Music controlled LED ring

Fredrik Knapskog[a], Henrik Storesund[a]

[a] *Department of Physics, Norwegian University of Science and Technology, N-7491 Trondheim, Norway.*

Abstract

A LED strip formed as a ring was programmed to light up with LED lights corresponding to dominating frequencies in a recorded audio signal. This was accomplished by performing a Fourier transform, which took place in the visual programming platform, LabVIEW. The LED strip was controlled by an Arduino Uno. The connection between LabVIEW and Arduino was established with the two expansion packs for LabVIEW called LINX and NI VISA, and the Arduino library called Serial.

1. Introduction

Most disco lights are pre-programmed, often to do repeated or random actions. We however, wanted to make a disco light that that reacts based on the frequencies in the music played. Not only is it supposed to follow the beat, but also show the different frequencies appearing as different LED lights on the disco light. For this we used a led ring with 24 LED lights. Each LED light corresponds to a frequency interval, and lights up when a note of that frequency is present. The lights were also coloured such that all together they make up the colour wheel.

2. Theory

The main tool used for determining which frequencies are present and their amplitudes is the Fourier transform. In its analytical form

$$\hat{F}(\omega) = \int_{-\infty}^{\infty} V(t) e^{-2\pi i t \omega} dt, \tag{1}$$

where $\hat{F}(\omega)$ is the complex amplitude of a given frequency, ω, and $V(t)$ which is the analytical voltage at time t. Computers on the other side use numerical methods and we don't want the integration limits to be infinite. Hence the discrete version of (1) is used

$$\hat{F}_\omega = \sum_{n=0}^{N-1} V_n e^{-2\pi i \omega \frac{n}{N}}. \tag{2}$$

An array of complex amplitudes, $\hat{F}_\omega$, are by (2) computed from measured voltage samples, V_n, for each frequency ω in our frequency array. In order to get a real amplitude

the complex Fourier transforms are multiplied by their complex conjugates. Equation (2) is implemented using FFT [2] in order to calculate more samples per time.

3. Method

The setup consists of four main parts as figure 1 presents. The first part is the microphone circuit. A simple microphone is connected to ground, a 5 V DC voltage, a resistor and a capacitor as shown in figure 2. This simple microphone circuit is then amplified through an operational amplifier as shown in figure 3. The theoretical magnification R_2/R_1, which is about 25 times, corresponds well with the Bode plot in figure 4.

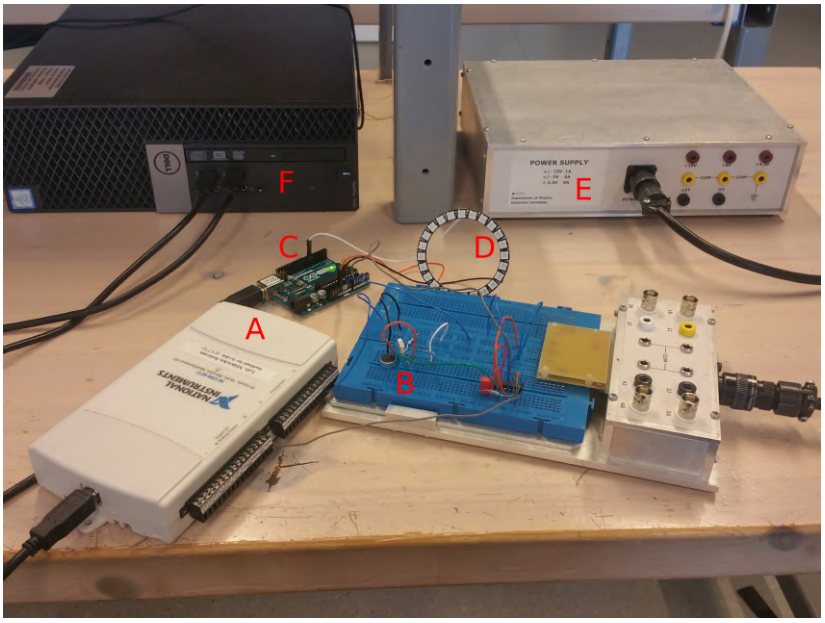

Figure 1: The project set up. A: A NI DAQ, B: A microphone circuit, C: An Arduino Uno, D: An Adafruit RGB 24 neopixel LED ring, E: A DC voltage supply and F: A computer with LabVIEW, LINX and Arduino.

