adafruit learning system

LED Ampli-Tie

Created by Becky Stern

Last updated on 2013-08-08 01:15:40 PM EDT

Guide Contents

Overview

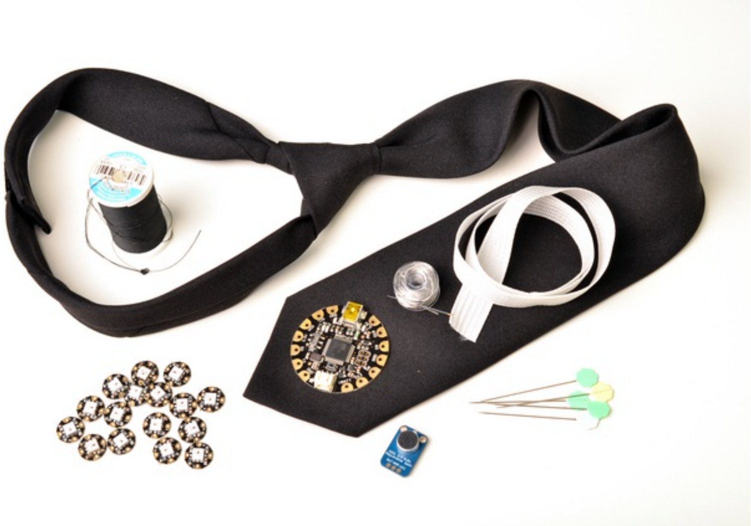
Make your necktie light up like a VU meter! This Flora project uses the Electret Microphone Amplifier to trigger 16 Flora NeoPixels sewn with conductive thread along the length of the tie.

Before you begin this project, we recommend reading the following guides:

- Getting Started with FLORA (http://adafru.it/aSZ)
- Adafruit Microphone Amplifier Breakout (http://adafru.it/aZS)
- Conductive Thread (http://adafru.it/aVx)

Photo by johngineer.

Tools & Supplies



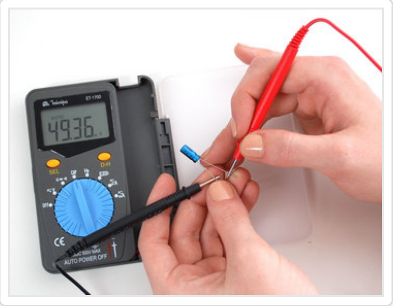
Bill of materials:

- 16 Flora NeoPixels (http://adafru.it/1260)
- Flora main board (http://adafru.it/659)
- Microphone amplifier breakout (http://adafru.it/1063)
- Lipoly battery (http://adafru.it/258)
- Scrap fabric for battery pouch
- 3-ply conductive thread (http://adafru.it/641)
- Standard cotton/poly thread
- Ribbon cable or conductive thread ribbon (http://adafru.it/1139)
- Break-away or otherwise clip-on tie (http://adafru.it/aZT)

You'll use a needle (http://adafru.it/615) and to stitch up the circuit.

Sharp scissors are a must! You'll also need a long ruler, some tailor's chalk, and a seam ripper.

Don't forget your wire strippers (http://adafru.it/527), pliers (http://adafru.it/146), and flush snips (http://adafru.it/152)!



You will need a good quality basic multimeter that can measure voltage and continuity.

Click here to buy a basic multimeter. (http://adafru.it/71)

Click here to buy a top of the line multimeter. (http://adafru.it/308)

Click here to buy a pocket multimeter. (http://adafru.it/850)

Don't forget to learn how to use your multimeter too! (http://adafru.it/aOy)

Any entry level 'all-in-one' soldering iron that you might find at your local hardware store should work. As with most things in life, you get what you pay for. Upgrading to a higher end soldering iron setup, like the Hakko FX-888 that we stock in our store (http://adafru.it/180), will make soldering fun and easy.

Do not use a "ColdHeat" soldering iron! They are not suitable for delicate electronics work and can damage the Flora (see here (http://adafru.it/aOo)).

