CS/EE 120B Custom Laboratory Project Report
Piano Hero

Bobby Lerias
March 17, 2022

Introduction

Piano Hero is a LED matrix game that emulates the nature of the legendary game, Guitar Hero. Utilizing a joystick, a buzzer, and the internal EEPROM, the player will be using the joystick to choose from a menu of songs that are available to play. Once a song is chosen, a countdown will be displayed and will shortly start the song. As the player hits the corresponding button to the light display correctly, the chosen song will come to life. The goal of the game is to have fun all while trying your best to obtain the highest score.

At the current state of Piano Hero, I have successfully implemented two songs, "Titanic Theme Song" and "7 Nation Army". However, in my song selection menu, I have listed a third song, "Supalonley". I did not plan on implementing more than two songs, however, I wanted to showcase the scrolling ability on the LCD and show its potential to have more than two songs.

Complexities

All three complexities were successfully implemented:

- Joystick (used to traverse the menu)
- Internal EEPROM (used to save the highest score)
- Buzzer (used to output the notes of the song)

User Guide

INSTRUCTIONS:

1. Click joystick to start game.
2. Using joystick, scroll through menu to select song.
3. Wait for countdown to end and the game will begin.
4. Hit the correct buttons to play the song and increase your score.
5. Wait or click the joystick for the End Screens to finish displaying. After, you will be returned to the Song Selection Menu to play the next song!

CONTROLS:

- Joystick
  - Move up and down to traverse through Song Selection Menu.
  - Click to select song and skip screens.

- Piano Keys (5 buttons)
    - Located below the LCD Display, when a light flashes onto the LED Matrix, hit the corresponding button to play the note and earn points.
- Reset Button
    - At any point in the game, hit this button located next to the LCD Display to return to the opening screen. If hit during song, score will not be saved.

VISUAL DISPLAY:

- 1602 Module LCD
    - Displays:
        - **Opening Screen**: displays until user clicks to continue
        - **Song Selection Menu**: can scroll and choose from three songs
        - **Countdown and Current High Score**: countdown from 3-1 on first row. Second row displays the current held high score.
        - **Score Keep**: will be displayed during gameplay. Score will increment by 1 if the user hits the correct key.
        - **End of Song Screen**: displays "YOU DID GREAT!" and the score received for 5 seconds. Click joystick to skip.
        - **New High Score Screen**: if current high score is beaten, new high score will be displayed here for 5 seconds. Click joystick to skip.
- 8x8 RBG LED Matrix
    - Initially starts off with two white columns and five different colors at the bottom row in the order: green, red, yellow, blue, and pink.
    - Once a song is chosen, it will flash a column of color which will signify to click that button.

SOUND:

- Buzzer
    - When the LED matrix displays a column of light and the corresponding button is hit, the Buzzer will play a note of the chosen song.

