Interactive Music System

Authors Name:
Samyuktha V-CB.EN.U4CCE22043
Shri Namrutha S-CB.EN.U4CCE22047
Kajith T-CB.EN.U4CCE22050
Department of electronics and communication Engineering,
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidhyapeetham, India

Abstract* — *This paper introduces an interactive music system leveraging a Tiva TM4C123GH6PM microcontroller for melody playback and LED activation with LCD display. Users select melodies via a keypad interface, with each key mapped to a distinct melody from the system's library. Real-time feedback is facilitated , and the messages are displayed on LCD, providing users with immediate updates on selections and playback status. The system offers an engaging platform for both educational and recreational purposes, encouraging exploration and experimentation with sound and light. Its modular design and user friendly interface make it suitable for a wide range of applications.

Keywords— Tiva TM4C123GH6PM, keypad, LCD.

I. INTRODUCTION

Interactive systems that combine auditory and visual elements have become increasingly popular for both educational and recreational applications. In today's world, integrating technology with education is very crucial in order to help the students better understand complex concepts and foster innovation. Systems like these help the students learn electronics, programming and music theory in a practical way. Interactive music system can be utilized in music therapy to help individuals with mental health issues such as autism, depression, and anxiety. These systems can also be used in physical and cognitive rehabilitation. It can also be used by researchers and developers for the study of sound and light prototyping.

The primary objective of interactive music system is to create an engaging and multifaceted experience that enhances learning, fosters creativity, facilitates intuitive interaction, provides therapeutic benefits and supports innovation. Integrating sound and light in a user-friendly way offers new and innovative learning experience and creative exploration.

The primary components of the system include a tiva tm4c123gh6pm microcontroller, a buzzer, leds, LCDs, and a 4x3 keypad. The microcontroller serves as the central unit which controls the led toggling and illumination and also controls the melody playback based on the user input from the keypad. Users can select a melody from the library to be played. Each melody is associated with specific note frequencies and durations.

The frequencies or the audio signal is generated using PWM (Pulse Width Modulation). This method enables the precise control of frequencies which is necessary for the tone production. In addition PWM also allows efficient use of digital outputs, simulating analog signals without requiring a DAC (Digital to analog convertor).

The rest of the work is described as follows: Chapter II lists a few research works; Chapter III depicts the block diagram of the system; Chapter IV gives a brief description of the components used; Chapter V illustrates the working of the system, chapter VI discusses the results and conclusions derived from the project and finally Chapter VII includes the papers referred to in the project work.

II. RELATED WORKS

The interactive music system has gained significant attention in the recent years, driven by advancements in microcontroller technology and innovative application areas such as education, therapy, and entertainment. This section reviews related works that focuses on the design and implementation of interactive music system, highlighting various approaches and technologies cited in contemporary research.

Jewale et al. (2023) designed a music rhythm LED flashlight circuit intended for entertainment and decoration purposes [1]. This system uses LEDs to synchronize with the music using a microcontroller.

Shaer et al. (2021) designed a system that presented a piano player embedded with microcontrollers, where the system automates piano key actuation based on pre-programmed

musical scores[3]. This represents the potential that embedded system possess to enhance traditional music instruments, offering automated performance capabilities.

Wu et al. (2023) designed a STM32 based music player, focusing on the integration of audio playback capabilities within a compact and efficient microcontroller platform[2]. This research points out the technical challenges and solutions associated with real-time audio processing and user interface integration.

Mehta and Kharote (2014) designed an ARM-7 based MP3 player, highlighting the applications of ARM microcontrollers in portable audio playback devices[5]. This project emphasizes the importance of efficient processing and low power consumption in interactive music system, which are important for portable and wearable applications.

