Discoteca

(modified from)

SIK Guide

ENGR102 Intro to Electronics

Shoreline Community College

October 2017

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

The circuit #11 presented on the SIK Guide titled “Piezo Buzzer” will only play a pre-recorded and fixed song made by the programmer; This circuit is entertaining, but it also lacks dynamics. Now, Discoteca will allow the user to make a melody at run-time with his own fingers by using both the flex sensor and the soft potentiometer to change the beats per second and select the note they would like to play, respectively.

What things can you do with this circuit?

You can make melodies at run-time by manipulating the flex sensor to select the speed of the melody from three speeds by changing the beating. The fastest is where beats is exactly one, the middle is where beats is exactly 4, and the slowest is where beats is exactly 8. You can also affect the note to play, there are 8 different notes that Discoteca supports: a, b, c, C, d, e, f, g. When both sensors are not touched it will play random notes with the fastest beating.

Knowledge from which prior experiments is required?

Knowledge from Circuit #3 to manipulate the RGB LED, #9 in order to operate the flex sensor, circuit #10 to operate the soft potentiometer and from circuit #11 to operate the piezo buzzer.

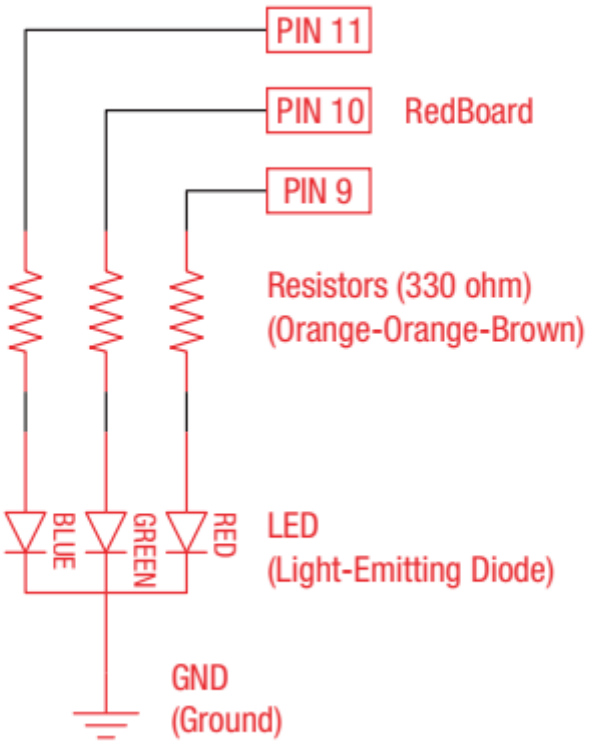

Figure 1. Schematic of circuit #3 (RGB LED).

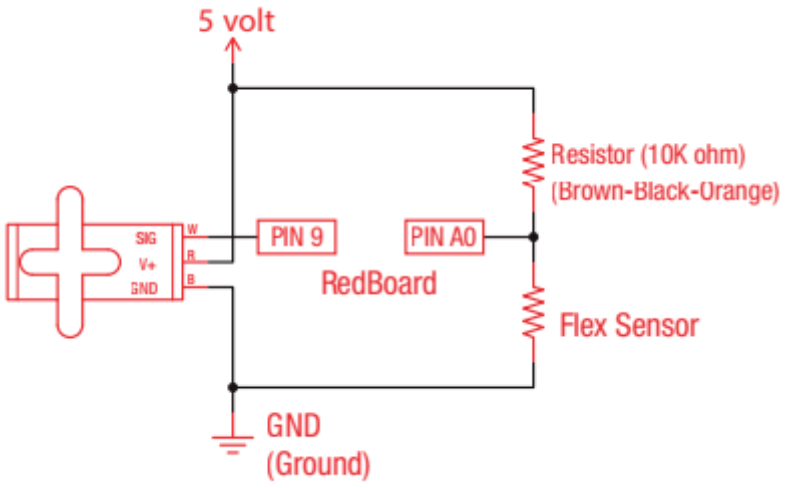

Figure 2. Schematic of circuit #9 (Flex Sensor).