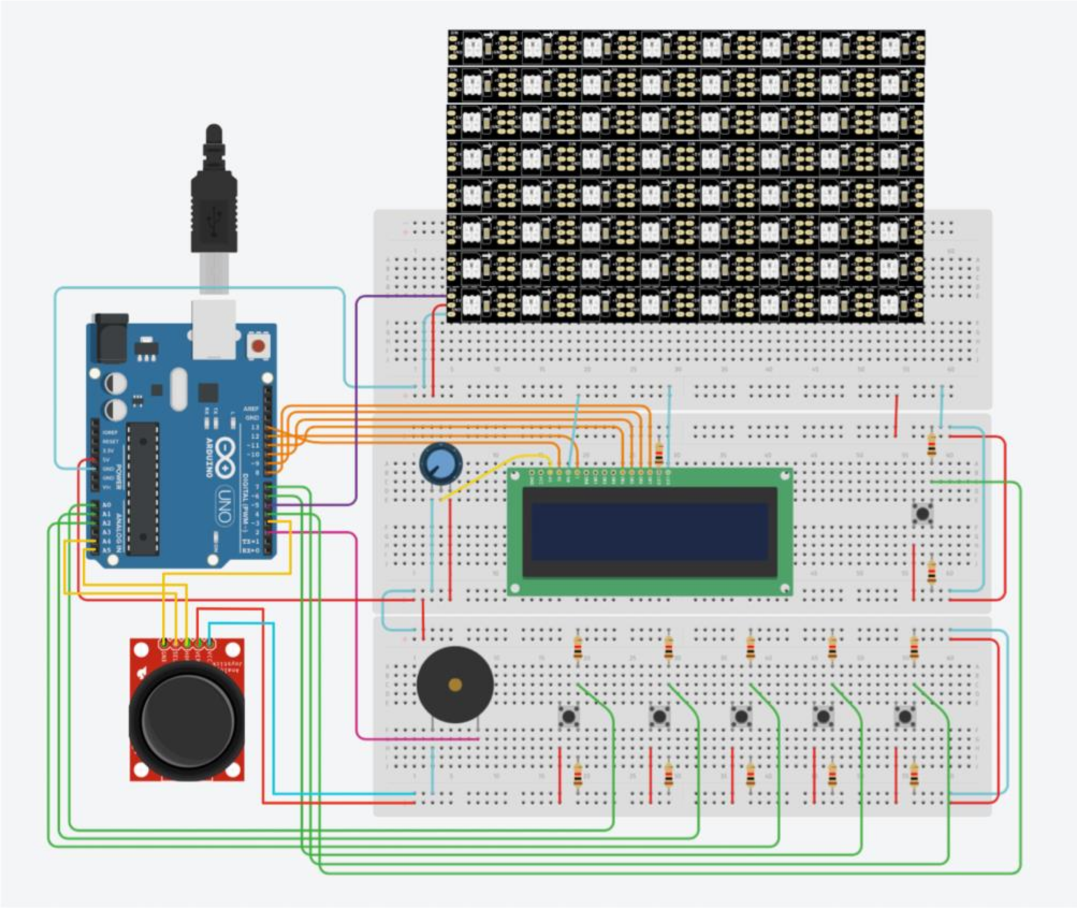
Hardware Components Used

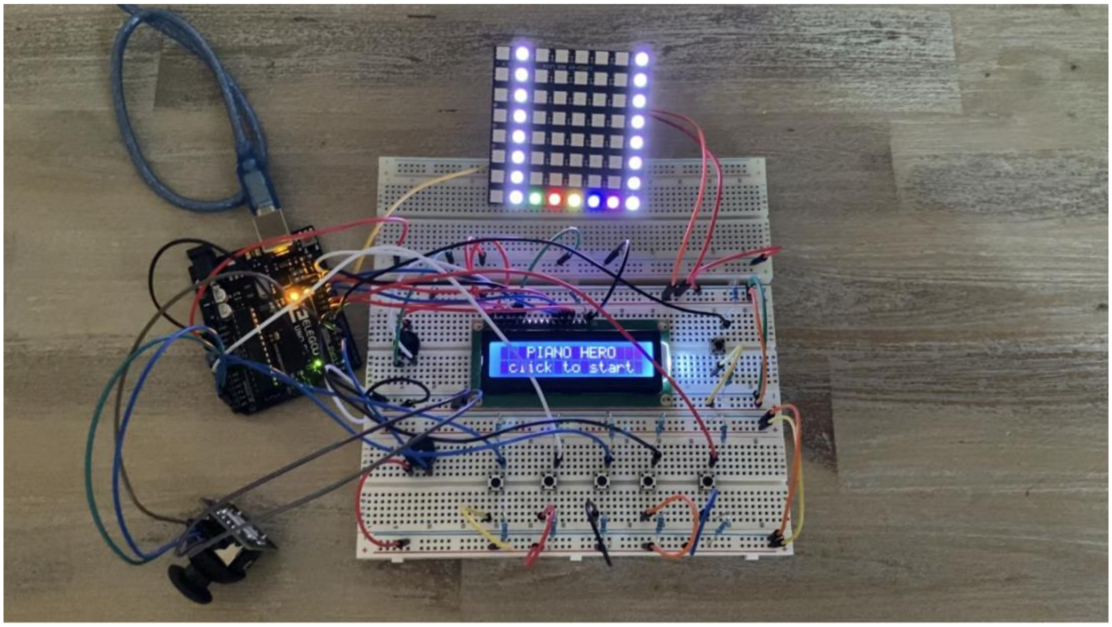
- ELEGOO UNO R3 Board
- Breadboard (x3)
- Button (6x)
- Resistor (13x)
- Buzzer
- 1602 Module LCD Display
- Potentiometer
- 8x8 RGB LED Matrix
- Joystick
- Wires

