MeloKeys

By:

Shagun Majotra, sm11304

Devika Kodi, dak9250

Andrew Dave, ad7068

Vineela Reddy, vp2504

New York University,

Tandon School of Engineering

Department of Mechanical and Aerospace Engineering

Advanced Mechatronics, ROB-GY 6103

ABSTRACT

In our pursuit to make music education more approachable and enjoyable, we introduce an interactive music system tailored for users at any proficiency level. With its straightforward push-button keys and illuminating LED feedback, the system offers an intuitive interface for users to engage with music effortlessly. A standout feature includes its recording and playback capabilities, allowing users to document their musical explorations and review their progress at their convenience. The overview provides a concise summary of our interactive music system, highlighting its user-centric design and potential to enrich the learning journey for musicians of diverse backgrounds. Through this project, our aim is to ignite passion for music and foster inclusive learning experiences for all.

INDEX

[1] INTRODUCTION

In our fast-paced digital age, learning music can sometimes feel like a daunting task. That's why we've developed an interactive music system to make it easier and more enjoyable for everyone, regardless of their musical background.

Our system is designed with simplicity in mind. Using easy-to-use push-button keys and LED lights, it allows users to interact with music in a hands-on way. Whether you're a complete beginner or an experienced musician, our system provides a fun and intuitive platform for learning and experimenting with music.

One of the coolest features of our system is its ability to record and playback musical sequences. This means you can capture your musical creations and listen back to them, helping you track your progress and refine your skills over time.

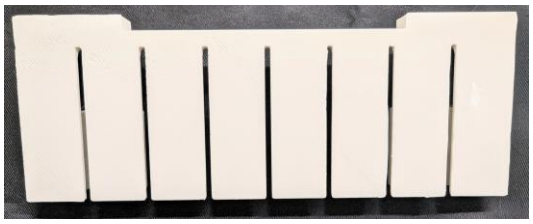
In this report, we'll take a closer look at how our interactive music system works and explore its various features. From its user-friendly design to its recording capabilities, we'll show you how our system can make learning music a breeze for everyone.

[2] METHOD

1. Design Approach

➢ Keys

Our 3D-printed keyboard design features a cantilever beam structure, providing robust support while allowing for smooth key presses. At the tip of each key, there are intricate node-like structures designed to enhance durability and precision in key movement. To ensure optimal light dispersion, each key is enclosed up to the LEDs, guaranteeing uniform illumination and enhancing the visual experience for users. This design not only improves aesthetics but also contributes to the overall functionality and longevity.



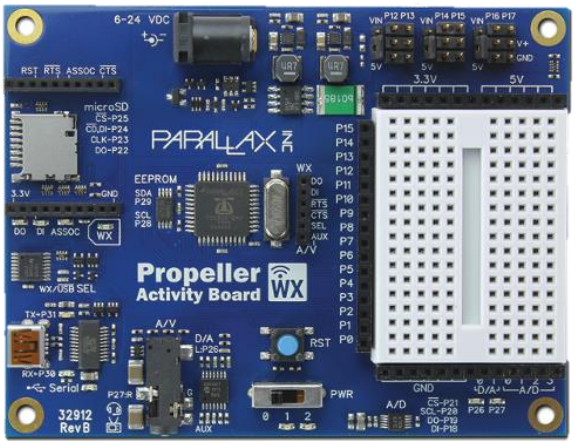
➢ Base

Our 3D-printed base serves as the central hub for all the components of our system, providing a sturdy and organized home for seamless integration. Within this base, the Zero PCB, momentary and latch push buttons, LEDs, LCD screen, and wiring system are meticulously arranged for optimal functionality and accessibility. The keys, fitted inside the base, are positioned with precision to accommodate the LEDs and push buttons, allowing for smooth and responsive user input during music playing. Moreover, the base design prioritizes cable management, ensuring that the wiring system is neatly organized and securely housed within the structure.

2. Circuit and Electronics

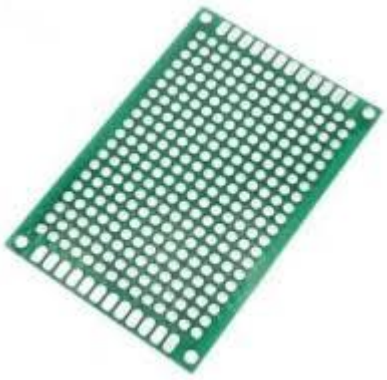
➢ Parallax Propeller Activity Board

The Parallax Propeller Activity Board serves as the central hardware platform for our interactive music system project. This board hosts a powerful Parallax Propeller microcontroller, which forms the heart of our system, providing the computational power necessary for its operation. In our setup, a Zero PCB acts as an intermediary interface, connecting directly to the Activity Board. All components used in our system are then connected to this Zero PCB. This arrangement simplifies the wiring process and ensures a neat and organized setup. By utilizing the Activity Board alongside the Zero PCB, we benefit from a robust and versatile hardware platform that facilitates seamless integration and expansion of components, ultimately enhancing the functionality and scalability of our system.