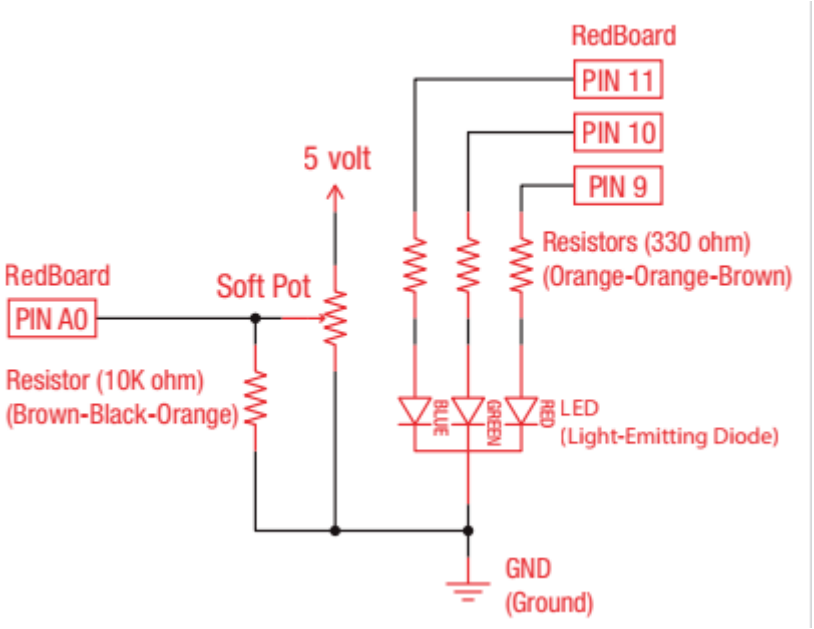

Figure 3. Schematic of circuit #10 (Soft Potentiometer).

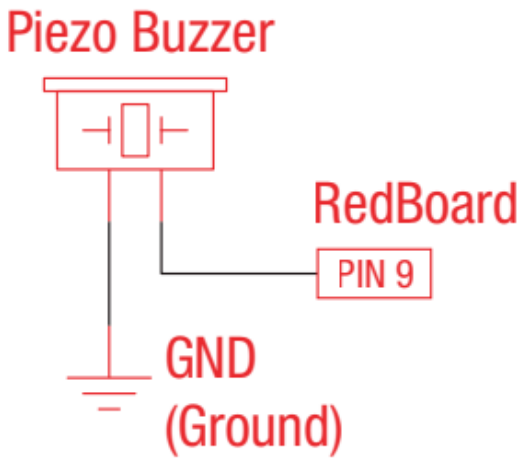

Figure 4. Schematic of circuit #11 (Piezo Buzzer).

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 Resistor.
- One (1) 5kΩ Resistor.
- One (1) Piezo Buzzer.
- Three (3) 330Ω Resistors.
- Jumpers/Wire as needed.

Problem-solving approach

I followed the following steps:

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.

Problems

- Lack of music knowledge: At first, I did not have a clue about how music is actually done at an electronics level. The most trouble-some part of the whole project was figuring out how to actually play decent melodies, what frequencies to take, what is a good “beating” and “tempo”, etc. The problem solved itself after a lot of testing and self-teaching. The code in Circuit #11 of the SIK Guide was very helpful too.
- Soft Potentiometer’s sensibility: In its first implementation I did mistakenly have a 330Ω Resistor (orange-orange-brown) mixed between the bag of 10kΩ (brown-black-orange) resistors. When using the sensor with the 330Ω resistor the sensibility breaks. See the figures below:

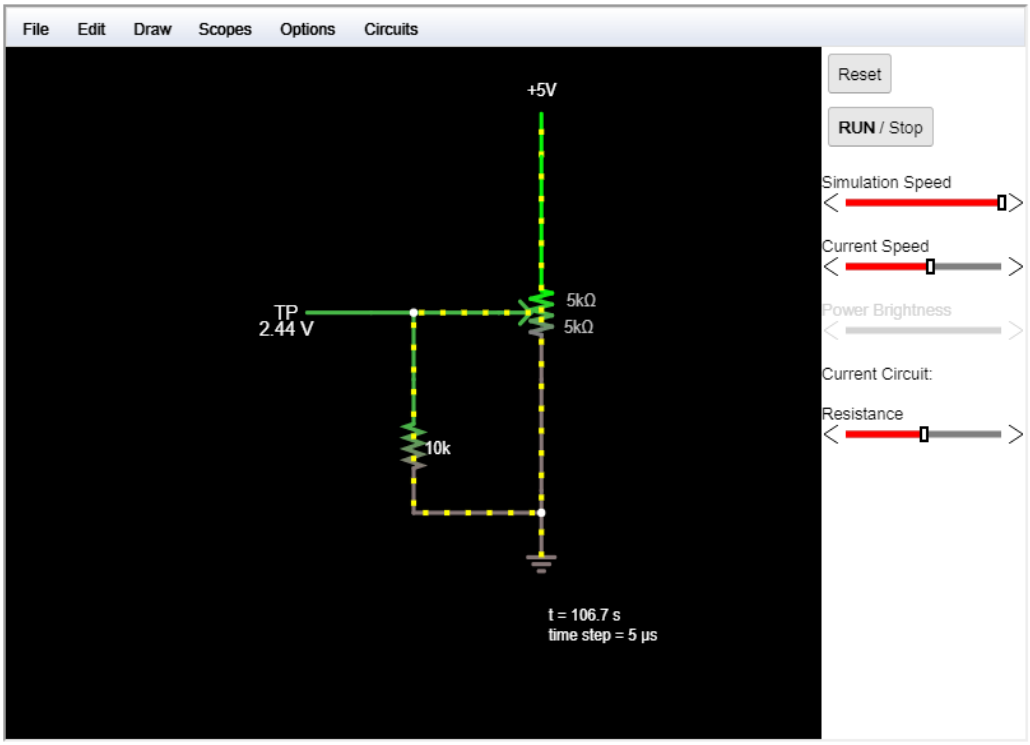

Figure 5. Falstad simulation of the soft potentiometer

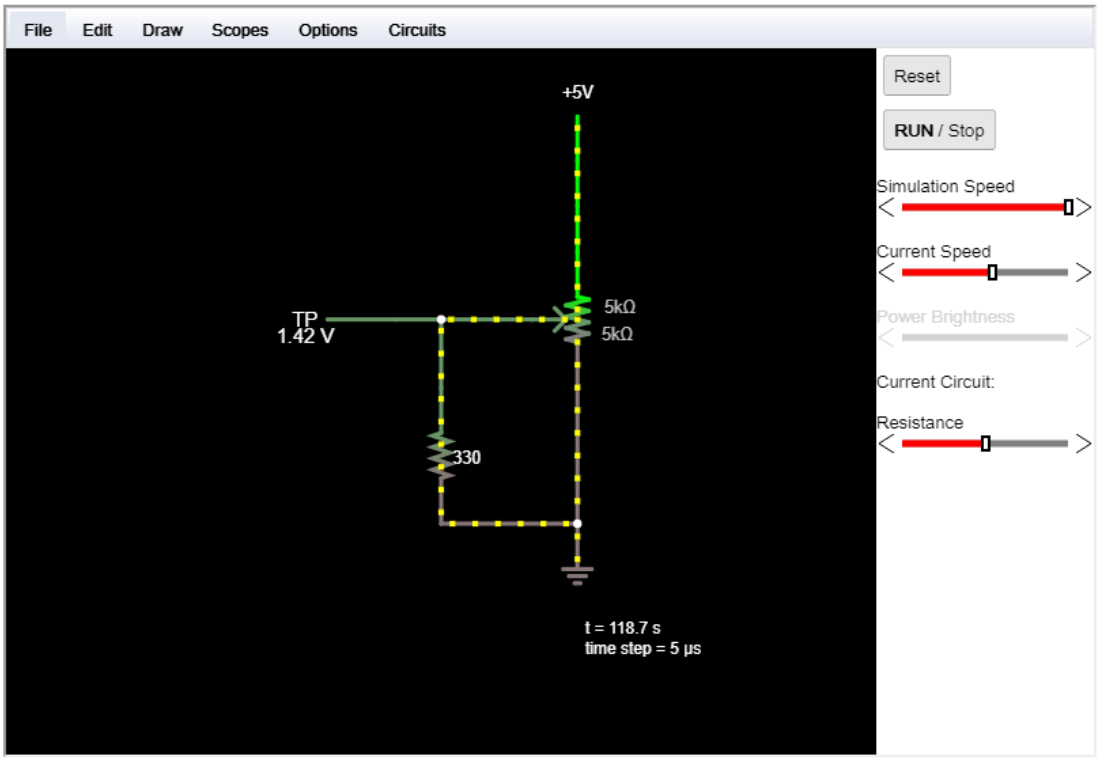

Figure 6. Another Simulation in Falstad of the soft potentiometer and a lower resistor

Then I used the intended 10kΩ resistor and sensibility became bearable, but I realized it could still be improved and so I spoke to Instructor Basham who suggested I use a 5kΩ resistor due to the application of the potentiometer in this specific circuit.