Software Libraries Used

- FastLED.h
    - Resources:
        - https://www.youtube.com/watch?v=T-LYYBJsu4Y
        - https://fastled.io/
    - The FastLED Library allowed me to manipulate each individual LED on the LED Matrix. Allowed me to write and obtain desired colors with ease.
- EEPROM.h
    - Resources:
        - https://www.youtube.com/watch?v=ShqvATqXA7g&t=1335s
        - https://www.arduino.cc/en/Reference/EEPROM
    - EEPROM.h allowed me to save the highest score into Arduino's internal EEPROM. By doing this, the highest score will be saved even if powered off.
- LiquidCrystal.h
    - Resources:
        - https://www.arduino.cc/en/Reference/LiquidCrystal
    - LiquidCrystal.h allowed me to display the Opening Menu and various screens on the 1602 Module LCD Display.
- pitches.h
    - Resources:
        - https://www.youtube.com/watch?v=DQsSiShI3vs&ab_channel=Williamo%27hara
        - https://www.arduino.cc/en/Tutorial/BuiltInExamples/toneMelody
    - Part of the Arduino library, I used the header file, pitches.h found on https://www.arduino.cc/en/Tutorial/BuiltInExamples/toneMelody, to output the sound of the current note in the song.

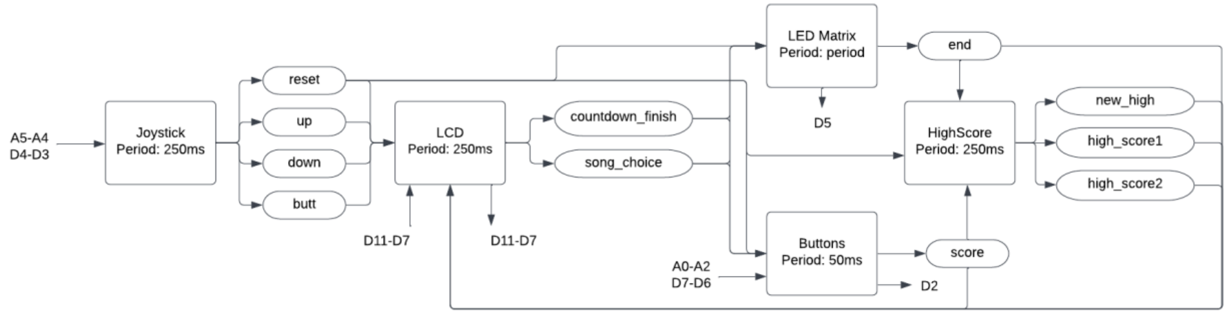
Wiring Diagram





Task Diagram

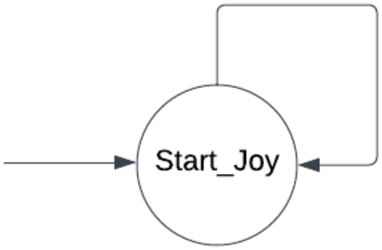
int reset, up, down, butt, countdown_finish, song_choice, end, score, new_high, high_score1, high_score2, t, period = 0;