III. BLOCK DIAGRAM

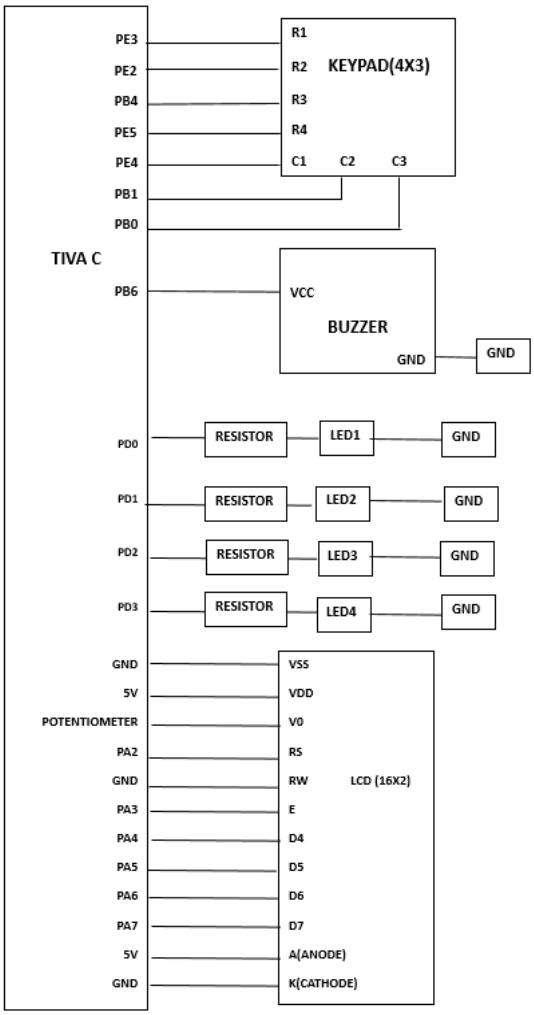

Fig 1. Block diagram of Interactive Music system

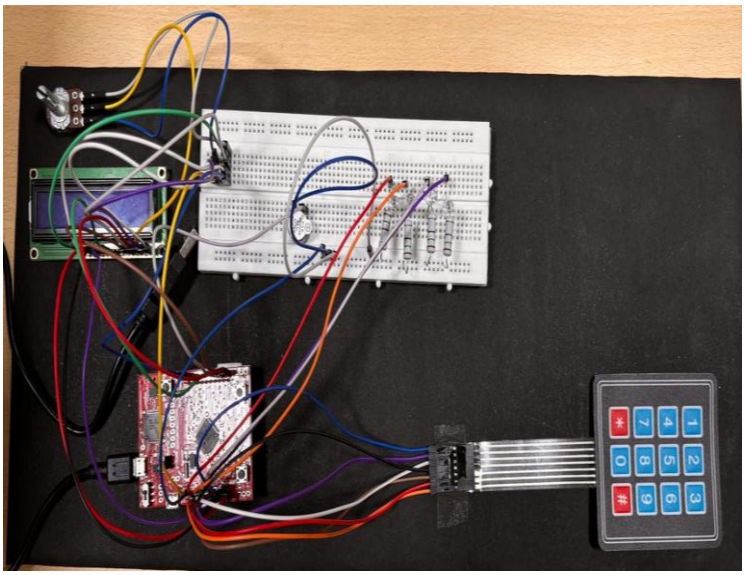

Fig 2. Interactive music system.

IV. BLOCK DIAGRAM DESCRIPTION

This section explains the components used in the implementation of interactive music system which includes Tiva tm4c123gh6pm microcontroller, a buzzer, a 4x3 keypad, LEDs, resistors,16x2 LCD, connecting wires and a breadboard.

A. Tiva TM4C123GH6PM Microcontroller :

• The Tiva TM4C123GH6PM is a microcontroller from Texas Instruments, part of the Tiva C Series.
• The microcontroller features an ARM Cortex-M4 core running at upto 80MHz, with 32 KB of SRAM, 256 KB of flash memory, and various peripherals.
• It is also equipped with 12-bit ADC, multiple UART, I2C, SPI interfaces, PWM modules, and USB 2.0 support.
• This versatile microcontroller is designed for performance, connectivity, and easy integration for diverse applications which includes embedded systems, industrial control, and consumer electronics.
• It has various low power modes to enhance the energy efficiency, making it suitable for battery-operated devices.

B. Energia :

• Energia IDE is an integrated development environment (IDE) designed to facilitate programming and development for microcontrollers, especially those from Texas Instruments (TI).
• It is inspired from Arduino IDE making it user-friendly and easy to use.

• Energia can run on multiple platforms (Windows, macOS, and Linux) which ensures that users can work in their preferred platform.
• Energia provides a pin mapping system that translates the pin numbers used in the sketches to the actual hardware pins of the microcontroller. This allows users to write hardware-agnostic code that can be easily transferred between different microcontrollers supported by Energia.

C. Buzzer:

- Buzzer is an electroacoustic device that is used to produce sound.
- It is many often used in various electronic projects for generating audio alerts or melodies.
- Buzzer converts electrical signals (PWM) into sound.
- It is used to play melodies by generating tones at specific frequencies defined in the code.
- It is connected to the microcontroller on a PWM-capable pin (pin 14).