Click here to buy our entry level adjustable 30W 110V soldering iron. (http://adafru.it/180)

Click here to upgrade to a Genuine Hakko FX-888 adjustable temperature soldering iron. (http://adafru.it/303)

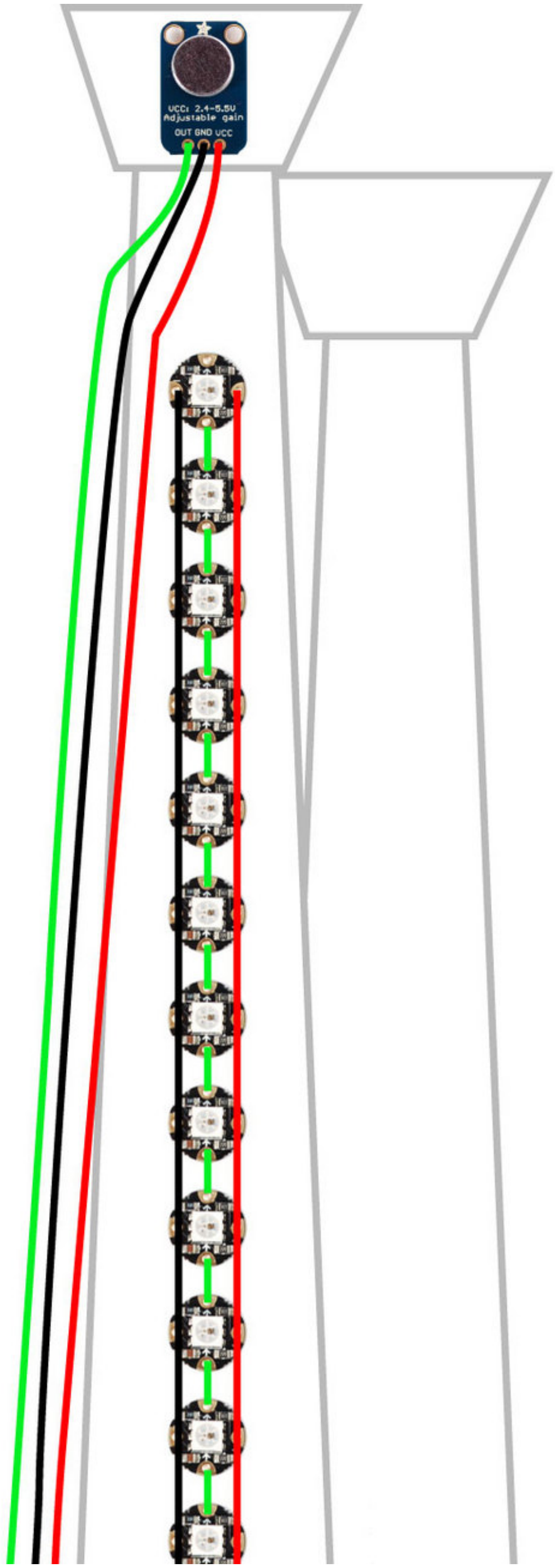
Learn how to solder with tons of tutorials! (http://adafru.it/aTk)

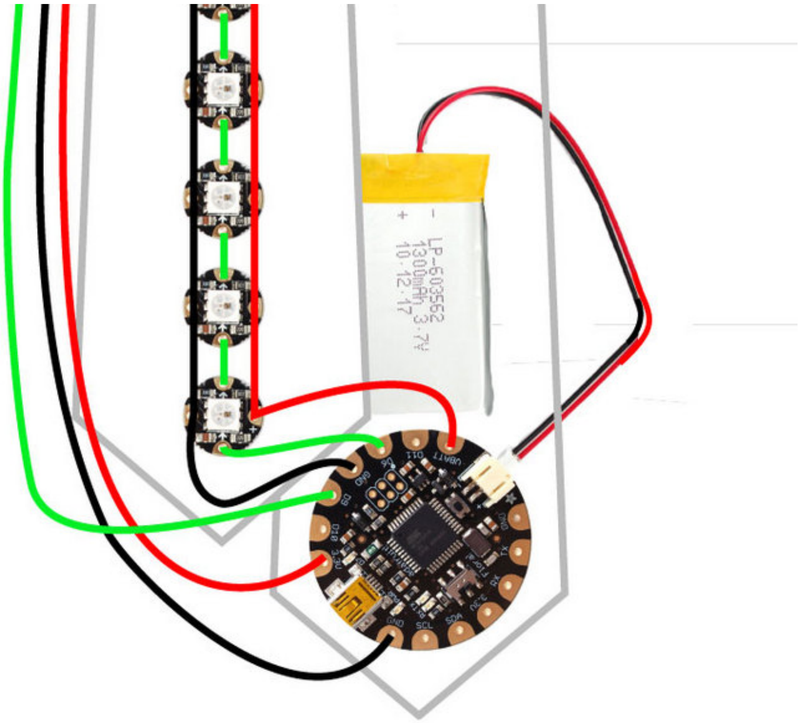
You will want rosin core, 60/40 solder. Good solder is a good thing. Bad solder leads to bridging and cold solder joints which can be tough to find.