SynchSM Diagrams

JOYSTICK

int xPosition, yPosition, SW_state, mapX, mapY = 0;
Period = 250ms

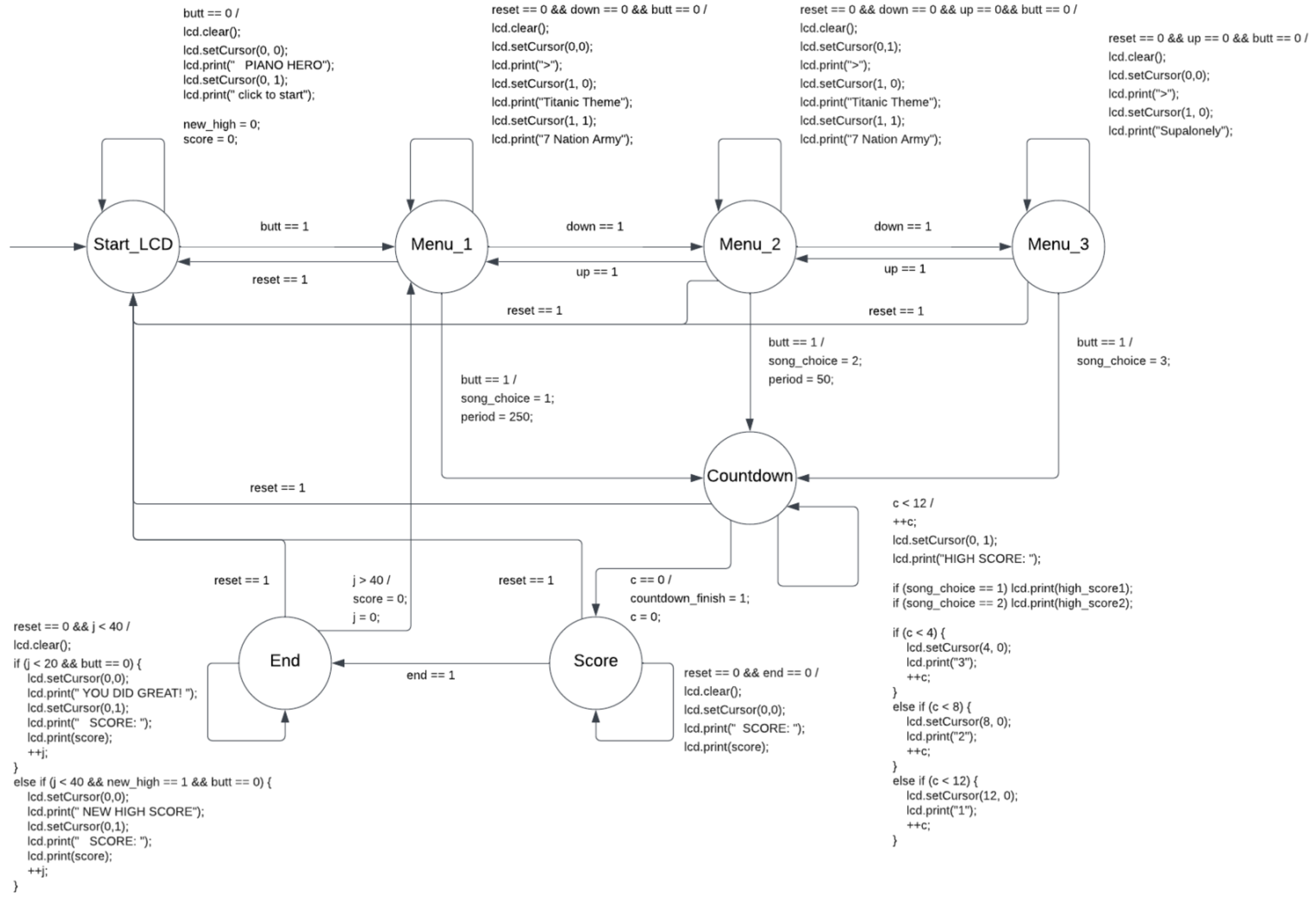


```
1 /
xPosition = analogRead(VRx);
yPosition = analogRead(VRy);
SW_state = digitalRead(SW);
mapX = map(xPosition, 0, 1023, -512, 512);
mapY = map(yPosition, 0, 1023, -512, 512);


if (mapX <= -400 && mapY < 60) {
   up = 1;
}
else if (mapX >= 400 && mapY < 60) {
   down = 1;
}
else {
   up = 0;
   down = 0;
}

if (SW_state == 0) {
   butt = 1;
}
else {
  butt = 0;
}

if (digitalRead(RESET) == 1) {
  reset = 1;
}

else if (digitalRead(RESET) == 0) {
  reset = 0;
}
```

