Arduino Music Reactive LEDs

# Introduction

The purpose of this project is to create an LED visualizer using Arduino Uno, paired with a sound sensor. The LEDs light up according to the intensity of the sound that’s being picked up by the sensor and change color based on the frequency recorded.

# Components

These are the components that I’ve used to create this project:

* Arduino Uno R3
* Sound Detection Module – OKY3131
* 6 LED Strips – WS2811

# Circuit Diagram

The circuit follows this schematic:

A diagram of a circuit board

Description automatically generated

The LED strips were soldered in order for the Arduino to perceive them as one long strip.

# Arduino Code

The code is relatively simple. I used two libraries:

* FastLED
* arduinoFFT

The FastLED library is very useful when controlling LEDs because it provides easy setup for colors, brightness, and patterns for a wide range of LED types.

The arduinoFFT library is used to perform Fast Fourier Transform on signals, enabling the conversion of time-domain data into the frequency domain for analysis.

The purpose of the code is to read the input from the sensor, process it using the FFT and display visualizations on the LED matrix. As I said in the beginning, the LEDs change color based on the frequency, starting with a red-ish to yellow color for a lower frequency and a magenta color for higher ones.

* Setup

A computer screen shot of a computer code

Description automatically generated

The setup() function initializes the hardware components (the sound sensor and the LED strips). A 3s delay is set as a safety delay for components to stabilize and to ensure reliable communication before the main program starts.

* Loop

A computer screen shot of a code

Description automatically generated

This function contains the main program logic. This is where I control the timing of the LED updates using “millis()” for periodic execution and change the LEDs at a given interval. This is where we call the Visualizer function to see the LEDs physically light up.

* Visualizer

A screenshot of a computer program

Description automatically generated

Using this function, we begin the process of collecting samples and displaying the LEDs.

* getSamples

Here is where we actually collect the sound samples from the microphone input, using FFT analysis on them. Then we extract the relevant frequency information and store the real and imaginary part of the samples in the vReal and vImag arrays. After collecting the samples, it applies a windowing function to minimize the distortions in the frequency analysis results and to prepare the FFT analysis. Finally, it computes the FFT, changes the data into magnitudes, and identifies the peak frequency and amplitude. Lastly, we use the variable smoothedFrequency to stabilize the frequency measurements and reduce fluctuations, applying a smoothing algorithm to the peak frequency.

A computer screen shot of a program code

Description automatically generated

* displayUpdate

This the part where we update the LED display based on the analyzed sound data from the previous function. We check if the smoothed peak frequency is within a range of 500 Hz and 3000 Hz. This ensures that only sounds within the desired frequency range are visualized. If the frequency condition is met, the function maps the smoothed frequency to a color hue. Increasing frequency mean increasing colors as specified in the picture below. Higher peak amplitudes result in more LEDs being lit up in each column. In case the frequency condition isn’t met, the function turns off all LEDs.

A screen shot of a computer program

Description automatically generated

# Conclusions

I am quite happy with what I have accomplished. There are some things that I did not manage to solve such as the frequency range in the displayUpdate function is hard coded. I tried setting them according to a video on YouTube in which we see a gradual increase in frequency from 20 Hz to 20 KHz, but the visuals and program did not work accordingly. There is another issue when it comes to coloring and the hues; when there is a sudden spike in frequency, it results in a hue that is above the limit of 255 so the LEDs turn off until the next frequency is registered.

# Bibliography

* <https://www.instructables.com/Arduino-FFT-Visualizer-With-Addressable-LEDs/>
* <https://www.arduino.cc/reference/en/language/functions/time/millis/>
* <https://fastled.io/docs/>