D. LEDs:

- LEDs (Light Emitting Diodes) are semiconductor devices that emit light when electric current passes through them.
- They are highly efficient, long-lasting and they also come in different colors and sizes.
- LEDs are widely used for display, indication, and illumination purposes.
- The LEDs are connected to dedicated pins in the microcontroller and the toggling effects can be observed in synchronization with the rhythm.

E. Keypad:

- Keypad is an input device which consists a grid of buttons, which are typically arranged in rows and columns..
- It is commonly used in calculators, security system, door locking system, and electronic locks.
- It allows the user to input command data into the system.
- In this project, the keypad allows the user to select different melodies and albums and control the playback.

F. LCD 16x2:

- A 16x2 LCD (Liquid Crystal Display) is a widely used electronic display module which is capable of displaying 16 characters per line on 2 lines, making it suitable for various application.
- Each character is usually displayed in a 5x8 pixel matrix.
- The physical dimensions are around 80mm x 36mm.
- LCD is widely used in televisions, smartphones, infotainment systems, head-up displays, control panels, diagnostic equipment, microwaves and ovens.

G. Potentiometer

- Potentiometer which is often referred as pot, is a three-terminal resistor with an adjustable center tap that functions as an adjustable voltage divider.
- It has two outer terminals (A and B) connected to the ends of the resistive element and one middle terminal (W) connected to the wiper.
- The potentiometer is connected to pin 3(V0) of the LCD to adjust LCD's contrast. Turning the potentiometer alters the voltage applied to V0, thereby changing the contrast.

H. Resistors:

- Resistor is a passive electronic component that resists the flow of electrical current, providing a specific amount of resistance in ohms.
- It is also used to divide voltages, and protect sensitive components by limiting the amount of current that can flow through them.
- Resistors come with color coded banks that indicate their resistance value or tolerance.
- In this project, resistors are used to limit current to the LEDs in order to prevent them from burning out.

I. Connecting wires:

- Connecting wires are essential components required to establish electrical connections between different wires.
- There are various types which include solid core, stranded, jumper wires, each serving a different purpose based on their flexibility.
- It is often color coded to help distinguish different signals or power lines within a circuit, such as red for positive, black for ground, etc.

J. Breadboard:

- Breadboard is a rectangular, plastic board that is used for constructing and testing electronic circuits without soldering.
- It consists of a grid of interconnected holes into which components and connecting wires can be inserted to create circuits.
- These are essential for prototyping and testing circuits due to their reusability and ease of use.

V. WORKING

In the interactive music system, PWM is utilized to control the buzzer, which allows the generation of different frequencies

corresponding to music notes. The system combines hardware and software components to create an engaging user experience allowing users to select and play melodies while synchronizing visual and auditory feedback. To enhance the user experience the albums and songs available are displayed on the LCD along with display of the song being currently played.

A. Hardware setup and components

- The buzzer is connected to PWM pin 14 (PB6), which is responsible for producing sound according to the music notes.
- Four LEDs are connected to pins 23,24,25,26 (PD0, PD1, PD2, PD3) which is used to provide visual feedback and synchronization with the music.
- A 4x3 keypad is used for user interaction where the rows and columns are connected to pins 29, 28, 7, 6 (PE3, PE2, PB4, PE5) and pins 5, 4, 3 (PE4, PB1, PB0) respectively.
- The resistors are used to limit the current flowing to the LEDs.
- The LCD connections are given as below:

LCD pin	**Tiva TM4C123GH6PM pin**
VSS	GND
VDD	5V
V0	Potentiometer
RS	PA2
RW	GND
E	PA3
D4	PA4
D5	PA5
D6	PA6
D7	PA7
A (Anode)	5V
K (Cathode)	GND

Table 1. LCD pin configurations

- The system is controlled by the microcontroller Tiva TM4C123GH6PM which processes the user input and controls the buzzer and LED and displays the melody being played on the LCD.