Click here to buy a spool of leaded solder (recommended for beginners). (http://adafru.it/145)

Click here to buy a spool of lead-free solder. (http://adafru.it/734)

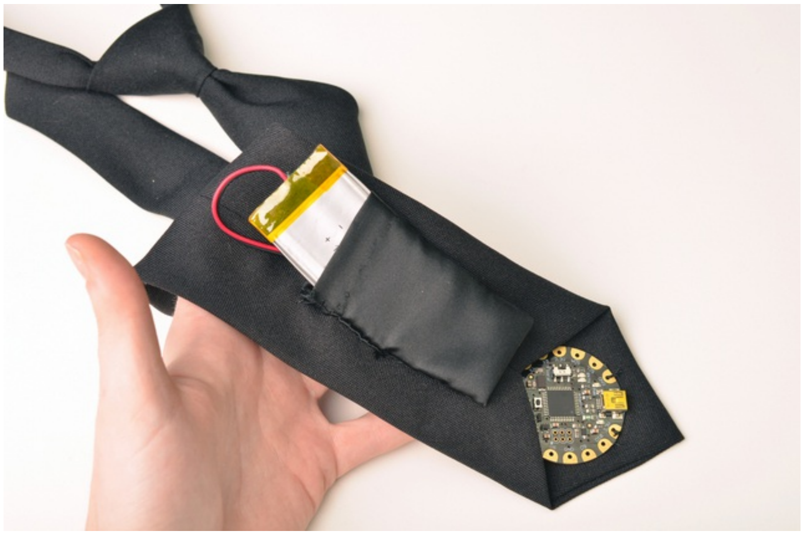
Circuit Diagram



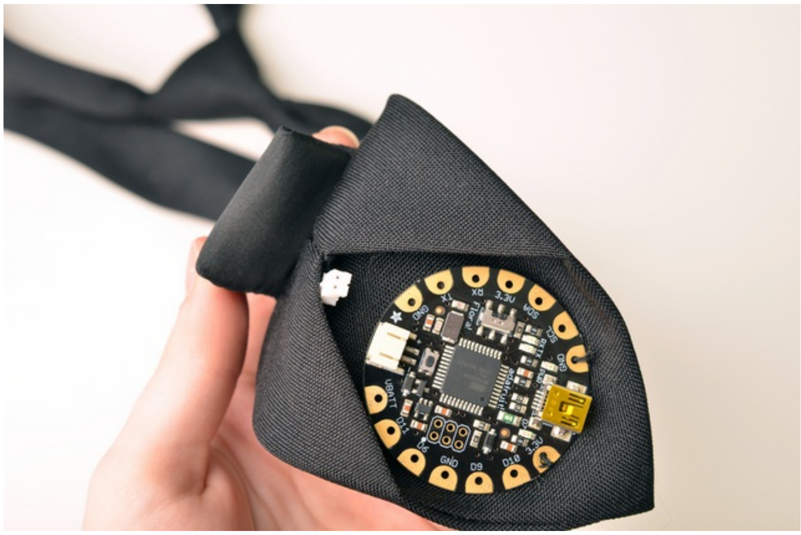


The Flora pixels are all connected to common ground bus, as well as a common power bus connected to VBATT. The Flora pixel data bus is connected to D6. The microphone amplifier is connected to 3.3V, GND, and D9.

