

User Manual: Monophonic Sound Analysis and Visualization System

By: Lucas Kang

Introduction:

Welcome to the Monophonic Sound Analysis and Visualization System user manual. This system is designed to analyze monophonic sound input in real-time using Max/MSP, employing the Fast Fourier Transform (FFT) algorithm through the Sigmund object to convert frequency data into MIDI note data. This MIDI data is further processed, scaled, and translated into RGB hue data, which is visualized through color representation. Finally, the RGB data is sent via the serial port to the Arduino IDE and an Arduino device, allowing you to control a Neopixel LED strip for captivating visual effects synchronized with the input sound.

System Requirements

- Max/MSP software
- Arduino IDE
- Arduino Nano
- Breadboard, 100 uF Capacitor, 300 Ohm resistor
- Neopixel LED strip
- Computer with available USB port
- Audio input device (microphone, audio interface, etc.)

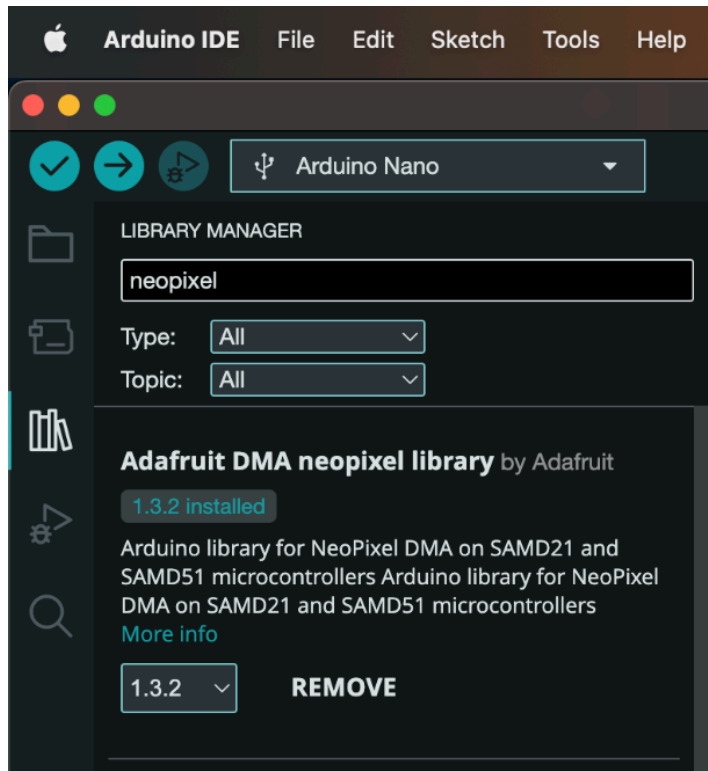
Setting Up the Sigmund Object (MAC OS):

Before using the Monophonic Sound Analysis and Visualization System on MAC OS, you need to download and install the Sigmund object as a third-party plugin. Follow these steps:

1. Visit <http://vud.org/max/> to download the Sigmund object.
2. At the top of the page, you'll find the 64-bit releases of the plugin. The older 32-bit versions are no longer compatible.
3. Download the Sigmund object from the provided GitHub link:
<https://github.com/v7b1/>
4. Once downloaded, extract the files. You should find a .mxo file and a .maxhelp file.
5. Move both the .mxo and .maxhelp files to the following directory: Documents > Max 8 > library.
6. Restart Max/MSP to use the newly installed Sigmund object.