The second part is the National Instruments' Data Acquisition, NI DAQ. It's combined with LabVIEW, a graphical programming program as seen in figure 5. This part is inspired by and adapted from [1]. Here the voltage from the microphone circuit is measured. A buffer is first set up in order to get enough samples before the Fourier transform is executed. By doing this the result becomes more stable as a fixed amount of values are sen each time. They are sent to the Fourier transform as in (2) and removed from the buffer. The measurements that weren't sent are then placed to the front in the buffer. After the Fourier transform the values are multiplied by their complex conjugates in order to form real amplitudes. Next in order to display the amplitudes on 24 LED lights some selection has to be done. We built a 24×5 matrix with the first $24 \cdot 5$ amplitudes from the Fourier transform. For each row if a sufficient amount of the amplitudes

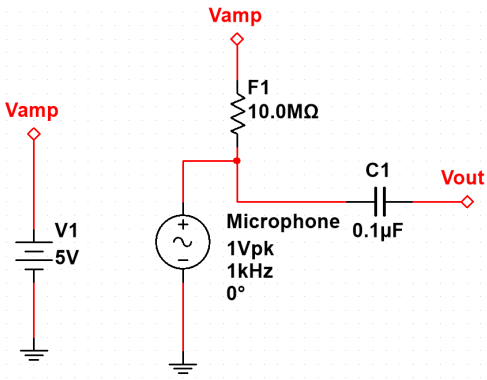

Figure 2: A Multisim sketch of the simple microphone circuit.

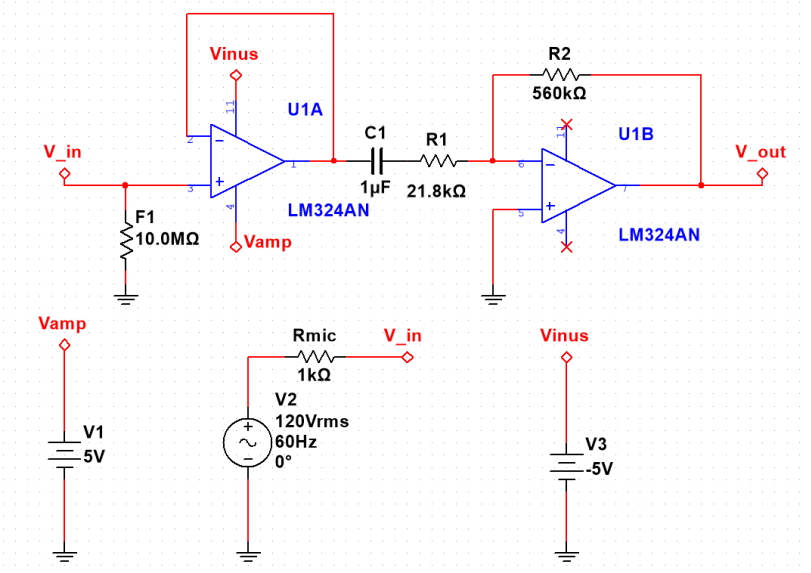

Figure 3: A Multisim sketch of the amplification circuit. The blue triangles are an operational amplifier.

are greater than a given threshold, the row's corresponding LED light will ignite. Our frequency intervals are of approximately 100 Hz and starts at 0 Hz. When the matrix is passed on, the process is repeated with new sound signals.

The third part is the Arduino Uno that is used as an intermediary between LabVIEW and the LED ring. In order to send the Boolean value for each row from LabVIEW to Arduino, an expansion for LabVIEW called LINX is used in order for LabVIEW to recognise the Arduino. Furthermore the library NI VISA's write command enables us to send a character from LabVIEW to Arduino. There the Serial library's read command

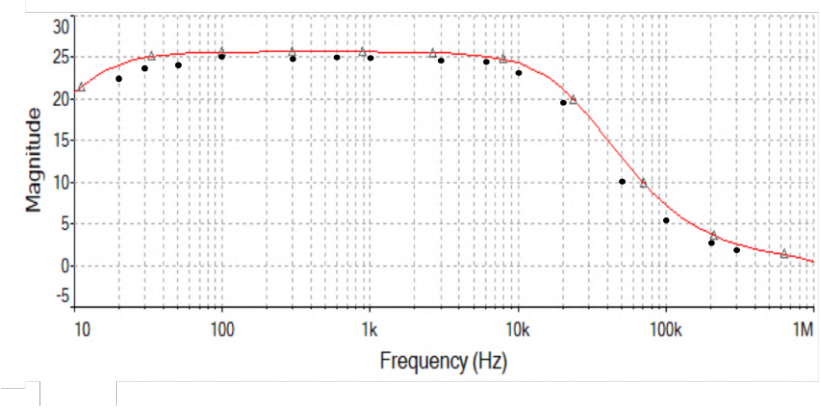

Figure 4: A Bode plot of the magnification from the amplifier circuit for frequencies 10 Hz to 1 MHz. The red line is the theoretical magnification calculated by Multisim and the black dots are the measured magnifications. The measured values are a bit lower than the theoretical values, but the measurements take the same form as the theoretical line.

allows the Arduino to read this character as appendix A shows. We chose "a" to present "true" and "b" to present "false". When all the rows are iterated through, a third character "c" is sent for the Arduino to display the array on the LED lights. A library from Adafruit, the producers of the LED ring, is downloaded and implemented in the Arduino. For each character sent, the "a"'s and "b"'s, a light is initialised corresponding to the frequency intervals, in other words the rows in our matrix.

