Discoteca 2.0

(modified from)

Discoteca

ENGR102 Intro to Electronics

Shoreline Community College

November 2017

Description of the problem. Why are you building this circuit?

The circuit Discoteca that was presented as make-it better report #3 would play a melody made at run-time by the user and relate every one of the 8 supported notes to a color in the RGB LED and the light would blink at the pace of the melody. Now this improved Discoteca 2.0 will output important information at run-time via an LCD screen to the user such as the current playing note, the current speed level of the song (out of three levels) and the frequency of the current playing note. Discoteca 2.0 also features a linear array of 8 LEDs thanks to the shift register which makes it possible to have so many digital outputs. The LEDs are on according to the current playing note leading to a much better animation than the single RGB LED.

What things can you do with this circuit?

You can make melodies at run-time by manipulating the soft-potentiometer in order to manipulate the current playing note (the system supports 8 notes: a, b, c, C, d, e, f, g and one no sound state). You can manipulate the flex sensor to change the pace of the melody based on three levels (1 is the fastest, 3 is the slowest). You can see the RGB LED blink along with the song with different colors which are related to each note. You can learn from Discoteca 2.0 by reading the useful information shown on the LCD screen (note, speed and frequency). The Linear array of 8 LEDs brings the user a nice animation that is a function of the notes playing.

Knowledge from which prior experiments is required?

Knowledge from the following circuits in the SIK guide: Circuit #3 to manipulate the RGB LED, circuit #4 to manipulate a big number of LEDs, #9 in order to operate the flex sensor, circuit #10 to operate the soft potentiometer, from circuit #11 to operate the piezo buzzer, #14 to operate the Shift register, #15 to know how to use the LCD screen and the knowledge from Original Discoteca.

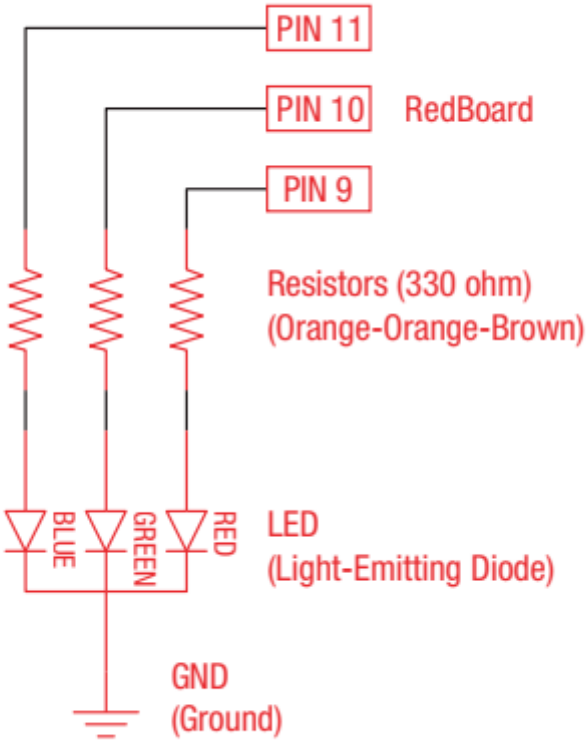


Figure 1. Schematic of circuit #3 (RGB LED).

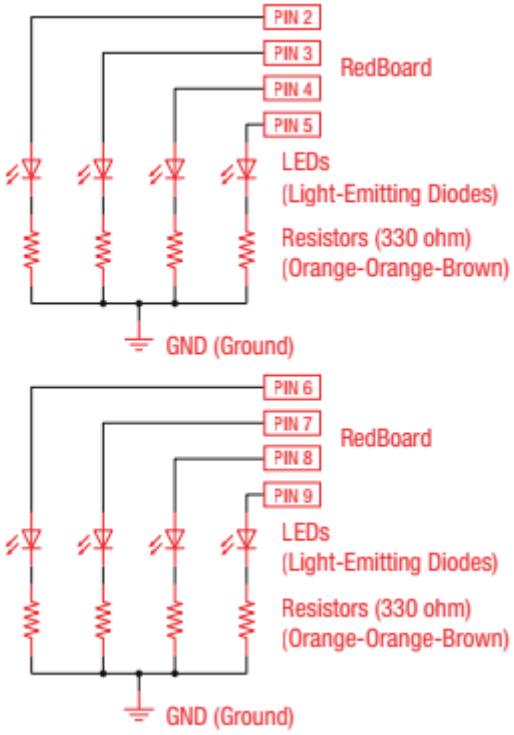


Figure 2. Schematic of circuit #4 (Multiple LEDs)

