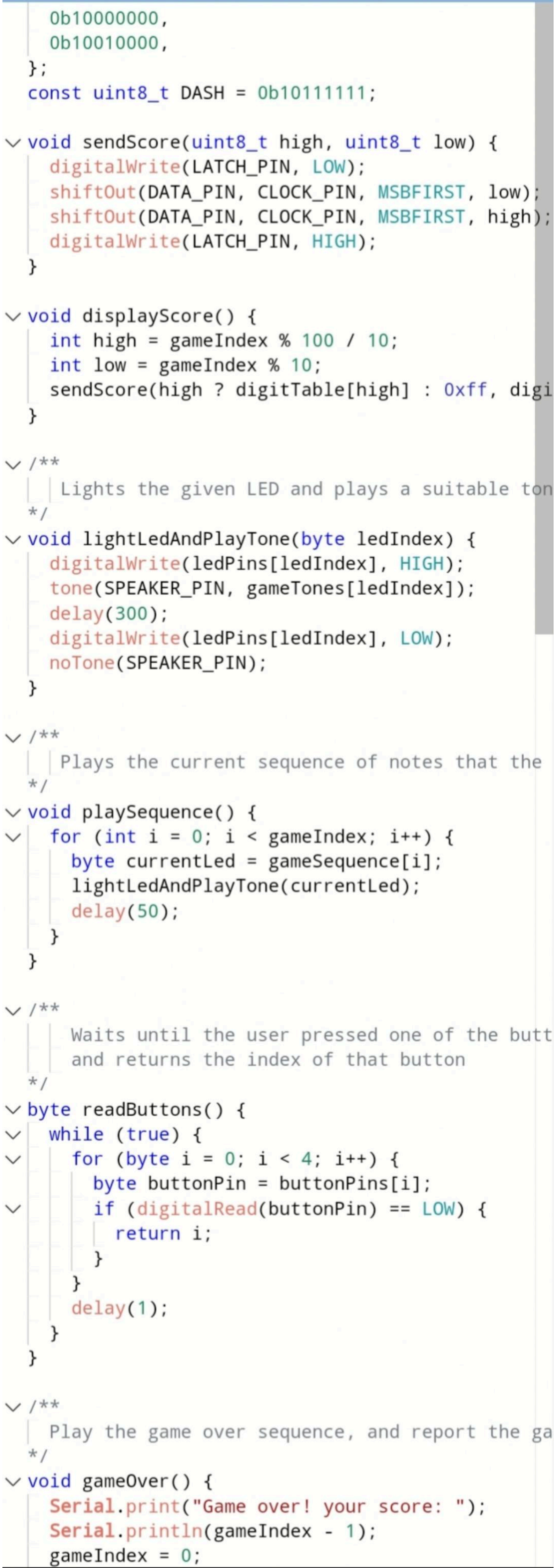
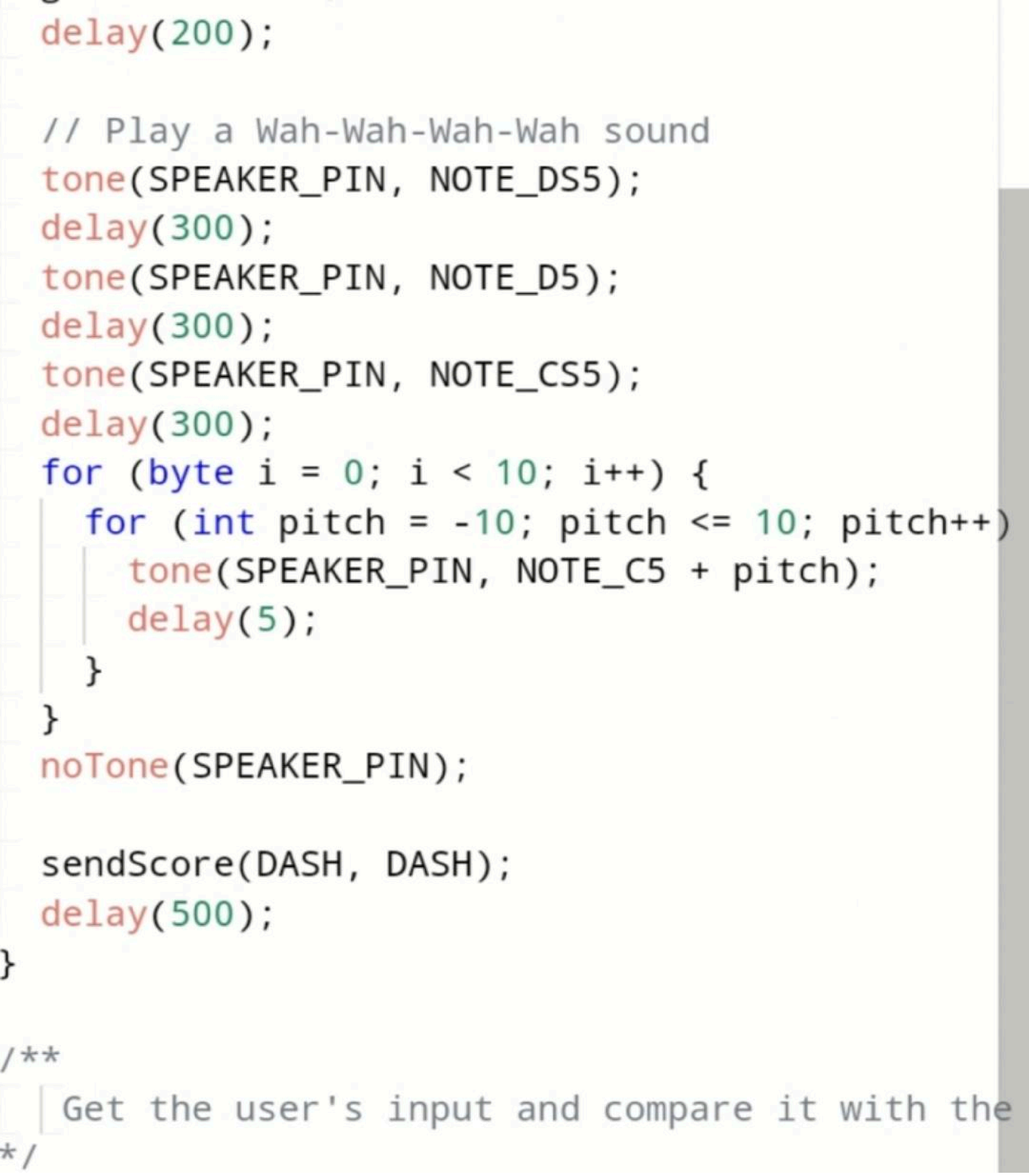
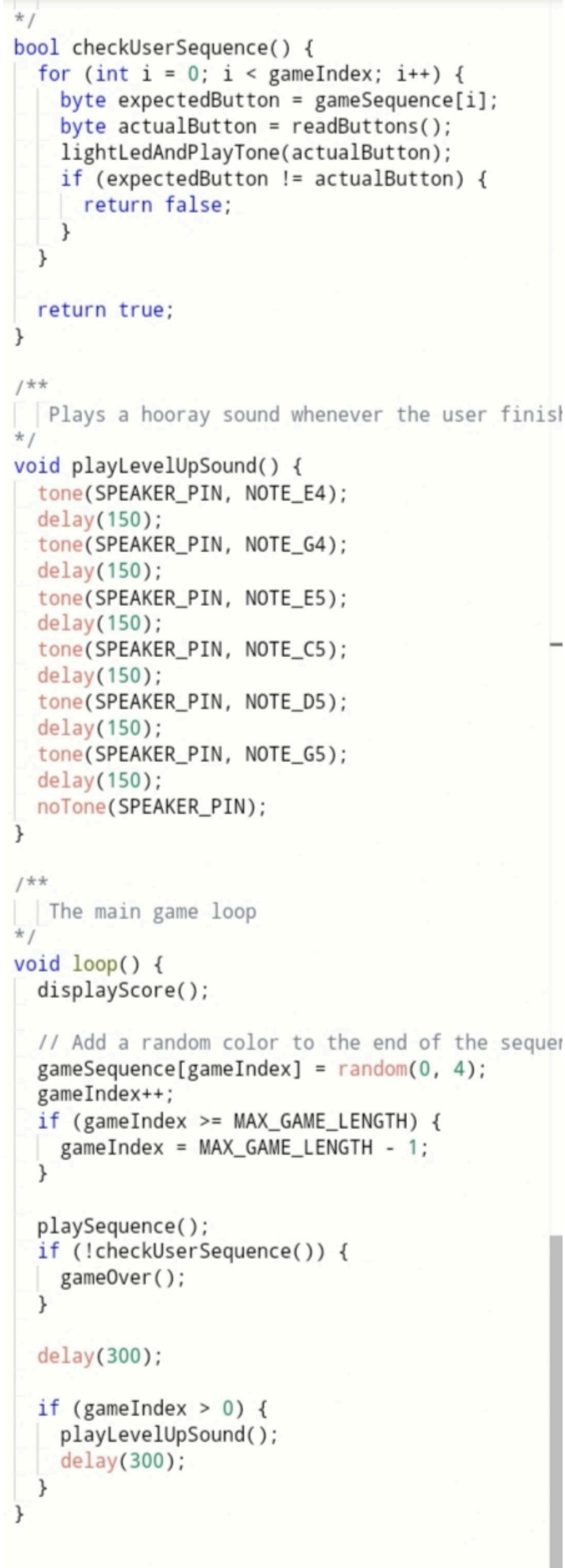
```
/**
   Simon Game for Arduino with Score display

   Copyright (C) 2022, Uri Shaked

   Released under the MIT License.
*/

#include "pitches.h"

/* Constants - define pin numbers for LEDs,
   buttons and speaker, and also the game tones:
const uint8_t ledPins[] = {9, 10, 11, 12};
const uint8_t buttonPins[] = {2, 3, 4, 5};
#define SPEAKER_PIN 8

// These are connected to 74HC595 shift register
const int LATCH_PIN = A1;  // 74HC595 pin 12
const int DATA_PIN = A0;  // 74HC595pin 14
const int CLOCK_PIN = A2;  // 74HC595 pin 11

#define MAX_GAME_LENGTH 100

const int gameTones[] = { NOTE_G3, NOTE_C4, NOTE

/* Global variables - store the game state */
uint8_t gameSequence[MAX_GAME_LENGTH] = {0};
uint8_t gameIndex = 0;

/**
   Set up the Arduino board and initialize Seria