Lastly the Arduino sends these values to the LED ring, which is the fourth part. The LED ring is soldered to three wires. One is ground, one is a 5 V DC voltage and the last one is the input data signals. The LED ring has RGB colours. In order to make it "disco-like", the LED lights follows RGB values of a colour wheel as in figure 6. The lowest frequencies are green. They then increase clock-wise all the way up to blue.

4. Results and discussion

Figure 7 presents an example of visualising audio frequencies on the LED ring. The LED lights shining show there are a frequency within their intervals with an amplitude greater than our set threshold. The frequencies seemed accurate, and blinked along to the beat of the music. If more lights were wanted the threshold could be lowered, and if only dominating frequencies were wanted the threshold could be raised.

The most time consuming task was definitely the communication between LabVIEW and Arduino. This was partially because the Arduino locks itself to LabVIEW once LabVIEW is running. Therefore the Serial monitor for Arduino becomes unavailable and knowing what is sent to the Arduino turns into a try and error in the dark. We also tried the LINX's custom command which allows a custom written Arduino command to be called from LabVIEW. This failed either because LINX don't operate well with the library from Adafruit, or because the Arduino Uno wasn't powerful enough. The error message in LabVIEW was "A timeout has occurred". The error happened first when we implemented the Adafruit library.

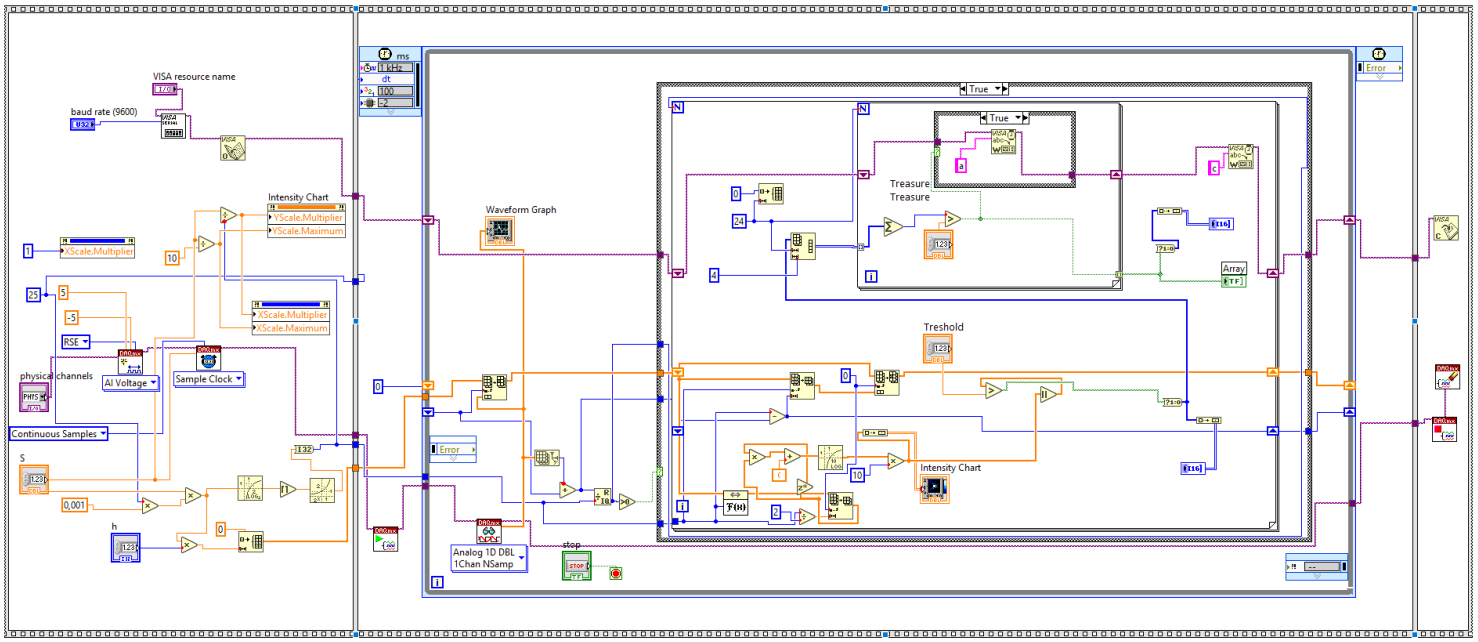

Figure 5: LabVIEW code for sampling an audio signal and displaying frequencies. Here the audio is sampled and stored in an array. Then the Fourier transform is performed, and the new array is multiplied by its complex conjugate. The formed amplitudes are tested against a threshold and with VISA's write function a Boolean character is sent to the Arduino for each amplitude depending on whether the amplitude is greater than the threshold.

