ECE 477 Final Project
LED Visual Music EQ

Christian Knight, Nikko Noble

September 16, 2020

1 Introduction

This report describes the design and development of the hardware and software required to produce an LED light show that responds to musical audio input. The goal is to take line-level audio input from a laptop or phone and split the audio signal into four separate frequency bands heard in music and pulse LEDs corresponding to each frequency band based on the music intensity. The microcontroller used for this project is the STM32L476VG Discovery board, since it was used in ECE 486 Digital Signal Processing to do real-time digital filtering.

Section 2 covers the design of the digital filters required to separate the frequency content of the audio, Section 3 covers the hardware required to sample the audio signal and drive the LEDs, Section 4 covers the software that was developed to process the audio signal and control the LEDs, and Section 5 wraps up the report.

2 Filter Design

For real-time filtering, digital biquad filtering routines are used on the STM board. Figure 1 shows the structure of a single Direct Form II biquad section.

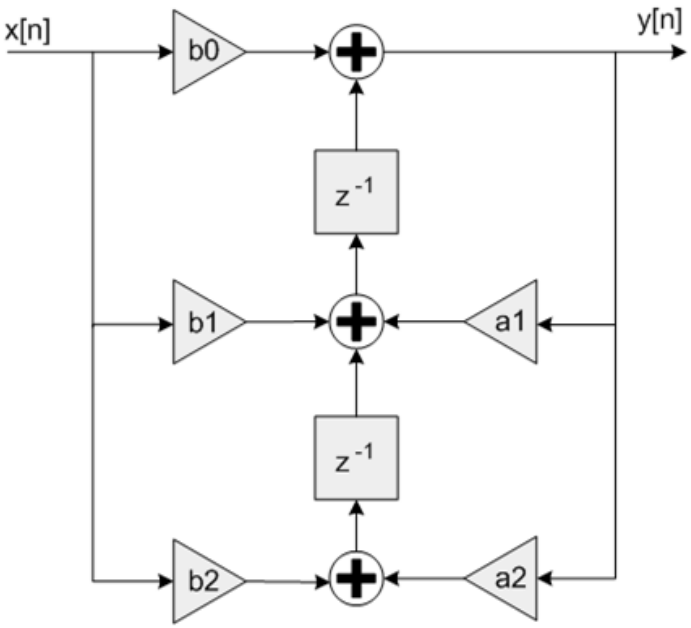

Figure 1: Single transposed Direct Form II Biquad structure

As shown in Figure 1, the structure works by holding the two previous samples (indicated by the z^{-1} blocks) and applying a gain to each branch (indicated by the triangular blocks) as each sample is processed. The filter gain coefficients b0, b1, b2, a1, and a2 were calculated using Matlab for each filter stage to get the desired frequency response. The following statements describe the algorithmic representation of the structure.

```
y[n] = b0 * x[n] + d1
d1 = b1 * x[n] + a1 * y[n] + d2
d2 = b2 * x[n] + a2 * y[n]
```

To implement the biquad filtering on the STM board, the ARM CMSIS-DSP library is used heavily as well as the ECE 486 sampling library to set up the board and collect input samples from the ADC. The ARM CMSIS-DSP library is a collection of DSP functions optimized for the ARM 32-bit architecture. Matlab's Filter Design and Analysis tool in the Signal Processing toolbox was used to generate useful filter coefficients for each stage that the ARM filtering functions could use. Table 1 shows the frequency range that each filter is designed to pass.

Table 1: Designed filter passband

Filter	Passband
Bass	0-60Hz
Low-mid	250-750Hz
Mid-high	2-2.8kHz
Treble	4.3-24kHz

Since the maximum frequency in music is about 20kHz, a sampling frequency of 48kHz was chosen. Figure 2 shows the frequency response of each designed filter from 0Hz to 5kHz.

