B31DD Mini Project Report - Conor O'Brien H00249212

1. Introduction

A basic audio spectrum analyzer built around an Arduino, an LCD display and a crude OpAmp audio amplifier.
The Arduino takes an amplified audio signal as an input to pin A0. In the main loop, this audio signal is sampled and a Fast Fourier Transform is computed, providing an array of complex Fourier samples, whose magnitudes are computed, scaled, trimmed and finally placed into 16 frequency "bins", whose contents are displayed on the LCD.

2. Design

2.1. Amplifier

In order to accept audio signals from devices such as a mobile phone or a computer, whose signal amplitude is typically +/-200mV, it is necessary to perform an amplification step. This is achieved by an amplifier circuit utilising an LM358 OpAmp in a non inverting configuration, which brings the signal to an amplitude of 0-5V, which can be read by the arduino's ADC. This amplifier is crude since it clips the negative half of the signal, which is acceptable for this application.

2.2. FFT

The Fast Fourier Transform algorithm that I used was implemented in the KissFFT library written by Mark Borgerding. After trying various libraries, I opted for KissFFT since it is small and simple, and crucially doesn't have many dependencies.
The result of the FFT algorithm is an array of complex Fourier samples, whose magnitudes are computed. We then have access to a Fourier signal representing magnitude over a bandwidth related to our sampling rate. The frequency components corresponding to the audible spectrum can be extracted by trimming the signal and ignoring the negative frequencies.

2.3. LCD

The magnitudes of the frequency samples are sorted into 16 frequency bins, which are mapped to the 16x2 LCD. Each frequency bin takes up two vertical characters on the LCD. The characters used to represent the magnitudes are custom characters written to the LCD's RAM. To write a custom character to the LCD's RAM, a bitmap of the character in the form of a 2D array of 1s and 0s is constructed, and each byte of this bitmap is written to an address in the LCD's memory, starting at offset 0x80. At this point the character is accessible just like any other character. I used the hd44780.h library since it allowed me to send arbitrary commands to the LCD.

2.4. Delays

Functions for both microsecond and millisecond delays were implemented. The microsecond delay function uses Timer0 with no prescaler to delay for 1us, and this is looped for as many times as the user specifies. The millisecond delay utilises the microsecond delay function, looping it 1000 times for each millisecond. The value of the offset in the TNCT0 register was calculated theoretically, and then tuned with the help of an oscilloscope to achieve a delay as accurate as possible.

2.5. Interrupts

A button attached to pin 2 of the arduino triggers external interrupt 0 when pin 2 is shorted to 5V. The associated interrupt routine modulates the magnitudes displayed on the LCD to simulate a volume button. Note that this does not change the volume of the amplified input signal, since that is all done in hardware. The interrupt routine also implements software debouncing using a delay.

3. Performance Evaluation

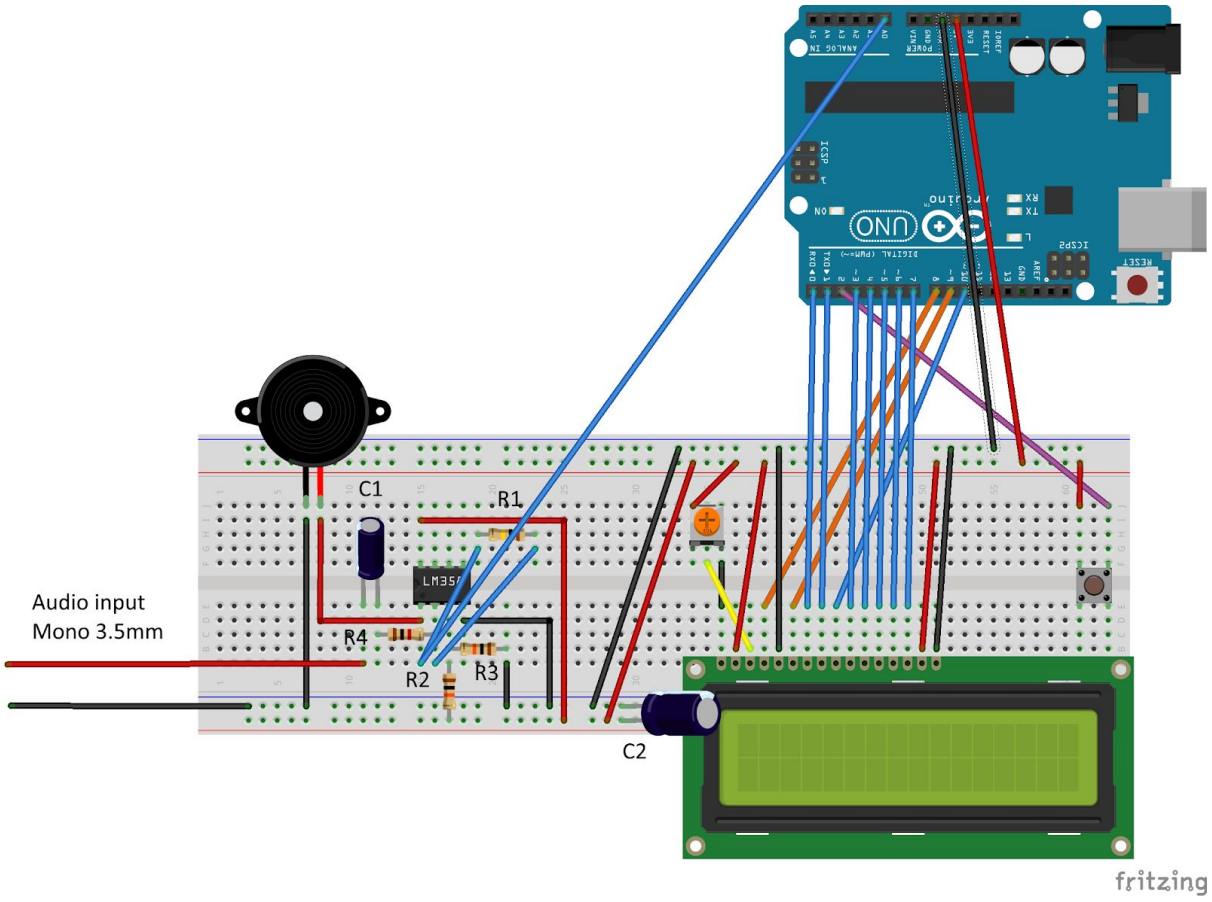
Although the system does its job as a simple audio spectrum visualiser, many of its features could be implemented in a better way. For instance, the Arduino is not suited for real-time signal processing, since the sampling rate is limited by the single-core and relatively slow processor, which for this application directly affects the bandwidth of the Fourier signal. Additionally, the size of the sample arrays is limited by the Arduino's small RAM, hence why I was only able to collect 64 audio samples at a time for the FFT. Despite this, the system is still able to compute Fourier components in a sufficiently quick and accurate manner for visualisation purposes.

The second shortcoming is the amplifier, based on a crude design which clips a significant amount of the audio signal and adds noise and distortion, although this is sufficient for visualisation. I suspect that the amplifier circuit also acts as a low pass filter, which affects the samples and the resulting spectrum displayed on the LCD.

The software debouncing of the button is unreliable and doesn't adhere to the best practice of spending as little time as possible in an interrupt routine. Hardware debouncing would have been a better alternative.

Display alternatives such as an LED matrix could have been used, but one of the advantages of the LCD is that it offers a compact way of displaying text and the spectrum on a single display.

4. Fritzing Diagram



C1 = 10uF
C2 = 100uF denoising cap
R1 = 1k
R2, R3 = 10k voltage divider resistors
R4 = 100k