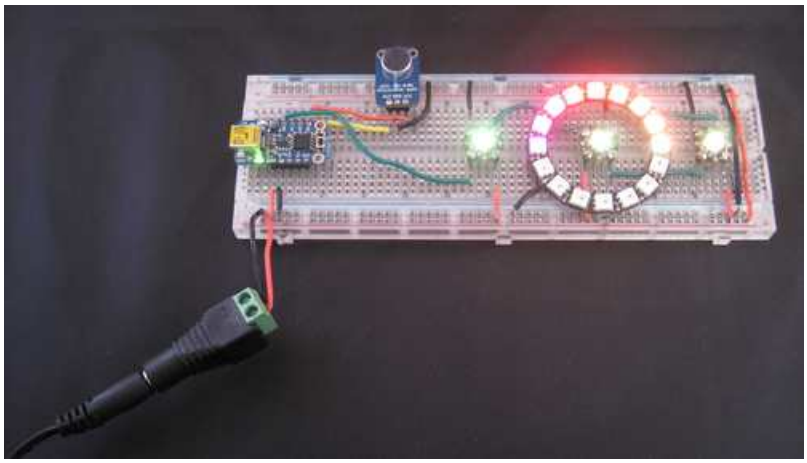
adafruit learning system

Trinket Sound-Reactive LED Color Organ

Created by Mike Barela



Last updated on 2018-08-22 03:37:44 PM UTC

Guide Contents

Overview

A color organ was a staple of the music scene in the 1970s and is still in use today at concerts and select home theaters.

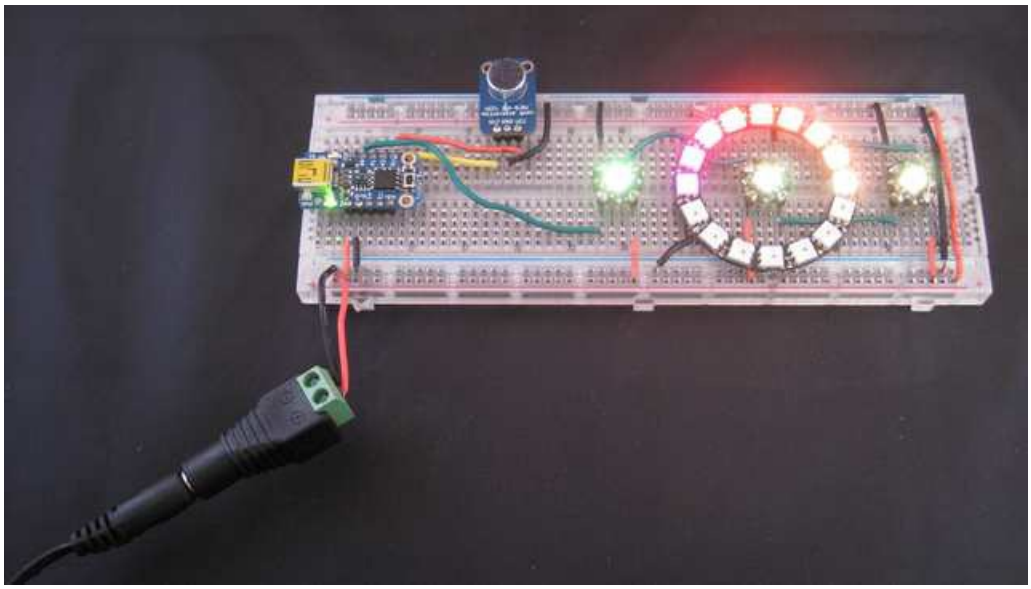
The principle is simple: flash colored lights in step with music or other sounds.

