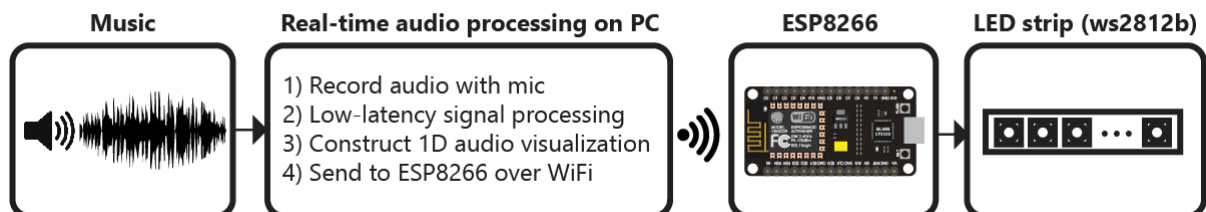
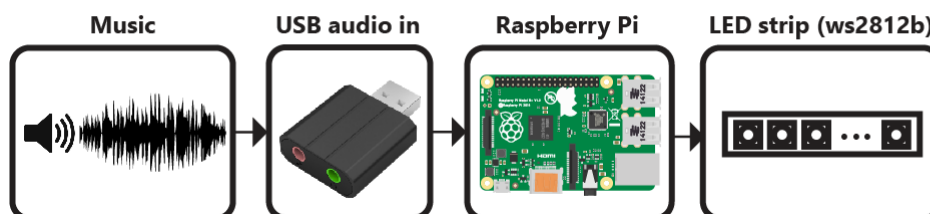

Audio Reactive LED Strip

Real-time LED strip music visualization using Python and the ESP8266 or Raspberry Pi.

Computer + ESP8266

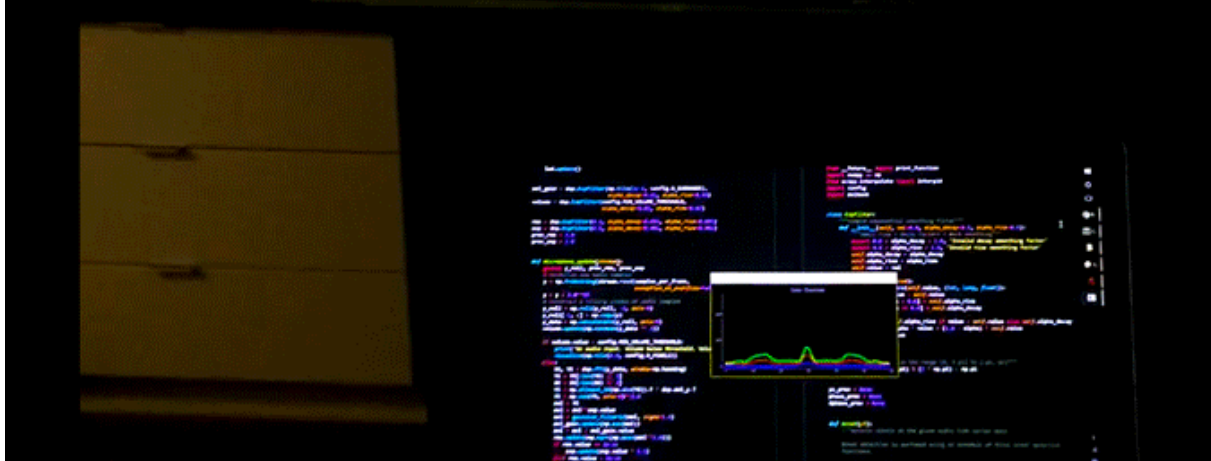


Standalone Raspberry Pi

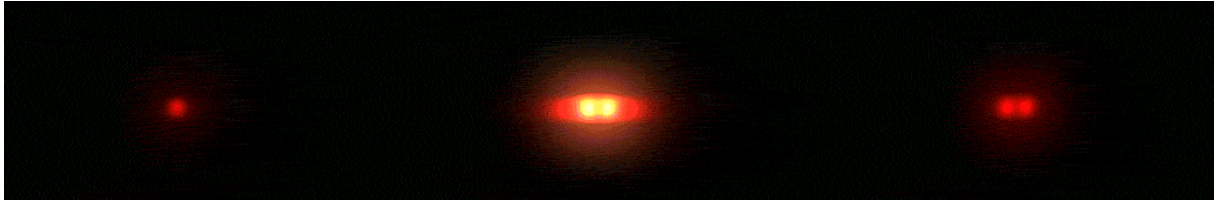


Real-time audio visualization

- ESP8266 controls LED strip (ws2812b)
- Computer processes real-time audio and streams commands to ESP8266 over WiFi