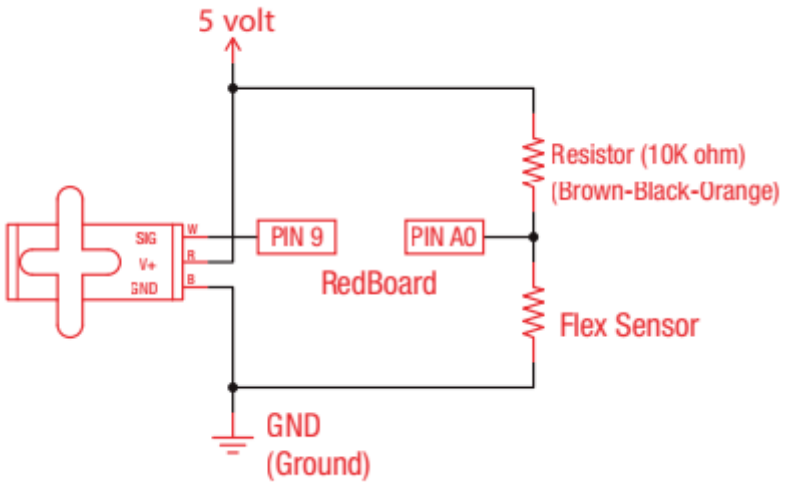


Figure 3. Schematic of circuit #9 (Flex Sensor).

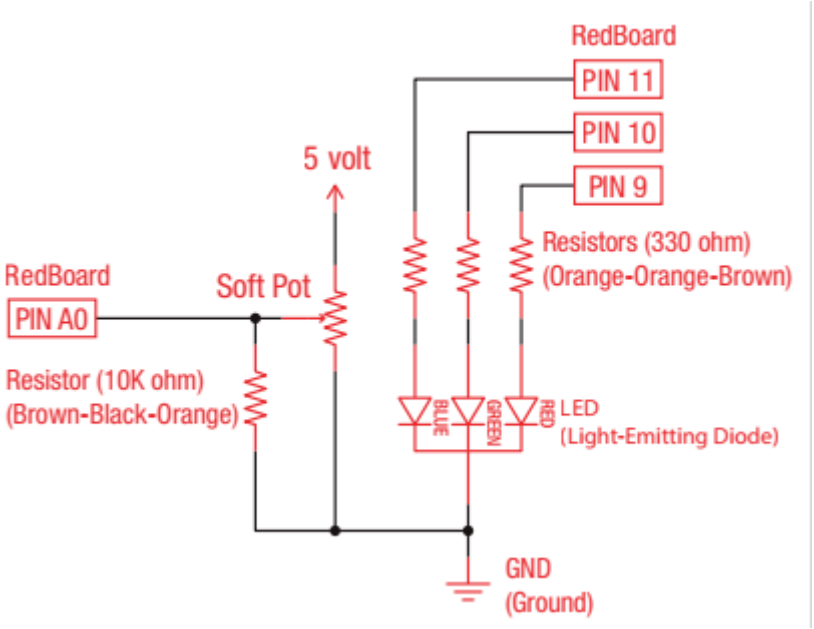


Figure 4. Schematic of circuit #10 (Soft Potentiometer).

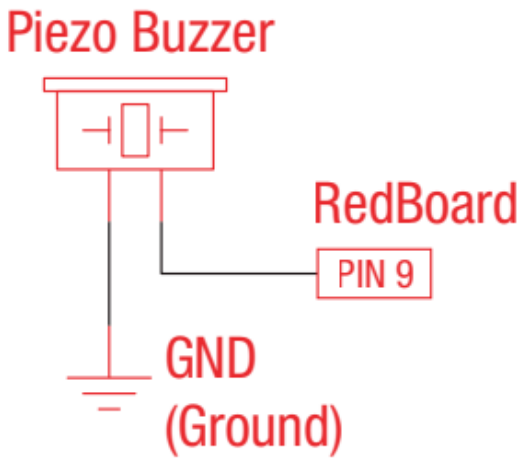


Figure 5. Schematic of circuit #11 (Piezo Buzzer)