➢ Zero PCB

The Zero PCB (Printed Circuit Board) serves as a vital component in our interactive music system, facilitating the connection and organization of various hardware components. Designed to interface with the Parallax Propeller Activity Board, the Zero PCB acts as a central hub for connecting all system components. By providing a structured layout and electrical pathways, it simplifies the wiring process and ensures a neat and organized assembly of the system. All components, including the push buttons, LEDs, and LCD display, are connected to the Zero PCB, allowing for efficient communication and interaction within the system. This centralized arrangement enhances the system's reliability and scalability, enabling easy expansion and modification of features as needed. Overall, the Zero PCB plays a crucial role in optimizing the functionality and performance of our interactive music system, providing a robust foundation for seamless integration and operation of hardware components.



➢ **Push Buttons**

➔ Momentary Push Buttons: The 8 momentary push buttons in our music system correspond to the eight musical notes: C, D, E, F, G, A, B, and another C octave. Each button represents a specific note, allowing users to trigger individual notes or combinations of notes to create melodies. These push buttons serve as the primary interface for users to interact with the system, providing a tactile and intuitive means of playing music. By pressing the buttons, users can produce sounds corresponding to the selected notes, enabling them to experiment with different musical compositions and arrangements. The layout of the push buttons follows the standard arrangement of musical notes, facilitating easy navigation between different notes and enhancing the user's ability to play music with accuracy and precision.

➔ Latching Push Buttons: The two latching push buttons in our interactive music system serve distinct yet essential functions: the record button and the playback button. The record button initiates the recording mode of the system when pressed. This enables users to capture their musical performances note by note, allowing for the creation of personalized compositions. Once activated, the system begins to record the sequence of notes played by the user, along with their respective durations. The latching mechanism ensures that the recording mode remains active until the button is pressed again to stop the recording process. On the other hand, the playback button facilitates the playback mode of the system. When pressed, it triggers the playback of the recorded musical sequence, allowing users to listen to their compositions in real-time. Similar to the record button, the latching mechanism ensures that the playback mode remains active until the button is pressed again to stop playback. Together, these latching push buttons provide users with control over the recording and playback functionalities of the interactive music system, enhancing their ability to create, edit, and review their musical compositions with ease and convenience.

➢ Speaker

The speaker in our interactive music system serves as the audio output device, allowing users to hear the musical notes and compositions played through the system. Connected to the Parallax Propeller Activity Board via Zero PCB, the speaker produces sound in response to user input, providing auditory feedback during performance, recording, and playback modes. Its compact size and efficient power consumption make it an ideal choice for our project, providing reliable audio output while maintaining portability and ease of use.

