Shivam Desai
Joseph Torres
Professor Wilken
05/30/19

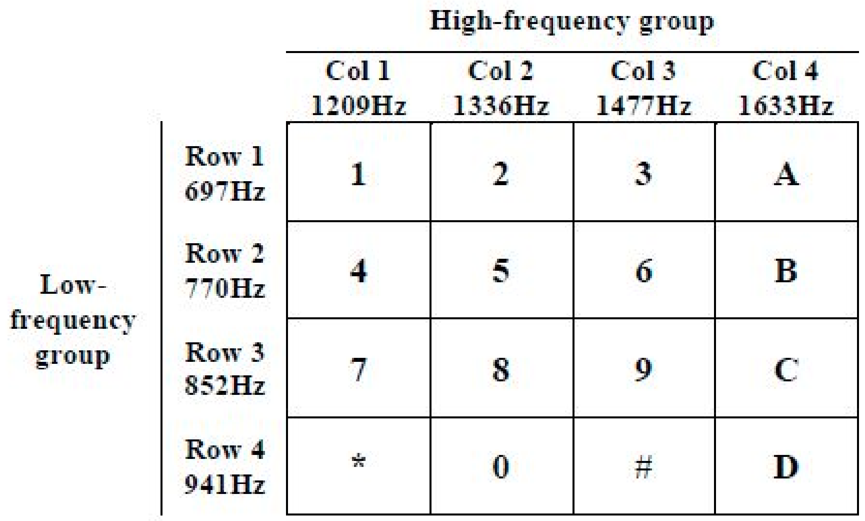
EEC 172: Lab 4 Report

Introduction

Something of a sibling to lab 3, the purpose of this lab is investigate the use of the CC3200 Launchpad's hardware interrupt capabilities in response to various triggers such as timers and UART receipts. In this case, however, the interface is audio, rather than infrared, based. That is, this lab permits the detection of dual tone multiple frequency (DTMF) waveforms via a microphone; in turn, the output is passed into an analog-to-digital converter, passed to the Launchpad by SPI, and decoded there using the Goertzel algorithm. By further interfacing the Launchpad with an OLED over SPI, we examine how such decodings can permit multi-tap text entry transcription and communication over UART.

High Level Description

On a DTMF-based keypad, each digit is produced by the superposition of two tones, or sinusoidal waveforms, according to the pairs possible wherein each each of the two tones is selected from one of two frequency groups. This allows for 16 possibilities, as shown:



		High-frequency group			
		Col 1 1209Hz	Col 2 1336Hz	Col 3 1477Hz	Col 4 1633Hz
Low-frequency group	Row 1 697Hz	1	2	3	A
	Row 2 770Hz	4	5	6	B
	Row 3 852Hz	7	8	9	C
	Row 4 941Hz	*	0	#	D

These tones have several special properties, including the property that no two of the frequencies are a multiple each other, no frequency can be derived from the sum or difference of the others, and all are in the range of human hearing. As the lab is specified multi-tap transcription, primary decoding interest was centered around digits 1-9, '*' for backspace, and '#' to send a message.

Similar to the FFT algorithm, the Goertzel can be used to take discrete samples from a function in the time domain and produce a function of of discrete samples in the frequency domain (determine its discrete Fourier transform). In our case, once a DTMF tone picked up by the microphone and converted into discrete samples by the ADC, the Goertzel is applied to produce a function whose highest peaks (power) represent the frequencies superimposed to generate the original tone. From here we can proceed in a similar manner as to Lab 3, with the microcontroller recognizing subsequent button presses of the same type within a certain time interval to facilitate multi-tap character production. Since both the OLED and ADC use SPI, care is taken to not interleave these functionalities.

Low Level Implementation

Our hardware setup made use of an Adafruit Electret Microphone Amplifier-MAX9814 for audio input and a MCP3001 10-bit ADC with SPI Serial Interface. Because the microphone requires an extremely clean power supply and the OLED is known to produce noise on the power line, the OLED received power separately from the microphone and ADC. Namely, the OLED received power from the 5V pin on the Launchpad, reduced through a LM1086CT voltage regulator down to 3.3V; the microphone and ADC pulled power from the 3.3V pin. The LM1086CT also featured two 10 μF capacitors between the input and ground and the output and ground.

As encouraged in its specification, our lab builds upon the implementation found at https://github.com/OmaymaS/DTMF-Detection-Goertzel-Algorithm-. Unlike in that code, we use a 16kHz sampling rate, a 400,000 bps SPI rate, and we set our blocks to contain 410 samples. To eliminate some calculation from our program, the coefficients used in the algorithm were hard coded as $2cos(2\pi f_i/16000)$, where f_i is each DTMF frequency and 16000 our sampling rate. Fixed point arithmetic (representations) were used to achieve improved performance from the microcontroller; thus, it was necessary to left shift our raw values by 14 to avoid floating points.

The operation of our code is as follows. The Launchpad continuously samples in audio input at 16kHz until it fills its buffer of size 410. This data is read in big endian byte order and the dc bias removed. At this point, sampling is stopped and the goertzel procedure called on the samples to calculate powers. The post_test function then settles on which which frequency from each of the two DTMF frequency groups is most strongly indicated by the powers and the corresponding button that would produce them.

The detected button is considered valid and returned only when one row and one column frequency exceed a certain threshold, otherwise, ‘x’ is returned. Assuming a valid detection, two cases are possible. If the button returned is not the same as the previous button encountered, a 1.75 second timer is started, a counter set to 0 and the findText() function called. The findText() function keeps track of those buttons having three versus four (7,9) possibilities, as well as which character is to be selected using the number of identical button presses as an array index. If the button returned is the same as the previous, the preexisting counter incremented. In both cases, printCurChar() is used to write to the OLED.

For responsiveness, individual characters are drawn to the OLED with each button press, and when the same button is pressed, the next character in that button’s possible character array overwrites the previous one. In the background, a buffer of the current and all previously selected characters is retained for use in UART; when the user presses “#” to send the message, the entire buffer is copied over to the appropriate location on the specified board’s OLED, as per the UART interrupt handler. Similarly, when the user presses “*” to delete a character, the entire buffer is reprinted to the OLED, with the final character simply set to null. Whenever an OLED operation completes, sampling resumes.

Discussion of Issues and Solutions

Thanks to the many similarities that this lab had with the previous lab, we ran into many fewer issues. We did have trouble selecting an appropriate threshold that would yield reliable results until we settled on 150000. Before realizing that we needed to flush out invalid post_test() return values (instances in which the returned character was ‘x’), we could not get acceptable results. Attempts to determine the timing between consecutive decodings was attempted, but abandoned. As in Lab 3, our TA found it sufficient that we should be able to complete intra-board message passing over UART instead of demonstrating it between both boards, so no attempt to that end was made.

Conclusion

In this lab, we explored the CC3200 Launchpad’s hardware interrupt capabilities in response to various triggers such as timers and UART receipts. We implemented an audio interface featuring an analog to digital converter over SPI to listen for DTMF tones. We use the Goertzel algorithm to take the discrete fourier transform of the function to decompose the superposition into its constituent frequencies. We deduced a button press from this, and realized a multi-tap text entry implementation similar to the previous lab. We printed the resulting characters to an OLED over SPI, and then passed these as messages over UART.