Demo (click gif for video)



Overview

The repository includes everything needed to build an LED strip music visualizer (excluding hardware):

- Python visualization code, which includes code for:
 - Recording audio with a microphone (microphone.py)
 - Digital signal processing (dsp.py)
 - Constructing 1D visualizations (visualization.py)
 - Sending pixel information to the ESP8266 over WiFi (led.py)
 - Configuration and settings (config.py)
- Arduino firmware for the ESP8266 (ws2812_controller.ino)

What do I need to make one?

Computer + ESP8266

To build a visualizer using a computer and ESP8266, you will need: - Computer with Python 2.7 or 3.5 (Anaconda is recommended on Windows) - ESP8266 module with RX1 pin exposed. These modules can be purchased for as little as \$5 USD. These modules are known to be compatible, but many others will work too: - NodeMCU v3 - Adafruit HUZZAH - Adafruit Feather HUZZAH - WS2812B LED strip (such as Adafruit Neopixels). These can be purchased for as little as \$5-15 USD per meter. - 5V power supply - 3.3V-5V level shifter (optional, must be non-inverting)

Limitations when using a computer + ESP8266: - The communication protocol between the computer and ESP8266 currently supports a maximum of 256 LEDs.

Standalone Raspberry Pi

You can also build a standalone visualizer using a Raspberry Pi. For this you will need: - Raspberry Pi (1, 2, or 3) - USB audio input device. This could be a USB microphone or a sound card. You just need to find some way of giving the Raspberry Pi audio input. - WS2812B LED strip (such as Adafruit Neopixels) - 5V power supply - 3.3V-5V level shifter (optional)

Limitations when using the Raspberry Pi: - Raspberry Pi is just fast enough to run the visualization, but it is too slow to run the GUI window as well. It is recommended that you disable the GUI when running the code on the Raspberry Pi. - The ESP8266 uses a technique called temporal dithering to improve the color depth of the LED strip. Unfortunately the Raspberry Pi lacks this capability.

Installation for Computer + ESP8266

Python Dependencies

Visualization code is compatible with Python 2.7 or 3.5. A few Python dependencies must also be installed: - Numpy - Scipy (for digital signal processing) - PyQtGraph (for GUI visualization) - PyAudio (for recording audio with microphone)

On Windows machines, the use of Anaconda is **highly recommended**. Anaconda simplifies the installation of Python dependencies, which is sometimes difficult on Windows.

Installing dependencies with Anaconda

Create a conda virtual environment (this step is optional but recommended)

```
1 conda create --name visualization-env python=3.5
2 activate visualization-env
```

Install dependencies using pip and the conda package manager

```
1 conda install numpy scipy pyqtgraph
2 pip install pyaudio
```

Installing dependencies without Anaconda

The pip package manager can also be used to install the python dependencies.

```
1 pip install numpy
2 pip install scipy
3 pip install pyqtgraph
4 pip install pyaudio
```

If `pip` is not found try using `python -m pip install` instead.

Installing macOS dependencies

On macOS, python3 is required and `portaudio` must be used in place of `pyaudio`. If you don't have brew installed you can get it here: <https://brew.sh>

```
1 brew install portaudio
2 brew install pyqt5
3 pip3 install numpy
4 pip3 install scipy
5 pip3 install pyqtgraph
6 pip3 install pyaudio
```

Running the visualization can be done using the command below.

```
python3 visualization.py /tmp
```

Arduino dependencies

ESP8266 firmware is uploaded using the Arduino IDE. See this tutorial to setup the Arduino IDE for ESP8266.

Install NeoPixelBus library

Download [Here](#) or using library manager, search for "NeoPixelBus".

