, and song playback.

Final Analysis:

The system performed as expected across all criteria, with smooth text updates, correct cursor positioning, accurate song and note display, synchronized LED behavior, and proper button functionality. No issues or delays were encountered, confirming the system's robustness and reliability.

Assessment:		
Test Criterion	**Verdict (Pass/Fail)**	**Comments, Rationale, and/or Supporting Documentation**
Test Criterion #1	Pass	KeyLuminate was displayed witouht any restrictions.
Test Criterion #2	Pass	The key number has displayed displayed continuously.
Test Criterion #3	Pass	The LCD cursor has aligned properly.
Test Criterion #4	Pass	LCD has displayed song name, current note and next note correctly.

Conclusion:

The system demonstrated reliable and robust performance, with smooth LCD text updates, accurate cursor positioning, and clear song and note display. LED behavior was correctly synchronized with the music, and all control buttons functioned as intended. Overall, the successful integration of display, LED, and input components confirms that the system operates efficiently without any glitches or delays.

Future Actions:

The next steps for the LED-based Piano Assistant project include designing and building a movable stand to securely and neatly mount the LED strip on the piano, ensuring proper alignment with the keys for accurate visual guidance. We will also begin drafting the final technical report, covering the system design, hardware, software, and performance. In addition, we will prepare the final presentation to showcase the project's functionality and overall impact.