"In the day", these could be obtained as ready to use accessories or as kits. Radio Shack sold units that were rather popular, these from a 1977 catalog:

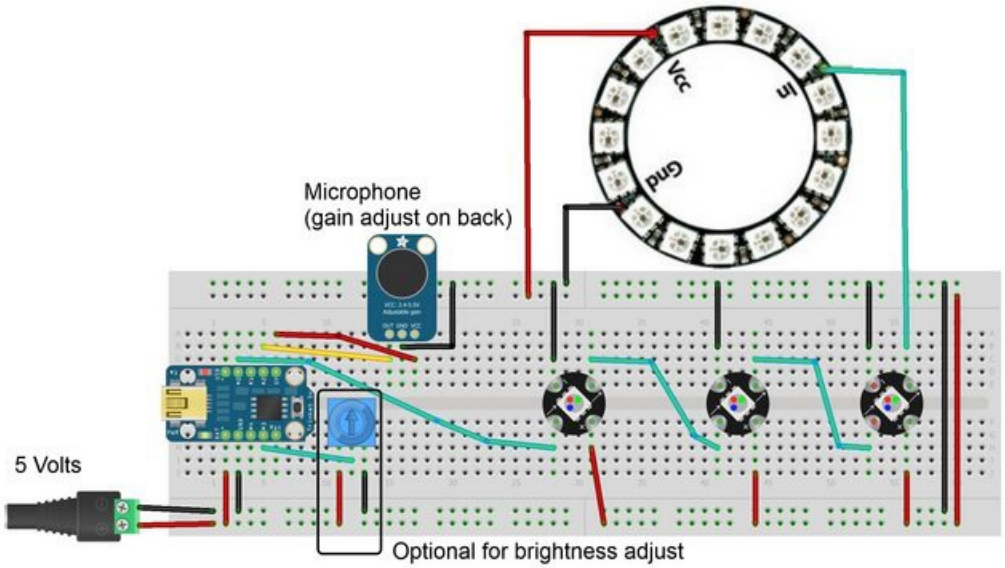
How It Works

Color organs sample sound and flash lights based either the sound intensity or frequency. The higher end units use analog or digital signal analysis to determine the sound energy in selective parts of the frequency spectrum and flash the lights accordingly.

The Adafruit Ampli-Tie project (https://adafru.it/c1u), using FLORA, has two different algorithms to light a string of Neopixel LEDs (https://adafru.it/cMZ) according to sound intensity. We will reuse much of the first Ampli-Tie algorithm's code. The more complex algorithm uses a good deal of floating point math, which is too large to fit on a Trinket or Gemma. The simpler algorithm fits with room to spare, using integer math. The code is slightly modified to give the effect one may want in a color organ. The user may alter this code to produce other effects for their own projects.



Build the Circuit



The best way to start is to breadboard first. You can then transfer the circuit to a small perma-proto board when you are satisfied with your circuit and want to consider a permanent mount.

You may solder the headers supplied with Trinket to facilitate breadboarding. A small 3 pin header was placed on the microphone breakout board for breadboard connection. For a more permanent circuit, you could use a servo extension cable to extend the microphone or wire your own three wires from the microphone breakout to the Trinket, power, and ground lines.

The Adafruit Neopixel UberGuide (https://adafru.it/cEz) shows the range of Neopixels available for your projects. The single Neopixels pictured have single header pins soldered on. The ring was placed over the middle single LED (pictured on the previous page). If you plan to build a permanent circuit, wiring to the pads would be a better choice.

A good 5 volt power supply is very important if powering more than three Neopixels. You can test the wiring with the USB power, just make sure its plugged directly into the computer USB port or a powered hub, not through the keyboard or monitor. If powering from a wall adapter, anything that can supply 5V and 1A or more should be fine. If you add additional Neopixels, more current may be needed to supply the power they need. Select a power supply (5 volt recommended) that can provide more than your anticipated maximum current. For large power-hungry strips, the Adafruit 5 volt, 4 amp or 10 amp supply could be required, although Trinket can only control approximately 100 pixels (to control more, consider using a larger Arduino with more memory).

Power for the microphone breakout is taken from the Trinket 5 volt regulated power pin. If you have a clean power supply, it can be wired to the 5 volt bus.

Trinket GPIO #2 is both an analog and digital pin. This circuit uses it as Analog Pin 1 to read the varying voltage from the microphone breakout.

Trinket GPIO #0 is used as the digital signal line out to the string of Neopixels.