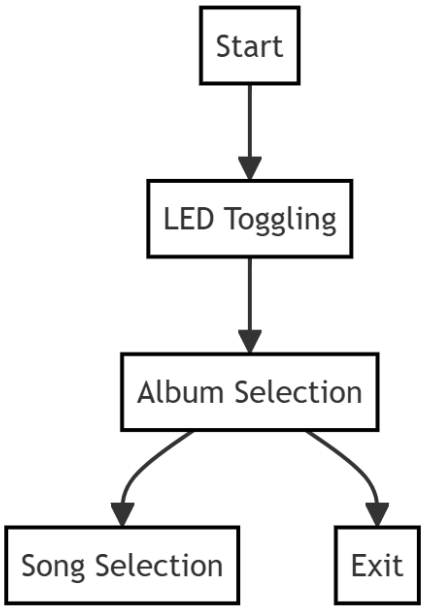

Fig 3.Flowchart of system working

B. Melody storage and selection

- The system stores several melodies each represented as an array of note frequencies and durations.
- The melodies are predefined that includes many popular tunes such as jingle bells and baby shark.
- The melodies are stored in separate arrays, along with their corresponding note durations.
- There is also a mapping between the melodies and LEDs, allowing users to select a melody for each LED (album).

C. User interaction and control

- The LCD is initialized and a welcome message is displayed showing the available songs and albums.
- The keypad serves as the primary interface for user interaction.
- The users are enabled to select different LEDs (albums) and different melodies by pressing keys corresponding to different melodies and LEDs.
- Pressing a key triggers the system to play the melody which has been selected through the buzzer and synchronize the LED with music.
- This system supports up to 4 albums comprising of 2 melodies each.

D. Working of PWM

The PWM pins are capable of producing digital pulses with varying widths, allowing for the simulation of analog signals, such as varying frequencies.

1. Tone Generation

    - The system iterates through the array of note frequencies and durations for the selected melody.
    - For each note in the melody, a tone is generated using the tone( ) function, which utilizes the PWM to produce a square wave of the desired frequency.
    - By modulating the pulse width of the digital signal at the PWM pin, the buzzer produces a tone at the specified frequency.

2. Note duration

    - The duration of each note decides the length of time corresponding tone is played.

3. Dynamic frequency adjustment

    - PWM works by switching the pin between high and low states at a high frequency. The duration of the high state within each cycle (duty cycle) can be varied to simulate an analog signal.
    - By rapidly toggling the pin state, PWM can generate a square wave signal at the desired frequency.

E. Playback and LED synchronization

- When a melody is selected, the system plays corresponding notes through the buzzer while synchronizing the LED with the music.
- The LED's on/off state is determined by each note's duration which creates a visual representation of the melody.
- This provides a dynamic and engaging user experience.
- A debounce delay is also introduced additionally, to prevent multiple selections.

F. Displaying messages on LCD

- The system cycles through different albums available in the system along with the songs available in it for an overview.
- When an album is selected particularly, songs available for the respective album is displayed again for user friendly working.
- Since the song can be played again from the same album, user is given a choice to choose the songs available from the album again or can exit from the current album.
- The above process will be repeated again until user desires to use the system.
- The LCD helps the user navigate and make selections easily.

G. System control and feedback

- The buzzer produces sound corresponding to the selected melody.
- The LEDs are controlled to provide visual feedback, indicating which melodies are selected and being played.
- The system communicates with an external device via serial communication, providing feedback on melody selection and playback status.
- This allows for debugging and monitoring system behavior.

VI. RESULTS AND DISCUSSION

1. Initially the LCD displays the available songs in each album and the keypad allows the user to select the melodies and control the playback.
2. The system allows to select the desired album and the desired song in each album. Users can select one of eight predefined melodies using the keypad.
3. A buzzer connected to the PWM pin plays the selected melody. The buzzer produces different frequencies corresponding to the predefined musical notes.the melody being played is displayed on the LCD.
4. The system integrates various components to create an interactive user experience. Its modular design makes it easy to modify individual part without affecting the entire system.
5. User interaction helps in the easy navigation through the albums available in the system. The LCD also provides clear instructions and feedback, making the system user friendly.
6. Proper debouncing of keypad inputs is crucial for accurate melody selection. The current implementation ensures that multiple presses are not registered, preventing unwanted behavior.

VII. REFERENCES

[1] Tanu Jewale, Piyush Harinkhede, Vaibhav There, Gauraw Sakore, Asst. Prof. Ashish, " Design and implementation of a music rhythm LED flashlight circuit for entertainment and decoration purposes", International Conference Proceeding ICTTSEM May 2023 https://ijaite.co.in/downloads/special_issue/volume_2/issue_1/101.pdf

[2] Heijing Wu, Hongtao Han, Qiang Wang, Jinqiu Lin, Fang Liu, "STM32-based music player", https://doi.org/10.59429/esta.v10i4.1623