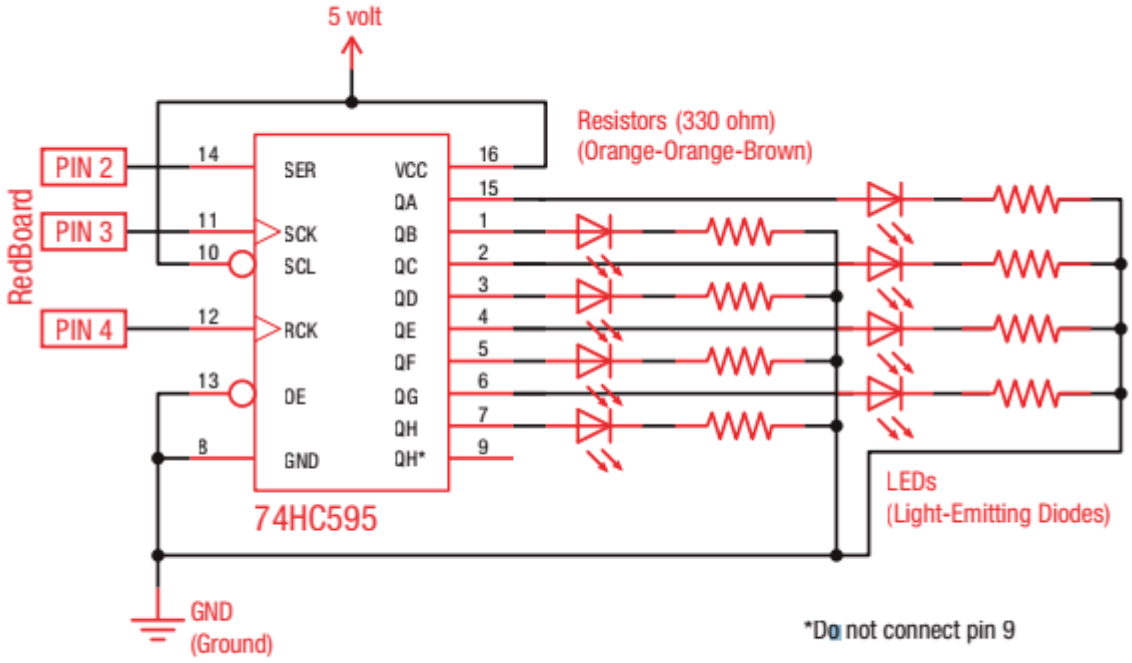


Figure 6. Schematic of Circuit #14 (Shift Register)

The codes for these circuits can be found at: https://www.sparkfun.com/sikcode

Parts Required

- One (1) Arduino Board from the Sparkfun kit.
- One (1) Flex Sensor.
- One (1) RGB LED.
- One (1) Soft Potentiometer.
- One (1) 10kΩ Resistors.
- One (1) 5kΩ Resistors.
- One (1) LCD Screen of 16x2.
- Two (2) Red LEDs.
- Two (2) Yellow LEDs.
- Two (2) Green LEDs.
- Two (2) Blue LEDs.
- One (1) Shift Register.
- Three (8) 330Ω Resistors.
- One (1) Piezo Buzzer
- Jumpers/Wire as needed.

Problem-solving approach

I followed the following steps (steps 1-7 take you to Original Discoteca, the last steps get you to Discoteca 2.0):

1) Learn how to control each element with the Arduino Board:
   1.1) Learn how to control the RGB LED.
   1.2) Learn how to measure flex using the Flex Sensor.
   1.3) Learn how to measure pressure with the Soft potentiometer.
   1.4) Learn how to play melodies with the Piezo Buzzer.
2) Start with the Piezo buzzer playing fixed pre-programmed sounds.
3) Control its speed with the Flex Sensor.
4) Control the beating with the Soft Potentiometer.
5) We should have enough variables figured out that can make the LED blink as desired, so add the LED.
6) Adjust ranges, intervals, constants and other values that might arise from empirical beta testing.
7) Remember to debug before and after every step from 1 to 6. Then debug at the very end.
8) Learn how to use the Shift Register.
9) Learn how to manipulate many LEDs at once.
10) Learn how to use the Shift Register to manipulate many LEDs at once.
11) Learn how to use the LCD Screen.
12) Mix all the knowledge to produce Discoteca 2.0.
13) Code and Debug.

Problems

- I had issues with the LCD at first. However, I repeated the wiring more carefully and managed to make it work properly.
- I had run out of digital pins, what I did is I realized I should use a shift register. The implementation of a shift register solved the issue. I am also using the red part of the RGB LED as an analog pin (the Arduino is enabled to let you do exactly that).
- Issues when coding the Linear array of 8 LEDs. At first it would do exactly the opposite animation. I ran the numbers more carefully and realized my mistake in the code which I then fixed (another solution would've been to re-wire the LEDS and the shift-register, but I prefer the way the LEDs are wired right now).
- I also forgot the buzzer to ground connection many times when building and re-building the circuit.
- The flex sensor seemed broken. Prayed to god and then it worked.

Schematic (The picture is big and couldn't find into this page, therefore the rest of this page is blank).