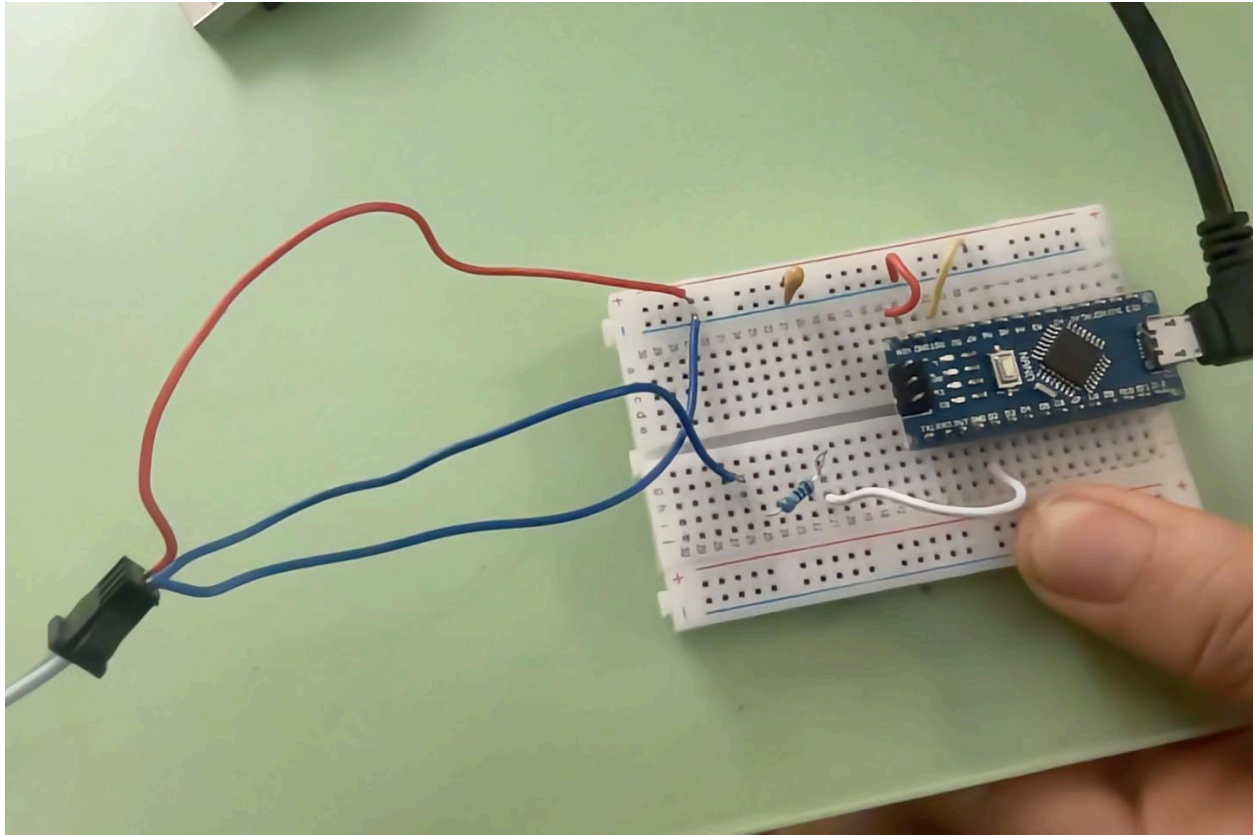
Downloading the Adafruit DMA Neopixel Library (Arduino):

To enable communication between the Max/MSP application and the Neopixel LED strip connected to your Arduino device, you need to download and install the Adafruit DMA Neopixel library. Follow these steps:



1. Open the Arduino IDE on your computer.
2. Navigate to "Sketch" > "Include Library" > "Manage Libraries..."
3. In the Library Manager window, type "Adafruit DMA Neopixel" in the search bar.
4. Locate the Adafruit DMA Neopixel library in the search results.
5. Click on the library, then click the "Install" button to download and install it.
6. Once the installation is complete, close the Library Manager window.