If you would like to control the brightness of the pixels, you can add a potentiometer (nominal 10 kiloohms, larger than 1k to 1 megaohm should be fine). The center wiper is connected to GPIO #3 (Analog 3 in the Arduino IDE). The changing voltage is read from 0 to 255 which is exactly the range the Neopixel setBrightness function uses to set the pixel brightness. If not used, comment out the value POT_PIN and the code will use maximum brightness. If you have

trouble loading programs after adding the potentiometer, temporarily remove it to load the program, then replace.

This leaves GPIO #1 and #4 free for your own use. #1 is also controls the red Trinket LED so it takes some care to use in some applications. #3 and #4 are shared with the USB port, so if you use them, disconnect wiring when programming to avoid signal line interference.

The wiring should also work for Gemma. The brightness control cannot be added to Gemma as pins 3 and 4 are not broken out.

Preparing to Code

Using the Introducing Trinket (https://adafru.it/cN0) tutorial, ensure you have the Arduino integrated development environment (IDE) downloaded and installed for your operating system.

Remember you must press the hardware reset button on the Trinket then quickly press upload in the Arduino software to upload a sketch. If you get an error, try the reset-upload process again. If you continually cannot load the blink sketch, check to make sure the Trinket is connected and the Arduino IDE software has all the required changes.

Before loading the color organ program, you need to install the Adafruit Neopixel Library (https://adafru.it/aZU). Installation of the library is as follows:

1. Visit the Adafruit_NeoPixel library page (https://adafru.it/aZU) at Github.com.
2. Select the “Download ZIP” button, or simply click this link (https://adafru.it/cDj) to download directly.
3. Uncompress the ZIP file after it’s finished downloading.
4. The resulting folder should contain the files “Adafruit_NeoPixel.cpp”, “Adafruit_NeoPixel.h” and an “examples” sub-folder. Sometimes in Windows you’ll get an intermediate-level folder and need to move things around.
5. Rename the folder (containing the .cpp and .h files) to “Adafruit_NeoPixel” (with the underscore and everything), and place it alongside your other Arduino libraries, typically in your (home folder)/Documents/Arduino/Libraries folder. Libraries should not be installed alongside the Arduino application itself.
6. Re-start the Arduino IDE if it’s currently running.

Here’s a tutorial (https://adafru.it/aYM) that walks through the process of correctly installing Arduino libraries.

Now you are ready to copy the sketch on the next page to control the color organ.

Debugging Issues

For errors in the Arduino IDE software:

- Ensure you have installed Trinket support in the boards manager, as listed in the Introducing Trinket (https://adafru.it/cEu) tutorial.
- Ensure you have installed the Adafruit NeoPixel library (https://adafru.it/aZU).
- Ensure you push the Trinket on board reset button before uploading your sketch, the red LED will blink when ready for upload, there is a 10 second window to do this.
- If you place a large amount of code or other libraries in a sketch, it is very easy to exceed the available code space on the Trinket. If your program absolutely will not fit, consider switching to an Arduino Uno, Adafruit Boarduino (https://adafru.it/cKj), or Adafruit Flora (https://adafru.it/aSZ) with standard libraries.
- If you get errors similar to the one below, you may have included decimal numbers and the floating point library was added by the Arduino IDE, exceeding the amount of program space available.

arduino-1.0.1/hardware/tools/avr/bin/../lib/gcc/avr/4.3.2/../../../../avr/lib/**avr25**/crttn85.o:(.init9+0x2):**relocation truncated to fit: R_AVR_13_PCREL** against symbol `exit' defined in .fini9 section in /arduino-1.0.1/hardware/tools/avr/bin/../lib/gcc/avr/4.3.2/**avr25**\libgcc.a(_exit.o)