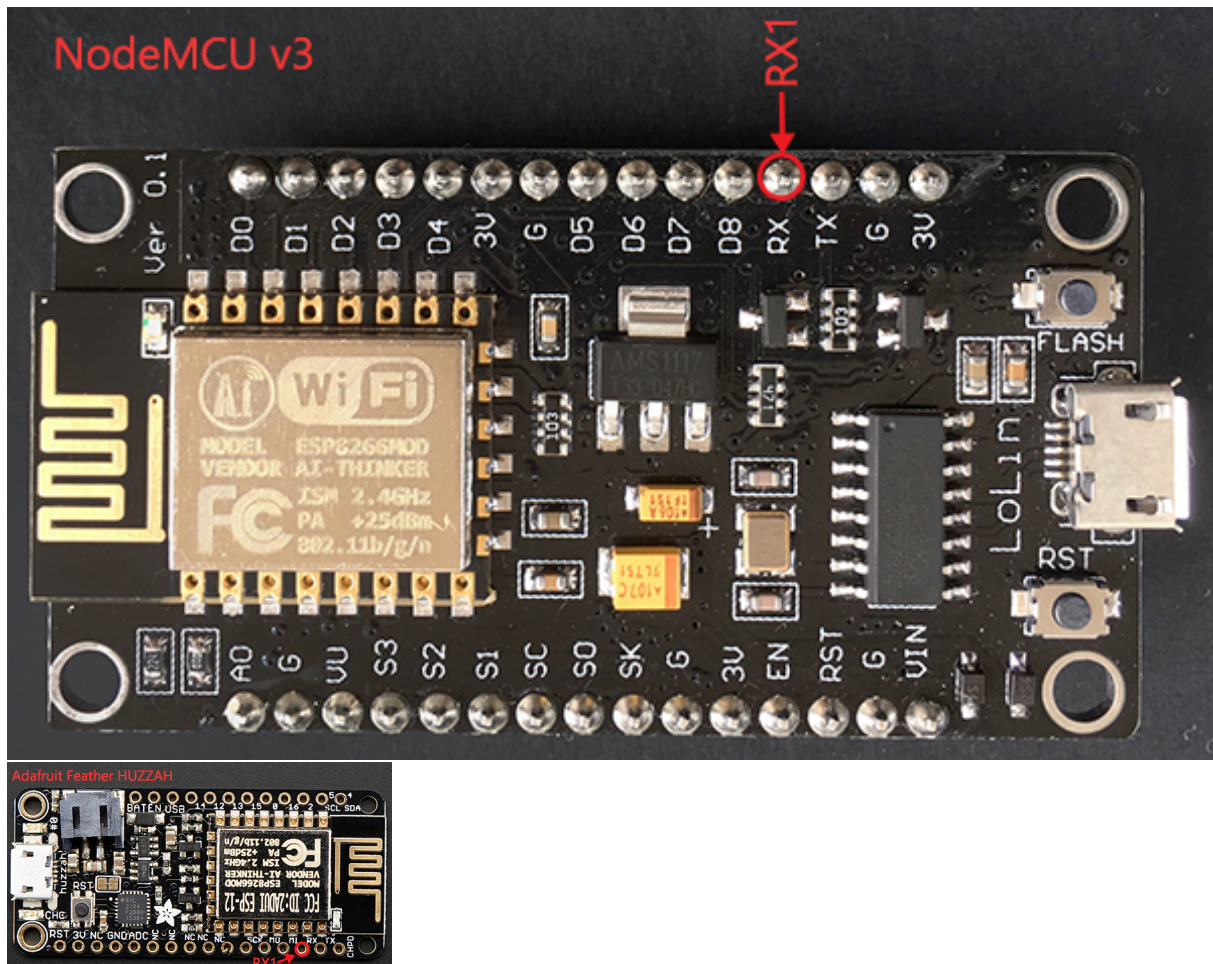
Hardware Connections

ESP8266

The ESP8266 has hardware support for I²S and this peripheral is used to control the ws2812b LED strip. This significantly improves performance compared to bit-banging the IO pin. Unfortunately, this means that the LED strip **must** be connected to the RX1 pin, which is not accessible in some ESP8266 modules (such as the ESP-01).

The RX1 pin on the ESP8266 module should be connected to the data input pin of the ws2812b LED strip (often labelled DIN or D0).

For the NodeMCU v3 and Adafruit Feather HUZZAH, the location of the RX1 pin is shown in the images below. Many other modules also expose the RX1 pin.



Raspberry Pi

Since the Raspberry Pi is a 3.3V device, the best practice is to use a logic level converter to shift the 3.3V logic to 5V logic (WS2812 LEDs use 5V logic). There is a good overview on the best practices [here](#).

Although a logic level converter is the best practice, sometimes it will still work if you simply connect the LED strip directly to the Raspberry Pi.