[3] Shaer, B., Gressett, G., Mitchell, P., Meeks, J., Barnes, W., Hewitt, S. (2021). Piano Player with Embedded Microcontrollers. In: Daimi, K., Arabnia, H.R., Deligiannidis, L., Hwang, MS., Tinetti, F.G. (eds) Advances in Security,

Networks, and Internet of Things. Transactions on Computational Science and Computational Intelligence. Springer, Cham. https://doi.org/10.1007/978-3-030-71017-0_55

[4] Mukhopadhyay, S., Kumar, A., Parashar, D., Singh, M. (2024). Enhanced music recommendation systems: A comparative study of content-based filtering and K-Means clustering approaches. Revue d'Intelligence Artificielle, Vol. 38, No. 1, pp. 365-376. https://doi.org/10.18280/ria.340138

[5] Ashiq V Mehta, Prashant R Kharote, "ARM7 based MP3 Player" Int. Journal of Engineering Research and Applications ISSN : 2248-9622, Vol. 4, Issue 2(Version 6), February 2014, pp.01-05, https://www.ijera.com/papers/Vol4_issue2/Version%206/A42060105.pdf

[6] M. A and R. Hegde, "Music Synthesis on Different Musical Instruments Using ARM Cortex-M4F Microcontroller," 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS), Bangalore, India, 2018, pp. 1-5, doi: 10.1109/ICNEWS.2018.8903918.

[7] Guo, X. (2013). MIDI Music Generator Based on STM32. In: Yang, Y., Ma, M., Liu, B. (eds) Information Computing and Applications. ICICA 2013. Communications in Computer and Information Science, vol 392. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-53703-5_13

[8] "Music to Light: A Study of Music Synchronized LED Lighting System" by S. S. Saini, S. K. Biswas, and S. Dey, International Journal of Scientific and Research Publications (IJSRP), 2017.

a.

