```
#include "pitches.h"
#include <Wire.h>
#include <avr/pgmspace.h>
#include "Adafruit_Trellis.h"

Adafruit_Trellis matrix0 = Adafruit_Trellis();
Adafruit_TrellisSet trellis =  Adafruit_TrellisSet(&matrix0);


// set to however many Trellis' working with here, up to 8
#define NUMTRELLIS 1
#define numKeys (NUMTRELLIS * 16)
#define MAX_NOTES 200

//matrix of notes
int freqs[]=
{NOTE_C6,NOTE_B5,NOTE_A5,NOTE_G5,NOTE_F5,NOTE_E5,NOTE_D5,NOTE_C5,NOTE_C5,NOTE_B4,NO
TE_A4,NOTE_G4,NOTE_F4,NOTE_E4,NOTE_D4,NOTE_C4};

//pause and beep times
int p = 100;
int time = 250;

byte index14 = 27;
byte indexes14[] = {11, 11, 11, 10, 9, 9, 10, 9, 8, 6, 3, 3, 3, 6, 6, 6,
9, 9, 9, 11, 11, 11, 6, 8, 9, 10, 11};
int beepTimes14[] =
{time*3,time*3,time*2,time,time*3,time*2,time,time*2,time,time*6,
time,time,time,time,time,time,time,time,time,time,time,time,time,
time*2,time,time*2,time,time*6};
byte pauseTimes14[] = {p, p, p, p, p, p, p, p, p, p, p, p, p, p, p, p,
p, p, p, p, p, p, p, p, p, p, p };

//booleans (true or false)
int recordingBeep;
int recordingPause;
int replaying;
int resetSong;
int keyPressed;
int seqRight;
int playerTurn;
int completed;
int done;
int song15;
int song14;
int song13;
int song12;
```

```
//indices of the arrays
int index = 0;
int beep = 0;
int pause = 0;

//time recording
long startTime;
long startPauseTime;
long elapsedTime;
long elapsedPauseTime;

//arrays
int beepTimes[MAX_NOTES];
int pauseTimes[MAX_NOTES];
int indexes[MAX_NOTES];

//pitch changer
double pitchChange;
int threshold;

//switch between modes
int mode;
int lastMode = 0;
int RecordingMode = 0;
int LearningMode = 1;

//counters
int totalNotes = 0;
int notesRight = 0;
int notesPlayed = 0;
int wins = 0;

//run only once
/* initialize hardware
 * LEDs, speaker, matrix
 */
void setup() {
  Serial.begin(9600);
  Serial.println("Trellis Demo + BOSS_BOT");

  //initializing the led matrix
  pinMode(INTPIN, INPUT);
  digitalWrite(INTPIN, HIGH);

  //initializing
  pinMode(3, INPUT);
  digitalWrite(3, HIGH);
  pinMode(12, INPUT);
```

```
  digitalWrite(12, HIGH);

  //LED Bars
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);


  trellis.begin(0x70);  // only one Trellis init

  // light up all the LEDs in order
  for (uint8_t i=0; i<numKeys; i++) {
    trellis.setLED(i);
    trellis.writeDisplay();
    delay(50);
  }
  // then turn them off
  for (uint8_t i=0; i<numKeys; i++) {
    trellis.clrLED(i);
    trellis.writeDisplay();
    delay(50);
  }
}


void loop() {
  delay(30); // 30ms delay is required, dont remove me!

  //check if the replaying button was pressed.
  replaying = digitalRead(3);

  Serial.println(notesPlayed);

  //switch between modes
  mode = digitalRead(12);

  if ( mode == RecordingMode ) {

    //if last mode was learning mode, reset the recording mode
    if ( lastMode == LearningMode ){

      resetRecordingMode();
```

```
}

// If a button was just pressed or released...
if (trellis.readSwitches()) {

  // go through every button
  for (int i=0; i < numKeys; i++) {

    // if it was pressed, turn it on
    if (trellis.justPressed(i) && !recordingBeep) {

      trellis.setLED(i);
      tone(2,freqs[i]);

      //put the index in the array
      indexes[index] = i;
      index++;

      //add +1 note played
      notesPlayed++;

      //start recording time
      recordingBeep = true;
      startTime = millis();

      //if it was recording the pause, stop it.