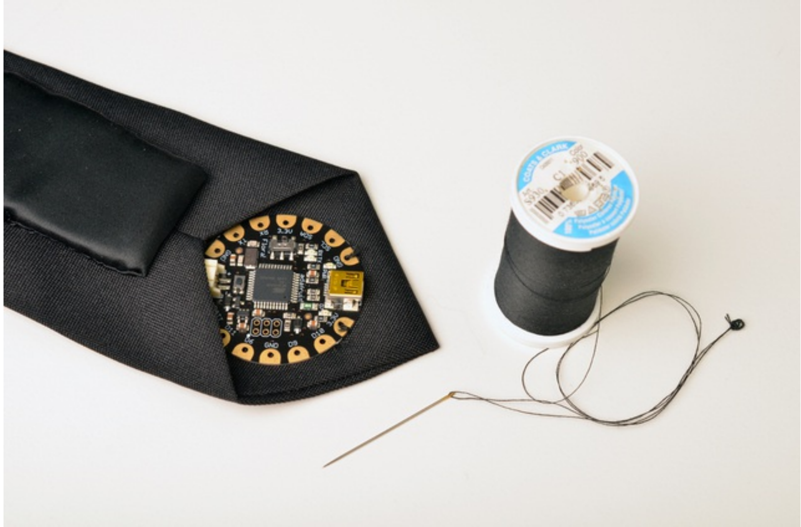
Battery Pouch & Flora



Use a piece of scrap fabric to stitch a small pouch for your lipoly battery. The pouch should be stitched to the back of the tie, just above where the Flora goes, and have an opening at the top for easy removal of the battery for charging. Use a seam ripper to open up the back seam of the tie just a little so you can thread the JST plug and wire inside the tie and down to the Flora.



The plug joins the Flora just inside the folds of the tie so it won't get caught on anything.

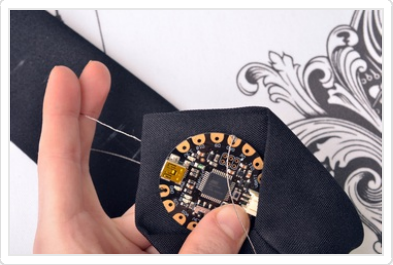


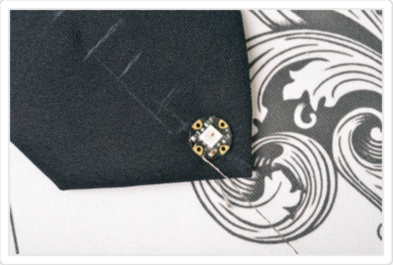
Tack your Flora in place with plain thread by stitching some unused pads to the tie. Try to just stitch to the back surface of the tie so the front fabric remains smooth.

Sewing Pixels

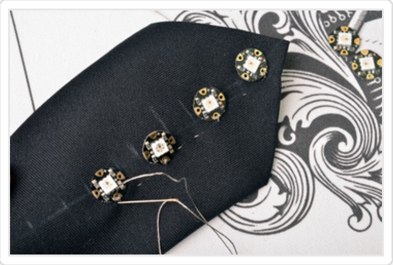
Use a ruler to draw a line down the center of your tie with tailor's chalk, and evenly distribute your 16 Flora pixels along this line.

Mark the position of each pixel with chalk.

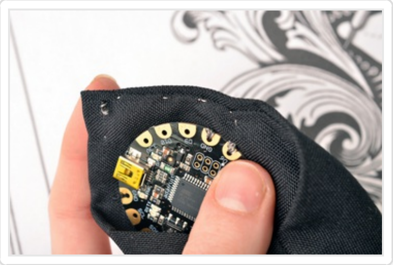


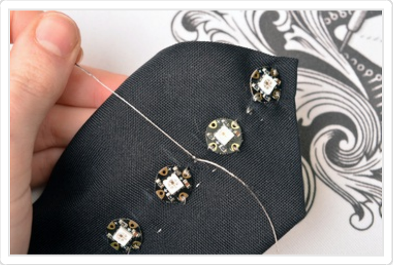


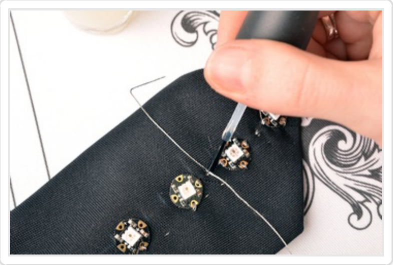
Stitch a long length of conductive thread to GND next to D6, only piercing the back surface of the tie. Stitch over to the (-) pad on your first pixel and secure (but don't cut the thread).



Add a few more pixels by connecting this long ground line to the (-) pads on the pixels.



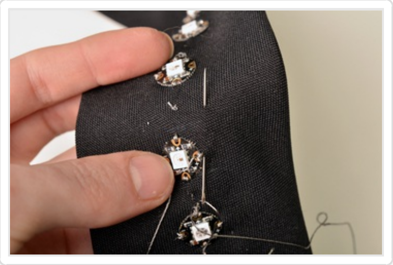


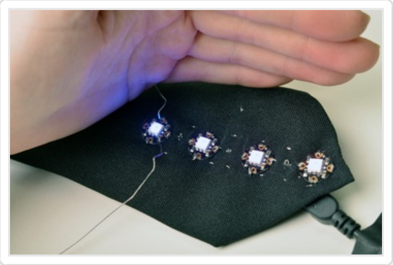


Stitch the data bus from D6 to the input pad on the first pixel (marked with an inward-facing arrow). Tie off, seal the knot, and snip the thread.

