Interrupt driven siren on a TI Stellaris LM4F120 Launchpad

Jeremy Koppenhaver

May 2018

Abstract

The goal of this project was to create an interrupt driven siren on the LM4F120 Stellaris Launchpad board. This project used hardware timers and interrupts to mimic 4 different siren sounds — Wail, Yelp, Phaser, and Horn.

Contents

1 Siren Sound Analysis

1.1 Introduction

Existing siren sounds were analyzed to get a general idea about the wave forms that need to be generated to create the different sounds. Sample sound files were taken from Whelen Engineering Company, Inc.[1] The analysis of the sound files was done using GNU Octave. The audio files were read into octave and the two stereo channels were combined into on mono channel for analysis. The first plot for each signal was an FFT that showed frequencies from 0 Hz to 5 kHz. This plot showed what frequencies were present in the audio sample. The second plot was a spectrogram that showed how the FFT changed over time. The spectrogram plots multiple FFTs at different times. The horizontal axis of the spectrogram is the time line in seconds. The vertical axis shows the frequency bands from 0 Hz to 2 kHz and the colors represent the intensity of each frequency at the given time. The third plot is a graph of the frequencies that have the maximum intensity at each point in time. This graph provides an approximate of the original sound that generated the audio clip. These three plots show the behavior of the signal in the sample audio clips and provide the base for what wave forms need to be generated to recreate the sounds. The main values that are important for recreating the first three sounds (Wail, Yelp, and Phaser) are minimum frequency, maximum frequency, rise time, and fall time. The horn sound is different than the other three because it does not change over time.

1.2 Wail

Figure 1 shows the FFT, spectrogram, and frequency vs time plot for the wail sample. The FFT shows that the main sounds start at around 710 Hz and ends around 1630 Hz. The frequency vs time plot shows the rise time to be about 1.9 seconds and the fall time to be about 3.1 seconds. These characteristics are summarized in table 1.

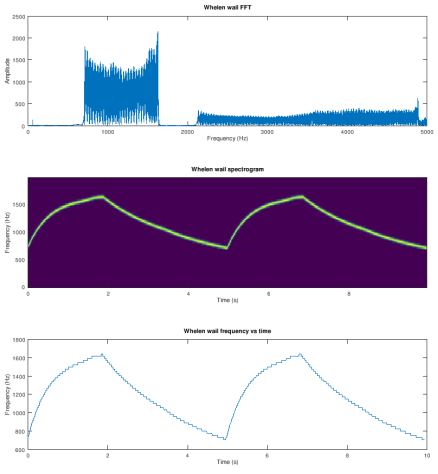


Figure 1: Analysis of the sample wail sound

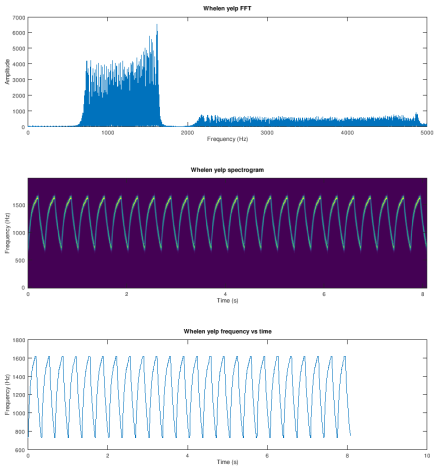
Min. Frequency (Hz)	710
Max. Frequency (Hz)	1630
Rise Time (sec)	1.9.
Fall Time (sec)	3.1

Table 1: Approximate Signal Characteristics for Wail.

[1] http://www.whelen.com/auto/sirentones.php

1.3 Yelp

Figure 2 shows the FFT, spectrogram, and frequency vs time plot for the yelp sample. The FFT shows that the main sounds start at around 730 Hz and ends around 1620 Hz. The frequency vs time plot shows the rise time to be about 0.18 seconds and the fall time to be about 0.16 seconds. These characteristics are summarized in table 2.



Min. Frequency (Hz)	730
Max. Frequency (Hz)	1620
Rise Time (sec)	0.18
Fall Time (sec)	0.16

Table 2: Approximate Signal Characteristics for Yelp.

Figure 2: Analysis of the sample wail sound

1.4 Phaser

Figure 3 shows the FFT, spectrogram, and frequency vs time plot for the yelp sample. The FFT shows that the main sounds start at around 750 Hz and ends around 1500 Hz. The frequency vs time plot shows the rise time to be about 0.038 seconds and the fall time to be about 0.038 seconds. These characteristics are summarized in table 3.