LCD

int c, j = 0;

Period = 250ms



```
butt == 0 /
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("   PIANO HERO");
lcd.setCursor(0, 1);
lcd.print(" click to start");

new_high = 0;
score = 0;
```

```
reset == 0 && down == 0 && butt == 0 /
lcd.clear();
lcd.setCursor(0,0);
lcd.print(">");
lcd.setCursor(1, 0);
lcd.print("Titanic Theme");
lcd.setCursor(1, 1);
lcd.print("7 Nation Army");
```

```
reset == 0 && down == 0 && up == 0&& butt == 0 /
lcd.clear();
lcd.setCursor(0,1);
lcd.print(">");
lcd.setCursor(1, 0);
lcd.print("Titanic Theme");
lcd.setCursor(1, 1);
lcd.print("7 Nation Army");
```

```
reset == 0 && up == 0 && butt == 0 /
lcd.clear();
lcd.setCursor(0,0);
lcd.print(">");
lcd.setCursor(1, 0);
lcd.print("Supalonely");
```

```
butt == 1 /
song_choice = 1;
period = 250;
```

```
butt == 1 /
song_choice = 2;
period = 50;
```

```
butt == 1 /
song_choice = 3;
```

```
c < 12 /
++c;
lcd.setCursor(0, 1);
lcd.print("HIGH SCORE: ");

if (song_choice == 1) lcd.print(high_score1);
if (song_choice == 2) lcd.print(high_score2);

if (c < 4) {
    lcd.setCursor(4, 0);
    lcd.print("3");
    ++c;
}
else if (c < 8) {
    lcd.setCursor(8, 0);
    lcd.print("2");
    ++c;
}
else if (c < 12) {
    lcd.setCursor(12, 0);
    lcd.print("1");
    ++c;
}
```

```
c == 0 /
countdown_finish = 1;
c = 0;
```