Stitch small segments of conductive thread between each pixel, connecting the output of one pixel to the input on the next.

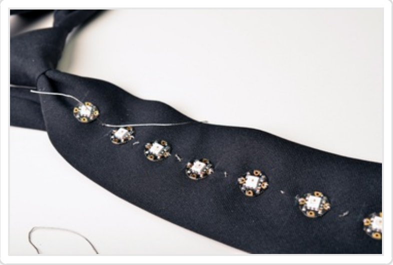
Check out our Conductive Thread guide (http://adafru.it/aVx) for more tips on working with conductive thread!





Use another long length of conductive thread to connect Flora's VBATT pad to the (+) pads on the pixels.

Once you've stitched a few pixels, test for shorts with your multimeter (make sure your long power and ground threads aren't touching), and fire up the NeoPixel library test code to ensure your fledgeling circuit is all good so far.

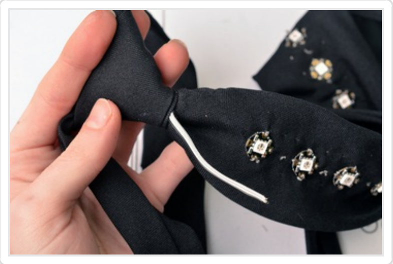


Stitch up the rest of the pixels - you'll have one long ground bus, one long power bus, and short segments between each input/output data pads.

Add Microphone

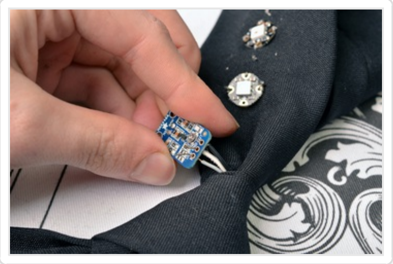
To match the tie, paint your microphone amplifier with a little black nail polish.

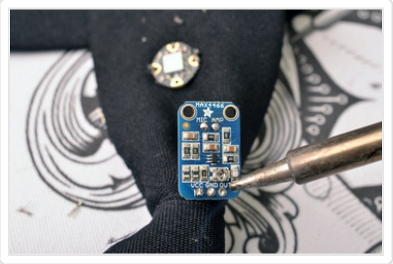
Cut a piece of ribbon cable longer than the main part of the tie.



Peel off three wires to use with the microphone amplifier, which will live at the knot of the tie.

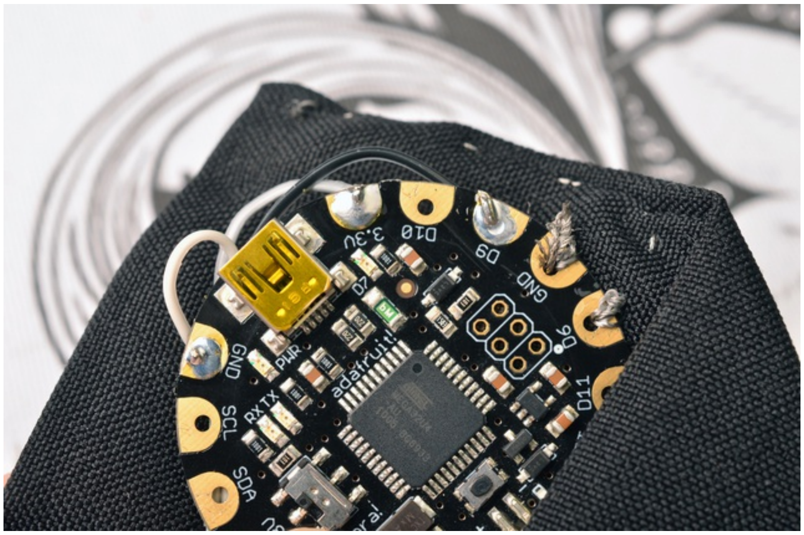
Insert the ribbon cable up through the inside of the tie. Cut a small hole with a seam ripper inside the knot of the tie and bring the ribbon cable through it.