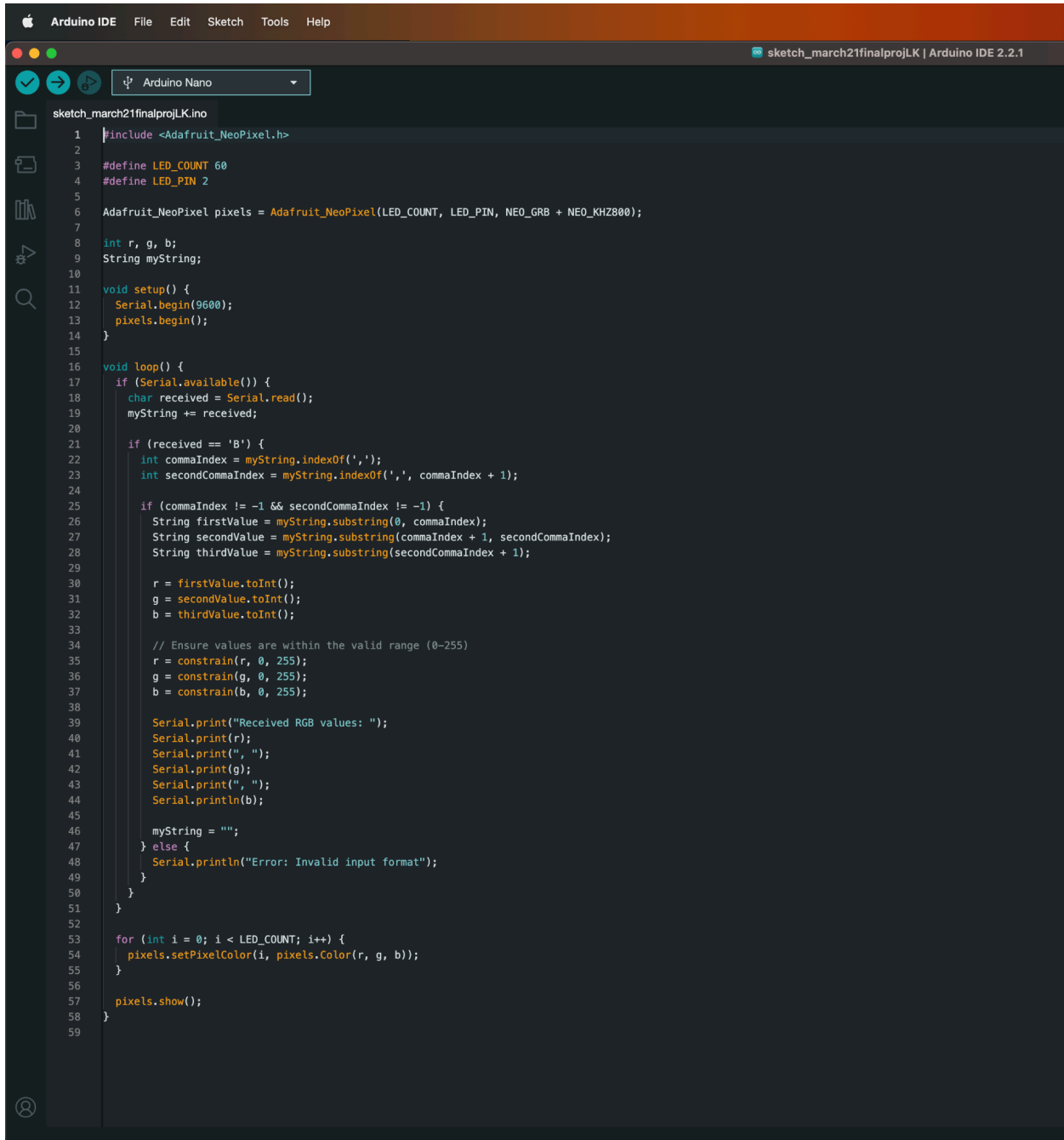
Arduino Nano Setup Instructions:



1. Connect the Arduino Nano or other Arduino device to your computer via a USB
2. Connect the Arduino to a breadboard, and wire the positive rail of the breadboard to the 5V pin, and the negative rail to the GND(ground) pin to establish a current.
3. Then place a 100 uF capacitor between the positive and negative rails to account for any sudden drops in current
4. Then connect the the D2 pin which is specified later in code to control the data stream, to a 300 Ohm resistor, which will account for sudden increases in current
5. Connect the negative and positive rails of the breadboard to the negative positive terminals of the LED strip on the outer edges, and in the medal, connect the 300 Ohm resistor to the middle data port of the LED strip.