*/
void setup() {
  Serial.begin(9600);
  for (byte i = 0; i < 4; i++) {
    pinMode(ledPins[i], OUTPUT);
    pinMode(buttonPins[i], INPUT_PULLUP);
  }
  pinMode(SPEAKER_PIN, OUTPUT);
  pinMode(LATCH_PIN, OUTPUT);
  pinMode(CLOCK_PIN, OUTPUT);
  pinMode(DATA_PIN, OUTPUT);

  // The following line primes the random number
  // It assumes pin A3 is floating (disconnected
  randomSeed(analogRead(A3));
}

/* Digit table for the 7-segment display */
const uint8_t digitTable[] = {
  0b11000000,
  0b11111001,
  0b10100100,
  0b10110000,
  0b10011001,
  0b10010010,
  0b10000010,
  0b11111000,
```



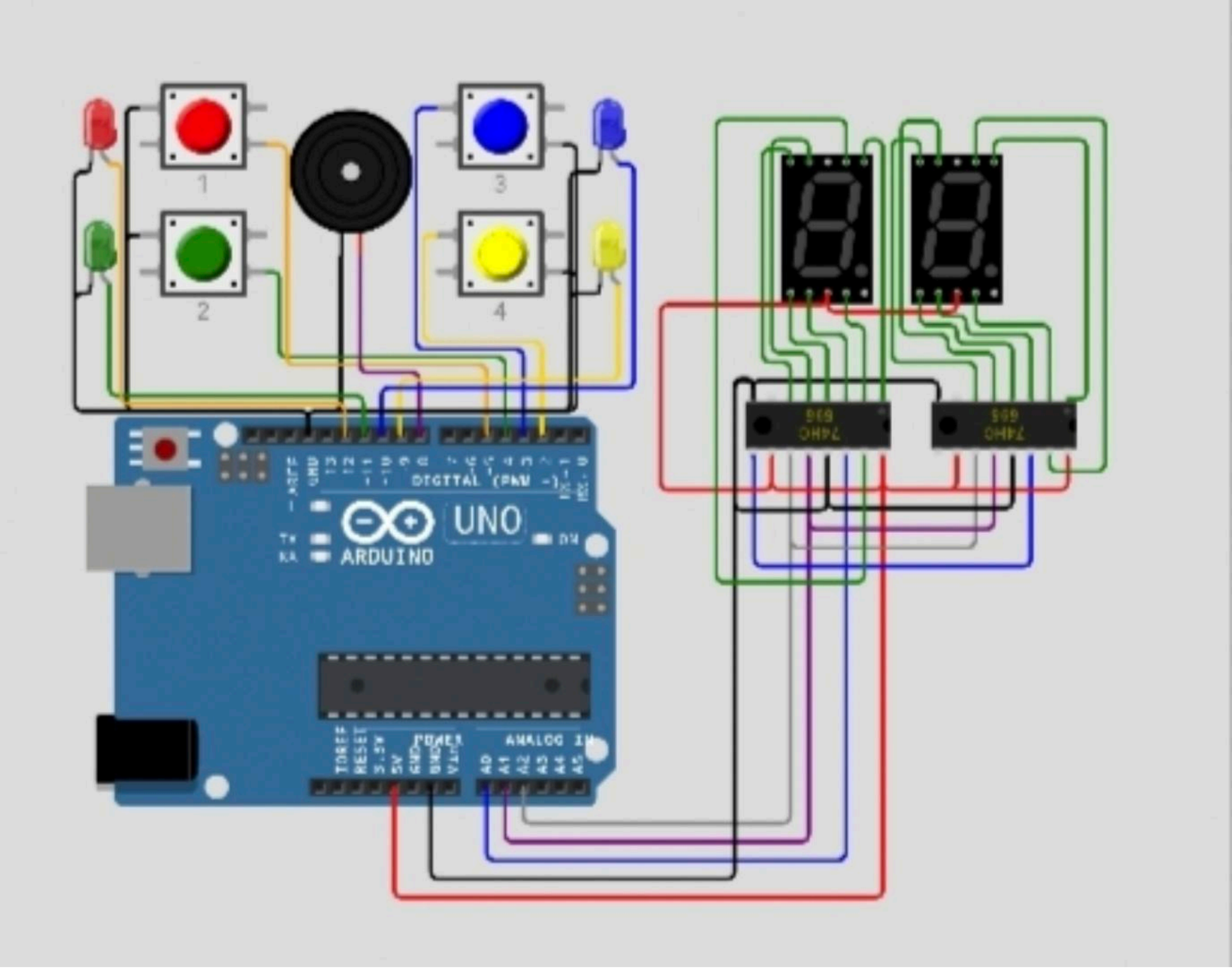
```
  0b10000000,
  0b10010000,
};
const uint8_t DASH = 0b10111111;

void sendScore(uint8_t high, uint8_t low) {
  digitalWrite(LATCH_PIN, LOW);
  shiftOut(DATA_PIN, CLOCK_PIN, MSBFIRST, low);
  shiftOut(DATA_PIN, CLOCK_PIN, MSBFIRST, high);