Figure 6: The LED ring with all LED lights on. When all the LED lights shine together they make up the complete colour wheel. Each LED light has its own constant colour.

Before we came up with the solution based on NI VISA we made two separate units instead of one. This was done by displaying the LabVIEW results on the computer

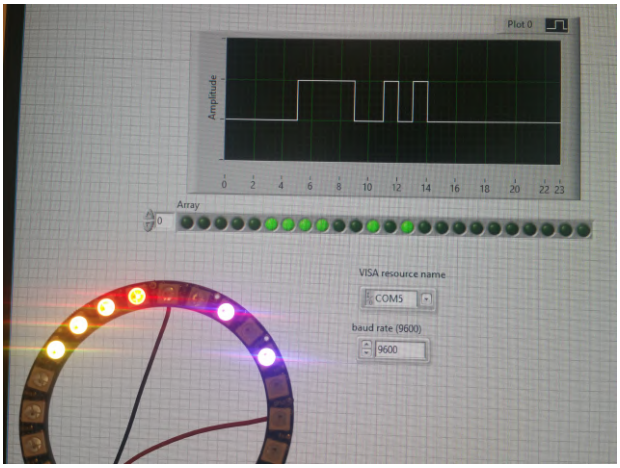

Figure 7: Both the LED ring and the monitor LED strip display the same result.

monitor with a LED strip, adding a Fourier transform to the Arduino as well and coupling both the Arduino and the NI DAQ to the microphone circuit. The LabVIEW results were more precise as the computer is more powerful, but the Arduino's LED ring was more elegant to observe. When we managed to combine the Arduino and LabVIEW as first planned the result turned out very pleasing. The calculations are done by the computer, and displayed on the LED ring. Other solutions could be finding a way to control the LED ring directly through the NI DAQ from LabVIEW, but we did not look into that. Also we do believe an Arduino Mega could replace LabVIEW and deliver decent results from the Fourier transform.

According to the Bode plot in figure 4 the magnification holds well for all frequencies in the human spectre. We expect little to none deviation due to the circuit, but the power supply however might be a source of noise. As long as the music is played loudly and the threshold is high the result on the ring holds up.

5. Conclusion

Combining Arduino and LabVIEW is perhaps not the easiest solution for making a disco light. However, it is completely doable and delivers great results. The Fourier transform computes quite precise audio to frequency amplitude conversion and the entire process happens quickly enough for fast beats to appear correctly. Combined the precision and rapidness were quite satisfying. We think our disco light could be extra useful for those who can't hear music anymore, but remembers the songs. Maybe the melody would come right back to them. Our recommendations for improving the experiment is to replace both the Arduino Uno and LabVIEW with an Arduino Mega, or finding a way to control the LED ring from LabVIEW's NI DAQ directly without an Arduino.

6. Bibliography

[1] Lab exercise 3 in the subject TFY 4190, Instrumentering, Department of Physics, NTNU, 2019.
[2] http://mathworld.wolfram.com/FastFourierTransform.html

Appendices

A. The Arduino code for controlling the LED ring

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif
#define led 13
#define lm A1
#define PIN 6
#define NUMPIXELS 24
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
char income;
int teller = 0;
void setup() {
	Serial.begin(9600);
	pinMode(led,OUTPUT);
	digitalWrite(led,LOW);
	pixels.begin();
}
int farger[24][3] = {{0,40,0},{5,35,0},{10, 30,0},{15,25,0},{20,20,0},{25,15,0},{30,10,0},{35,5,0},
{40,0,0},{35,0,5},{30,0,10},{25,0,15},{20,0,20},{15,0,25},{10,0,30},{5,0,35},{0,0,40},{0,5,35},
{0,10,30},{0,15,25},{0,20,20},{0,25,15},{0,30,10},{0,35,5}};
void loop() {
	if (Serial.available ()> 0) {
	income=Serial.read();
	if(income=='a') {
	pixels.setPixelColor(teller, pixels.Color(farger[teller][0],farger[teller][1], farger[teller][2]));
	teller++;
	}
	else if(income=='b') {
	pixels.setPixelColor(teller, pixels.Color(0,0,0));
	teller++;
	}
	else {
	pixels.show();
	teller = 0;
	}
	}
	delay(1);
}
```