Using the Arduino System:

1. Now that the Arduino is wired correctly and you have the Adafruit Neopixel library, open up the Arduino IDE, and import the code in the project folder titled: sketch_march21finalprojLK

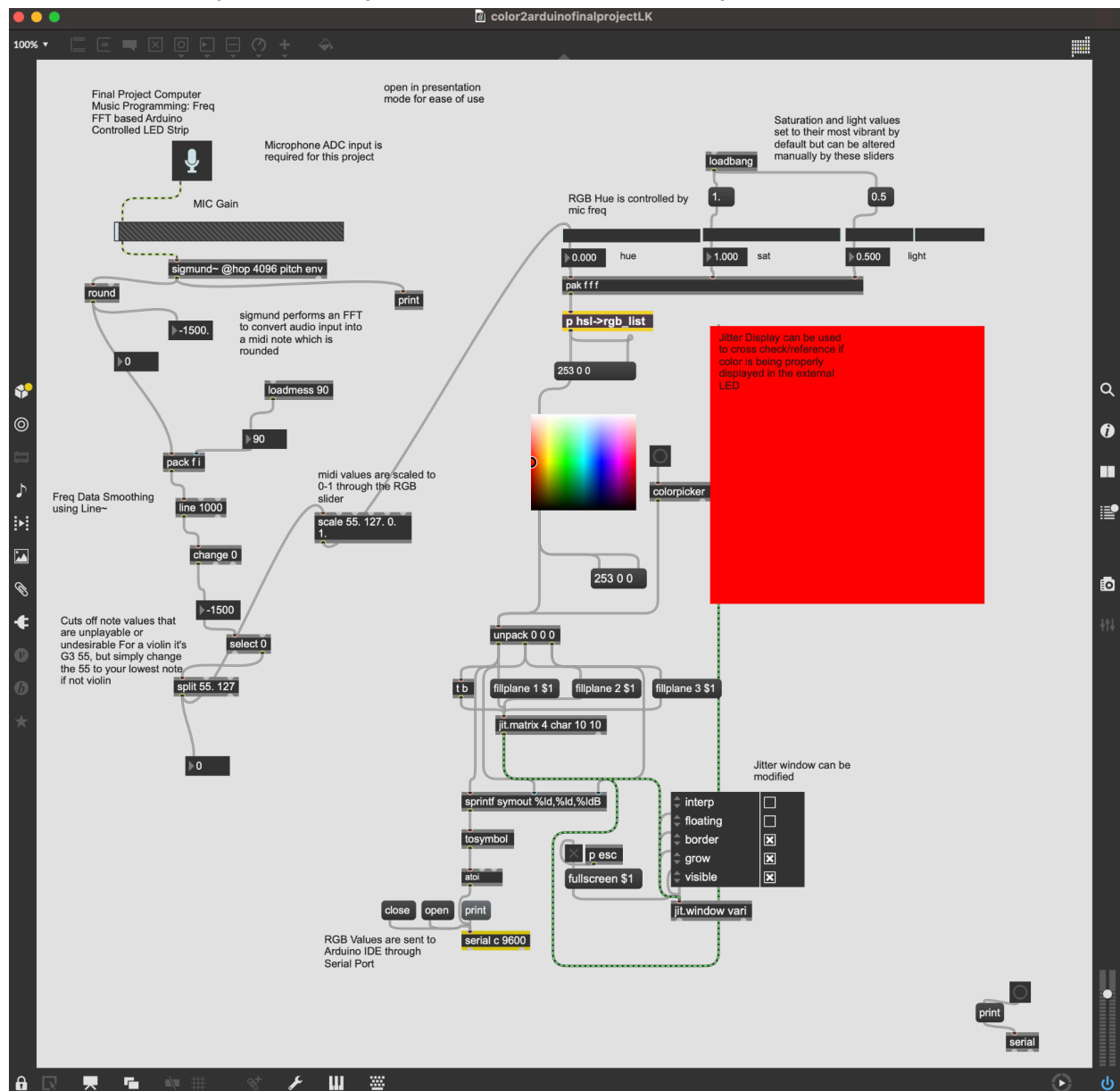


```
sketch_march21finalprojLK.ino
1  #include <Adafruit_NeoPixel>
2
3  #define LED_COUNT 60
4  #define LED_PIN 2
5
6  Adafruit_NeoPixel pixels = Adafruit_NeoPixel(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
7
8  int r, g, b;
9  String myString;
10
11 void setup() {
12   Serial.begin(9600);
13   pixels.begin();
14 }
15
16 void loop() {
17   if (Serial.available()) {
18     char received = Serial.read();
19     myString += received;
20
21     if (received == 'B') {
22       int commaIndex = myString.indexOf(',');
23       int secondCommaIndex = myString.indexOf(',', commaIndex + 1);
24
25       if (commaIndex != -1 && secondCommaIndex != -1) {
26         String firstValue = myString.substring(0, commaIndex);
27         String secondValue = myString.substring(commaIndex + 1, secondCommaIndex);
28         String thirdValue = myString.substring(secondCommaIndex + 1);
29
30         r = firstValue.toInt();
31         g = secondValue.toInt();
32         b = thirdValue.toInt();
33
34         // Ensure values are within the valid range (0-255)
35         r = constrain(r, 0, 255);
36         g = constrain(g, 0, 255);
37         b = constrain(b, 0, 255);
38
39         Serial.print("Received RGB values: ");
40         Serial.print(r);
41         Serial.print(", ");
42         Serial.print(g);
43         Serial.print(", ");
44         Serial.println(b);
45
46         myString = "";
47       } else {
48         Serial.println("Error: Invalid input format");
49       }
50     }
51   }
52
53   for (int i = 0; i < LED_COUNT; i++) {
54     pixels.setPixelColor(i, pixels.Color(r, g, b));
55   }
56
57   pixels.show();
58 }
59
```