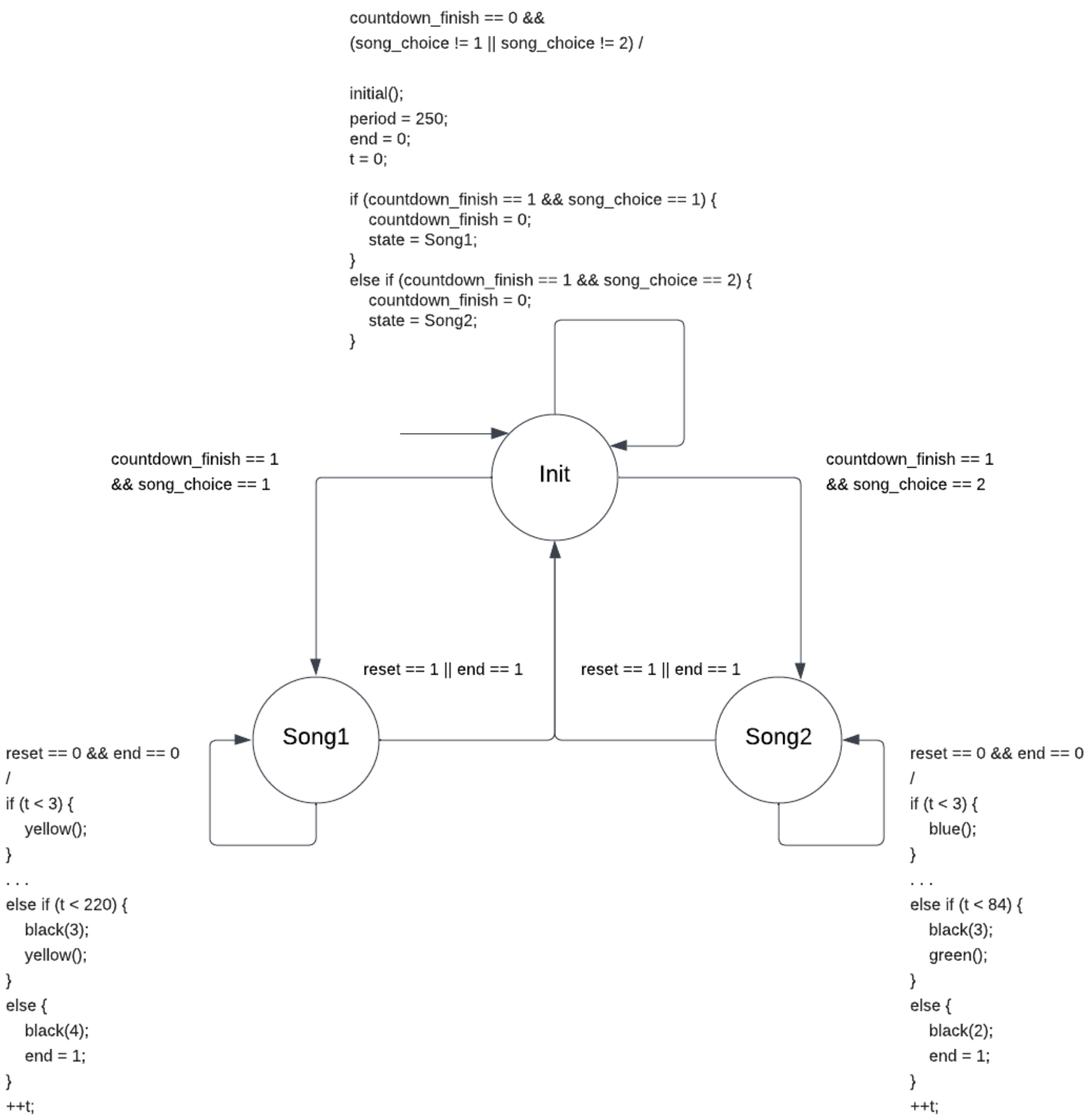
```
reset == 0 && end == 0 /
lcd.clear();
lcd.setCursor(0,0);
lcd.print("  SCORE: ");
lcd.print(score);
```

```
j > 40 /
score = 0;
j = 0;
```

```
reset == 0 && j < 40 /
lcd.clear();
if (j < 20 && butt == 0) {
    lcd.setCursor(0,0);
    lcd.print(" YOU DID GREAT! ");
    lcd.setCursor(0,1);
    lcd.print("   SCORE: ");
    lcd.print(score);
    ++j;
}
else if (j < 40 && new_high == 1 && butt == 0) {
    lcd.setCursor(0,0);
    lcd.print(" NEW HIGH SCORE");
    lcd.setCursor(0,1);
    lcd.print("   SCORE: ");
    lcd.print(score);
    ++j;
}
```

