NTNU | Norwegian University of Science and Technology

TFY4190 Instrumentering
October 16, 2024

Audio-reactive Discolights by FFT

Abel Gangeskar Horneland[1] and Thyra Rolfseng[1]

[1]Department of Physics, NTNU, Trondheim

Abstract

Audio-reactive disco lights, powered by Fast Fourier Transform (FFT) analysis of microphone signals, embody a fusion of technology and art. These lights can enrich sensory experiences through synchronized light displays. This project utilizes a microcontroller (Arduino Uno), along with the ArduinoFFT and FastLED libraries, Arduino software, and a custom-made printed circuit board with an bandpass filter with amplification to achieve the objective.

Introduction

This project aims to use audio data from a microphone to control the color and number of LEDs active, mimicking an equalizer. The goal is to do this through the integration and Arduino Uno with a custom made printed circuit board that amplifies and filters the audio signal, and powers the LED lights. The objective is to dissect the incoming audio signal via Fast Fourier Transform analysis, providing an understanding of the frequency components and manipulating and using it to control the lights. An ilustration of the concept can be seen in Figure 1.

Figure 1: Audio reactive discolights concept figure.

Theory

Fast Fourier Transform

The Cooley-Tukey Fast Fourier Transform (FFT) algorithm has opened new possibilities when it comes to digital signal processing [1]. It reduces the complexity of crucial computational tasks, like the Fourier transform (FT). Direct computations of discrete Fourier transform (DFT) requires on the order of N^2 operations, while Cooley-Tukey FFT brings down the complexity to an order of $Nlog_2(N)$ operations [1].

The general DFT of an N-point signal $\{x[n], 0 \leq n \leq N-1\}$ is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{-kn} \tag{1}$$

where

$$W_N = e^{j\frac{2\pi}{N}} = cos(\frac{2\pi}{N}) + jsin(\frac{2\pi}{N}) \tag{2}$$

is the principal N-th rooth of unity [2]. The radix-2 Decimation In Time (DIT) case of the Cooley Tukey FFT proceed by division of DFT into two DFTs of length $N/2$. The FFT will utilize two properties of W_N,

$$W_N^2 = W_{N/2} \tag{3}$$

and

$$W_N^{k+N/2} = -W_N^k. \tag{4}$$

Then by considering a N-point signal $x[n]$ of even length, the DIT radix-2 FFT derivation begins by splitting equation 1 into two sums [2]. One part has the even indexed values $x[2n]$, while the other the odd indexed values $x[2n+1]$. Equation 1 can then be written as

$$X[k] = \sum_{n=0}^{N/2-1} x[2n]W_N^{-2nk} + \sum_{n=0}^{N/2-1} x[2n+1]W_N^{-(2n+1)k} \tag{5}$$

or

$$X[k] = \sum_{n=0}^{N/2-1} x_0[n]W_{N/2}^{-nk} + W_N^{-k}\sum_{n=0}^{N/2-1} x_1[n]W_{N/2}^{-nk} \tag{6}$$

where $x_0[n] = x[2n]$ and $x_1[n] = x[2n+1]$. Leading to the result

$$X[k] = X_0 + W_N^{-k}X_1 \tag{7}$$

and

$$X[k + \frac{N}{2}] = X_0 - W_N^{-k}X_1, \tag{8}$$

where $X_0 = DFT_{N/2}\{x_0[n]\}$ and $X_1 = DFT_{N/2}\{x_1[n]\}$. This result shows the essence of the radix-2 DIT FFT, expressing the DFT of length N recursively, utilizing two DFTs of size N/2. This approach optimizes the speed by efficiently utilizing intermediate computation results to compute multiple DFT outputs.

Inverting bandpass filter with amplification

A bandpass filter is a circuit crucial in various electronic applications for its ability to selectively transmit frequencies within a specific range while attenuating those outside this range. This section delves into the fundamental characteristics of the inverting bandpass filter with amplification described in section , specifically focusing on the cutoff frequencies and the amplification of the signal.

An inverting bandpass filter is a filter that offsets the phase of the signal by 180 degrees [3]. The cutoff frequencies of the filter can be found individually from the standard formula for the cutoff frequency f_c, for an RC filter, given by

$$f_c = \frac{1}{2\pi RC}. \tag{9}$$

Where R is the resistance and C is the capacitance of the filter. For the high-pass part of the filter, this becomes

$$f_L = \frac{1}{2\pi R_1 C_1} \tag{10}$$

where f_L is the lower cutoff frequency of the filter. For the low-pass part of the filter, the upper cutoff frequency is

$$f_H = \frac{1}{2\pi R_2 C_2}. \tag{11}$$

The amplification of the filter can be calculated using

$$A = -\frac{R_1}{R_2} \tag{12}$$

where the negative sign indicates that the signal is inverted [3].