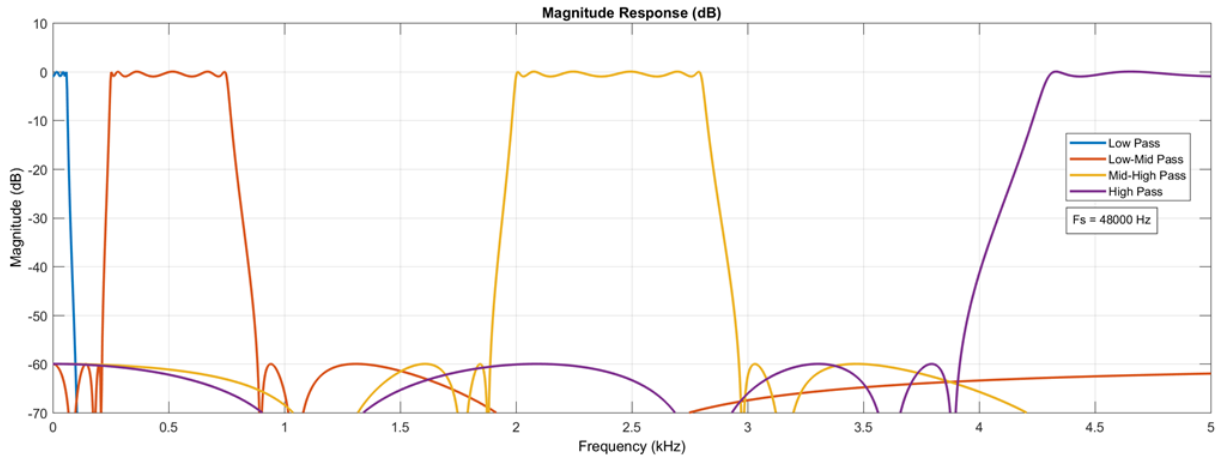

Figure 2: Frequency response of all filters

As shown in Figure 2, each filters' passband covers a different region of the audible spectrum; the low frequencies below 60Hz correspond to the thump of kick drums, the mid regions cover vocals and fill-in sounds, and the high frequencies correspond to fast-moving high pitched sounds.

3 Hardware

The hardware for this project is composed of the microcontroller, audio input conditioning circuitry, LED driving circuitry, and a power supply. The hardware schematic for this project is shown in Figure 3.

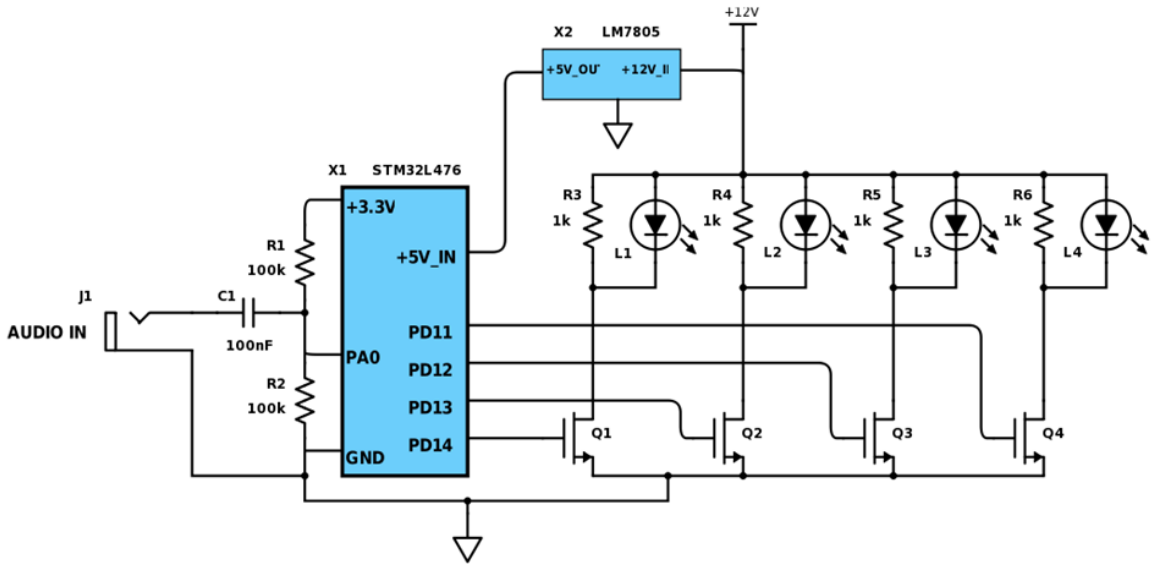

Figure 3: Hardware schematic for LED Visual Music EQ

On the far left of Figure 3 is an audio input from a 3.5mm headphone jack. This audio signal needs to be fed into the ADC of the STM board X1 on pin PA0 to sample the audio, but the signal has negative voltages which the ADC can't measure. Therefore, a DC offset is applied to the signal halfway between the board's supplies. The audio signal is AC coupled onto the DC voltage. Once the shifted audio signal is sampled and processed, pins PD11, PD12, PD13, and PD14 are used to drive the LEDs.

LEDs L1, L2, L3, and L4 are all individual RGB LED strips that are driven by tying the anode of each LED to +12V and pulling either the R, G, or B lines connected to the cathode of each LED low to turn on the respective color. The LEDs are turned on by driving an NMOS transistor with logic high from the board, which turns on the transistor, allowing the LED to conduct current. The strips have internal current limiting resistors. Lastly, the STM board is powered from a +5V supply provided by X2, an LM7805 linear voltage regulator that steps the +12V supply for the LEDs down to +5V.

4 Software

The software for this project is contained within the `LEDVisualEQ.c`, `LEDVisualEQ_init.c`, and `LEDVisualEQ_init.h` source code files; `LEDVisualEQ.c` holds the main program, `LEDVisualEQ_init.c` holds a supporting initialization function, and `LEDVisualEQ_init.h` holds some defined macros and coefficients for the filters. The flow diagram for the main program of this project is shown in Figure 4.