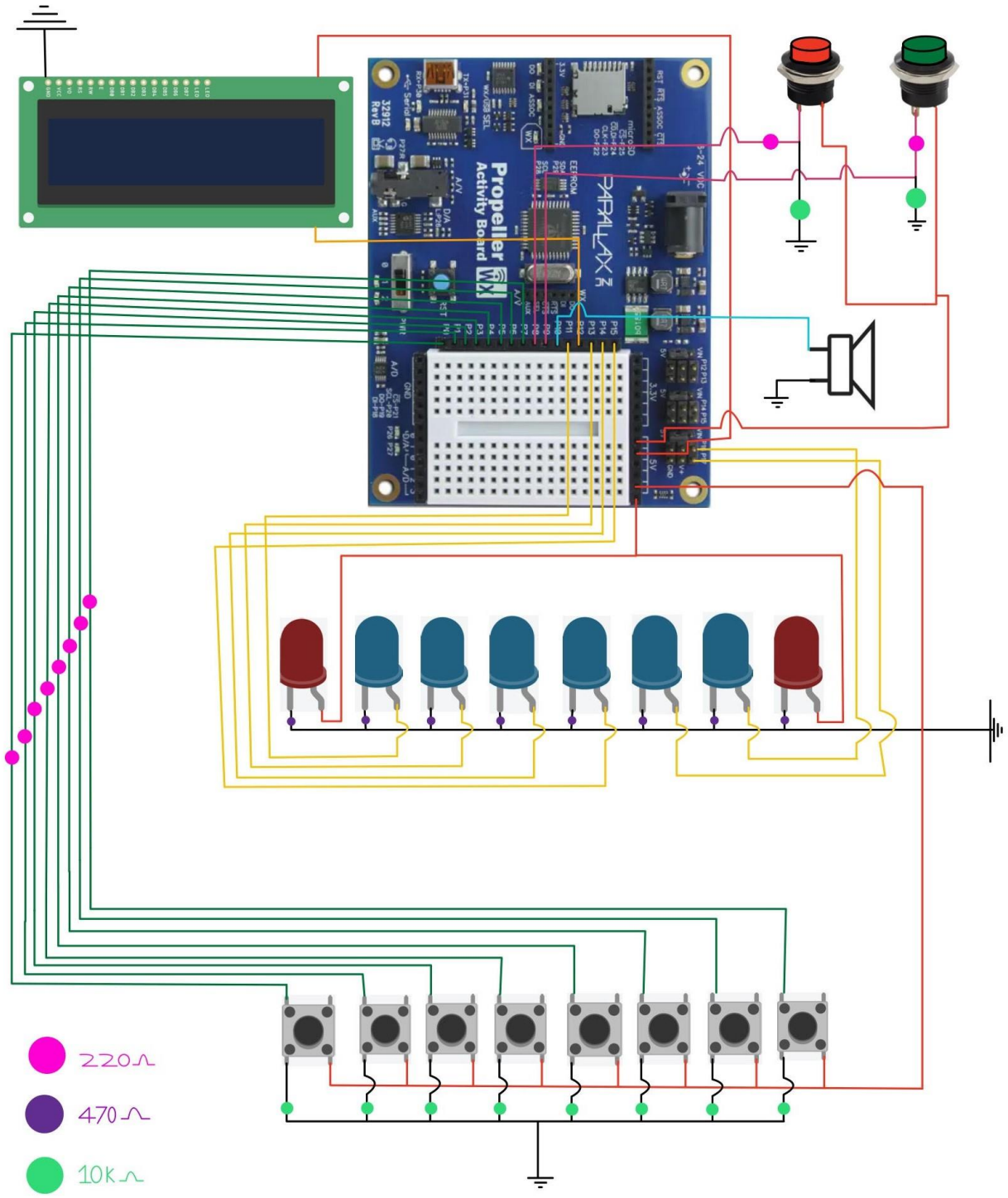
➢ LCD Screen

The LCD (Liquid Crystal Display) screen in our interactive music system serves a dual purpose: providing real-time feedback on notes played during performance and displaying recorded sequences during playback mode. During performance, the LCD screen dynamically updates to show the notes that are being pressed on the keyboard in real-time. This feature allows users to visually track their performance and better understand the music they are playing. In record mode, the LCD screen displays the notes that are being recorded as the user plays them. This visual feedback enables users to confirm that their input is being accurately captured by the system, ensuring a seamless recording process. During playback mode, the LCD screen shows the recorded sequence of notes, allowing users to review their performance and analyze the musical composition they have created. This feature provides valuable insight into their progress and allows for experimentation and refinement of their musical skills. Overall, the LCD screen plays a vital role in enhancing the user experience of our system by providing real-time feedback and visual representation of both live performances and recorded sequences.

