EEC 172 Lab 4 Report

Alejandro Torres
almtorres@ucdavis.edu

ABSTRACT

In this paper, I will discuss how I interfaced a microphone amplifier with a DTMF tone generator app to output text. I will then go into detail about how I interfaced the CC3200 LaunchPad with an OLED screen to display the messages.

I also discuss my learnings when attempting to connect two CC3200 LaunchPad's via asynchronous serial communication (UART).

General Terms

UART means Universal Asynchronous Receiver Transmitter. OLED means Color Organic Light-emitting Diode. DTMF means Dual Tone Multi Frequency. ADC means Analog Digital Converter.

1. INTRODUCTION

This laboratory experiment provided a valuable opportunity for me to thoroughly analyze the transmission of the DTMF tone generator from a smartphone. My primary objective was to discern the data format of the signals generated when specific buttons were pressed, as they each produced 2 frequencies (low and high frequency).

Once the decoding process was successfully accomplished, I proceeded to establish a connection between the CC3200 LaunchPad and an OLED display. This involved the implementation of relevant functions that facilitated the real-time composition and display of text messages on the upper portion of the OLED screen. In addition, I endeavored to establish a link between two CC3200 LaunchPads via the UART1 channel.

2. Procedures for Part 1

In the initial phase of the laboratory experiment, my focus was on characterizing the DTMF signals by the power array which was produced by the Goertzel algorithm. To achieve this, I began by constructing the necessary circuit for the microphone on a breadboard. My objective was to establish a setup that allowed the microphone to effectively receive signals.

2.1 Decoding DTMF Signals

In order to decode the DTMF signals, it was necessary to implement the circuit. The microphone was connected, and its output was linked to the ADC. Subsequently, the digital signal from the ADC was routed to the CC3200 LaunchPad for processing.

Subsequent to this, a generic timer handler was created to extract data from the ADC at a rate of 16KHz. The data obtained from the ADC was then stored in an array buffer with a size of 410. In the main function, a condition was added to the while loop to disable the timer interrupt when the buffer reached its maximum capacity.

Once the buffer was filled, I could proceed with the analysis of all frequencies. To achieve this, I calculated 7 coefficients for each targeted frequency associated with a specific button. These coefficients were then utilized as inputs for the Goertzel algorithm, which generated a power array. The power array consisted of magnitude values corresponding to each calculated coefficient. Subsequently, the power array was parsed to identify the two highest magnitudes, representing the low and high tone frequencies associated with the respective button.

Finally, a switch case was implemented to determine which button had been pressed. Upon identifying the pressed button, the timer interrupts were re-enabled to detect future button presses.

2.2 Texting

To integrate the text messaging, I took the decoded value of the DTMF signal and associated it with the appropriate button. Depending on which button was pressed, it would either select a character, erase a character, or print and send a character/string.

My implementation for selecting a character involves calling a get letter function when the user selects any number from 0-9, excluding 1. When a letter was selected, the function would be called, and the button that had been pressed would be evaluated by the switch statement. The cases within the switch statement determined which letter to be set. It determines which letter to set by analyzing the previous letter.

When the current letter was chosen, it was not actually selected until the user pressed a different button. This was my way of confirming that the current letter was the one I actually wanted to select. For example, if I wanted to access the letter 'c', I would press the button '2' three times (to reach 'c') and then press the button '3' to confirm the current letter. After this was determined, the letter was stored into a string and printed onto the board.

The send message function was constructed by calling the UART functions, storing the characters into the buffer, and calling the print message function. The message string within the program was then cleared so the next message sent would not hold the previous characters. Send is only called when the '#' button is pressed.

The print message function outputted the characters with a distance of 7 pixels between the start of each character, so they wouldn't overlap each other. When the end of the row was reached, I moved the x-axis back to 0 and moved the y-axis down by 7 pixels. Each time a character was selected, it was printed to the board, even if you are iterating through the same button. For example, pressing button '2' multiple times would show 'a', 'b', 'c', and again until a different button was pressed.

Erasing a character was only called when the button '*' was pressed. This triggered the delete function, which retrieved

the message string and deleted the most recent character by setting its value to null.

The post-test function created a representation of the DTMF buttons by mapping them to a 2-D array. The max frequency was determined by narrowing the max value in each row and column. When a button was pressed, the function then determined whether to print, delete, or send.

3. Problems and Solutions

I will discuss the problems I encountered during the lab and how they were resolved..

3.1 Decoding DTMF Signals

The primary challenge encountered during the decoding of DTMF signals was gaining a comprehensive understanding of the Goertzel algorithm and determining the appropriate thresholds for extracting values from the power array. Once I had grasped the functioning of the algorithm, I was able to proficiently implement my frequency analysis.

Likewise, I established a threshold to mitigate the impact of background noise. Fine-tuning the threshold was essential because there were instances when a value was returned even though no button was pressed. However, if the threshold was set too high, it would impede consistent button detection. After identifying the optimal threshold for our laboratory environment, I achieved consistent results.

3.2 Texting

I had issues when selecting the characters, where it seemed that the microphone would not receive the correct button information. This was assumed due to the issue occurring in a previous lab; however, after further troubleshooting, it was determined that the signals were being received properly. The issue was that the incorrect letters would be selected in our GetLetter function. Another reason for the issue was that the printing of the message on the screen was accidentally replacing the previous character. These issues were eventually resolved by making minor tweaks to the respective functions.