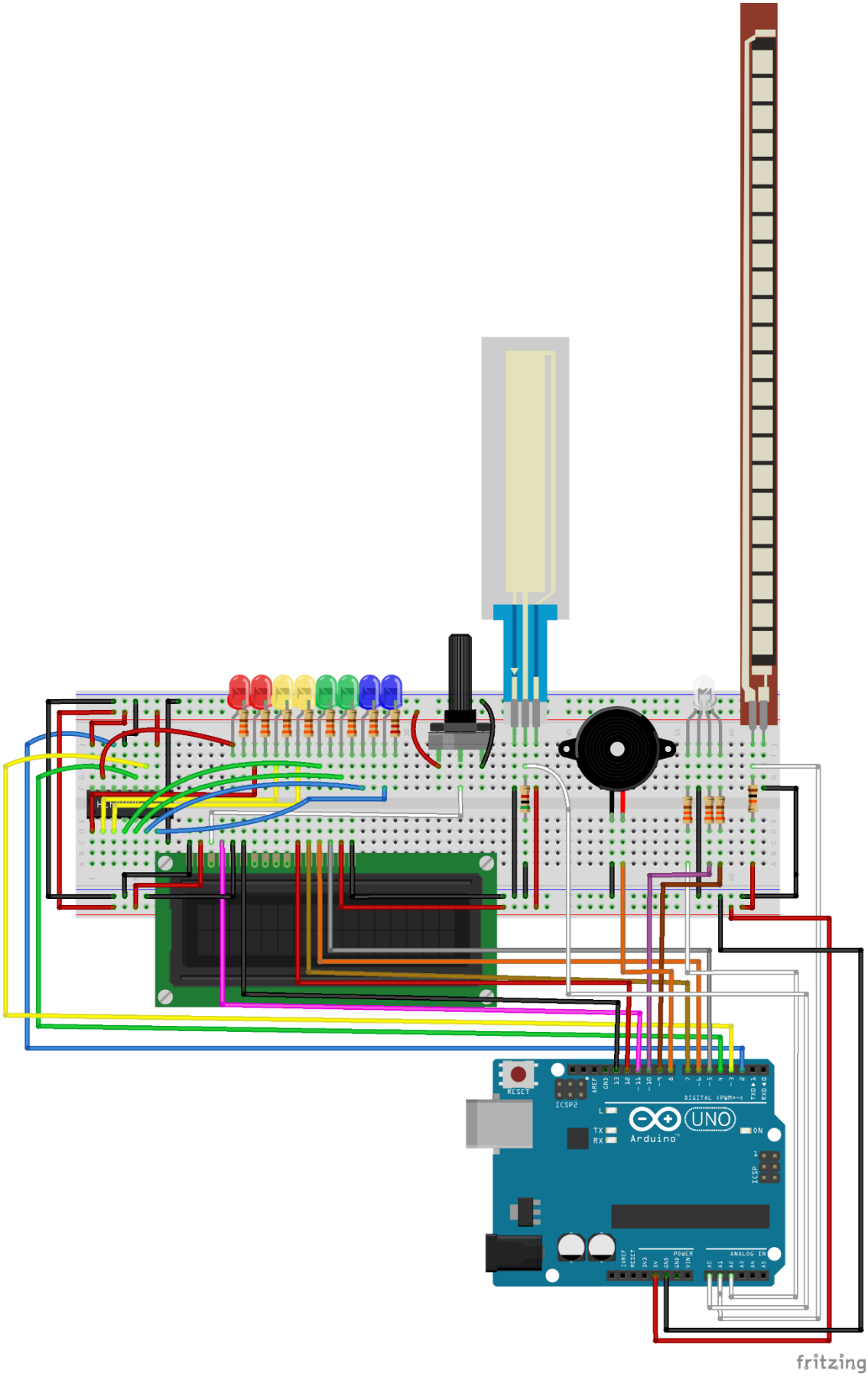


Figure 7. Breadboard View

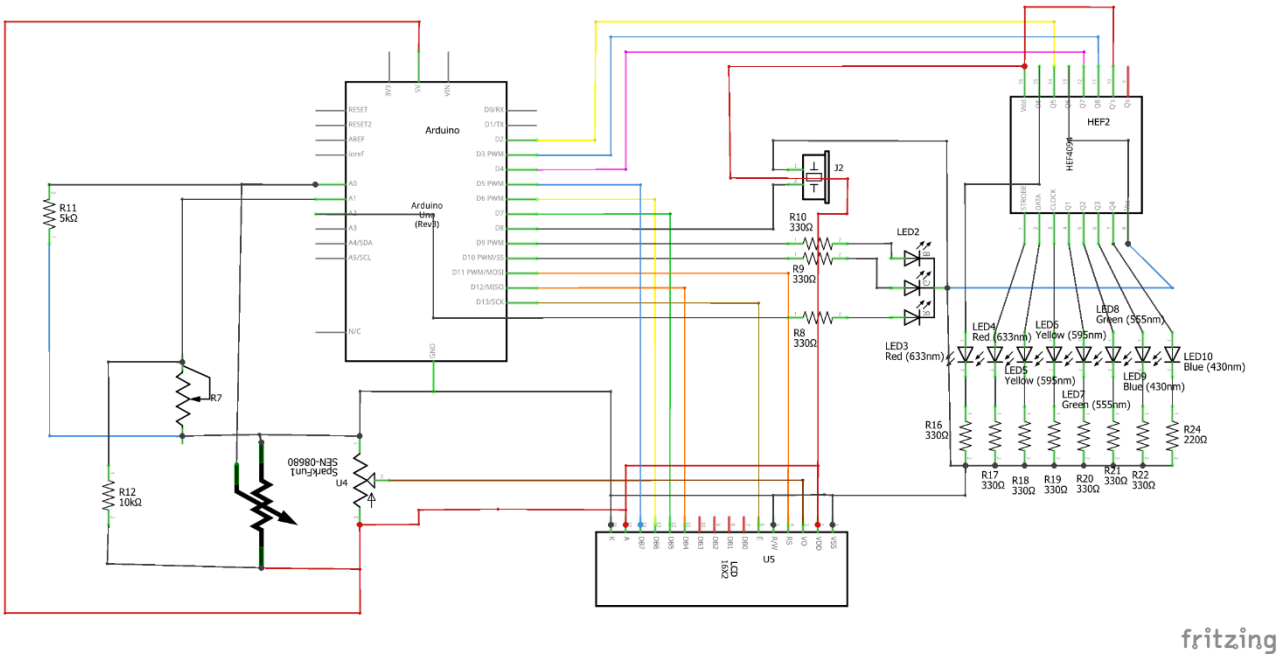


Figure 8. Schematic View

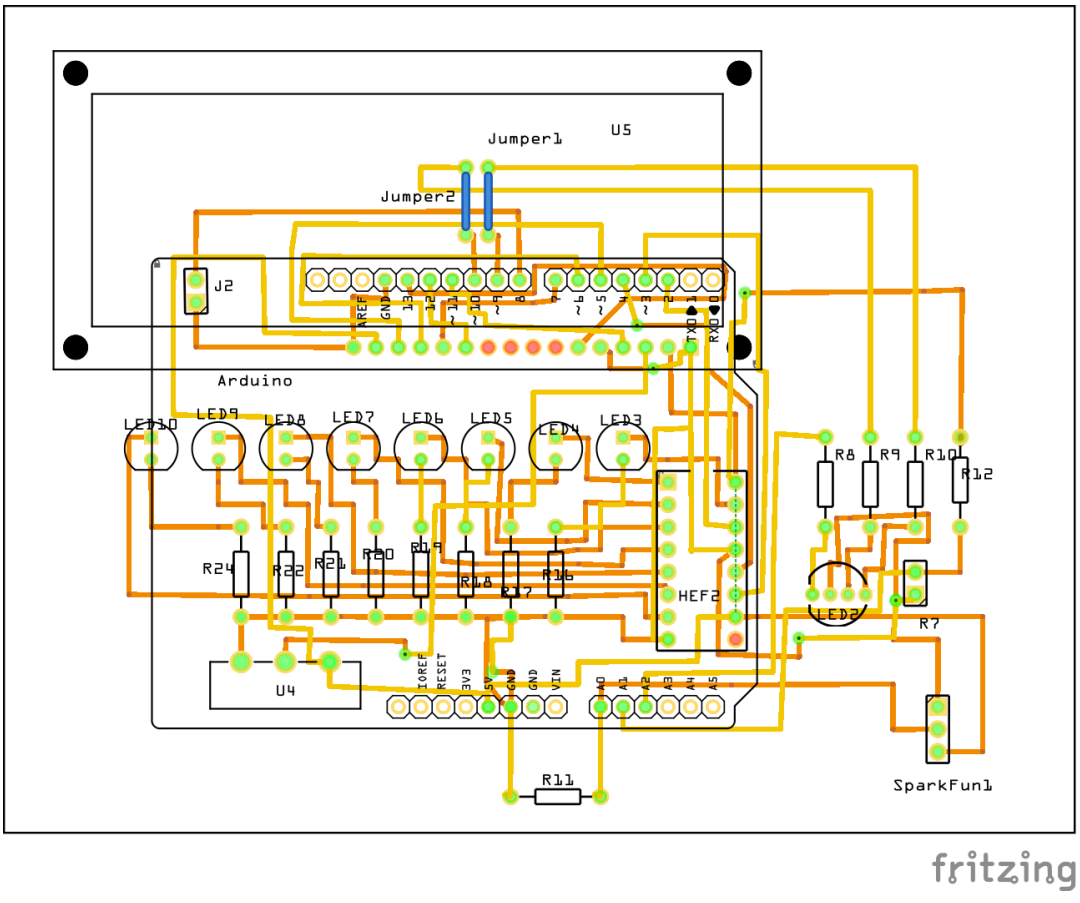


Figure 9. PCB View

As Fritzing did not have a Soft Potentiometer I put a regular potentiometer to model the soft potentiometer I used in real life.