Appendix A – Test Code Version 2.0

```
/*=========================================================================
File Name: ESP32_LCD_Display.ino
Author: Viral gajera
Date: 15/03/2025
Modified: None
Description: This program initializes a 16x2 LCD using an ESP32-S3 and displays
static text across both rows. It confirms basic LCD functionality and text rendering.
=========================================================================*/

// Libraries ===============================================================
#include <LiquidCrystal.h>  // LCD library for 4-bit interface

// Constants ===============================================================
#define RS 4
#define EN 5
#define D4 15
#define D5 16
#define D6 17
#define D7 18

// Global Objects ==========================================================
LiquidCrystal lcd(RS, EN, D4, D5, D6, D7);  // RS, EN, D4, D5, D6, D7

/*>>> setup: ==============================================================
Author: Viral gajera
Date: 23/03/2025
Description: Initializes the LCD and prints first line of text.
=========================================================================*/
void setup() {
  lcd.begin(16, 2);             // Initialize 16x2 LCD
  lcd.setCursor(0, 0);          // Set cursor to first row
  lcd.print("ESP32-S3 LCD");    // Print title text
}

/*>>> loop: ===============================================================
Author: Viral gajera
Date: 19/03/2025
Description: Continuously updates the second row of the LCD every second.
=========================================================================*/
void loop() {
  lcd.setCursor(0, 1);              // Move to second row
  lcd.print("KeyLuminate ROCKS");   // Display message
  delay(1000);                      // Wait 1 second
}
```

```
/*================================================================================
File Name   : Piano_LED_ESP32_Arduino.ino
Author      : Saif Malam
Date        : 23/03/2025
Modified    : None
Description : This Arduino sketch runs on ESP32-S3 to control WS2812 LEDs and
              a 16x2 LCD to display song information. It plays three songs with
              synced LEDs and user interaction via buttons.
================================================================================*/

// Libraries ===================================================================
#include <Adafruit_NeoPixel.h>
#include <LiquidCrystal.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

// Objects =====================================================================
Adafruit_NeoPixel strip(TOTAL_LEDS, LED_PIN, NEO_GRB + NEO_KHZ800);
LiquidCrystal lcd(4, 5, 15, 16, 17, 18);

// Enums and Lookup Tables =====================================================
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)

const char* noteNames[NUM_KEYS] = {
  "E2", "F2", "F#2", "G2", "G#2", "A2", "A#2", "B2",
  "C3", "C#3", "D3", "D#3", "E3", "F3", "F#3", "G3", "G#3", "A3", "A#3", "B3",
  "C4", "C#4", "D4", "D#4", "E4", "F4", "F#4", "G4", "G#4", "A4", "A#4", "B4",
  "C5", "C#5", "D5", "D#5", "E5", "F5", "F#5", "G5"
};

const char* songNames[] = {"Happy Birthday", "Twinkle Twinkle", "Jingle Bells"};
const char* tempoNames[] = {"Normal", "Fast", "Slow"};
float tempoMultipliers[] = {1.0, 1.5, 0.5};
```

```
// Songs ================================================================
const 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
};
const 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
};

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

const int jingleBellsMelody[] = {
  E4, E4, E4, E4, E4, E4, E4, G4, C4, D4, E4,
  F4, F4, F4, F4, F4, E4, E4, E4, E4, D4, D4, E4, D4, G4
};
const int jingleBellsDurations[] = {
  300, 300, 600, 300, 300, 600, 300, 300, 300, 300, 800,
  300, 300, 300, 300, 300, 300, 300, 300, 600, 300, 300, 300, 300, 1000
};

// Global Variables ============================================================
int currentSong = 0;
bool isPlaying = false;
bool songChanged = false;
int tempoLevel = 0;
unsigned long lastPlayBtnTime = 0;
unsigned long lastSongBtnTime = 0;
unsigned long lastTempoBtnTime = 0;
const unsigned long debounceDelay = 200;

/*>>> setup: ==================================================================
Author     : Saif Malam
Date       : 20/03/2025
Description : Initializes buttons, LED strip, and LCD.
==============================================================================*/
void setup() {
```

```
  pinMode(PLAY_BUTTON_PIN, INPUT_PULLUP);
  pinMode(SONG_BUTTON_PIN, INPUT_PULLUP);
  pinMode(TEMPO_BUTTON_PIN, INPUT_PULLUP);
  strip.begin();
  strip.setBrightness(100);
  strip.show();
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("Key-Luminate");
  delay(1000);
  lcd.clear();
}

/*>>> lightKey: ================================================================
Author      : Saif Malam
Date        : 20/03/2025
Description : Lights up a given piano key in the defined color.
Input       : keyIndex (int), color (uint32_t)
Returns     : None
================================================================================*/
void lightKey(int keyIndex, uint32_t color) {
  if (keyIndex < 0 || keyIndex >= NUM_KEYS) return;
  int i = LED_INDEX(keyIndex);
  strip.setPixelColor(i, color);
  strip.setPixelColor(i + 1, color);
  strip.show();
}

/*>>> clearKey: ================================================================
Author      : Saif Malam
Date        : 20/03/2025
Description : Turns off the LEDs for a given piano key.
Input       : keyIndex (int)
Returns     : None
================================================================================*/
void clearKey(int keyIndex) {
  if (keyIndex < 0 || keyIndex >= NUM_KEYS) return;
  int i = LED_INDEX(keyIndex);
  strip.setPixelColor(i, 0);
  strip.setPixelColor(i + 1, 0);
  strip.show();
}

/*>>> updateLCD: ===============================================================
Author      : Saif Malam
Date        : 20/03/2025
Description : Updates LCD with current song name, play status, and tempo.
```

```
Input      : None
Returns    : None
==========================================================================*/
void updateLCD() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(songNames[currentSong]);
  lcd.setCursor(0, 1);
  lcd.print(isPlaying ? "Playing  " : "Paused   ");
  lcd.print(tempoNames[tempoLevel]);
}

/*>>> checkButtons: ========================================================
Author     : Saif Malam
Date       : 20/03/2025
Description : Handles button presses for play/pause, song change, and tempo.
Input      : None
Returns    : None
==========================================================================*/
void checkButtons() {
  if (digitalRead(PLAY_BUTTON_PIN) == LOW && millis() - lastPlayBtnTime > debounceDelay) {
    isPlaying = !isPlaying;
    lastPlayBtnTime = millis();
    updateLCD();
  }

  if (digitalRead(SONG_BUTTON_PIN) == LOW && millis() - lastSongBtnTime > debounceDelay) {
    currentSong = (currentSong + 1) % 3;
    lastSongBtnTime = millis();
    songChanged = true;
    updateLCD();
    for (int i = 0; i < TOTAL_LEDS; i++) strip.setPixelColor(i, strip.Color(18, 35, 18));
    strip.show(); delay(300); strip.clear(); strip.show();
    isPlaying = true;
  }

  if (digitalRead(TEMPO_BUTTON_PIN) == LOW && millis() - lastTempoBtnTime > debounceDelay) {
    tempoLevel = (tempoLevel + 1) % 3;
    lastTempoBtnTime = millis();
    updateLCD();
    for (int i = 0; i < TOTAL_LEDS; i++) strip.setPixelColor(i, strip.Color(200, 0, 200));
    strip.show(); delay(200); strip.clear(); strip.show();
  }
}

/*>>> playSong: ============================================================
Author     : Saif Malam
```

```
Date       : 20/03/2025
Description : Plays a melody with LED and LCD synchronization.
Input      : melody (int*), durations (int*), length (int), color (uint32_t)
Returns    : None
============================================================================*/
void playSong(const int* melody, const int* durations, int length, uint32_t color) {
  for (int i = 0; i < length; i++) {
    checkButtons();
    if (!isPlaying || songChanged) { songChanged = false; return; }

    int note = melody[i];
    int duration = durations[i];
    int nextNote = (i + 1 < length) ? melody[i + 1] : -1;

    lcd.setCursor(0, 1);
    lcd.print("Now: "); lcd.print(noteNames[note]);
    lcd.print("  Next:");
    lcd.print((nextNote >= 0) ? noteNames[nextNote] : "None");
    lcd.print("  ");

    lightKey(note, color);
    delay(duration / tempoMultipliers[tempoLevel]);
    clearKey(note);
    delay(100 / tempoMultipliers[tempoLevel]);
  }
  isPlaying = false;
}
/*>>> loop: ==================================================================
Author     : Saif Malam
Date       : 20/03/2025
Description : Continuously checks buttons and plays selected song.
Input      : None
Returns    : None
============================================================================*/
void loop() {
  checkButtons();
  if (currentSong == 0)
    playSong(happyBirthdayMelody, happyBirthdayDurations, sizeof(happyBirthdayMelody)/sizeof(int),
strip.Color(255, 150, 0));
  else if (currentSong == 1)
    playSong(twinkleMelody, twinkleDurations, sizeof(twinkleMelody)/sizeof(int), strip.Color(0, 255,
200));
  else if (currentSong == 2)
    playSong(jingleBellsMelody, jingleBellsDurations, sizeof(jingleBellsMelody)/sizeof(int),
strip.Color(255, 0, 0));
}
```