You cannot power the LED strip using the Raspberry Pi GPIO pins, you need to have an external 5V power supply.

The connections are:

-
- Connect GND on the power supply to GND on the LED strip and GND on the Raspberry Pi (they MUST share a common GND connection)
 - Connect +5V on the power supply to +5V on the LED strip
 - Connect a PWM GPIO pin on the Raspberry Pi to the data pin on the LED strip. If using the Raspberry Pi 2 or 3, then try Pin 18(GPIO5).

Setup and Configuration

1. Install Python and Python dependencies
2. Install Arduino IDE and ESP8266 addon
3. Download and extract all of the files in this repository onto your computer
4. Connect the RX1 pin of your ESP8266 module to the data input pin of the ws2812b LED strip. Ensure that your LED strip is properly connected to a 5V power supply and that the ESP8266 and LED strip share a common electrical ground connection.
5. In `ws2812_controller.ino`:
 - Set `const char* ssid` to your router's SSID
 - Set `const char* password` to your router's password
 - Set `IPAddress gateway` to match your router's gateway
 - Set `IPAddress ip` to the IP address that you would like your ESP8266 to use (your choice)
 - Set `#define NUM_LEDS` to the number of LEDs in your LED strip
6. Upload the `ws2812_controller.ino` firmware to the ESP8266. Ensure that you have selected the correct ESP8266 board from the boards menu. In the dropdown menu, set `CPU Frequency` to 160 MHz for optimal performance.
7. In `config.py`:
 - Set `N_PIXELS` to the number of LEDs in your LED strip (must match `NUM_LEDS` in `ws2812_controller.ino`)
 - Set `UDP_IP` to the IP address of your ESP8266 (must match `ip` in `ws2812_controller.ino`)
 - If needed, set `MIC_RATE` to your microphone sampling rate in Hz. Most of the time you will not need to change this.

Installation for Raspberry Pi

If you encounter any problems running the visualization on a Raspberry Pi, please open a new issue. Also, please consider opening an issue if you have any questions or suggestions for improving the installation process.

Download and extract all of the files in this repository onto your pi to begin.

Installing the Python dependencies

Install python dependencies using apt-get

```
1 sudo apt-get update
2 sudo apt-get install python-numpy python-scipy python-pyaudio
```

Audio device configuration

For the Raspberry Pi, a USB audio device needs to be configured as the default audio device.

Create/edit `/etc/asound.conf`

```
1 sudo nano /etc/asound.conf
```

Set the file to the following text

```
1 pcm.!default {
2     type hw
3     card 1
4 }
5 ctl.!default {
6     type hw
7     card 1
8 }
```

Next, set the USB device to as the default device by editing `/usr/share/alsa/alsa.conf`

```
1 sudo nano /usr/share/alsa/alsa.conf
```

Change

```
1 defaults.ctl.card 0
2 defaults.pcm.card 0
```

To

```
1 defaults.ctl.card 1
2 defaults.pcm.card 1
```

Test the LED strip

1. `cd rpi_ws281x/python/examples`

-
2. `sudo nano strandtest.py`
 3. Configure the options at the top of the file. Enable logic inverting if you are using an inverting logic-level converter. Set the correct GPIO pin and number of pixels for the LED strip. You will likely need a logic-level converter to convert the Raspberry Pi's 3.3V logic to the 5V logic used by the ws2812b LED strip.
 4. Run example with `'sudo python strandtest.py'`

Configure the visualization code

In `config.py`, set the device to `'pi'` and configure the GPIO, LED and other hardware settings. If you are using an inverting logic level converter, set `LED_INVERT = True` in `config.py`. Set `LED_INVERT = False` if you are not using an inverting logic level converter (i.e. connecting LED strip directly to GPIO pin).

Audio Input

The visualization program streams audio from the default audio input device (set by the operating system). Windows users can change the audio input device by following these instructions.

Examples of typical audio sources: * Audio cable connected to the audio input jack (requires USB sound card on Raspberry Pi) * Webcam microphone, headset, studio recording microphone, etc

Virtual Audio Source

You can use a “virtual audio device” to transfer audio playback from one application to another. This means that you can play music on your computer and connect the playback directly into the visualization program.