Code

```
/*
 * Name: Rafael Antonio Laya Alfonzo
 * November 2017
 * Class: ENGR 102 intro to electronics (Fall 2017)
 * Software Version: Arduino 1.8.2
 * Source code for Discoteca 2.0 more information on the report
 * The reader can use this code as they please but I won't be liable of anything
*/


#include <LiquidCrystal.h>
LiquidCrystal lcd(11, 13, 12, 7, 6, 5);

// some constants for abstraction and readability
const int softPot = A0;
const int flexSensor = A1;
const int buzzerPin = 8;
const int blueLed = 9;
const int greenLed = 10;
const int redLed = 16;
const int numLeds = 8;

int dataPin = 2;
int clockPin = 3;
int latchPin = 4;

byte data = 0;

// tempo is constant, and is an empirical obtained value
const int tempo = 113;

void setup()
{
  Serial.begin(9600);
  pinMode(buzzerPin, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(greenLed, OUTPUT);

  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(latchPin, OUTPUT);


  lcd.begin(16, 2); //Initialize the 16x2 LCD
```

```
  lcd.clear();  //Clear any old data displayed on the LCD
}

void loop()
{
  int beats;
  int flexVal;
  int softPotVal;
  int note;

  // colors is an array of three integers, interpret as following:
  // - colors[i] = 1 if we want to light the LED related to i ON
  // i = 0 is red, i = 1 is green, i = 2 is blue
  int colors[3];

  // a list of notes is just an array of characters
  // interpret each character as a note
  char notesList[] = {'a', 'b', 'c', 'C', 'd', 'e', 'f', 'g', ' '};
  const int numNotes = 9;

  // read the sensors
  softPotVal = analogRead(softPot);
  flexVal = analogRead(flexSensor);

  Serial.println(flexVal);

  // beats is a function of the flex sensor
  switch (flexVal)
  {
    case 800 ... 900:
      beats = 1;
      break;
    case 720 ... 799:
      beats = 4;
      break;
    default:
      beats = 8;
      break;
  }

  Serial.println(softPotVal);
  // the notes are in function of the reading on the soft pot
  // also the colors match the notes
  switch (softPotVal)
  {
    case 0 ... 49:
      note = pickRandomNote(notesList, numNotes);

      // do not turn the leds on when note means stop
      if (note == ' ')
      {
        colors[0] = 0;
```

```
    colors[1] = 0;
    colors[2] = 0;
  }
  else
  {
    // if we succesfully get a note, pick a color at random that is not 000
    do
    {
      colors[0] = random(0, 2);
      colors[1] = random(0, 2);
      colors[2] = random(0, 2);
    } while ((colors[0] == 0) &&
             (colors[1] == 0) &&
             (colors[2] == 0));
  }
  break;

case 50 ... 140:
  note = 'c';
  colors[0] = 0;
  colors[1] = 0;
  colors[2] = 1;
break;

case 141 ... 254:
  note = 'd';
  colors[0] = 0;
  colors[1] = 0;
  colors[2] = 1;
  break;

case 255 ... 381:
  note = 'e';
  colors[0] = 0;
  colors[1] = 1;
  colors[2] = 0;
  break;

case 382 ... 508:
  note = 'f';
  colors[0] = 0;
  colors[1] = 1;
  colors[2] = 1;
  break;

case 509 ... 635:
  note = 'g';
  colors[0] = 1;
  colors[1] = 0;
  colors[2] = 0;
  break;
```

```
    case 636 ... 762:
      note = 'a';
      colors[0] = 1;
      colors[1] = 0;
      colors[2] = 1;
      break;

    case 763 ... 889:
      note = 'b';
      colors[0] = 1;
      colors[1] = 1;
      colors[2] = 0;
      break;

    default:
      note = 'C';
      colors[0] = 1;
      colors[1] = 1;
      colors[2] = 1;
      break;
  }

  printOnLCD(note, frequency(note, numNotes), beats);

  // play the buzzer
  tone(buzzerPin, frequency(note, numNotes), beats * tempo);

  // lights blink along with the buzzer
  blinkLightsON(colors, beats * tempo);

  // LEDs dance along with the frequency of the music
  danceLeds(note);

  delay(beats * tempo / 2);

  blinkLightsOFF();

  delay(beats * tempo / 2);

  // this is also taken empirically
  delay(beats * tempo / 5);
}

void blinkLightsOFF()
{
  digitalWrite(redLed, LOW);
  digitalWrite(blueLed, LOW);
  digitalWrite(greenLed, LOW);
}

void blinkLedsOFF()
{
```