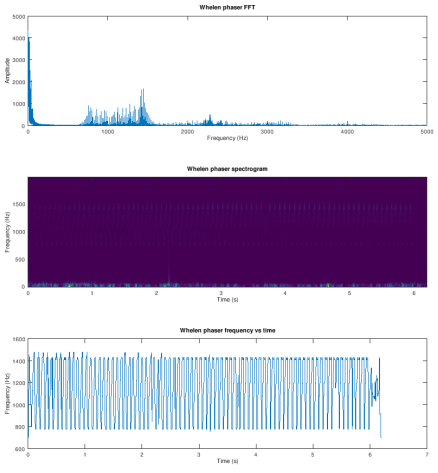


Figure 3: Analysis of the sample wail sound

Min. Frequency (Hz)	750
Max. Frequency (Hz)	1500
Rise Time (sec)	0.038
Fall Time (sec)	0.038

Table 3: Approximate Signal Characteristics for Phaser.

1.5 Horn

Figure 4 shows the FFT and spectrogram for the horn sample. The frequency vs time plot has been omitted because it did not show any useful information. Unlike the other signals, the frequency of the horn does not fluctuate. Therefore, the FFT is the most useful plot for this signal. The plots were limited to frequencies between 0 and 1 kHz. This is helps to isolate and identify the fundamental frequency. The FFT shows frequency spikes around 440 Hz, 660 Hz, and 880 Hz. The fundamental frequency appears to be 440 Hz, however, the interval between spikes appears to be 220 Hz. Due to the complexity of generating complex tones on a digital device, a single frequency was chosen for the horn implementation. 220 Hz was chosen because it was the interval between harmonics found in the original sample. Even though the frequency was not found in the original signal, it produces a similar sound to the sample. A sum of the intervals of without the fundamental frequency as demonstrated in equation 1 produces a much more accurate sound, however, is more complicated to generate. The implementation of the horn sound could be a point of further research and work.

$$f = \sum_{n=2}^{\infty} n f_0 \qquad f_0 = 220Hz \tag{1}$$

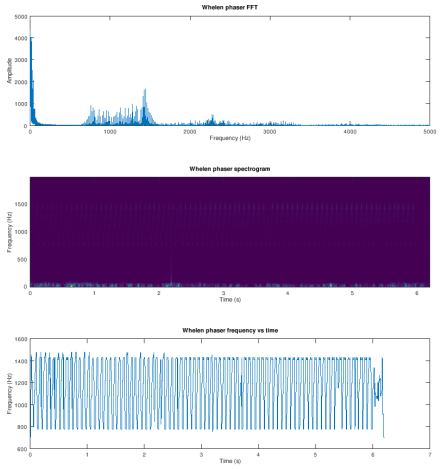


Figure 4: Analysis of the sample horn sound

1.6 Conclusion

Three of the four siren tones showed similarities in their frequency ranges. For simplicity of this project, one frequency range was picked to be used in all of these three sounds. This allows the frequency steps to be calculated once and the rise and fall times are the only variable characteristics. These sample sounds were only to get a starting point for the tones generated in this project. Rounded values were picked for the characteristics for this project based upon these samples. These rounded characteristics are found in Table 4.

	Wail	Yelp	Phaser
Min. Frequency (Hz)	700		
Max. Frequency (Hz)	1650		
Rise Time (sec)	2	0.18	0.04
Fall Time (sec)	3	0.18	0.04

Table 4: Rounded Signal Characteristics for this project.

2 Project Design

2.1 Requirements and Specifications

Siren Sounds

1. The system must be able to generate 4 siren sounds — Wail, Yelp, Phaser, and Horn

2. The variable siren sounds, Wail, Yelp, and Phaser, will all start at 700 Hz, rise to 1650 Hz, fall back to 700 Hz, and then repeat for as long as that mode is selected.