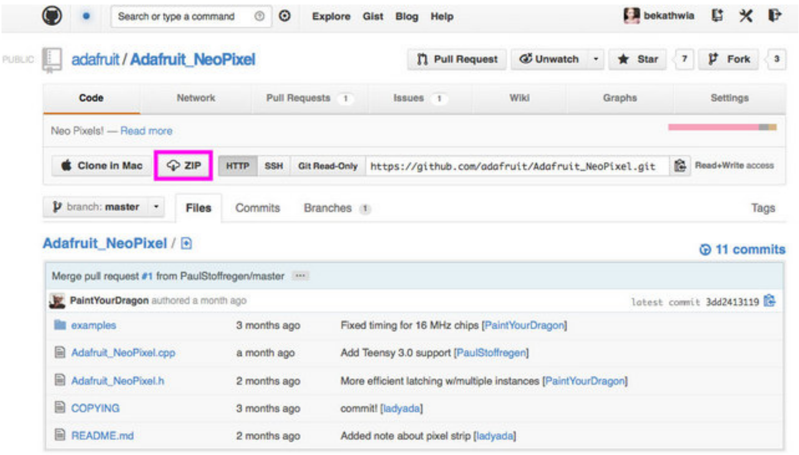
Strip the ends of the wires and solder them up to the three holes on the microphone amplifier.

Use plain thread to anchor the mic to the tie knot using the large mounting holes.



Back at the Flora end of the board, solder the corresponding wires to 3.3V, GND, and D9.

The Code



Merge pull request #1 from PaulStoffregen/master

PaintYourDragon authored a month ago — latest commit 3dd2413119

examples	3 months ago	Fixed timing for 16 MHz chips [PaintYourDragon]
Adafruit_NeoPixel.cpp	a month ago	Add Teensy 3.0 support [PaulStoffregen]
Adafruit_NeoPixel.h	2 months ago	More efficient latching w/multiple instances [PaintYourDragon]
COPYING	3 months ago	commit! [ladyada]
README.md	2 months ago	Added note about pixel strip [ladyada]

You'll need the NeoPixel library for this project. Download by clicking the ZIP button on the NeoPixel Github repository page (http://adafru.it/aZU), and rename the resulting folder "Adafruit_NeoPixel" and move to your Arduino libraries folder.

For more information on programming your Flora board including the software you need to do so, head over to the Getting Started with Flora (http://adafru.it/aSZ) guide.

We got so excited about this project we made TWO Arduino sketches to meter the volume in the room (by Phil Burgess, James DeVito, and Andy Doro). You can download them both at the LED Ampli-Tie Github repo (http://adafru.it/aZV) or copy from below. The first dynamically adjusts to whatever volume is happening:

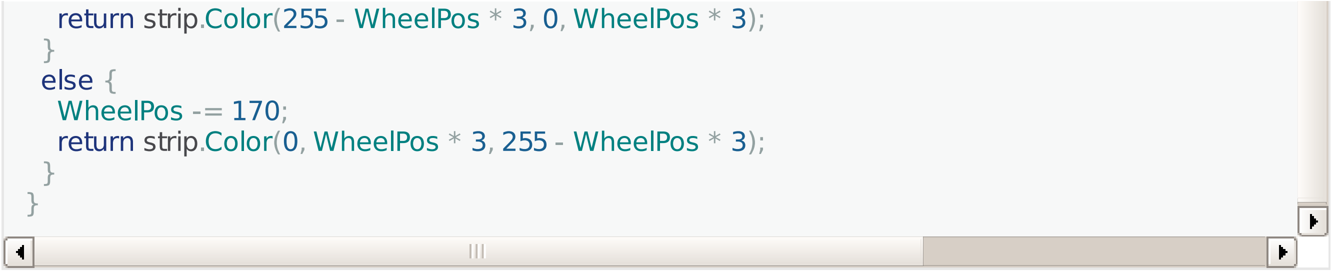
```
/*
LED VU meter for Arduino and Adafruit NeoPixel LEDs.

Hardware requirements:
 - Most Arduino or Arduino-compatible boards (ATmega 328P or better).
 - Adafruit Electret Microphone Amplifier (ID: 1063)
 - Adafruit Flora RGB Smart Pixels (ID: 1260)
   OR
 - Adafruit NeoPixel Digital LED strip (ID: 1138)
 - Optional: battery for portable use (else power through USB or adapter)
Software requirements:
 - Adafruit NeoPixel library

Connections:
 - 3.3V to mic amp +
 - GND to mic amp -
 - Analog pin to microphone output (configurable below)
 - Digital pin to LED data input (configurable below)
 See notes in setup() regarding 5V vs. 3.3V boards - there may be an
 extra connection to make and one line of code to enable or disable.
```

```
Written by Adafruit Industries.  Distributed under the BSD license.
This paragraph must be included in any redistribution.
*/

#include <Adafruit_NeoPixel.h>

#define N_PIXELS  16  // Number of pixels in strand
#define MIC_PIN   A9  // Microphone is attached to this analog pin
#define LED_PIN    6  // NeoPixel LED strand is connected to this pin
#define DC_OFFSET  0  // DC offset in mic signal - if unusure, leave 0
#define NOISE     10  // Noise/hum/interference in mic signal
#define SAMPLES   60  // Length of buffer for dynamic level adjustment
#define TOP       (N_PIXELS + 2) // Allow dot to go slightly off scale
#define PEAK_FALL 40  // Rate of peak falling dot

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 only needed on 5V Arduinos (Uno, Leonardo, etc.).
  // Connect 3.3V to mic AND TO AREF ON ARDUINO and enable this
  // line.  Audio samples are 'cleaner' at 3.3V.
  // COMMENT OUT THIS LINE FOR 3.3V ARDUINOS (FLORA, ETC.):
//  analogReference(EXTERNAL);

  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
  // 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)));

  }



  // Draw peak dot
  if(peak > 0 && peak <= N_PIXELS-1) strip.setPixelColor(peak,Wheel(map(peak,0,strip.numPixels()

   strip.show(); // Update strip

// Every few frames, make the peak pixel drop by 1:

    if(++dotCount >= PEAK_FALL) { //fall rate

      if(peak > 0) peak--;
      dotCount = 0;
    }



  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);
  }
}
```

The second allows you to adjust the sensitivity of the VU meter:

```
/*
LED VU meter for Arduino and Adafruit NeoPixel LEDs.

 Hardware requirements:
 - Most Arduino or Arduino-compatible boards (ATmega 328P or better).
 - Adafruit Electret Microphone Amplifier (ID: 1063)
 - Adafruit Flora RGB Smart Pixels (ID: 1260)
 OR
 - Adafruit NeoPixel Digital LED strip (ID: 1138)
 - Optional: battery for portable use (else power through USB or adapter)
 Software requirements:
 - Adafruit NeoPixel library

 Connections:
 - 3.3V to mic amp +
 - GND to mic amp -
 - Analog pin to microphone output (configurable below)
 - Digital pin to LED data input (configurable below)
 See notes in setup() regarding 5V vs. 3.3V boards - there may be an
 extra connection to make and one line of code to enable or disable.

 Written by Adafruit Industries.  Distributed under the BSD license.
 This paragraph must be included in any redistribution.

 fscale function:
 Floating Point Autoscale Function V0.1
 Written by Paul Badger 2007
 Modified from code by Greg Shakar

 */

#include <Adafruit_NeoPixel.h>
#include <math.h>

#define N_PIXELS  16  // Number of pixels in strand
#define MIC_PIN   A9  // Microphone is attached to this analog pin
#define LED_PIN    6  // NeoPixel LED strand is connected to this pin
#define SAMPLE_WINDOW   10  // Sample window for average level
#define PEAK_HANG 24 //Time of pause before peak dot falls
#define PEAK_FALL 4 //Rate of falling peak dot
#define INPUT_FLOOR 10 //Lower range of analogRead input
#define INPUT_CEILING 300 //Max range of analogRead input, the lower the value the more sensitiv



byte peak = 16;      // Peak level of column; used for falling dots
unsigned int sample;

byte dotCount = 0;  //Frame counter for peak dot
byte dotHangCount = 0; //Frame counter for holding peak dot
```

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

void setup()
{
  // This is only needed on 5V Arduinos (Uno, Leonardo, etc.).
  // Connect 3.3V to mic AND TO AREF ON ARDUINO and enable this
  // line.  Audio samples are 'cleaner' at 3.3V.
  // COMMENT OUT THIS LINE FOR 3.3V ARDUINOS (FLORA, ETC.):
  //  analogReference(EXTERNAL);

  // Serial.begin(9600);
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'