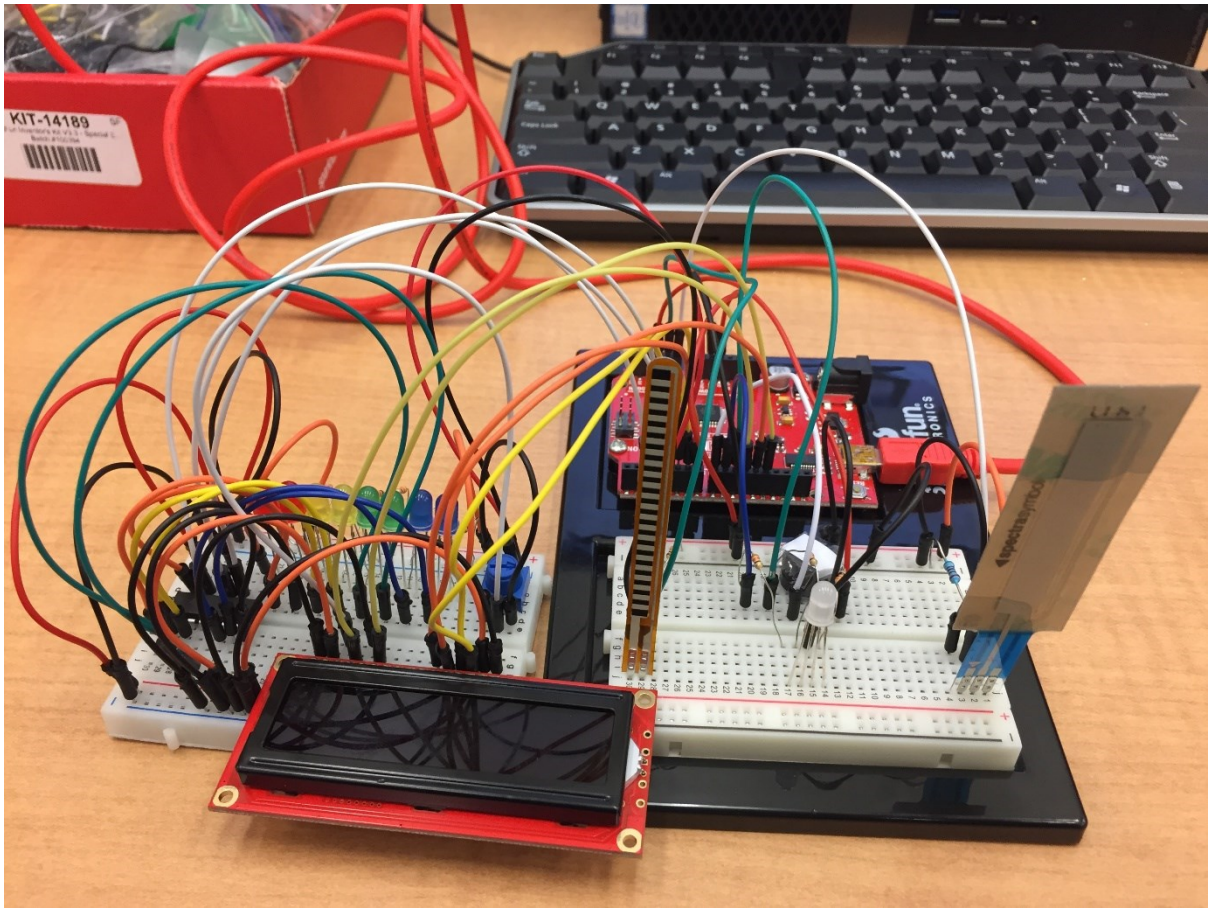
```
  for(int i = 0; i < 8; i++)
  {
    shiftWrite(i, LOW);
  }
}

// this function takes an array of colors and the speed in which to blink
// then makes the RGB Led blink as the parameters specify
void blinkLightsON(int colors[], int vel)
{
  digitalWrite(redLed, colors[0]);
  digitalWrite(greenLed, colors[1]);
  digitalWrite(blueLed, colors[2]);
}

// this function takes a note and assigns to it a frequency
// thanks to the SIK guide for this piece of code
int frequency(char note, int numNotes)
{
  int i;

  // these are the notes
  char notesList[numNotes] = {
    'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};

  // frequency[i] matches names[i]
  int frequencies[numNotes] = {
    262, 294, 330, 349, 392, 440, 494, 523};

  // Now we'll search through the letters in the array, and if
  // we find it, we'll return the frequency for that note.
  for (i = 0; i < numNotes; i++)  // Step through the notes
  {
    if (notesList[i] == note)         // Is this the one?
    {
      return (frequencies[i]);    // Yes! Return the frequency and exit function.
    }
  }
  return 0;  // We looked through everything and didn't find it,
  // but we still need to return a value, so return 0.
}

char pickRandomNote(char notes[], int numNotes)
{
  return notes[random(0, numNotes)];
}

void shiftWrite(int desiredPin, boolean desiredState)
{
  //Serial.println(data);
  bitWrite(data, desiredPin, desiredState);
  //Serial.println(data);
```