3. The constant siren sound, Horn, will generate a constant 220 Hz tone for as long as that mode is selected.

4. The Wail mode should be asymmetrical. It should take 2 seconds to rise and 3 seconds to fall.

5. The yelp mode should be symmetrical. It should take 0.18 seconds for both the rise and fall.

6. The phaser mode should be symmetrical. It should take 0.04 seconds for both the rise and fall.

7. The variable siren tones should rise and fall following a logarithmic curve.

Input/Output

1. The system shall take user input from two push buttons. These push buttons are active low.

2. The system shall output the siren tones as square waves on one GPIO pin.

Siren Tone Generation Design

The siren tones were generated by using a pulse width modulation(PWM) module on the processor. This allows the actual tone generation to be done in the background and not require processor time once it has been configured. The rise and fall of the siren is not linear. This posed a challenge because logarithmic calculations are not as fast as linear math on an embedded system. However, because all the variable siren sounds have the same beginning and end, a look-up table could be used. By calculating a number of intermediary frequency values between the start and end frequency, all the variable siren sounds could be generated using by changing the time interval between the points. This technique is explained in more detail in section 3.1.

Input/Output Design

The two push buttons on the Stellaris Launchpad board provided the user input for this project. A user interface that allows the selection of all 4 modes with two buttons was needed. The solution was for each button to have two possible commands. The user can either press or hold each button to move through the modes. A button press was defined as an activation of the button for less than 0.25 seconds. A button hold was defined as an activation of the button for greater than 0.25 seconds. Table 5 shows the functions of the two buttons.

	Press	**Hold**
Button 1	Turn on wail	Activate phaser until released
Button 2	If siren is on, switch between wail and yelp	Activate horn until released

Table 5: Behavior of the two buttons on the board.

3 Implementation

3.1 Siren Tones

Look-up Table

The frequency values were calculated and stored in memory on the Stellaris Launchpad. This allowed the more complex logarithmic equations to be run once and a full size computer instead of doing the calculation on the device every time it runs. An octave script was used to calculate the values and stored them in a header file. This header file was then imported into the main project.

Timers

Two timers were used to generate the siren tones. One timer was configured as a PWM timer. This timer toggled the output pin to create a square wave of a given frequency. The duty cycle is constant at 50%. The PWM frequency is changed to set the tone on the output pin. The PWM timer register is loaded with a value from the look-up table. The values in the look-up table are the number of clock cycles in one period of the output waveform. The second timer that was configured as a periodic timer. This timer triggers an interrupt when it times out and that interrupt is used to move to the next frequency in the look-up table. By changing the second timer value, the speed at which the siren tone oscillates can be set. This value is changed to alter the output siren mode. The rise and fall times for each mode are defined at the top of the generated look-up table. These values are the number of clock cycles in between each look-up value. This is based on the values determined in table 4, the number of frequency values in the look-up table, and the clock frequency. A pointer to a value in the look-up table is defined. This pointer is then incremented or decremented each time the second timer interrupts. The value of the pointer is then loaded into the PWM load register to change the output tone.

3.2 Input/Output

Input

The input for this project used the two buttons on the Stellaris Launchpad. The pins that the buttons connected to were configured as inputs with pull-up resistors attached. Therefore, a falling edge meant that a button had been pushed and a rising edge meant a button had been released. Each button had interrupts enabled for both rising and falling edges. Each button also had a 32 bit one shot timer associated with it. When the button was pressed, and a falling edge was generated, the processor enabled the one shot timer associated with that button and the timer started counting down. When the timer timed out, the current state of the button was read. If the button was high when the timer reached zero, that meant that the button had already been released. This signified a button press. If the button was still low when the timer reached zero, then that signified a button hold. The siren modes were changed accordingly. When the button is released, the button interrupt is triggered again. If the siren was in a hold mode when the button was released, then the siren needs to be placed back in whichever mode was active before the hold occurred. If the siren is not in a hold mode, then nothing needs to be done. All action for button presses is handled in the timer interrupt, not the button interrupt.

Output

The output for this project is on PB6. This pin was chosen because it is linked to the alternate function of the PWM timer and it is broken out on the headers. This pin can either be used to directly drive a small speaker or to drive the input of an audio amplifier if more volume is needed. The output waveform is a square wave with an amplitude of 3.3V.

4 Conclusion

Overall, this project turned out well and all of the project requirements were able to be met. There are some areas that could be improved upon such as the horn feature. This feature was simplified because it was a relatively small feature in the scope of this project. In the future, a more accurate horn could be produced by mixing multiple frequencies. Another area that could be improved is the user interface. For this project two push buttons were used to switch between all the modes. The two push buttons were all that was available on the Stellaris Launchpad board. However, this interface can be a little confusing. A much more intuitive interface could be created by adding more controls to the project. The goal of the project was to create a siren capable of producing multiple sounds and be entirely driven by interrupts and this was accomplished. Most of the sounds were accurately reproduced and the project is run by interrupts. Although there is always room for improvement, this project was a success.