VIII. APPENDIX

```
[1]#include <LiquidCrystal.h>
[2]#define RS PA_2
[3]#define E  PA_3
[4]#define D4 PA_4
[5]#define D5 PA_5
[6]#define D6 PA_6
[7]#define D7 PA_7
[8]#define NOTE_C4  262
[9]#define NOTE_D4  294
[10]#define NOTE_E4  330
[11]#define NOTE_F4  349
[12]#define NOTE_G4  392
[13]#define NOTE_A4  440
[14]#define NOTE_B4  494
[15]#define NOTE_C5  523
[16]#define NOTE_AS4 466
[17]#define LED1_PIN 23
[18]#define LED2_PIN 24
[19]#define LED3_PIN 25
[20]#define LED4_PIN 26
[21]#define BUZZER_PIN 14
[22]#define NO_KEY '\0'
[23]int melody1[] = { NOTE_C4, NOTE_C4, NOTE_D4,
NOTE_C4, NOTE_F4, NOTE_E4, NOTE_C4, NOTE_C4,
NOTE_D4, NOTE_C4, NOTE_G4, NOTE_F4, NOTE_C4,
NOTE_C4, NOTE_C5, NOTE_A4, NOTE_F4, NOTE_E4,
NOTE_D4, NOTE_AS4, NOTE_AS4, NOTE_A4, NOTE_F4,
NOTE_G4, NOTE_F4 };
[24]int noteDurations1[] = { 4, 4, 2, 2, 2, 1, 4, 4, 2, 2, 2, 1, 4,
4, 2, 2, 2, 2, 2, 4, 4, 2, 2, 2, 1 };
[25]int melody2[] = { NOTE_E4, NOTE_E4, NOTE_E4,
NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_G4,
NOTE_C4, NOTE_D4, NOTE_E4, NOTE_F4, NOTE_F4,
NOTE_F4, NOTE_F4, NOTE_F4, NOTE_E4, NOTE_E4,
NOTE_E4, NOTE_E4, NOTE_E4, NOTE_D4, NOTE_D4,
NOTE_E4, NOTE_D4, NOTE_G4 };
[26]int noteDurations2[] = { 4, 4, 2, 4, 4, 2, 4, 4, 4, 4, 2, 4, 4,
4, 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 2 };
[27]int melody3[] = { NOTE_C4, NOTE_C4, NOTE_G4,
NOTE_G4, NOTE_A4, NOTE_A4, NOTE_G4, NOTE_F4,
NOTE_F4, NOTE_E4, NOTE_E4, NOTE_D4, NOTE_D4,
NOTE_C4, NOTE_G4, NOTE_G4, NOTE_F4, NOTE_F4,
NOTE_E4, NOTE_E4, NOTE_D4, NOTE_G4, NOTE_G4,
NOTE_F4, NOTE_F4, NOTE_E4, NOTE_E4, NOTE_D4,
NOTE_C4, NOTE_C4, NOTE_G4, NOTE_G4, NOTE_A4,
NOTE_A4, NOTE_G4, NOTE_F4, NOTE_F4, NOTE_E4,
NOTE_E4, NOTE_D4, NOTE_D4, NOTE_C4 };
[28]int noteDurations3[] = { 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4,
2, 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 2,
4, 4, 4, 4, 2 };
[29]int melody4[] = { NOTE_C4, NOTE_C4, NOTE_D4,
NOTE_D4, NOTE_E4, NOTE_E4, NOTE_C4, NOTE_C4,
NOTE_D4, NOTE_D4, NOTE_E4, NOTE_E4, NOTE_C4 };
[30]int noteDurations4[] = { 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4 };
[31]int melody5[] = { NOTE_E4, NOTE_D4, NOTE_C4,
NOTE_D4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_D4,
NOTE_D4, NOTE_D4, NOTE_E4, NOTE_G4, NOTE_G4,
NOTE_E4, NOTE_D4, NOTE_C4, NOTE_D4, NOTE_E4,
NOTE_E4, NOTE_E4, NOTE_E4, NOTE_D4, NOTE_D4,
NOTE_E4, NOTE_D4, NOTE_C4 };
[32]int noteDurations5[] = { 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 };
[33]int melody6[] = { NOTE_E4, NOTE_E4, NOTE_F4,
NOTE_G4, NOTE_G4, NOTE_F4, NOTE_E4, NOTE_D4,
NOTE_C4, NOTE_C4, NOTE_D4, NOTE_E4, NOTE_E4,
NOTE_D4, NOTE_D4, NOTE_E4, NOTE_E4, NOTE_F4,
NOTE_G4, NOTE_G4, NOTE_F4, NOTE_E4, NOTE_D4,
NOTE_C4, NOTE_C4, NOTE_D4, NOTE_E4, NOTE_D4,
NOTE_C4, NOTE_C4 };
[34]int noteDurations6[] = { 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 };
[35]int melody7[] = { NOTE_G4, NOTE_A4, NOTE_G4,
NOTE_E4, NOTE_G4, NOTE_A4, NOTE_G4, NOTE_E4,
```

```
NOTE_D4, NOTE_D4, NOTE_B4, NOTE_C4, NOTE_D4,
NOTE_D4, NOTE_B4, NOTE_C4 };
[36]int noteDurations7[] = { 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4 };
[37]int melody8[] = { NOTE_G4, NOTE_G4, NOTE_A4,
NOTE_G4, NOTE_C4, NOTE_B4, NOTE_G4, NOTE_G4,
NOTE_A4, NOTE_G4, NOTE_D4, NOTE_C4, NOTE_G4,
NOTE_G4, NOTE_G4, NOTE_E4, NOTE_C4, NOTE_B4,
NOTE_A4, NOTE_F4, NOTE_F4, NOTE_E4, NOTE_C4,
NOTE_D4, NOTE_C4 };
[38]int noteDurations8[] = { 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 };
[39]int* melodies[] = { melody1, melody2, melody3,
melody4, melody5, melody6, melody7, melody8 };
[40]int* noteDurations[] = { noteDurations1, noteDurations2,
noteDurations3, noteDurations4, noteDurations5,
noteDurations6, noteDurations7, noteDurations8 };
[41]int melodyLengths[] = { sizeof(melody1) /
sizeof(melody1[0]), sizeof(melody2) / sizeof(melody2[0]),
sizeof(melody3) / sizeof(melody3[0]), sizeof(melody4) /
sizeof(melody4[0]), sizeof(melody5) / sizeof(melody5[0]),
sizeof(melody6) / sizeof(melody6[0]), sizeof(melody7) /