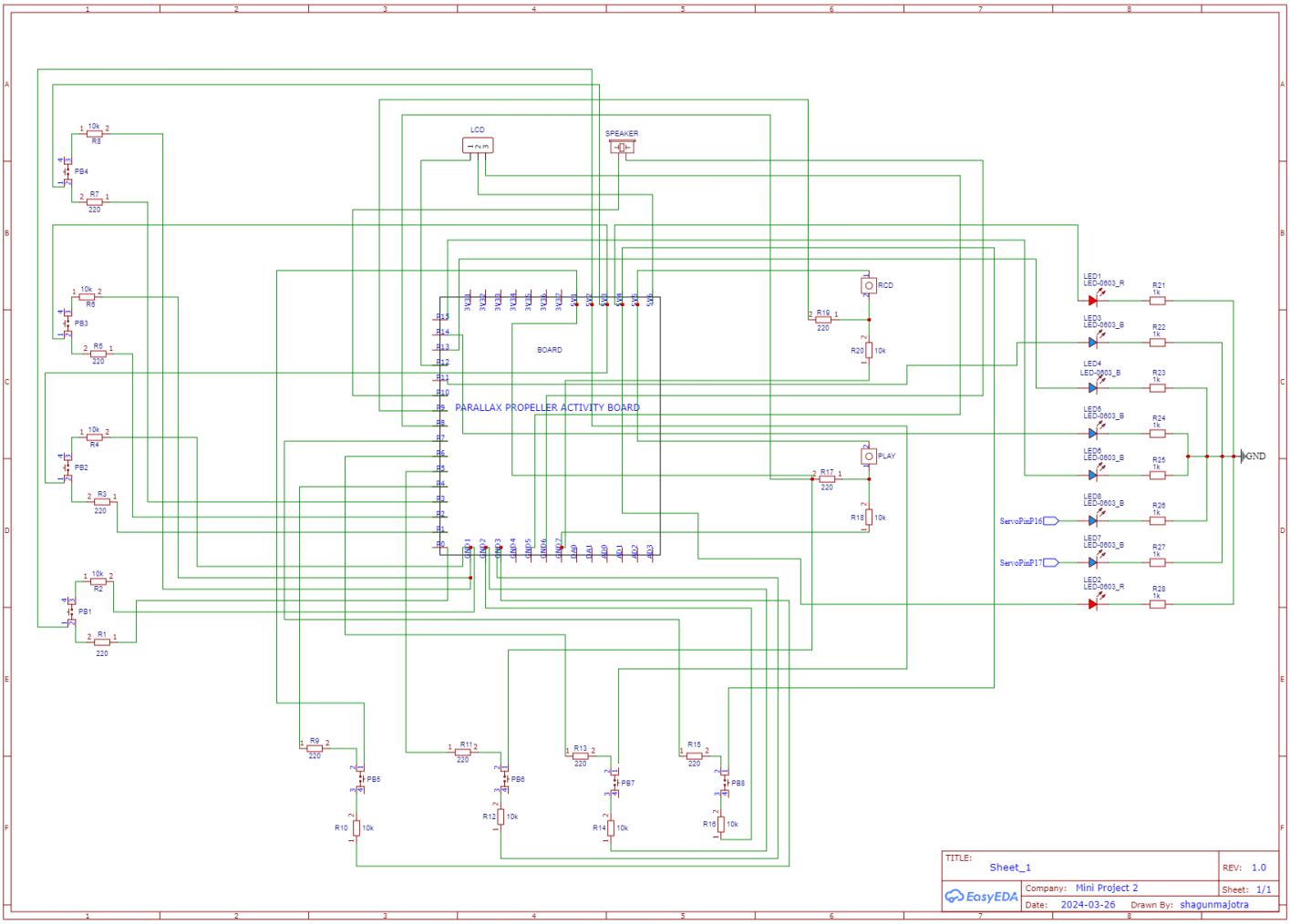
➢ LEDs

The LEDs in our interactive music system serve as visual indicators, providing feedback to the user during performance and playback modes. When a key corresponding to the notes D, E, F, G, A, or B is pressed or played, the respective LED associated with that key illuminates (blue), indicating the activation of the note. This visual feedback enhances the user experience by confirming the successful input of a note and aiding in the recognition of the notes being played. Additionally, both LEDs corresponding to the notes C octave remain permanently illuminated (red) throughout the operation of the system. This serves as a reference point for users, indicating the starting point of each octave and providing a visual cue for orientation on the keyboard. Overall, the LEDs play a crucial role in our interactive music system, providing real-time visual feedback to users and enhancing the usability and engagement of the system during both performance and playback modes.

3. Circuit Diagram



Schematic Diagram:



- **Push Buttons:**

➢ Momentary Push Buttons: The momentary push buttons for notes C to another C octave are connected to pins P0 to P7 on the Board. Each button is paired with a 220-ohm resistor to stabilize input signals and connected to ground via a 10k ohm resistor. The 5V power source connects to the other side of the buttons, allowing for reliable triggering of musical notes.

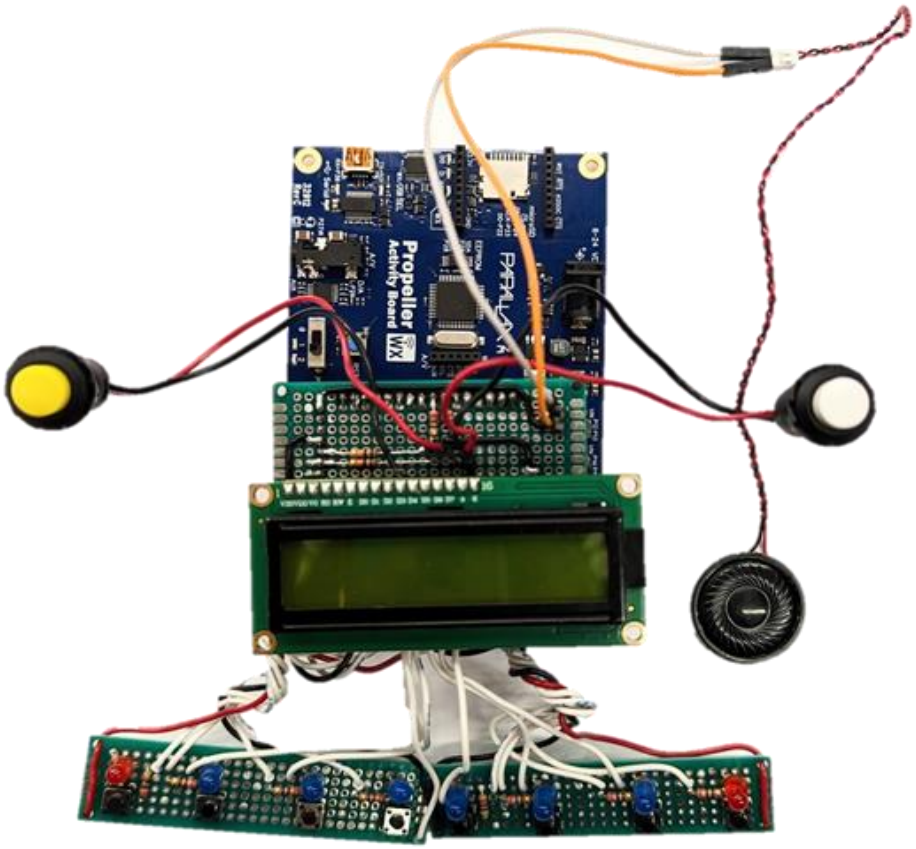
➢ Latching Push Buttons: The latching push buttons for record and playback functionalities are connected similarly to the momentary push buttons. The record button is linked to pin P8, while the playback button is connected to pin P9 Board. Each button is paired with a 220-ohm resistor for stability and connected to ground through a 10k ohm resistor. The 5V power source connects to the other side of the buttons, enabling reliable operation for recording and playback actions in the interactive music system.