      if (recordingPause) {
        recordingPause = false;
        elapsedPauseTime = millis() - startPauseTime;

        //put the pause in the array
        pauseTimes[pause] = elapsedPauseTime;
        pause++;

      }

    }

    // if it was released, turn it off
    if (trellis.justReleased(i) && recordingBeep) {
      trellis.clrLED(i);
      noTone(2);

      //stop beep timer
      recordingBeep = false;
      elapsedTime = millis() - startTime;

      //put the beep time in the array
```

```
      beepTimes[beep] = elapsedTime;
      beep++;


      //start recording pause
      recordingPause = true;
      startPauseTime = millis();

     }
    }
    // tell the trellis to set the LEDs we requested
    trellis.writeDisplay();
  }

    //update the pitch LEDs
    changePitch();

//if user play more too many notes, reset the array.
if ( notesPlayed > 190) {
  noTone(2);
  resetRecordingMode();
}

//if the replay button is pressed, replay the sequence
if ( replaying == 0 ) {

  //delay between the user's input and the
  delay(1000);

  //replay the array
  for ( int x = 0; x < beep; x++ ) {

    //check for changes in the pitch
    changePitch();

    //play the beep
    trellis.setLED((indexes[x]));
    trellis.writeDisplay();
    tone(2,freqs[indexes[x]]*pitchChange);
    delay(beepTimes[x]);

    //play the pause
    trellis.clrLED(indexes[x]);
    trellis.writeDisplay();
    noTone(2);
    delay(pauseTimes[x]);

  }
```

```
  //reset the recording mode after it replays
  resetRecordingMode();

}

  //update last mode
  lastMode = RecordingMode;

}//end of recording mode

//learning mode
else {

    //set LEDs according to the number of wins
    setLEDs( wins );

    //if last mode was Recording mode, reset the Learning Mode
    if (lastMode == RecordingMode){
      index = 0;
      notesRight = 0;
      seqRight = true;
      keyPressed = false;
      playerTurn = false;
      resetSongChoice();
    }

    //while a song has not been chosen, blink possible LEDs
    while (keyPressed == false) {

    //light LEDs
    trellis.setLED(15);
    trellis.setLED(14);
    trellis.setLED(13);
    trellis.setLED(12);
    trellis.writeDisplay();
    delay(500);

    //turn off LEDs
    trellis.clrLED(15);
    trellis.clrLED(14);
    trellis.clrLED(13);
    trellis.clrLED(12);
    trellis.writeDisplay();
    delay(500);

      //check  for buttons pressed
      if ( trellis.readSwitches() ) {
```

```
    //if button 15 was pressed, choose the song
    if ( trellis.justPressed(15) ) {
      song15 = true;
      totalNotes = index14; //index15, if you have another song.
      keyPressed = true;
    }

    //if button 14 was pressed, choose the song
    else if (trellis.justPressed(14)) {
      song14 = true;
      totalNotes = index14;
      keyPressed = true;
    }

    //if button 13 was pressed, choose the song
    else if (trellis.justPressed(13)) {
      song13 = true;
      totalNotes = index14; //index13, if you have another song.
      keyPressed = true;
    }

    //if button 12 was pressed, choose the song
    else if (trellis.justPressed(12)) {
      song12 = true;
      totalNotes = index14; //index12, if you have another song.
      keyPressed = true;
    }

  }

  //check the mode
  mode = digitalRead(12);

  //if the mode changes, break from this loop
  if (mode == RecordingMode) {
    break;
  }

}
  // if song 15 was chosen, play it.
  if (song15 && mode == LearningMode && playerTurn == false) {

    //player turn after song was played
    playerTurn = true;

    //play song
    for ( int x = 0; x < index14; x++ ) {
```

```
   //play the beep
   tone(2,freqs[indexes14[x]]);
   trellis.setLED((indexes14[x]));
   trellis.writeDisplay();
   delay(beepTimes14[x]);

   //play the pause
   noTone(2);
   trellis.clrLED(indexes14[x]);
   trellis.writeDisplay();
   delay(pauseTimes14[x]);

  }

}

// if song 14 was chosen, play it.