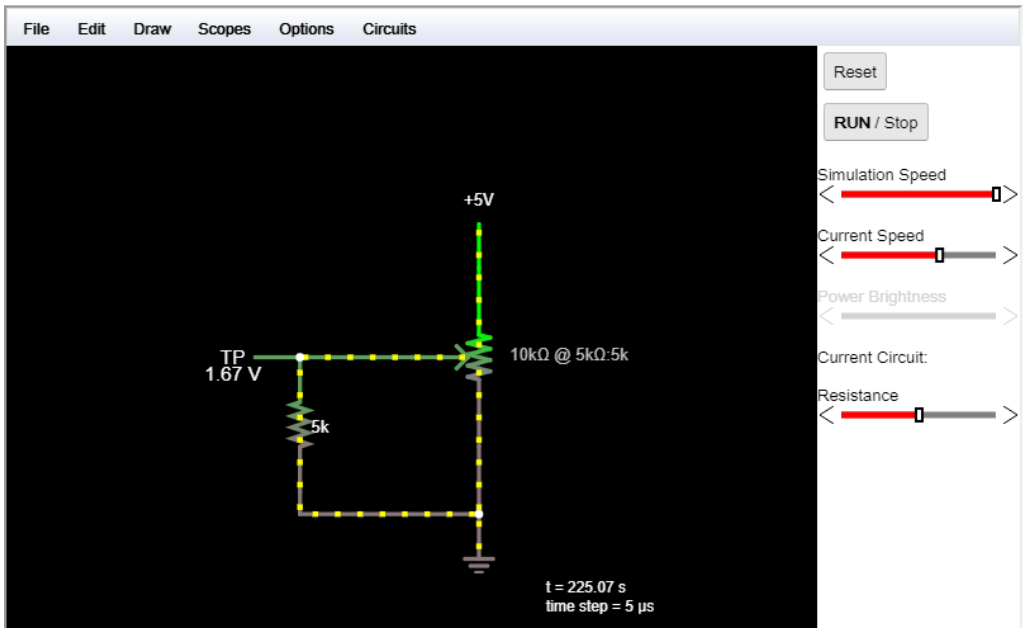

Figure 7. Simulation using the right resistor

Schematic.