2. Run the program and make sure it compiles correctly, if you run into issues compiling check if the arduino is properly wired and connected to the computer

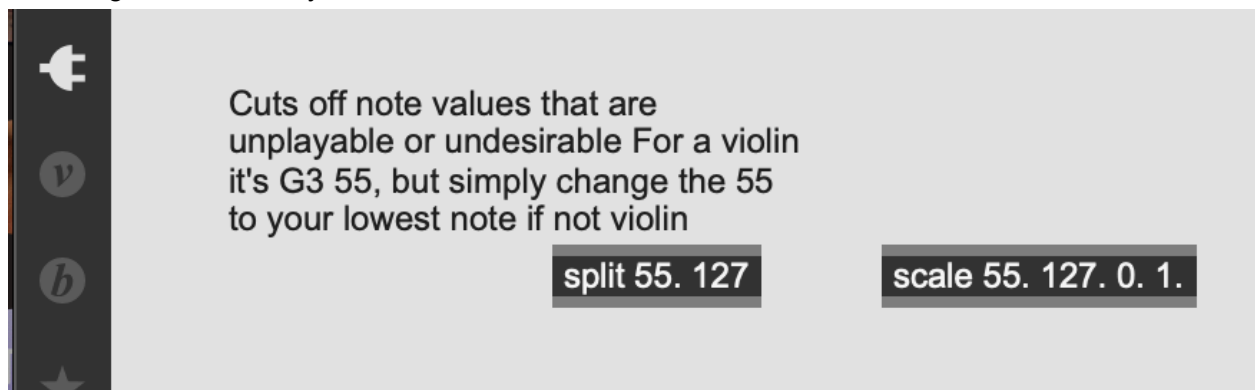
Using the Max/MSP System:

1. While the Arduino IDE code is running, then open up Max/MSP and open up the project folder and open my maxpat project titled color2arduinofinalprojectLK.