LED Matrix

Period = period



```
countdown_finish == 0 &&
(song_choice != 1 || song_choice != 2) /

initial();
period = 250;
end = 0;
t = 0;

if (countdown_finish == 1 && song_choice == 1) {
    countdown_finish = 0;
    state = Song1;
}
else if (countdown_finish == 1 && song_choice == 2) {
    countdown_finish = 0;
    state = Song2;
}
```

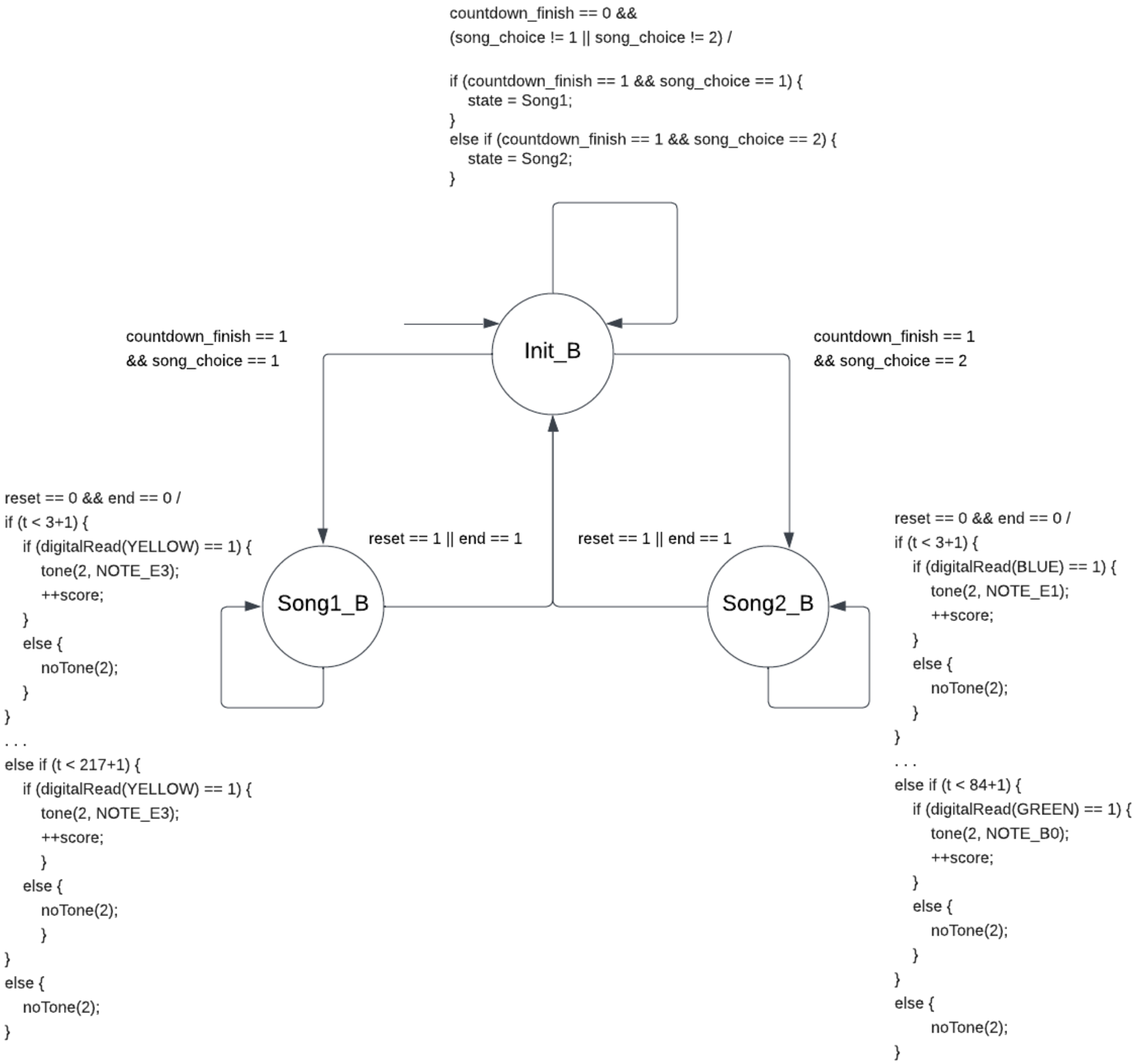
```
reset == 0 && end == 0
/
if (t < 3) {
    yellow();
}
. . .
else if (t < 220) {
    black(3);
    yellow();
}
else {
    black(4);
    end = 1;
}
++t;
```

```
reset == 0 && end == 0
/
if (t < 3) {
    blue();
}
. . .
else if (t < 84) {
    black(3);
    green();
}
else {
    black(2);
    end = 1;
}
++t;
```