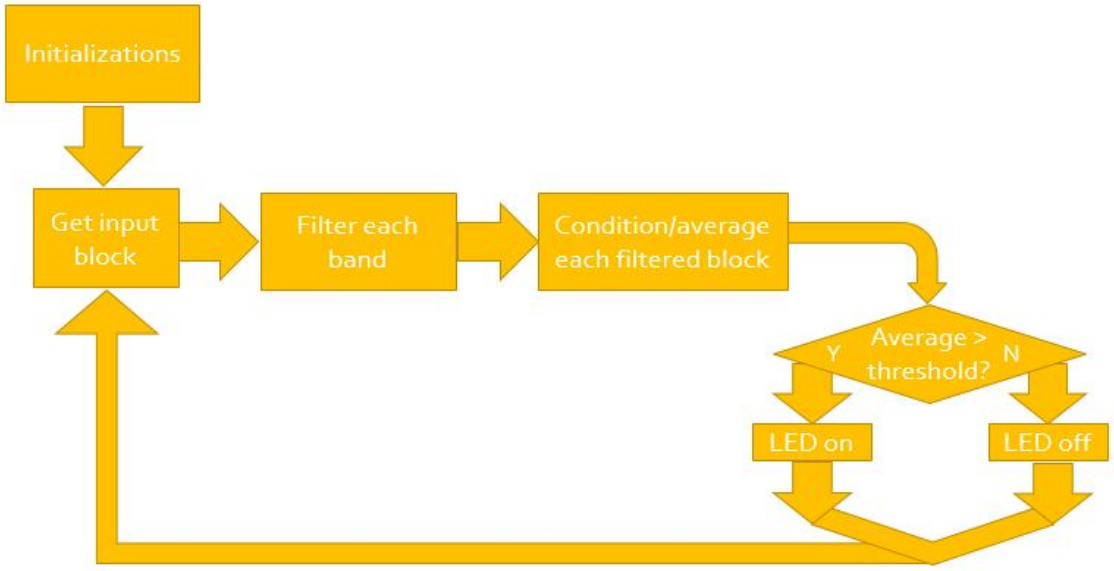

Figure 4: Software flow diagram for LED Visual Music EQ

From Figure 4, the first task the main program performs is the initialization block. This involves setting up the processor clocks and the ADC peripherals for sampling at 48kHz using the `initialize()` function in the ECE 486 sampling library. Also, the four GPIO lines that control the LEDs are initialized using the `gpio_init()` function developed for this project and all of the filter structures are initialized using the `arm_biquad_cascade_df2T_init_f32()` function in the ARM CMSIS-DSP library.

In the main loop of the program, which runs as long as the project has power, an input block of 20 samples is first acquired from the ADC using the `getblock()` function in the ECE 486 library. Then, the four filters are executed on the input block using the `arm_biquad_cascade_df2T_f32()` function. Following this, the filtered signals are manipulated to scale up the low input voltage, take the absolute value to prepare for averaging, and remove the DC offset applied to sample the full signal. This is accomplished using the `arm_scale_f32()`, `arm_abs_f32()`, and `arm_offset_f32()` functions respectively. The average value of the filtered signal is taken using the `arm_mean_f32()` function to gauge the intensity of the music. The program then compares the average with a set threshold to determine whether to turn the LEDs on or off. Since only 20 samples are collected every block, the loop is able to update the LEDs quickly enough to keep up with fast-paced music.

5 Conclusion

The development of the hardware and software required to produce an LED light show that responds to musical audio input has been described. The goal was to take line-level audio from a laptop or phone and split the audio signal into four separate frequency bands heard in

music and pulse an LED corresponding to each frequency band based on the music intensity. The microcontroller used for this project is the STM32L476VG Discovery board, which was used in ECE 486 Digital Signal Processing.

For real-time filtering, digital biquad filters were used on the STM board. To implement the biquad filtering on the STM board, the ARM CMSIS-DSP library was used heavily as well as the ECE 486 sampling library to set up the board and collect input samples from the ADC. The ARM CMSIS-DSP library is a collection of DSP functions optimized for the ARM 32-bit architecture. Matlab's Filter Design and Analysis tool in the Signal Processing toolbox was used to generate useful filter coefficients for each stage.

A Source Code

Attached: `LEDVisualEQ.c`, `LEDVisualEQ_init.c`, `LEDVisualEQ_init.h`, `makefile`

Sources

[1] Don Hummels. *ECE 486 Support Libraries: STM32L476G-Discovery support library for ECE 486 Digital Signal Processing*. May 5, 2017. URL: http://web.eece.maine.edu/~hummels/classes/ece486/docs/libece486_doc/html/index.html.

[2] ARM. *CMSIS DSP Software Library*. May 5, 2017. URL: http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html.