- **Speaker:** The speaker in our system is connected to pin P10 on the Board. One terminal of the speaker is connected to pin P10, while the other terminal is connected directly to ground. This configuration allows the speaker to receive electrical signals from the microcontroller through pin P10.

- **LCD Screen:** The LCD screen in our system connects one terminal to ground and another to the 5V power source. The data transmission pin (Rx) of the LCD is linked to pin P12 on the board, enabling data transfer for display. These connections provide the necessary power and data transmission for accurate information display in our interactive music system.

- **LEDs:**

- Red LEDs: These LEDs represent both C notes in our system connected with the positive (anode) terminal of each LED connected to the 5V power source, and the negative (cathode) terminal connected directly to ground via a 470-ohm resistor. This setup ensures that the LEDs remain constantly illuminated, serving as visual indicators for users without the need for external control.
- Blue LEDs: The blue LEDs representing notes D, E, F, G, A, and B in our system are connected with their positive (anode) terminals linked to individual pins on the Parallax Propeller Activity Board: pins P11, P13, P14, P15, servo pin P16, and servo pin P17, respectively. The negative (cathode) terminal of each LED is connected to ground through a 460-ohm resistor. This setup ensures that each LED illuminates whenever the corresponding button is pushed, providing visual feedback to the user.



All these components are connected to the Parallax Propeller Activity Board via the Zero PCB, ensuring seamless integration and efficient operation of our interactive music system.

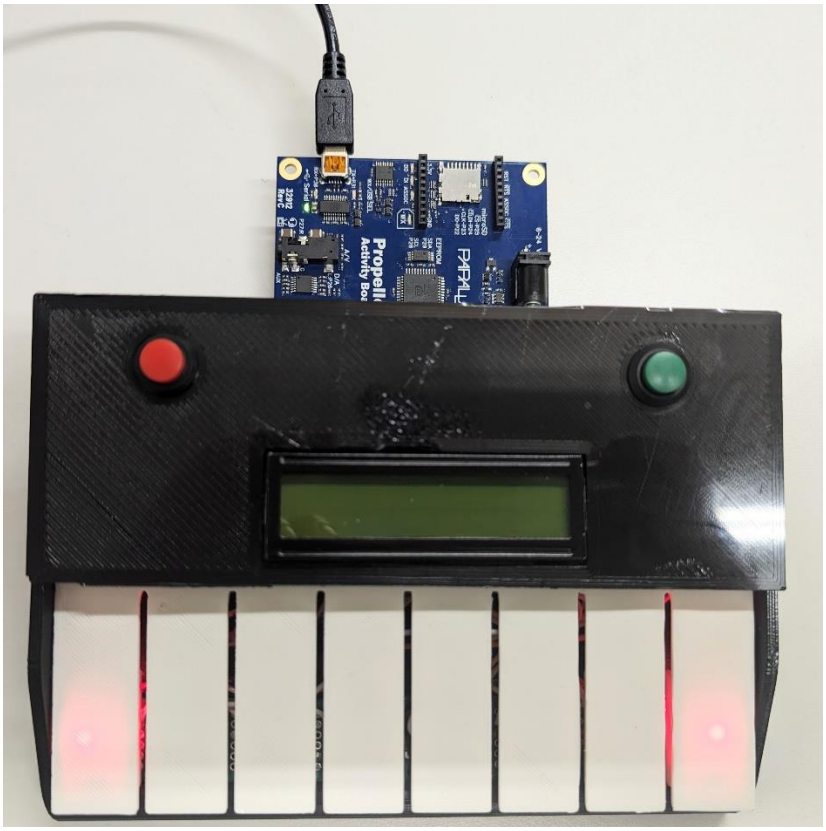
4. Operation

Our interactive music system operates through a series of intuitive and seamless functionalities designed to simplify the process of music learning and exploration. The following steps outline the working operation of the system:

- **Power On:** Upon powering on the system, all functions, variables and the LCD are initialized.
- **Default Mode (For Practice):** The user can start by pressing any of the eight momentary push buttons, each representing a musical note (C, D, E, F, G, A, B, C). When a button is pressed, the corresponding note is triggered to be played through the speaker, providing immediate auditory feedback to the user. The program continuously monitors button pushes, so when playback and recording are disabled, the user can still play notes and view them on the LCD.
- **Button state check:** When the user is done practicing, and decides to record or playback a sequence, pressing the button for either of the buttons activates a button state check. If either of the buttons is pressed, the input of it is taken as the current button state and the function is executed.
- **Recording Mode:** To record a musical sequence, the user presses the record button, initiating recording mode. Upon activation:
  - The record button's button state is updated to reflect input; if the button is high or 1, recording begins.
  - First, the LCD clears and displays "Recording".
  - Notes and durations are recorded according to how long the corresponding button is pressed and these notes can be seen on the LCD screen.
  - These sequences and durations are stored in the program for the playback.
  - When the record button is released, i.e., it's Low or 0, recording stops.
  - The LCD displays "Record Stopped".
- **Playback Mode:** To playback a musical sequence, users press the playback button, initiating the playback mode. Upon activation:
  - The playback button's button state is updated to reflect input; if the button is pressed or is at 1, the playback process begins.
  - The LCD clears and displays "Playing".
  - The recorded notes index are played sequentially, and the notes that are played are displayed on the screen.
  - Pause and Resume: If the user wants to pause the sequence, they can release the button so when the playback button is at 0, it 'Pauses'. If the playback button is pressed again, i.e it is at 1, it 'Resumes' until all the notes are played. Then "Playback Over!" displays and pauses for 5 seconds. Here, if the playback is not stopped, it starts over again with the message "Starting over".

- **Visual Representation:** All these modes share a LED cog. The LEDs associated with the notes lights up to provide real-time visual feedback, enhancing the user's understanding and engagement. Whenever a button is pressed, the corresponding LED lights up in both default and record modes. While in playback mode, when a note is played from the recording process' memory, LEDs light.

Through this working operation, our system offers users a versatile and engaging platform for learning and exploring music, fostering creativity and experimentation in a user-friendly manner.