sizeof(melody7[0]), sizeof(melody8) / sizeof(melody8[0]) };
[42]int ledPins[] = { LED1_PIN, LED2_PIN, LED3_PIN,
LED4_PIN };
[43]int melodyCount = 8;
[44]bool toggleLEDs = true;
[45]const byte ROWS = 4;
[46]const byte COLS = 3;
[47]char keys[ROWS][COLS] = {
[48]  {'1', '2', '3'},
[49]  {'4', '5', '6'},
[50]  {'7', '8', '9'},
[51]  {'*', '0', '#'}
[52]};
[53]byte rowPins[ROWS] = {29, 28, 7, 6};
[54]byte colPins[COLS] = {5, 4, 3};
[55]bool melodySelected[] = { false, false, false, false };
[56]LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
[57]void setup() {
[58]  Serial.begin(9600);
[59]  pinMode(BUZZER_PIN, OUTPUT);
[60]  for (int i = 0; i < 4; i++) {
[61]    pinMode(ledPins[i], OUTPUT);
[62]  }
[63]  for (int i = 0; i < ROWS; i++) {
[64]    pinMode(rowPins[i], INPUT_PULLUP);
[65]  }
[66]  for (int i = 0; i < COLS; i++) {
[67]    pinMode(colPins[i], OUTPUT);
[68]    digitalWrite(colPins[i], HIGH);
[69]  }
[70]  lcd.begin(16, 2);
[71]  lcd.clear();
[72]  printAvailableSongs();
[73]}
[74]void loop() {
[75]  if (toggleLEDs) {
[76]    initialToggleLEDs();
[77]  }
[78]  char key = getKeypadKey();
[79]  if (key >= '1' && key <= '4') {
[80]    toggleLEDs = false;
[81]    int ledIndex = key - '1';
[82]    if (!melodySelected[ledIndex]) {
[83]      selectMelodyForLED(ledIndex);
[84]      melodySelected[ledIndex] = true;
[85]    }
[86]    delay(300);
[87]  } else if (key == '#') {
[88]    stopMelodies();
[89]    delay(300);
[90]  }
[91]}
[92]void initialToggleLEDs() {
[93]  static unsigned long lastToggleTime = 0;
[94]  unsigned long currentTime = millis();
[95]  if (currentTime - lastToggleTime >= 500) {
[96]    lastToggleTime = currentTime;
[97]    for (int i = 0; i < 4; i++) {
[98]      digitalWrite(ledPins[i], !digitalRead(ledPins[i]));
[99]    }
[100]  }
[101]}
[102]void selectMelodyForLED(int ledIndex) {
[103]  lcd.clear();
[104]  lcd.setCursor(0, 0);
[105]  lcd.print("Select songs for");
[106]  lcd.setCursor(0, 1);
[107]  lcd.print("album ");
[108]  lcd.print(ledIndex + 1);
[109]  delay(2000);
[110]  lcd.clear();
[111]  bool exitSelection = false;
[112]  while (!exitSelection) {
[113]    lcd.clear();
[114]    lcd.setCursor(0, 0);
[115]    lcd.print("Album ");
[116]    lcd.print(ledIndex + 1);
[117]    lcd.print(" songs:");
[118]    delay(2000);
[119]    lcd.clear();
[120]    switch (ledIndex) {
[121]      case 0:
[122]        lcd.setCursor(0, 0);
[123]        lcd.print("1) Happy Bday");
[124]        lcd.setCursor(0, 1);
[125]        lcd.print("2) Jingle Bells");
[126]        delay(3000);
[127]        lcd.clear();
[128]        break;
[129]      case 1:
[130]        lcd.setCursor(0, 0);
[131]        lcd.print("1) Twinkle star");
```

```
      lcd.setCursor(0, 1);
      lcd.print("2) Baby Shark");
      delay(3000);
      lcd.clear();
      break;
    case 2:
      lcd.setCursor(0, 0);
      lcd.print("1) Mary Lamb");
      lcd.setCursor(0, 1);
      lcd.print("2) Ode to Joy");
      delay(3000);
      lcd.clear();
      break;
    case 3:
      lcd.setCursor(0, 0);
      lcd.print("1) Silent Night");
      lcd.setCursor(0, 1);
      lcd.print("2) Merry Xmas");
      delay(3000);
      lcd.clear();
      break;
    default:
      lcd.setCursor(0, 0);
      lcd.print("Unknown Album");
      delay(3000);
      lcd.clear();
      break;
  }
  lcd.setCursor(0, 0);
  lcd.print("press 1/2 for song");
  lcd.setCursor(0, 1);
  lcd.print("or '*' for exit");
  delay(2000);
  lcd.clear();
  char key = NO_KEY;
  while (key == NO_KEY) {
    key = getKeypadKey();
  }
  if (key == '*') {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Exiting from ");
    lcd.setCursor(0, 1);
    lcd.print("album ");
    lcd.print(ledIndex + 1);
    exitSelection = true;
    delay(2000);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Press 1/2/3/4 to");
    lcd.setCursor(0, 1);
    lcd.print("select album");
    delay(3000);
    lcd.clear();
  } else if (key >= '1' && key <= '2') {
    int melodyIndex = ledIndex * 2 + (key - '1');