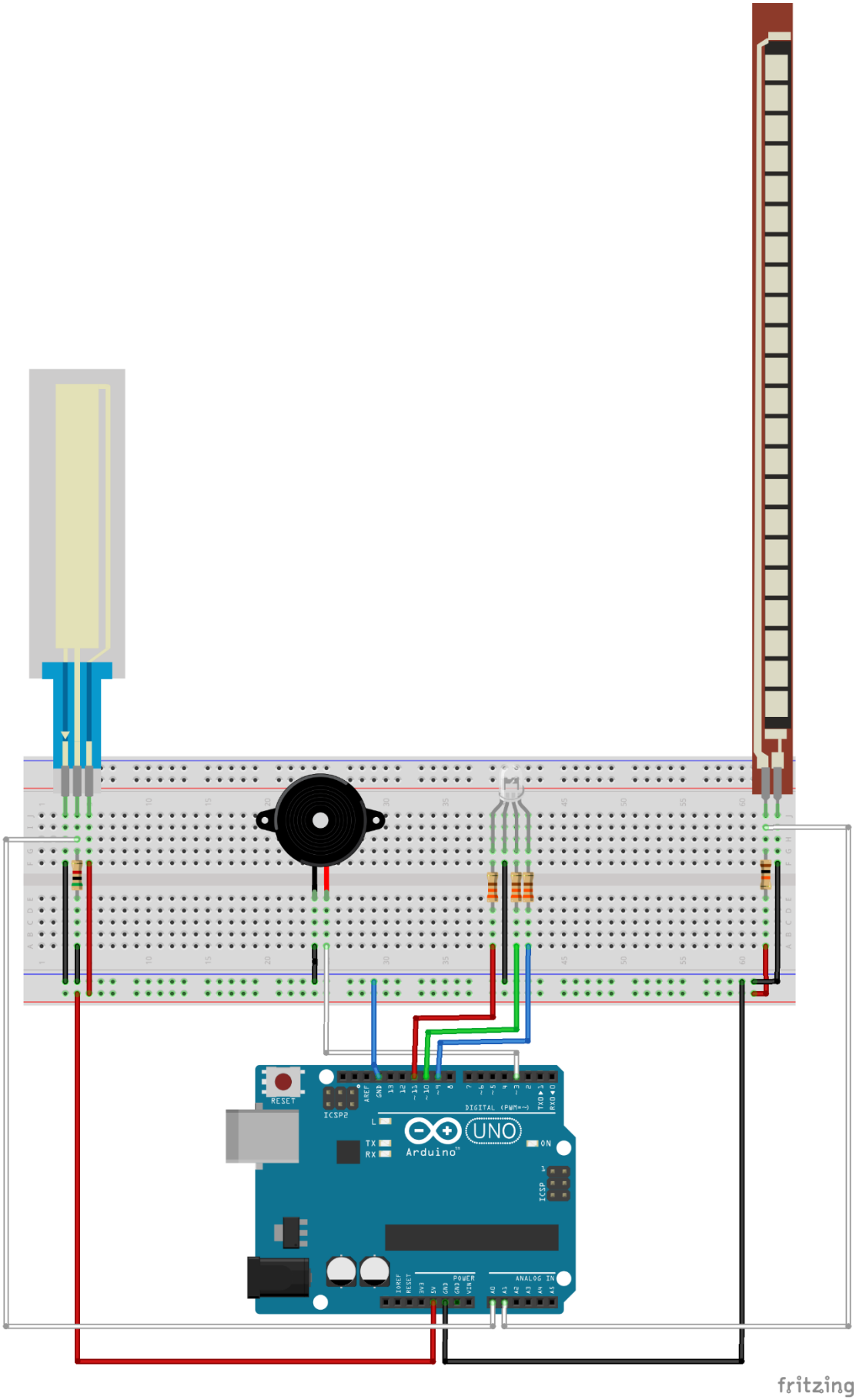

Figure 8. Breadboard View

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

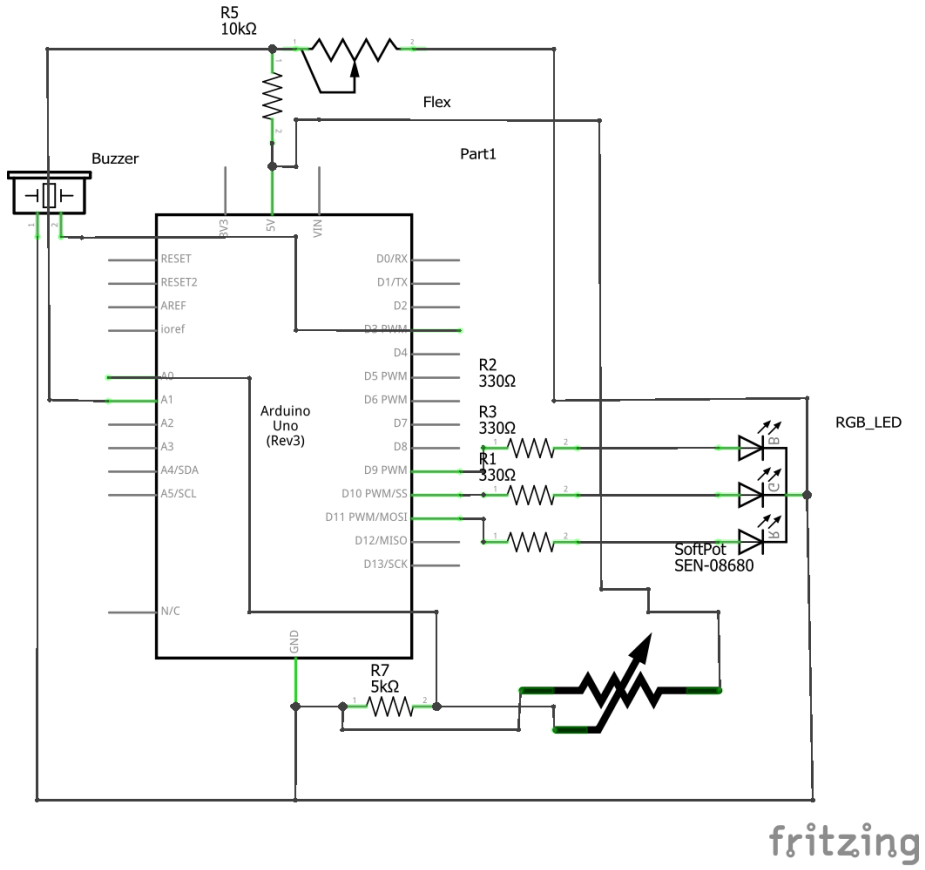

Figure 9. Schematic View

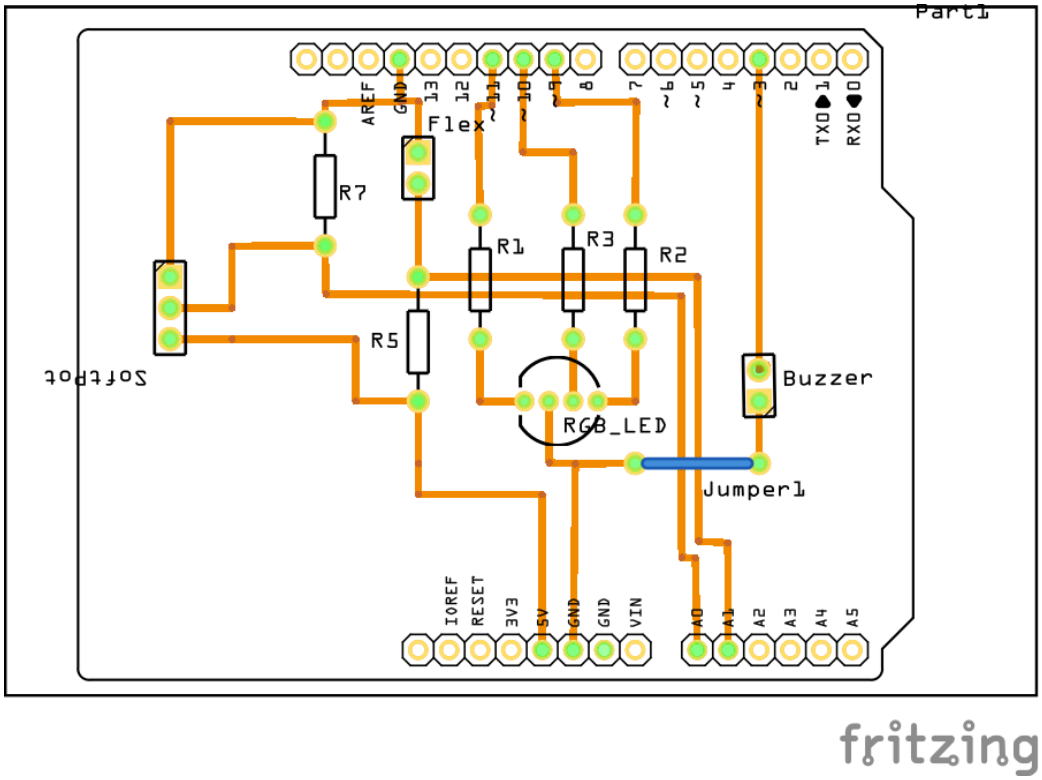

Figure 10. PCB View.

Code

```
/*
* Rafael Laya
* Professor Basham's ENGR102 class at Shoreline Community College
* The reader can use this code as they please but I won't be liable of anything
*/

// some constants for abstraction and readability
const int softPot = A0;
const int flexSensor = A1;
const int buzzerPin = 3;
const int blueLed = 9;
const int greenLed = 10;
const int redLed = 11;

// 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);
}

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;
}

// the notes are in function of the reading on the soft pot
// also the colors match the notes
switch (softPotVal)
{
case 0 ... 127:
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 128 ... 254:
note = 'b';
colors[0] = 0;
colors[1] = 0;
colors[2] = 1;
break;

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

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

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

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

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

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

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

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

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

// 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 blinkLights(int colors[], int vel)
{
digitalWrite(redLed, colors[0]);
digitalWrite(greenLed, colors[1]);
digitalWrite(blueLed, colors[2]);
delay(vel / 2);
digitalWrite(redLed, LOW);
digitalWrite(blueLed, LOW);
digitalWrite(greenLed, LOW);
delay(vel / 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)];
}
```

The code acts as follows:

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 LED will blink at the speed of the melody.