Method

Main Program

Behind the audio-reactive disco lights lies the FFT, a mathematical algorithm used to analyze signals in the frequency domain, as presented in the theory. By applying FFT to the microphone input signals captured, it was possible to break down the audio data into its constituent frequencies. This process allowed for the lights to synchronize and respond to the frequency and intensity of the music.

By comparing the calculated frequency peaks to a measurement by the application phyphox, we confirmed that the frequency by FFT was not far off.

The LED strip was programmed by the FastLED library. FastLED is a fast, efficient, easy-to-use Arduino library for programming addressable LED strips [4]. The amplitude by the microphone signal decided the number of LEDs reacting to sound, while the frequency f_peaks[0] decided the color of these LEDs.

Custom PCB

For this project, a custom printed circuit board (PCB) was made. The PCB was made to fit directly on top of the Arduino Uno, and included the microphone, the band pass filter and an MPC3201 analog to digital converter to convert the filtered microphone signal. The finished PCB i pictured in Figure 2.

The bandpass filter circuit is depicted in Figure 3. It is an inverting bandpass filter with amplification, and is typically used for sound where the offset from

:

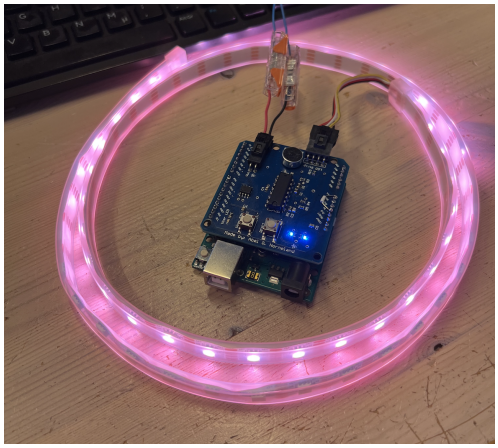

Figure 2: Image of the project setup with a PCB.

the inverting factor doesent matter [3]. The values for C_1, C_2, R_1 and R_2 were changed to increase the amplification of the signal from 20 dB to 30 dB. The final values are listed in table 1.

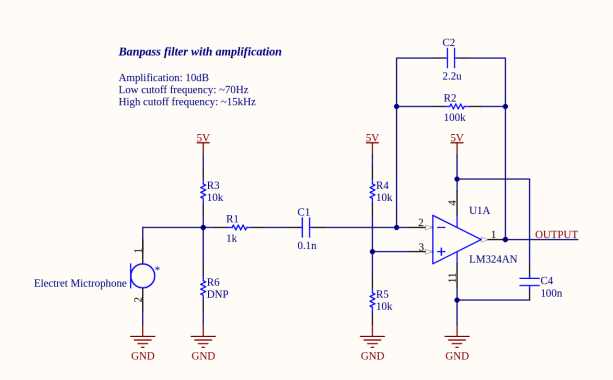

Figure 3: Schematic of the bandpass filter circuit

The low cutoff frequency was chosen to be 150 Hz to filter some of the 50 Hz noise, and frequencies that are not typically in music. The high cutoff frequency was set to 15 900 Hz. The cutoff frequencies were also informed by available resistor and capacitor values. The filter amplification was set to 30 dB. This comes from the equations 12, 11, and 10 and the component values in table 1.

Table 1: Final component values for bandpass filter

Component	Value	Unit
C_1	10	μF
R_1	100	Ω
C_2	100	pF
R_2	100	$k\Omega$

The analog to digital converter on the PCB was used to sample the audio signal at a higher resolution and frequency than what was possible with the internal ADC on the arduino uno. It has a resolution of 12 bit, compeared to the Arduino Uno's internal 10 bit ADC improving the resolution of the measured audio. The ADC uses a 3 wire SPI protocol to communicate with the arduino through the dedicated hardware SPI pins on the arduino.