else if (song14 && mode == LearningMode && playerTurn == false) {

  //player turn after song was played
  playerTurn = true;

  //play song
  for ( int x = 0; x < index14; x++ ) {

   //play the beep
   tone(2,freqs[indexes14[x]]);
   trellis.setLED((indexes14[x]));
   trellis.writeDisplay();
   delay(beepTimes14[x]);

   //play the pause
   noTone(2);
   trellis.clrLED(indexes14[x]);
   trellis.writeDisplay();
   delay(pauseTimes14[x]);

  }

}

// if song 13 was chosen, play it.
else if (song13 && mode == LearningMode && playerTurn == false) {

  //player turn after song was played
  playerTurn = true;
```

```
        //play song
        for ( int x = 0; x < index14; x++ ) {

          //play the beep
          tone(2,freqs[indexes14[x]]);
          trellis.setLED((indexes14[x]));
          trellis.writeDisplay();
          delay(beepTimes14[x]);

          //play the pause
          noTone(2);
          trellis.clrLED(indexes14[x]);
          trellis.writeDisplay();
          delay(pauseTimes14[x]);

        }

      }

      // if song 12 was chosen, play it.
      else if (song12 && mode == LearningMode && playerTurn == false) {

        //player turn after song was played
        playerTurn = true;

        //play song
        for ( int x = 0; x < index14; x++ ) {

          //play the beep
          tone(2,freqs[indexes14[x]]);
          trellis.setLED((indexes14[x]));
          trellis.writeDisplay();
          delay(beepTimes14[x]);

          //play the pause
          noTone(2);
          trellis.clrLED(indexes14[x]);
          trellis.writeDisplay();
          delay(pauseTimes14[x]);

        }

      }

  //if the song was chosen and already played, get user input
  if ( keyPressed == true && playerTurn == true && mode == LearningMode &&
seqRight == true) {
```

```
//while user has the right sequence, get his input
while (seqRight == true && completed == false){
delay(30);

//check the mode
mode = digitalRead(12);

//if the mode changes, break from this loop
if (mode == RecordingMode) {
  break;
}

//if all notes were correct, wait for user to press button
if (notesRight == totalNotes) {

  //get button's input
  done = digitalRead(3);

  //if button was pressed
  if (done == 0){

    //count +1 win
    wins++;
    wins = wins % 9; //once you reach 8 LEDs, come back to 0.

    //song conpleted
    completed = true;

    //set show number of wins with LEDs
    setLEDs( wins );

  }
}

// get user's sequence
if (trellis.readSwitches()) {

  // go through every button
  for (int i=0; i < numKeys; i++) {

    // if it was pressed, turn it on
    if (trellis.justPressed(i)) {
       //put the index in the array
       indexes[index] = i;

       //compare the original song's note with user's, if wrong, start over
       if( indexes[index] != indexes14[index] ){
         seqRight = false;
```

```
        }

        //if right, count +1 right note
        else {
         notesRight++;
        }

        //update index
        index++;

        //turn on LEDs and play note
        trellis.setLED(i);
        trellis.writeDisplay();
        tone(2,freqs[i]);
      }

      //when button is released, turn off sound and LED
      if (trellis.justReleased(i)) {
       trellis.clrLED(i);
       trellis.writeDisplay();
       noTone(2);
      }

      //if the note was wrong, make animation and start over
      if (seqRight == false) {
       refresh();
      }

    }

  }


}//end of while

}//end of player mode

//when sequence completed, if button pressed, do animation
if ( completed == true ) {
  refresh();
  completed = false;
}

//if not on player turn, reset right sequence
if ( playerTurn == false ) {
  seqRight = true;
}
```

```
  //update last mode
  lastMode = LearningMode;

  }//end of learning mode

}//end of loop

/*
 * get the button's input, calculate the change in the pitch
 * and update the LED's according to the change.