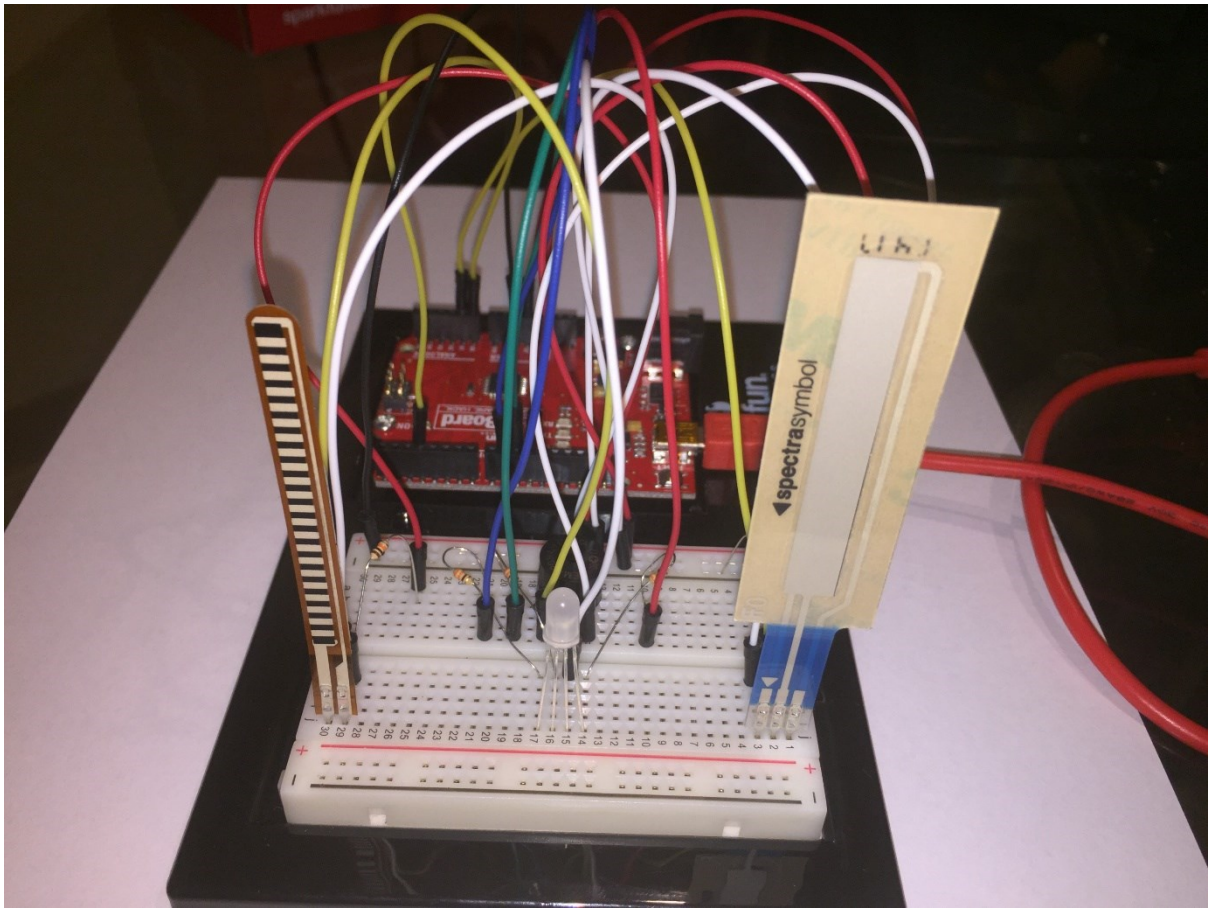

Figure 11. Photo of the circuit

Graphs

The quantities measured in this section were measured using an Etekcity MSR-R500 Digital Multimeter. The datasheets are found in the resources section. When reading the datasheets take into account this soft potentiometer is the 7182 (65.22mm of part length) and the flex sensor is the standard (10kΩ). Take also into account that in the pages of each of the sensors you can find many complains of inconsistencies between the datasheets and the products people bought.

When measuring the flex sensor I found the value is of 0.7kΩ without any bending, and 10kΩ at full bent of 180°. This graph shows the resistance between the two terminals of the flex sensor measured in kΩ as a function of the degree of bending of the flex sensor measured in sexagesimal degrees (the common measurement for degrees besides radians).

Measuring the resistance of the soft potentiometer was very similar. Instead, this time it is the resistance measured in kΩ as a function of the displacement of the finger from the lower part of the active part of the soft potentiometer measured in millimeters (mm). Note: The point (0,0) that lies in the graph means the resistance is measured to be zero when the device is not touched at all. The vertical line test is also intact (the line begins very closely to x = 0 as the sensor is doing its job properly).