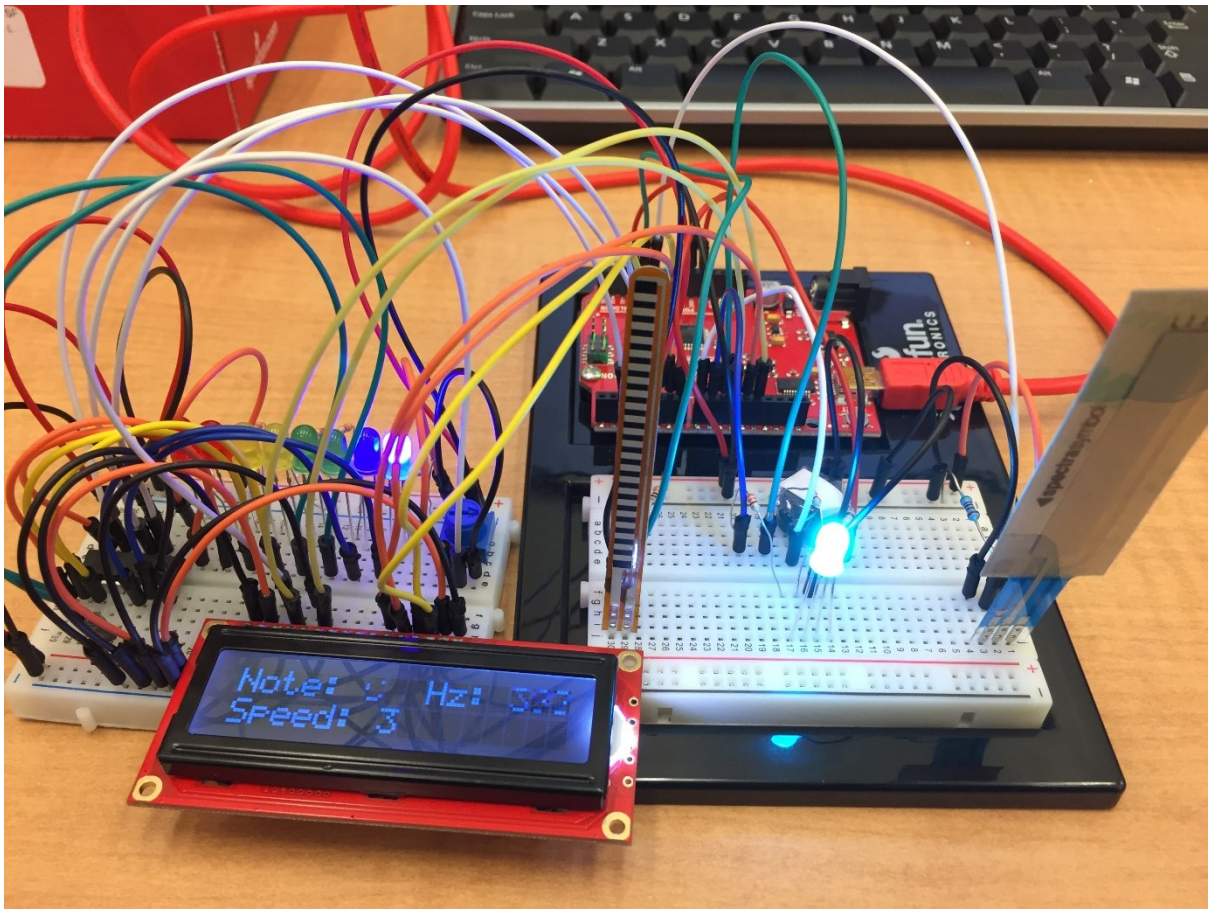
```
  shiftOut(dataPin, clockPin, MSBFIRST, data);

  digitalWrite(latchPin, HIGH);
  digitalWrite(latchPin, LOW);
}

void danceLeds(char note)
{
  Serial.println(note);

  for(int index = 0; index < numLeds; index++)
  {
    Serial.println(noteValue(note, 8));
    if(shouldBlink(numLeds - 1 - index, noteValue(note, 8)))
    {
      shiftWrite(numLeds - 1 - index, HIGH);
    }
    else
    {
      shiftWrite(numLeds - 1 - index, LOW);
    }
  }
}

boolean shouldBlink(int LED, int noteValue)
{
  return (LED >= noteValue);
}

int noteValue(char note, int numNotes)
{
  if (note == ' ')
  {
    return numNotes + 100;
  }

  char notesList[numNotes] = {
    'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};

  for(int i = 0; i < 8; i++)
  {
    if (notesList[i] == note)
    {
      return i;
    }
    else
    {
      continue;
    }
  }

  return 0;
```

```
}

void printOnLCD(int note,int frequency,int beats)
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Note: ");
  lcd.print(char(note));
  lcd.print("  Hz: ");
  lcd.print(frequency);
  lcd.setCursor(0, 1);
  lcd.print("Speed: ");

  int speedLevel;
  if(beats == 1)
  {
     speedLevel = 3;
  }
  else if (beats == 4)
  {
    speedLevel = 2;
  }
  else
  {
    speedLevel = 1;
  }
  lcd.print(speedLevel);
}
```

Discoteca 2.0 acts as follow:

1) When the soft potentiometer is left untouched, it will play notes at random, including stops.
2) The RGB colors will be random when the notes are random too.
3) When the user touches the soft potentiometer, he will be playing one note depending on where they are touching. Each note is related to one color.
4) The user can choose between three speeds (the beat) by manipulating the flex sensor.
5) The RGB LED will blink at the speed of the melody.
6) The LCD Screen will show information about the Speed of the melody, the current playing note and the frequency of the current playing note.
7) The shift register is used to manage the 8 LEDs with only 3 PINs.
8) The linear array of eight LEDs will do an animation that is a function of the frequency of the current playing note.





Figures 10 and 11. Photos of the circuit

Resources

https://www.sparkfun,com/sikcode

https://www.sparkfun,com/sikguide

http://www.falstad.com/circuit/

http://fritzing.org/home/

https://www.sparkfun.com/datasheets/Sensors/Flex/SoftPot-Datasheet.pdf

https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEX%20SENSOR%20DATA%20SHEET%202014.pdf

https://www.sparkfun.com/datasheets/Components/YSL-R596CR3G4B5C-C10.pdf

https://images-na.ssl-images-amazon.com/images/I/A1agxml6wXL.pdf