Code

The code below is the Ampli-Tie modification. It does not use a peak dot. With my microphone, I had noise issues, so the noise level is rather high. You can adjust values in the code to suit your project.

```
/* LED "Color Organ" for Adafruit Trinket and NeoPixel LEDs.

Hardware requirements:
 - Adafruit Trinket or Gemma mini microcontroller (ATTiny85).
 - Adafruit Electret Microphone Amplifier (ID: 1063)
 - Several Neopixels, you can mix and match
   o Adafruit Flora RGB Smart Pixels (ID: 1260)
   o Adafruit NeoPixel Digital LED strip (ID: 1138)
   o Adafruit Neopixel Ring (ID: 1463)

Software requirements:
 - Adafruit NeoPixel library

Connections:
 - 5 V to mic amp +
 - GND to mic amp -
 - Analog pinto microphone output (configurable below)
 - Digital pin to LED data input (configurable below)

Written by Adafruit Industries.  Distributed under the BSD license.
This paragraph must be included in any redistribution.
*/
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>

#define N_PIXELS  19  // Number of pixels you are using
#define MIC_PIN    1  // Microphone is attached to Trinket GPIO #2/Gemma D2 (A1)
#define LED_PIN    0  // NeoPixel LED strand is connected to GPIO #0 / D0
#define DC_OFFSET  0  // DC offset in mic signal - if unusure, leave 0
#define NOISE     100  // Noise/hum/interference in mic signal
#define SAMPLES   60  // Length of buffer for dynamic level adjustment
#define TOP       (N_PIXELS +1) // Allow dot to go slightly off scale
// Comment out the next line if you do not want brightness control or have a Gemma
#define POT_PIN    3  // if defined, a potentiometer is on GPIO #3 (A3, Trinket only)

byte
  peak      = 0,      // Used for falling dot
  dotCount  = 0,      // Frame counter for delaying dot-falling speed
  volCount  = 0;      // Frame counter for storing past volume data

int
  vol[SAMPLES],       // Collection of prior volume samples
  lvl       = 10,     // Current "dampened" audio level
  minLvlAvg = 0,      // For dynamic adjustment of graph low & high
  maxLvlAvg = 512;

Adafruit_NeoPixel  strip = Adafruit_NeoPixel(N_PIXELS, LED_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  // This is the auto-speed doubler line, keep it in, it will
  // automatically double the speed when 16Mhz is selected!
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);

  memset(vol, 0, sizeof(vol));
```

```
  memset(vol, 0, sizeof(vol));
  strip.begin();
}
void loop() {
  uint8_t  i;
  uint16_t minLvl, maxLvl;
  int      n, height;
  n   = analogRead(MIC_PIN);                  // Raw reading from mic
  n   = abs(n - 512 - DC_OFFSET);             // Center on zero
  n   = (n <= NOISE) ? 0 : (n - NOISE);       // Remove noise/hum
  lvl = ((lvl * 7) + n) >> 3;    // "Dampened" reading (else looks twitchy)

  // Calculate bar height based on dynamic min/max levels (fixed point):
  height = TOP * (lvl - minLvlAvg) / (long)(maxLvlAvg - minLvlAvg);

  if(height < 0L)       height = 0;      // Clip output
  else if(height > TOP) height = TOP;
  if(height > peak)     peak   = height; // Keep 'peak' dot at top

// if POT_PIN is defined, we have a potentiometer on GPIO #3 on a Trinket
//    (Gemma doesn't have this pin)
  uint8_t bright = 255;
#ifdef POT_PIN
   bright = analogRead(POT_PIN);  // Read pin (0-255) (adjust potentiometer
                                  //   to give 0 to Vcc volts
#endif
  strip.setBrightness(bright);    // Set LED brightness (if POT_PIN at top
                                  //  define commented out, will be full)
  // Color pixels based on rainbow gradient
  for(i=0; i<N_PIXELS; i++) {
    if(i >= height)
       strip.setPixelColor(i,   0,   0, 0);
    else
       strip.setPixelColor(i,Wheel(map(i,0,strip.numPixels()-1,30,150)));
    }

   strip.show(); // Update strip

  vol[volCount] = n;                      // Save sample for dynamic leveling
  if(++volCount >= SAMPLES) volCount = 0; // Advance/rollover sample counter