  digitalWrite(LATCH_PIN, HIGH);
}

void displayScore() {
  int high = gameIndex % 100 / 10;
  int low = gameIndex % 10;
  sendScore(high ? digitTable[high] : 0xff, digi
}

/**
  Lights the given LED and plays a suitable ton
*/
void lightLedAndPlayTone(byte ledIndex) {
  digitalWrite(ledPins[ledIndex], HIGH);
  tone(SPEAKER_PIN, gameTones[ledIndex]);
  delay(300);
  digitalWrite(ledPins[ledIndex], LOW);
  noTone(SPEAKER_PIN);
}

/**
  Plays the current sequence of notes that the
*/
void playSequence() {
  for (int i = 0; i < gameIndex; i++) {
    byte currentLed = gameSequence[i];
    lightLedAndPlayTone(currentLed);
    delay(50);
  }
}

/**
   Waits until the user pressed one of the butt
   and returns the index of that button
*/
byte readButtons() {
  while (true) {
    for (byte i = 0; i < 4; i++) {
      byte buttonPin = buttonPins[i];
      if (digitalRead(buttonPin) == LOW) {
        return i;
      }
    }
    delay(1);
  }
}

/**
  Play the game over sequence, and report the ga
*/
void gameOver() {
  Serial.print("Game over! your score: ");
  Serial.println(gameIndex - 1);
  gameIndex = 0;
```



```
  delay(200);

  // Play a Wah-Wah-Wah-Wah sound
  tone(SPEAKER_PIN, NOTE_DS5);
  delay(300);
  tone(SPEAKER_PIN, NOTE_D5);
  delay(300);
  tone(SPEAKER_PIN, NOTE_CS5);
  delay(300);
  for (byte i = 0; i < 10; i++) {
    for (int pitch = -10; pitch <= 10; pitch++)
      tone(SPEAKER_PIN, NOTE_C5 + pitch);
      delay(5);
    }
  }
  noTone(SPEAKER_PIN);

  sendScore(DASH, DASH);
  delay(500);
}

/**
   Get the user's input and compare it with the
*/
```



```
*/
bool checkUserSequence() {
  for (int i = 0; i < gameIndex; i++) {
    byte expectedButton = gameSequence[i];
    byte actualButton = readButtons();
    lightLedAndPlayTone(actualButton);
    if (expectedButton != actualButton) {
