LED Visualizer Project

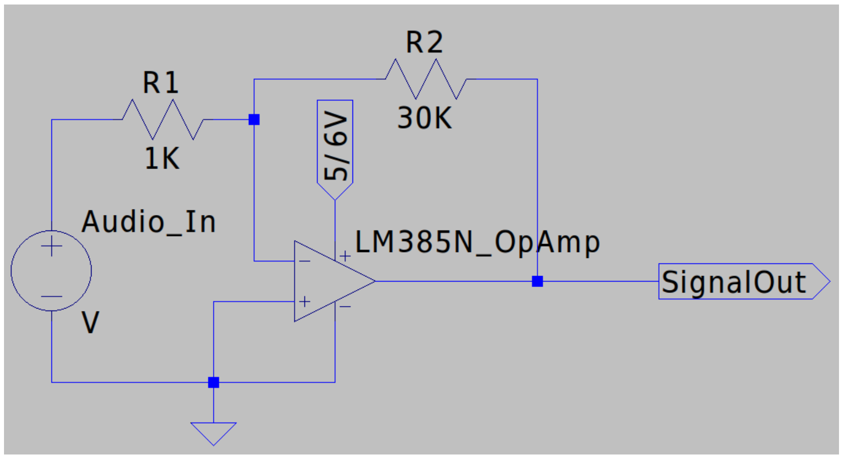
By Samuel Pack, Colby Hatton, Jordi Kloosterboer, and Wade Belanger

GitHub Project: https://github.com/Vetaz/LED_Light_Show

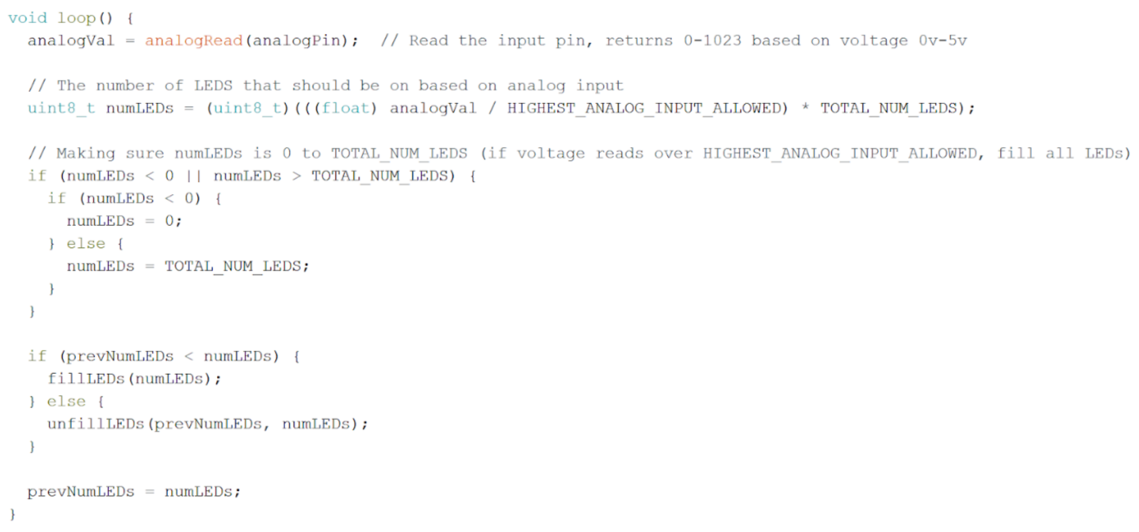
Nothing makes a great song really shine like a visualizer. This tool adds excitement to music by adding an additional medium to enjoy while you listen. Our project is an LED Visualizer combining our favorite hobbies with our pursuits in education.

Our project consists of 2 parts: the hardware and the software. Our hardware consists of an Arduino Uno, WS2812B LED Strip, 30 kΩ and 1 kΩ Resistor, LM385N Op-Amp, Audio Splitter, and an External Speaker.

The circuit takes the output from a 3.5mm audio jack and amplifies it with a custom-built inverting Op-Amp circuit. The resistor ratio yields a gain magnitude of 30, which removes negative voltage and increases the signal enough to be read. SignalOut goes to the Analog to Digital converter on the Arduino Uno.



Loop and fillLEDs code:


```
void loop() {
  analogVal = analogRead(analogPin);  // Read the input pin, returns 0-1023 based on voltage 0v-5v

  // The number of LEDS that should be on based on analog input
  uint8_t numLEDs = (uint8_t)(((float) analogVal / HIGHEST_ANALOG_INPUT_ALLOWED) * TOTAL_NUM_LEDS);

  // Making sure numLEDs is 0 to TOTAL_NUM_LEDS (if voltage reads over HIGHEST_ANALOG_INPUT_ALLOWED, fill all LEDs)
  if (numLEDs < 0 || numLEDs > TOTAL_NUM_LEDS) {
    if (numLEDs < 0) {
      numLEDs = 0;
    } else {
      numLEDs = TOTAL_NUM_LEDS;
    }
  }

  if (prevNumLEDs < numLEDs) {
    fillLEDs(numLEDs);
  } else {
    unfillLEDs(prevNumLEDs, numLEDs);
  }

  prevNumLEDs = numLEDs;
}
```

Loop gets the signal (a number from 0 to 1023) from the ADC of the Arduino Uno and sets it equal to analogVal. A number of 0 corresponds to 0 V and a number of 1023 corresponds to 5 V. The higher the voltage, the higher the volume of the music at that time.

The number of LEDs (numLEDs) corresponds to the analogVal with 0 corresponding to 0 number of LEDs to turn on and HIGHEST_ANALOG_INPUT_ALLOWED corresponding to all the LEDS to turn on. The HIGHEST_ANALOG_INPUT_ALLOWED is a number from 0 to 1023 to cap the highest analog input. It is used to increase or decrease how many LEDs are turned on at a specific voltage. We made HIGHEST_ANALOG_INPUT_ALLOWED equal to 746 because that was the highest number we got from the analogPin consistently. Therefore, anything above 746 or equal to it would turn on all the LEDs at that time.

After the numLEDs is calculated, if it is greater than it was before (as set in the previous loop or at the beginning as 0), the remaining LEDs are turned on, otherwise, the difference of LEDs are turned off. Then the prevNumLEDs equals what numLEDs is at the end of this loop and the next loop starts.

```
/* Fill LEDs up to the numLEDs */
void fillLEDs(uint8_t numLEDs)
{
  uint8_t hue;
  uint8_t brightness;

  for(int i = 0; i < numLEDs; i++) {
    hue = (uint8_t) ((float)i / TOTAL_NUM_LEDS * 250); // LED colors go from Red (0) to Pink (250)
    brightness = (uint8_t) ((float)i / numLEDs * 55 + 200); // Brightness from 200 to 255.
    leds[i] = CHSV(hue, 255, brightness);
    if (numLEDs < 48 && i % 6 == 0){ // Show filled LEDs every 6 LEDS
        FastLED.show();
    }
    else if (numLEDs < 96 && i % 8 == 0){ // Show filled LEDs every 8 LEDS
        FastLED.show();
    }
    else if (i % 12 == 0){ // Show filled LEDs every 12 LEDS
        FastLED.show();
    }
  }
}
```

In fillLEDs, 6, 8, or 12 LEDs (depending on where the loop is on turning on LEDs in the LED strip) are turned on at a time with their specific hue and brightness. The hue is dependent on where the LED is in the LED strip. Red LEDs are at the beginning of the strip and pink LEDs are at the end of the strip. unfillLEDs is almost the same as fillLEDS but with the LEDs turning off.