5. Code

```
#include "simpletools.h"

serial *lcd;// Initializing LCD

const int ON = 22;

const int CLR = 12;

const int CRT = 13;


#define MAX_NOTES 30 // Increased to accommodate more notes

#define BTN_RECORD 8 // Button to start and stop the record

#define BTN_PLAYBACK 9//Button to play, pause and stop the Recorded Notes



void recordNotes(char* recordedNotes, int* noteDurations);// Function to record notes

void playBack(char* recordedNotes, int* noteDurations, int* leds);//Function to play back the recorded notes

void normalPlay(char* recordedNotes);// Function to playnotes

void led(void *par1);//cog for lighting up the LEDs while playnotes and recording them

//void led1(void* par2);

static volatile int leds[6] = {11, 13, 14, 15, 16, 17};// Initializing LED array

static volatile char notes[8] = {'c','D','E','F','G','A','B','C'};// Initializing Notes Array

int noteFrequencies[8] = {2093,2349,2637,2793,3136,3520,3951,4186};// Initializing Frequency Array

static volatile char recordedNotes[MAX_NOTES];

unsigned int stack1[128 + 128];//Stack variables for cog 1
```

```
int main() {
  cogstart(led, NULL, stack1, sizeof(stack1));// Initiating cog 1

  lcd = serial_open(12, 12, 0, 9600); // Initialize the LCD
  writeChar(lcd, ON); // Turn on the LCD
  writeChar(lcd, CLR); // Clear the LCD
  pause(10);

  volatile int buttonState_1;// Record Button State
  volatile int buttonState_2;// Playback Button State

  int noteDurations[MAX_NOTES];

  for(int i = 0; i < 6; i++) {
    low(leds[i]); // Initialize LEDs to low
  }
  while(1){
    buttonState_1 = input(BTN_RECORD);
    buttonState_2 = input(BTN_PLAYBACK);
    if (buttonState_1 == 1 && buttonState_2 == 0) {
      writeChar(lcd, CLR); // Clear the LCD
```

```
  writeChar(lcd, ON); // Turn on the LCD
  dprint(lcd, "Recording..\n");
  pause(500); // Pause for readability
  writeChar(lcd, CRT); // Move cursor to the second line

  // Clear recorded notes and durations
  memset(recordedNotes, 0, sizeof(char) * MAX_NOTES);
  memset(noteDurations, 0, sizeof(int) * MAX_NOTES);

  // recordNotesfunction Call
  recordNotes(recordedNotes, noteDurations);
  writeChar(lcd, CLR);
  dprint(lcd, "Record Stopped\n");
  pause(500); // Pause for readability
 }
 if (buttonState_2 == 1 && buttonState_1 == 0) {
  writeChar(lcd, CLR); // Clear the LCD
  writeChar(lcd, ON); // Turn on the LCD
  dprint(lcd, "Playing..\n");
  pause(250); // Pause for readability
  writeChar(lcd, CRT); // Move cursor to the second line

  // playBack function Call
```

```
      playBack(recordedNotes, noteDurations, leds);
      writeChar(lcd, CLR); // Clear the LCD
      pause(250); // Pause for readability
    }
    if(buttonState_2 == 0 && buttonState_1 == 0){
      writeChar(lcd, CLR); // Clear the LCD
      writeChar(lcd, ON); // Turn on the LCD
      pause(100); // Pause for readability
      writeChar(lcd, CRT); // Move cursor to the second line

      //normalPlay function call
      normalPlay(recordedNotes);

      writeChar(lcd, CLR); // Clear the LCD
      pause(100); // Pause for readability
    }
  }
  serial_close(lcd); // Close the serial connection
  return 0;
}
//function normalPlay
void normalPlay(char* recordedNotes){
  for (int i = 0; i < 8; i++) {
```

```
// to check buttons pressed and playing
if (input(i) == 1) {
  char note = ' ';
  switch (i) {
    case 0:
      note = 'c';
      break;
    case 1:
      note = 'D';
      break;
    case 2:
      note = 'E';
      break;
    case 3:
      note = 'F';
      break;
    case 4:
      note = 'G';
      break;
    case 5:
      note = 'A';
      break;
    case 6:
```

```
          note = 'B';
          break;
        case 7:
          note = 'C';
          break;
        }
      while(input(i)==1){
        // to play
        freqout(10, 100, noteFrequencies[i]);
        dprint(lcd, "Normal Play: %c\n", note);
      }
    }
  }
}
//function recordNotes
void recordNotes(char* recordedNotes, int* noteDurations){
  volatile int index = 0;
  volatile int lastButtonState[8] = {0};  // Array to store the last state of each button
  volatile int buttonPressStart[8] = {0}; // Array to store the start time of each button press
  int c;// variable to count number of times button pressed
  while (index < MAX_NOTES && input(BTN_RECORD) == 1) {
    int buttonState[8] = {input(0), input(1), input(2), input(3), input(4), input(5), input(6),
input(7)}; // Array to store the current state of each button
    // Check if any button is pressed
```

```
    for (int i = 0; i < 8; i++) {
     if (buttonState[i] == 1 && lastButtonState[i] == 0) { // Button transitioned from released to pressed
       c= 0;
       // Display the note being recorded
       char note = ' ';
       switch (i) {
         case 0:
           note = 'c';
           break;
         case 1:
           note = 'D';
           break;
         case 2:
           note = 'E';
           break;
         case 3:
           note = 'F';
           break;
         case 4:
           note = 'G';
           break;
         case 5:
           note = 'A';
```

```
        break;
      case 6:
        note = 'B';
        break;
      case 7:
        note = 'C'; // Octave change, so we have another 'C'
        break;
    }
    while (input(i) == 1) { // Keep playing the note as long as the button is pressed
      c++;
      freqout(10, 100, noteFrequencies[i]); // Play the note continuously
    }
    dprint(lcd, "Note:%c\n", note);
    pause(250); // Pause for readability
    writeChar(lcd, CLR); // Clear the second line
    writeChar(lcd, CRT); // Move cursor to the second line