Discussion

The primary challenge with audio-reactive lights lies in effectively sampling sound while simultaneously manipulating the lights. Sampling sound halt the rest of the program, preventing the lights from reacting to sound signals in real-time. Moreover, the inclusion of the built-in wait() function in Arduino makes this issue worse further delaying the light response. To address this challenge, we developed a custom function for time management and scheduling sound sampling, and light control, allowing for seamless synchronization between audio input and light manipulation. Ideally, the program for audio-reactive lights should perform both sound sampling and light manipulation concurrently, but this is not possible with our setup.

PCB

There were several issues with the PCB. Firstly the formulas for the lower and upper cutoff frequencies given in [3] are swapped, so the planned values for the circuit board were wrong. Secondly, the Vin voltage input for the board that was designed to power both the arduino and the LED strip was connected to the Vin port on the arduino and the Vin of the LED strip. Theese require different voltages, with the arduino taking $7 - 12V$, while the LED strip having a max rated voltage of $5V$. This was solved by removing the Vin header pin connecting the PCB to the arduino, and jumping it to the $5V$ pin to power it through there instead. This is not a recommended solution as the $5V$ pin on the arduino offers no protection.

It also turned out that the arduino uses pin 0 and pin 1 to upload new programs. Theese pins were routed to the CS and MISO pins of the external ADC, meaning the arduino could not be programed with the shield on. This was fixed by again removing the header pins connecting the PCB to the arduino and soldering jumper wires from those pins to other unused pins on the PCB. It was however still not

possible to program the arduino with the PCB on, which was traced to the $100nF$ capacitor connected to the reset button being too big, so it was soldered off.

After this the PCB worked well and gave sensible audio readings, but the signal was still a little weak, so the R_1 resistor and C_1 capacitor was swapped to amplify the signal more and to readjust the low cutoff frequency. The ADC worked well for sampling the signal, a plot of the raw audio data from the ADC can be seen in Figure 4. The LED strip pinout was also incorrect compared to the connector on the LED strip, and the connectors did not fit each other so the connector on the LED strip had to be changed for a nanofit connector.

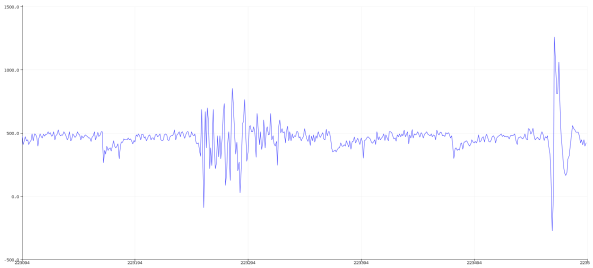

Figure 4: Plot of filtered and amplified audio data recorded using the PCB

Program

Despite prioritizing speed, ArduinoFFT maintains a balance between accuracy and computational efficiency, making it a valuable tool for FFT analysis on Arduino platforms. However, one should be mindful of memory constraints, especially on boards with limited resources. This meant that the data buffer used in the FFT computation was very small at only 64 samples which gave inaccuracies in estimating the peak frequency. While ArduinoFFT offers performance improvements compared to traditional FT, its effectiveness may be limited by memory limitations, particularly for sample sizes exceeding the board's capacity.

It was also difficult to translate the calculated peak frequency into a good color palate. In the end the frequency was mapped to a hue value between 0 and 360 degrees, which yielded good enough results. The number of active LEDs was controlled by the amplitude of the signal. The number of active LEDs were directly mapped to the typical range of the signal, which meant some LEDs were never used. More work could be done to improve the colors of the LED strip and the animation of turning on and off the active LEDs. It still worked well as a proof of concept, and gave a good visualization of the recorded audio signal even though it didn't mimic an equaliser as originally planned.

Testing of the program was done mostly on a breadboard assembly of the circuit while the PCB was being developed and fixed. This meant that in the end when the software was used together with the PCB, some issues with the FFT and control of the LED strip arose that could not be fixed in time. The PCB should have been prioritised earlier to allow time to fix the issues and test it beforehand.

Conclusion

The design, soldering and fixing of the PCB took longer than expected, and should have been prioritised earlier in the project. This in turn affected the software implementation, which couldn't be tested with the final assembly in time, and therefore didn't work.

The program itself worked fairly well during testing, and gave a good visualization of the audio signal, even though it didnt mimic an equalizer as originally planned. More work should generally have been done on the final aesthetics of the project to sell the idea better.

We learned a lot about the FFT algorithm and PCB design in this project. And while it didn't work in the end, we are happy with the project and everything we have learned from it.