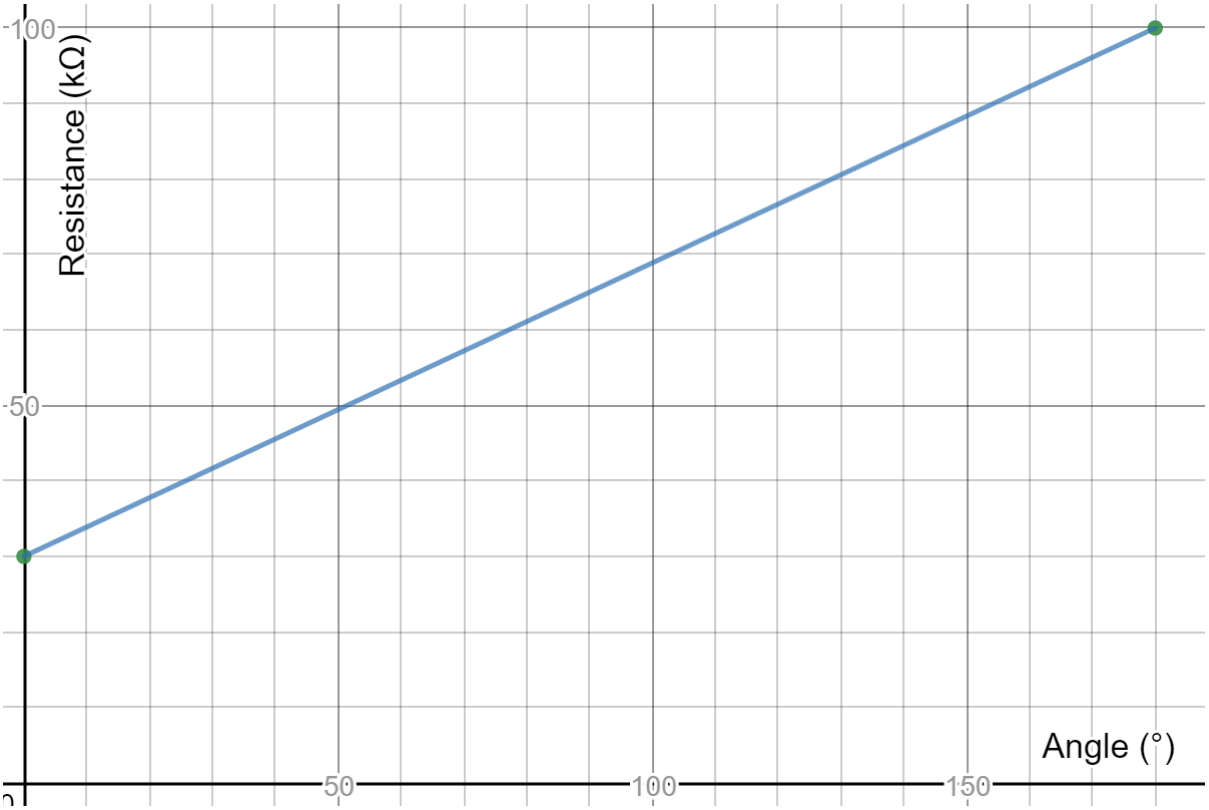

Figure 12. Resistance vs Angle – Flex Sensor

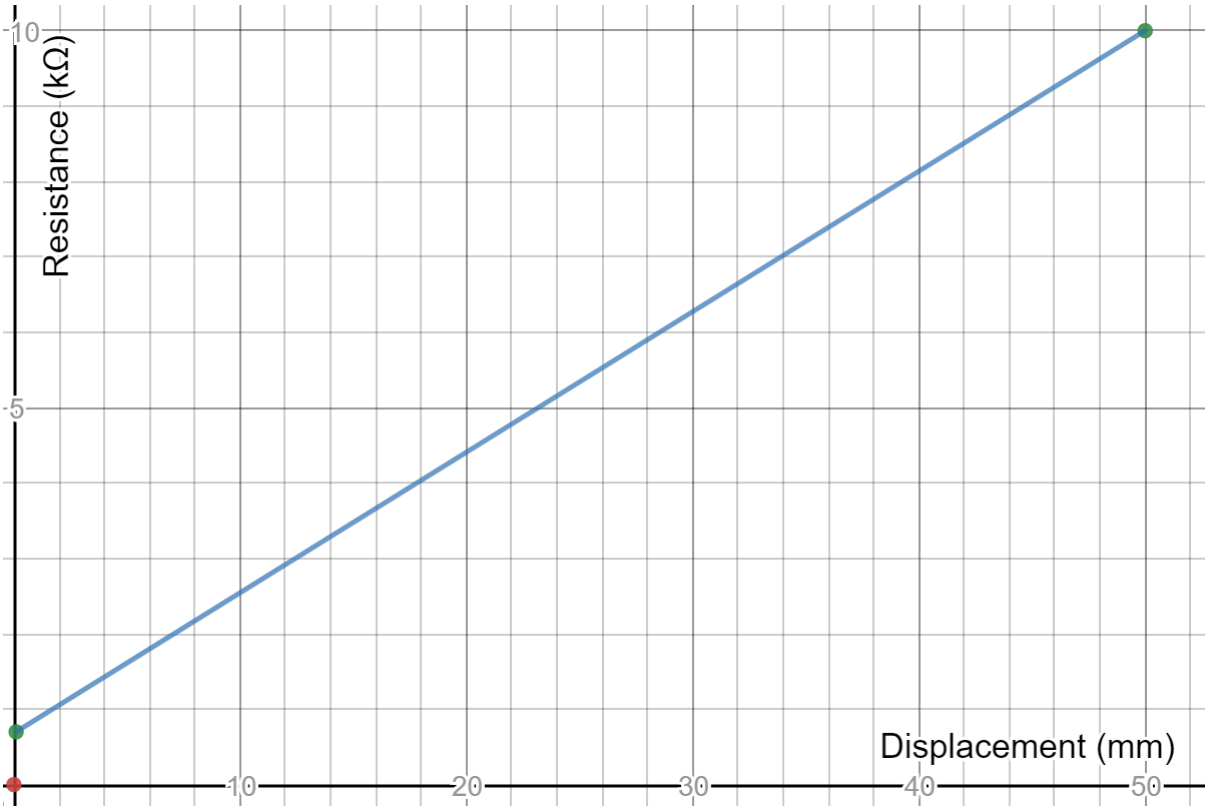

Figure 13. Resistance vs Displacement – Soft Potentiometer

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