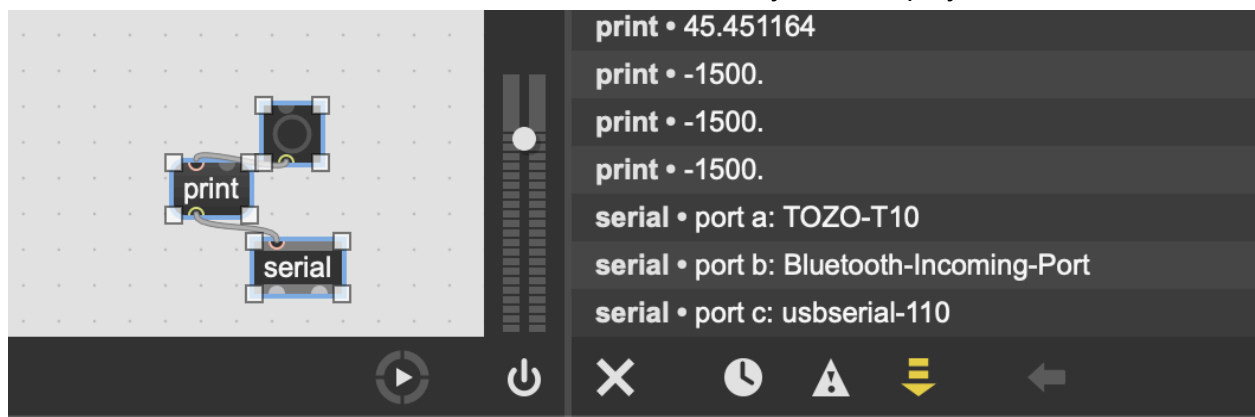


2. For ease of use, open the project in presenter mode, and simply provide monophonic audio input to the system. This could be from a microphone or any other audio source connected to your computer. Adjust the microphone gain input for sensitivity
3. The system will analyze the incoming sound using the Fast Fourier Transform algorithm implemented through the Sigmund object. This will convert frequency data into MIDI note data.
4. The MIDI note data is processed, scaled, and converted into RGB hue data. This color data is then visualized through a Jitter object, displaying vibrant colors corresponding to the analyzed sound. These objects control the cutoff point of the midi values that are accepted, so adjust the

midi range as necessary for non violin instruments.



4. The RGB hue data is sent via the serial port to the connected Arduino Nano. In the Arduino IDE, upload the provided sketch to your Arduino Nano. This sketch will receive the RGB data and control the Neopixel LED strip accordingly, check with the jitter object to see if the correct colors or if any color should be sent. If there isn't any data being sent to the Arduino, check if the serial port of your Arduino is correctly labeled in the serial object. In my case, the serial object is serial c, but it may be different if you use a different Arduino. Use this print statement to double check. Just make sure the baud rate of the serial object in the project is set to 9600



5. Once the system is set up and running, enjoy the captivating visual effects produced by the synchronized LED strip, responding to the input sound in real-time.

Troubleshooting:

- If the Arduino IDE doesn't upload correctly, make sure you compile/run the Arduino IDE before opening the Maxpatch, for some reason
- If the LED strip doesn't display any colors:
 - Check the wiring connections between the Arduino Nano and the LED strip.
 - Ensure that the Arduino IDE sketch is uploaded correctly to the Arduino Nano.
- If the Max patch doesn't display any visualizations:
 - Verify that the audio input device is connected and working properly.
 - Check the Max patch for any errors or missing objects.
- If the Max patch doesn't load the message objects correctly
 - Check in the project folder for a test patch called color2arduino, and open that up first and re-open the project file

Sources:

- NeoPixel Library
- Sound Simulator Youtube Channel
 - [Sending Data Back & Forth Between Arduino And Pure Data \(also works wi...](#)
 - Used to help create the interface between max and Arduino with the serial object and serial reference value
 - [Let's Create a Ribbon Synth! \(Arduino + Max/MSP Or Pure Data\)](#)
 - Used as inspiration for wiring the Arduino
- Inspiration for Jitter [Audio Reactive: Amplitude to Color - Audiovisual Max/MSP Tut...](#)
- Data smoothing: [Data Smoothing in Pure Data & MaxMSP](#) with line~

Thank you for using the Monophonic Sound Analysis and Visualization System. We hope you enjoy exploring the realms of sound and color with this interactive and immersive experience.