}

void loop()
{
  unsigned long startMillis= millis();  // Start of sample window
  float peakToPeak = 0;   // peak-to-peak level

  unsigned int signalMax = 0;
  unsigned int signalMin = 1023;
  unsigned int c, y;


  // collect data for length of sample window (in mS)
  while (millis() - startMillis < SAMPLE_WINDOW)
  {
    sample = analogRead(MIC_PIN);
    if (sample < 1024)  // toss out spurious readings
    {
      if (sample > signalMax)
      {
        signalMax = sample;  // save just the max levels
      }
      else if (sample < signalMin)
      {
        signalMin = sample;  // save just the min levels
      }
    }
  }
  peakToPeak = signalMax - signalMin;  // max - min = peak-peak amplitude

  // Serial.println(peakToPeak);


  //Fill the strip with rainbow gradient
  for (int i=0;i<=strip.numPixels()-1;i++){
    strip.setPixelColor(i,Wheel(map(i,0,strip.numPixels()-1,30,150)));
  }


  //Scale the input logarithmically instead of linearly
  c = fscale(INPUT_FLOOR, INPUT_CEILING, strip.numPixels(), 0, peakToPeak, 2);
  if(c < peak) {
    peak = c;        // Keep dot on top
    dotHangCount = 0;    // make the dot hang before falling
  }
  if (c <= strip.numPixels()) { // Fill partial column with off pixels
    drawLine(strip.numPixels(), strip.numPixels()-c, strip.Color(0, 0, 0));
  }

  // Set the peak dot to match the rainbow gradient
  y = strip.numPixels() - peak;

  strip.setPixelColor(y-1,Wheel(map(y,0,strip.numPixels()-1,30,150)));

  strip.show();

  // Frame based peak dot animation
  if(dotHangCount > PEAK_HANG) { //Peak pause length
    if(++dotCount >= PEAK_FALL) { //Fall rate
      peak++;
      dotCount = 0;
    }
  }
  else {
    dotHangCount++;
  }
}

//Used to draw a line between two points of a given color
void drawLine(uint8_t from, uint8_t to, uint32_t c) {
  uint8_t fromTemp;
  if (from > to) {
    fromTemp = from;
    from = to;
    to = fromTemp;
  }
  for(int i=from; i<=to; i++){
    strip.setPixelColor(i, c);
  }
}


float fscale( float originalMin, float originalMax, float newBegin, float
newEnd, float inputValue, float curve){

  float OriginalRange = 0;
  float NewRange = 0;
  float zeroRefCurVal = 0;
  float normalizedCurVal = 0;
  float rangedValue = 0;
  boolean invFlag = 0;


  // condition curve parameter
  // limit range

  if (curve > 10) curve = 10;
  if (curve < -10) curve = -10;
  curve = (curve * -.1) ; // - invert and scale - this seems more intuitive - postive numbers give mo
  curve = pow(10, curve); // convert linear scale into lograthimic exponent for other pow function

  /*
   Serial.println(curve * 100, DEC);   // multply by 100 to preserve resolution
   Serial.println();
   */

  // Check for out of range inputValues
  if (inputValue < originalMin) {
    inputValue = originalMin;
  }
  if (inputValue > originalMax) {
    inputValue = originalMax;
  }

  // Zero Refference the values
  OriginalRange = originalMax - originalMin;

  if (newEnd > newBegin){
    NewRange = newEnd - newBegin;
  }
  else
  {
    NewRange = newBegin - newEnd;
    invFlag = 1;
  }

  zeroRefCurVal = inputValue - originalMin;
  normalizedCurVal  =  zeroRefCurVal / OriginalRange;   // normalize to 0 - 1 float

  // Check for originalMin > originalMax  - the math for all other cases i.e. negative numbers seems
  if (originalMin > originalMax ) {
    return 0;
  }

  if (invFlag == 0){
    rangedValue =  (pow(normalizedCurVal, curve) * NewRange) + newBegin;

  }
  else     // invert the ranges
  {
    rangedValue =  newBegin - (pow(normalizedCurVal, curve) * NewRange);
  }

  return rangedValue;
}


// Input a value 0 to 255 to get a color value.
// The colours 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);
  }
}
```

Wear it!

Take your tie out on the town! It's perfect for parties, concerts, weddings, Bar Mitzvahs...

If you need to wash your tie, remove the battery and gently spot clean-- the pixels, thread and Flora board can handle getting wet (then dry throrougly), but water should not get in the microphone.