References

[1] Duhamel P, Vetterli M. Fast fourier transforms: A tutorial review and a state of the art. Signal Processing. 1990;19(4):259-99. Available from: https://www.sciencedirect.com/science/article/pii/016516849090158U.

[2] Selesnick I. The Fast Fourier Transform (FFT) NYU. 2012. Available from: https://eeweb.engineering.nyu.edu/iselesni/EL713/zoom/fft.

[3] Electronics LA. Bandpass filter calculator. 2018. Available from: http://www.learningaboutelectronics.com/Articles/Bandpass-filter-calculator.php.

[4] Gracia D, Kriegsman M. FastLED. 2012. Available from: https://fastled.io/.

:

Appendix
PCB schematic

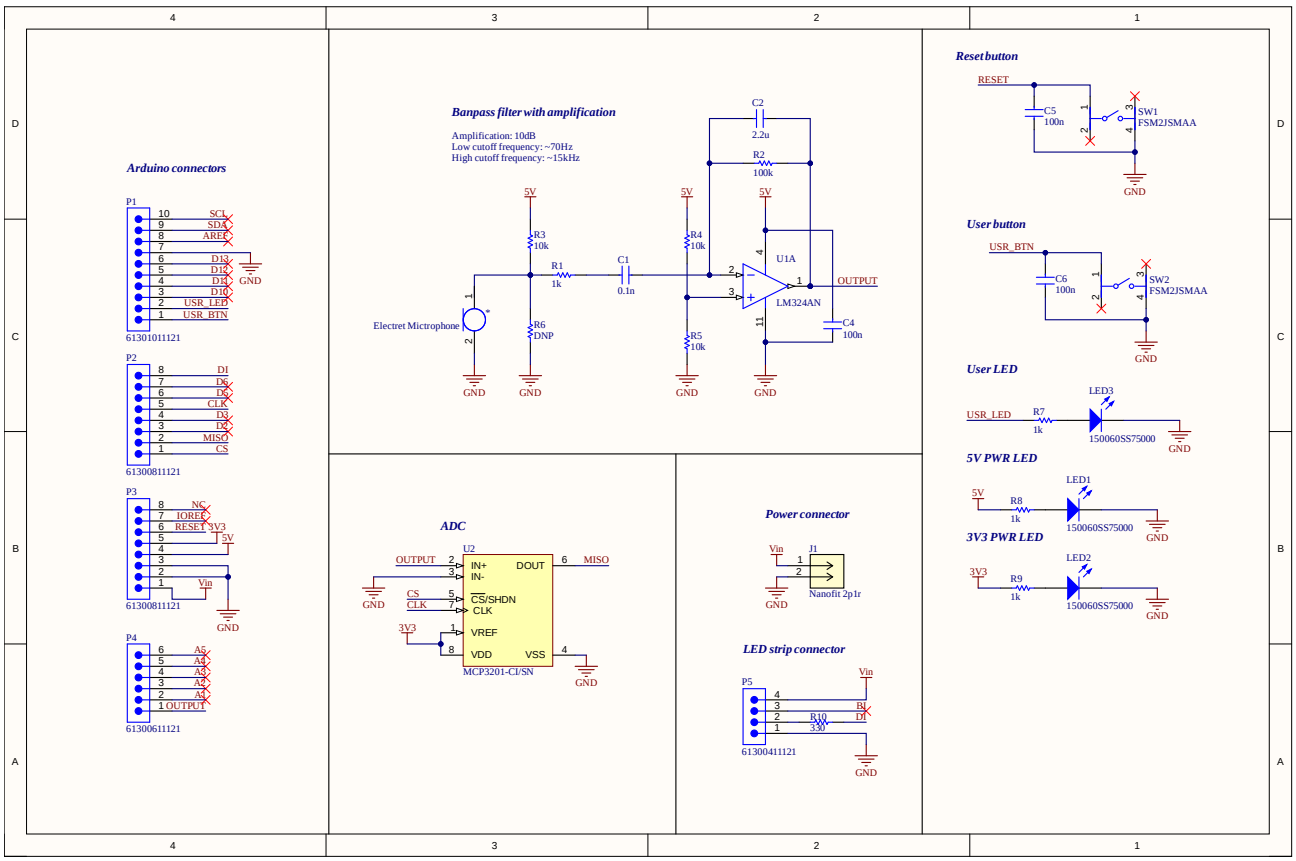

Figure 5: Schematic of the PCB