      return false;
    }
  }

  return true;
}

/**
  Plays a hooray sound whenever the user finis
*/
void playLevelUpSound() {
  tone(SPEAKER_PIN, NOTE_E4);
  delay(150);
  tone(SPEAKER_PIN, NOTE_G4);
  delay(150);
  tone(SPEAKER_PIN, NOTE_E5);
  delay(150);
  tone(SPEAKER_PIN, NOTE_C5);
  delay(150);
  tone(SPEAKER_PIN, NOTE_D5);
  delay(150);
  tone(SPEAKER_PIN, NOTE_G5);
  delay(150);
  noTone(SPEAKER_PIN);
}

/**
  The main game loop
*/
void loop() {
  displayScore();

  // Add a random color to the end of the seque
  gameSequence[gameIndex] = random(0, 4);
  gameIndex++;
  if (gameIndex >= MAX_GAME_LENGTH) {
    gameIndex = MAX_GAME_LENGTH - 1;
  }

  playSequence();
  if (!checkUserSequence()) {
    gameOver();
  }

  delay(300);

  if (gameIndex > 0) {
    playLevelUpSound();
    delay(300);
  }
}
```


1
2
3
4
74HC595
74HC595
UNO
ARDUINO
DIGITAL (PWM ~)
ANALOG IN
POWER