 */
void changePitch() {

    //read the input from the button
    pitchChange = analogRead(0);

    //Serial.print("PitchChange: ");Serial.println(pitchChange);

    //make it an acceptable value to multiply the frequency
    pitchChange = (pitchChange/200) + 1;
    threshold = 191; //Frequencies: (C8 - C4)/7

    //light LEDs depending on the pitch
    if( freqs[15] * pitchChange <= NOTE_C4) {
      digitalWrite(11, HIGH);
      digitalWrite(10, LOW);
      digitalWrite(9, LOW);
      digitalWrite(8, LOW);
      digitalWrite(7, LOW);
      digitalWrite(6, LOW);
      digitalWrite(5, LOW);
      digitalWrite(4, LOW);
    }
    else if( freqs[15] * pitchChange <= NOTE_C4 + threshold) {
      digitalWrite(11, HIGH);
      digitalWrite(10, HIGH);
      digitalWrite(9, LOW);
      digitalWrite(8, LOW);
      digitalWrite(7, LOW);
      digitalWrite(6, LOW);
      digitalWrite(5, LOW);
      digitalWrite(4, LOW);
    }
    else if( freqs[15] * pitchChange <= NOTE_C4 + threshold*2) {
      digitalWrite(11, HIGH);
      digitalWrite(10, HIGH);
      digitalWrite(9, HIGH);
      digitalWrite(8, LOW);
```

```
  digitalWrite(7, LOW);
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
 }
 else if( freqs[15] * pitchChange <= NOTE_C4 + threshold*3) {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, LOW);
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
 }
 else if( freqs[15] * pitchChange <= NOTE_C4 + threshold*4) {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
 }
 else if( freqs[15] * pitchChange <= NOTE_C4 + threshold*5) {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
 }
 else if( freqs[15] * pitchChange <= NOTE_C4 + threshold*6) {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);
 }
 else {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
```

```
      digitalWrite(9, HIGH);
      digitalWrite(8, HIGH);
      digitalWrite(7, HIGH);
      digitalWrite(6, HIGH);
      digitalWrite(5, HIGH);
      digitalWrite(4, HIGH);
    }

}

// reset all booleans and do animation
void refresh() {

  playerTurn = false;
  seqRight = true;
  keyPressed = false;
  index = 0;
  notesRight = 0;

  // light up all the LEDs in order
  for (uint8_t i=0; i<numKeys; i++) {
    trellis.setLED(i);
    trellis.writeDisplay();
    delay(50);
  }

  // then turn them off
  for (uint8_t i=0; i<numKeys; i++) {
    trellis.clrLED(i);
    trellis.writeDisplay();
    delay(50);
  }

}

//unselect all the songs selected
void resetSongChoice() {
 song12 = false;
 song13 = false;
 song14 = false;
 song15 = false;
}

//reset all used variables
void resetRecordingMode() {
  beep = 0;
  index = 0;
  pause = 0;
```

```
  notesPlayed = 0;
  recordingPause = false;
  recordingBeep = false;
  replaying = false;
}

//turn set LEDs on if the sequence was succesfully completed
void setLEDs( int wins ) {

  if ( wins <= 0 ) {
    digitalWrite(11, LOW);
    digitalWrite(10, LOW);
    digitalWrite(9, LOW);
    digitalWrite(8, LOW);
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
  }
  else if ( wins <= 1 ) {
    digitalWrite(11, HIGH);
    digitalWrite(10, LOW);
    digitalWrite(9, LOW);
    digitalWrite(8, LOW);
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
  }
  else if ( wins <= 2 ) {
    digitalWrite(11, HIGH);
    digitalWrite(10, HIGH);
    digitalWrite(9, LOW);
    digitalWrite(8, LOW);
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
  }
  else if ( wins <= 3 ) {
    digitalWrite(11, HIGH);
    digitalWrite(10, HIGH);
    digitalWrite(9, HIGH);
    digitalWrite(8, LOW);
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
```

```
}
else if ( wins <= 4 ) {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, LOW);
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
}
else if ( wins <= 5 ) {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
}
else if ( wins <= 6 ) {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
}
else if ( wins <= 7 ) {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);
}
  else {
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
```

```
    digitalWrite(5, HIGH);
    digitalWrite(4, HIGH);
  }


}
```