  // Get volume range of prior frames
  minLvl = maxLvl = vol[0];
  for(i=1; i<SAMPLES; i++) {
    if(vol[i] < minLvl)      minLvl = vol[i];
    else if(vol[i] > maxLvl) maxLvl = vol[i];
  }
  // minLvl and maxLvl indicate the volume range over prior frames, used
  // for vertically scaling the output graph (so it looks interesting
  // regardless of volume level).  If they're too close together though
  // (e.g. at very low volume levels) the graph becomes super coarse
  // and 'jumpy'...so keep some minimum distance between them (this
  // also lets the graph go to zero when no sound is playing):
  if((maxLvl - minLvl) < TOP) maxLvl = minLvl + TOP;
  minLvlAvg = (minLvlAvg * 63 + minLvl) >> 6; // Dampen min/max levels
  maxLvlAvg = (maxLvlAvg * 63 + maxLvl) >> 6; // (fake rolling average)
}

// Input a value 0 to 255 to get a color value.
// The colors are a transition r - g - b - back to r.
```

```
uint32_t Wheel(byte WheelPos) {
  if(WheelPos < 85) {
   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos < 170) {
   WheelPos -= 85;
   return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

Adjustments and Mounting

Adjustments

The main adjustment you will want to make is the gain on the microphone breakout. This is a tiny silver potentiometer on the back of the board. Use a small phillips screwdriver to make small adjustments while you make sounds or play music with both some loud and soft passages. This might take a bit of patience.

You may want to rearrange your pixels to produce colors in a pattern that you like.

You can change some of the constants at the beginning of the program to also adjust the behavior.

Finally, the brightness potentiometer is optional - in a cabinet you may want maximum brightness although in a dark room, it is beneficial to tone down the light a bit.

Here is the circuit without a cabinet, the Neopixels brightness was reduced to allow for photographing without oversaturating the camera:

Mounting

The typical color organ cabinet of the 1970s has a wood grain or black plastic box and a clear diffuser. Of course the wood grain was typically faux.

To create your own cabinet, you can chose nearly anything. A clear plastic case works well but the light will not diffuse through a clear lid, it will go straight through and you will not get that fuzzy light look.

You may select a cabinet size to suit your decor. Repurposing a box made of nearly any material is ideal. To give it that faux wood grain look, there are a number of contact papers available marketed as shelf liners that would do nicely. To make an inexpensive diffuser, a replacement fluorescent light fixture plastic cover is ideal and inexpensive.

Supplies may be found at home stores. In the US, Home Depot (https://adafru.it/cN3) has the following:

- OPTIX (https://adafru.it/cN5) 23.75 in. x 47.75 in. Prismatic Clear Acrylic Light Panel

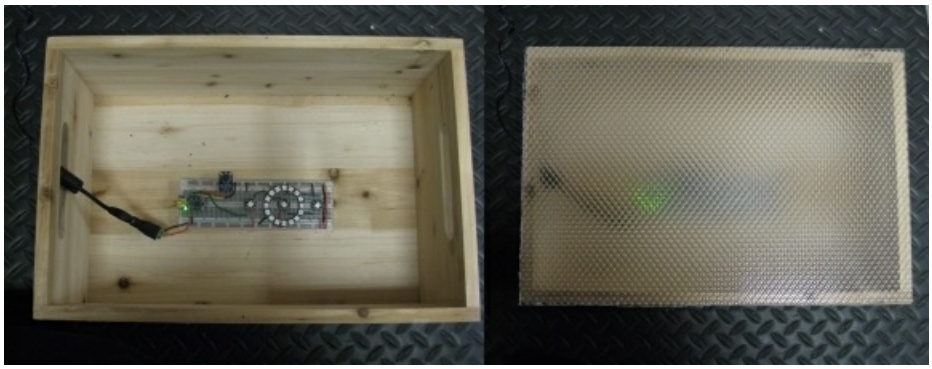
For the faux wood look, optionally you may chose:

- Con-Tact Creative Covering (https://adafru.it/cN4) 18 in. x 75 ft. Light Oak Multipurpose Shelf Liner, roll

Cover your box in the self adhesive liner and cut a section of diffuser to cover the open end of the box, allowing the LED lights to shine through.

If you want a more professional look, a well made wood cabinet is hard to beat.

Place your electronics in the back of the box, spacing your LEDs to suit your desired pattern. Test the project before securing the LEDs to the back of the box to ensure you like the light pattern when sound is made. If you find the sketch is not producing the ideal light pattern, you can change some of the parameters to get a reaction more suitable to your taste. Ensure you adjust the microphone gain to pick up the sound at the levels you want. When done, cover the front with the diffuser, and place in an area to add that special ambiance.



https://adafru.it/cNv

https://adafru.it/cNv