NOTE: depending on song selection, period of this task will change. If Song1 is chosen, period will be 250ms. If Song2 is selected, period will be 50ms. Note 'period' is modified in LCD task.
NOTE: ' . . . ' reflects repetitive code of turning on and off lights to display song on matrix
NOTE: yellow(), black(), etc. call helper functions to light up the designated column

Button

Period = 50ms



```
countdown_finish == 0 &&
(song_choice != 1 || song_choice != 2) /

if (countdown_finish == 1 && song_choice == 1) {
    state = Song1;
}
else if (countdown_finish == 1 && song_choice == 2) {
    state = Song2;
}
```

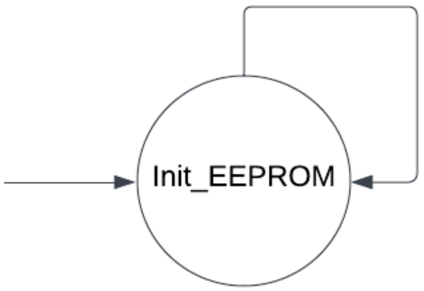
```
reset == 0 && end == 0 /
if (t < 3+1) {
    if (digitalRead(YELLOW) == 1) {
        tone(2, NOTE_E3);
        ++score;
    }
    else {
        noTone(2);
    }
}
. . .
else if (t < 217+1) {
    if (digitalRead(YELLOW) == 1) {
        tone(2, NOTE_E3);
        ++score;
        }
    else {
        noTone(2);
        }
}
else {
    noTone(2);
}
```

```
reset == 0 && end == 0 /
if (t < 3+1) {
    if (digitalRead(BLUE) == 1) {
        tone(2, NOTE_E1);
        ++score;
    }
    else {
        noTone(2);
    }
}
. . .
else if (t < 84+1) {
    if (digitalRead(GREEN) == 1) {
        tone(2, NOTE_B0);
        ++score;
    }
    else {
        noTone(2);
    }
}
else {
        noTone(2);
}
```

NOTE: ‘ . . . ‘ reflects repetitive code of time windows to hit certain buttons to earn points

High Score

Period = 250ms



```
1 /
 if (song_choice == 1) {
          if (score > high_score1 ) {

            if (reset == 0) {
              new_high = 1;

              if (score > 510) {
                EEPROM.write(2, score - 510);
              }
              else if (score > 255) {
                EEPROM.write(1, score - 255);
              }
              else {
                EEPROM.write(0, score);
              }

              if (end == 1) {
                high_score1 = EEPROM.read(0) + EEPROM.read(1) + EEPROM.read(2);
              }
            }

          }
        }
        else if (song_choice == 2) {
          if (score > high_score2) {

            if (reset == 0) {
              new_high = 1;

              if (score > 510) {
                EEPROM.write(5, score - 510);
              }
              else if (score > 255) {
                EEPROM.write(4, score - 255);
              }
              else {
                EEPROM.write(3, score);
              }

              if (end == 1) {
                Serial.println("REACHED");
                high_score2 = EEPROM.read(3) + EEPROM.read(4) + EEPROM.read(5);
              }
            }

          }

        }
```