Physical Audio Source

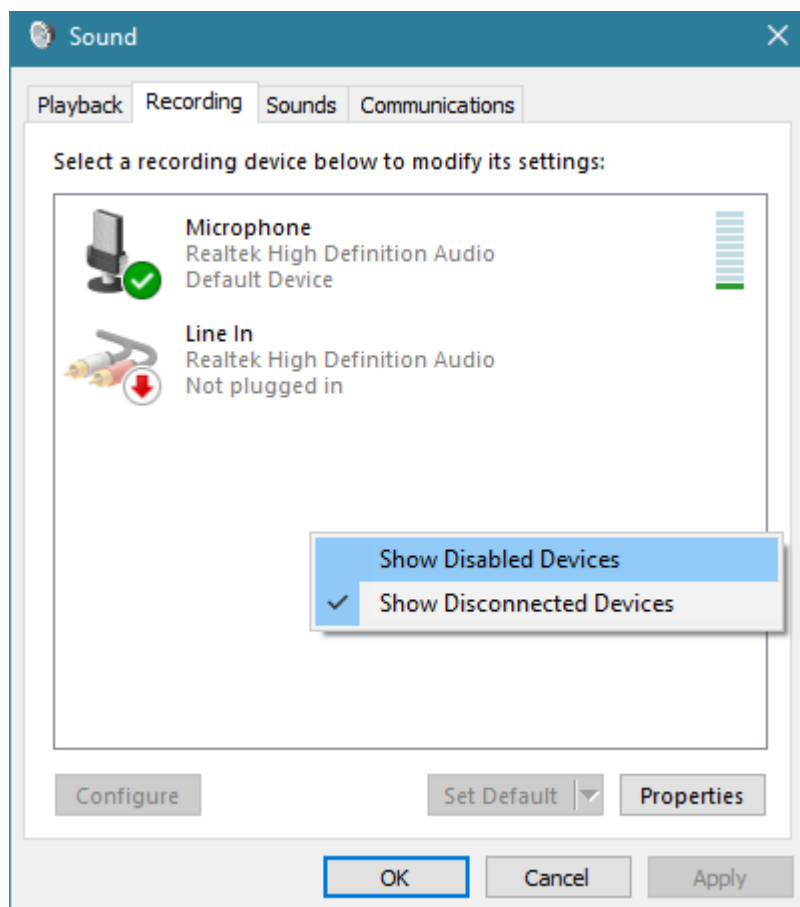


Virtual Audio Source

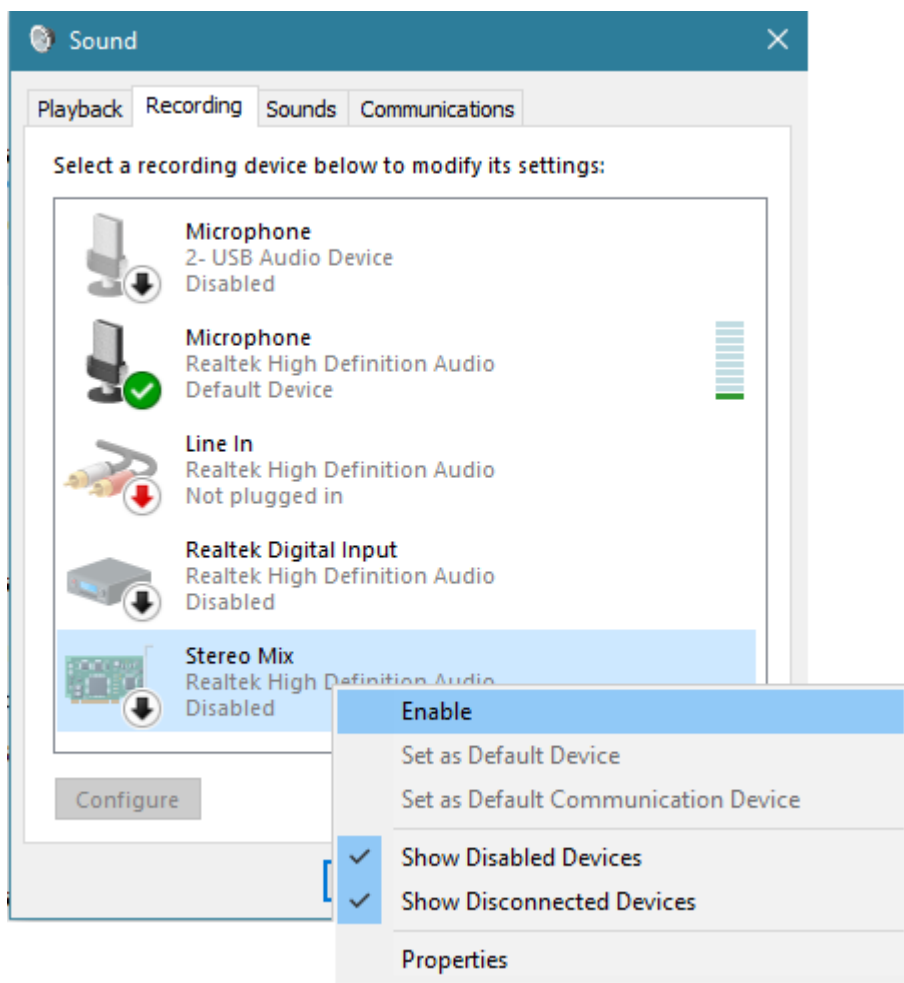


Windows

On Windows, you can use “Stereo Mix” to copy the audio output stream into the audio input. Stereo Mix is only supported on certain audio chipsets. If your chipset does not support Stereo Mix, you can use a third-party application such as Voicemeeter.



Go to recording devices under Windows Sound settings (Control Panel -> Sound). In the right-click menu, select "Show Disabled Devices".



Enable Stereo Mix and set it as the default device. Your audio playback should now be used as the audio input source for the visualization program. If your audio chipset does not support Stereo Mix then it will not appear in the list.

Linux

Linux users can use Jack Audio to create a virtual audio device.

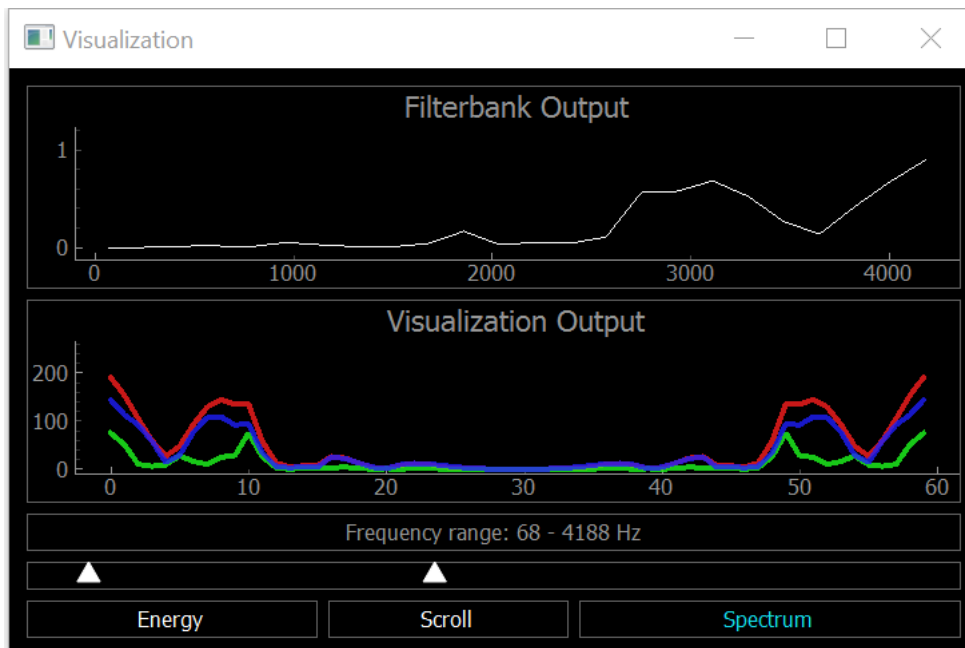
OSX

On OSX, Loopback can be use to create a virtual audio device.

Running the Visualization

Once everything has been configured, run `visualization.py` to start the visualization. The visualization will automatically use your default recording device (microphone) as the audio input.

A PyQtGraph GUI will open to display the output of the visualization on the computer. There is a setting to enable/disable the GUI display in `config.py`



If you encounter any issues or have questions about this project, feel free to open a new issue.

Limitations

- ESP8266 supports a maximum of 256 LEDs. This limitation will be removed in a future update. The Raspberry Pi can use more than 256 LEDs.
- Even numbers of pixels must be used. For example, if you have 71 pixels then use the next lowest even number, 70. Odd pixel quantities will be supported in a future update.

License

This project was developed by Scott Lawson and is released under the MIT License.