  }
  else if (buttonState[i] == 0 && lastButtonState[i] == 1) { // Button transitioned from pressed to released
    int duration=c*100;//Duration of button pressed
    // Record the note corresponding to the button and its duration
    switch (i) {
      case 0:
        recordedNotes[index] = 'c';
```

```
    break;
  case 1:
    recordedNotes[index] = 'D';
    break;
  case 2:
    recordedNotes[index] = 'E';
    break;
  case 3:
    recordedNotes[index] = 'F';
    break;
  case 4:
    recordedNotes[index] = 'G';
    break;
  case 5:
    recordedNotes[index] = 'A';
    break;
  case 6:
    recordedNotes[index] = 'B';
    break;
  case 7:
    recordedNotes[index] = 'C'; // Octave change, so we have another 'C'
    break;
}
```

```
        noteDurations[index] = duration; // Store the duration of the note
        index++;
        // Pause for a short duration to prevent accidental button presses
        pause(100);
      }
      // Update the last state of the button
      lastButtonState[i] = buttonState[i];
    }
  }
}
//function playBack
void playBack(char* recordedNotes, int* noteDurations, int* leds) {
  volatile int index = 0;
  volatile int playbackButtonState;// playBack button State
  while (index < MAX_NOTES && recordedNotes[index] != '\0') {
    playbackButtonState = input(BTN_PLAYBACK);
    if (playbackButtonState == 0) {
      dprint(lcd, "Paused\n");
      while (input(BTN_PLAYBACK) == 0) {
        pause(100); // Wait until the button is released to resume playback
      }
      dprint(lcd, "Resuming\n");
      pause(500); // Pause for readability
```

```
 writeChar(lcd, CLR); // Clear the second line
 writeChar(lcd, CRT); // Move cursor to the second line
}
else {
 // Play note
 int noteDuration = noteDurations[index];
 char note = recordedNotes[index];
 switch (note) {
   case 'c':
     dprint(lcd, "Playing: c\n");
     freqout(10, noteDuration, 2093); // Play C note
     break;
   case 'D':
     dprint(lcd, "Playing: D\n");
     high(leds[0]);
     freqout(10, noteDuration,  2349);//Play D note
     low(leds[0]);
     break;
   case 'E':
     dprint(lcd, "Playing: E\n");
     high(leds[1]);
     freqout(10, noteDuration, 2637); // Play E note
     low(leds[1]);
```

```
  break;
case 'F':
  dprint(lcd, "Playing: F\n");
  high(leds[2]);
  freqout(10, noteDuration, 2793); // Play F note
  low(leds[2]);
  break;
case 'G':
  dprint(lcd, "Playing: G\n");
  high(leds[3]);
  freqout(10, noteDuration,  3136); // Play G note
  low(leds[3]);
  break;
case 'A':
  dprint(lcd, "Playing: A\n");
  high(leds[4]);
  freqout(10, noteDuration, 3520); // Play A note
  low(leds[4]);
  break;
case 'B':
  dprint(lcd, "Playing: B\n");
  high(leds[5]);
  freqout(10, noteDuration, 3951); // Play B note
```

```
        low(leds[5]);
        break;
      case 'C' :
        dprint(lcd, "Playing: C\n");//Play C note
        freqout(10, noteDuration, 4186);
        break;
    }
    index++;
    pause(50);
    writeChar(lcd, CLR); // Clear the second line
    writeChar(lcd, CRT); // Move cursor to the second line
    // Turn off all LEDs after note is played
    for(int i = 0; i < 6; i++) {
      low(leds[i]);
    }
  }
}
 // Display "Playback Over!" after all recorded notes have been played
  dprint(lcd, "Playback Over!\n");
  pause(5000);
 //Display "Starting over.." to start playing
  dprint(lcd, "Starting over..\n");
  pause(5000); // Pause for 500 seconds before starting again
```

```
}

// cog for led lighting up

void led(void *par1){

  while(1){

    int buttonState[8] = {input(0), input(1), input(2), input(3), input(4), input(5), input(6),
input(7)}; // Array to store the current state of each button

    for(int i = 1; i < 7; i++){

      if(buttonState[i] == 1){

        high(leds[i-1]);

      }

      else{

        low(leds[i-1]);

      }

    }

  }

}
```

[3] RESULTS

- Successful integration of various hardware components with the Parallax Propeller Activity Board.
- Responsive functionality observed during testing, with push buttons triggering corresponding notes and LEDs providing real-time visual feedback.
- Smooth operation of recording and playback functionalities, enabling users to capture and review musical compositions effortlessly.
- Effective communication of relevant information through the LCD display during performance, recording, and playback modes.
- Demonstrated stability and scalability, showcasing the system's potential as a valuable tool for music education and exploration.
- Overall, the result is a robust and versatile interactive music system that simplifies music learning and fosters creativity, with potential for significant impact in music education and beyond.

[4] CONCLUSION

In conclusion, our project has successfully achieved its goal of creating an interactive music system that provides users with a user-friendly platform for music education and exploration. Through the integration of various hardware components with the Parallax Propeller Activity Board, we have developed a system that responds reliably to user input, facilitates smooth recording and playback functionalities, and effectively communicates relevant information through the LCD display.

The culmination of our efforts represents a significant step forward in simplifying music learning and fostering creativity among users of diverse backgrounds. With its demonstrated stability, scalability, and potential for further refinement, our interactive music system has the capacity to make a meaningful impact in the realm of music education and beyond.

[5] REFERENCE

➔ What’s A Microcontroller