    printMelodyName(melodyIndex);
    playMelody(melodyIndex);
  } else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Invalid choice!");
    delay(1000);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Select songs for");
    lcd.setCursor(0, 1);
    lcd.print("Album ");
    lcd.print(ledIndex + 1);
  }
}
}
char getKeypadKey() {
  char key = NO_KEY;
  for (byte col = 0; col < COLS; col++) {
    digitalWrite(colPins[col], LOW);
    for (byte row = 0; row < ROWS; row++) {
      if (digitalRead(rowPins[row]) == LOW) {
        key = keys[row][col];
        while (digitalRead(rowPins[row]) == LOW);
      }
    }
    digitalWrite(colPins[col], HIGH);
  }
  return key;
}
void playMelody(int melodyIndex) {
  int* melody = melodies[melodyIndex];
  int* durations = noteDurations[melodyIndex];
  int melodyLength = melodyLengths[melodyIndex];
  int ledPin = ledPins[melodyIndex / 2];
  for (int i = 0; i < 4; i++) {
    digitalWrite(ledPins[i], LOW);
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Playing:");
  lcd.setCursor(0, 1);
  printMelodyName(melodyIndex);
  for (int i = 0; i < melodyLength; i++) {
    int noteDuration = 1000 / durations[i];
    tone(BUZZER_PIN, melody[i], noteDuration);
    digitalWrite(ledPin, HIGH);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    digitalWrite(ledPin, LOW);
    noTone(BUZZER_PIN);
  }
}
void printAvailableSongs() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Available albums:");
  delay(3000);
```

```
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Songs in album 1:");
  delay(3000);
  lcd.clear();
  lcd.print("1) Happy Bday");
  lcd.setCursor(0, 1);
  lcd.print("2) Jingle Bells");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Songs in album 2:");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("1) Twinkle star");
  lcd.setCursor(0, 1);
  lcd.print("2) Baby Shark");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Songs in album 3:");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("1) Mary Lamb");
  lcd.setCursor(0, 1);
  lcd.print("2) Ode to Joy");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Songs in album 4:");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("1) Silent Night");
  lcd.setCursor(0, 1);
  lcd.print("2) Merry Xmas");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Press 1/2/3/4 to");
  lcd.setCursor(0, 1);
  lcd.print("select album");
  delay(3000);
  lcd.clear();
}
void printMelodyName(int melodyIndex) {
  lcd.clear();
  switch (melodyIndex) {
    case 0:
      lcd.setCursor(0, 0);
      lcd.print("Playing now:");
      lcd.setCursor(0, 1);
      lcd.print("1.1.Happy Bday");
      break;
    case 1:
      lcd.setCursor(0, 0);
      lcd.print("Playing now:");
      lcd.setCursor(0, 1);
      lcd.print("1.2.Jingle Bells");
      break;
    case 2:
      lcd.setCursor(0, 0);
      lcd.print("Playing now:");
      lcd.setCursor(0, 1);
      lcd.print("2.1.Twinkle star");
      break;
    case 3:
      lcd.setCursor(0, 0);
      lcd.print("Playing now:");
      lcd.setCursor(0, 1);
      lcd.print("2.2.Baby Shark");
      break;
    case 4:
      lcd.setCursor(0, 0);
      lcd.print("Playing now:");
      lcd.setCursor(0, 1);
      lcd.print("3.1.Mary Lamb");
      break;
    case 5:
      lcd.setCursor(0, 0);
      lcd.print("Playing now:");
      lcd.setCursor(0, 1);
      lcd.print("3.2.Ode to Joy");
      break;
    case 6:
      lcd.setCursor(0, 0);
      lcd.print("Playing now:");
      lcd.setCursor(0, 1);
      lcd.print("4.1.Silent Night");
      break;
    case 7:
      lcd.setCursor(0, 0);
      lcd.print("Playing now:");
      lcd.setCursor(0, 1);
      lcd.print("4.2.Merry Xmas");
      break;
    default:
      lcd.print("Unknown");
      break;
  }
}
void stopMelodies() {
  noTone(BUZZER_PIN);
  for (int i = 0; i < 4; i++) {
    digitalWrite(ledPins[i], LOW);
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Melodies stopped");
  delay(1000);
  printAvailableSongs();
}
```