DTMF Texting Over an Asynchronous Serial Link

Laboratory 4 - EEC 172
Kendall Lui and Mary Florek
2/27/18

INTRODUCTION

In recent years it has become more and more common to see audio inputs used in embedded system designs. Thus, it is becoming important to learn how to decode and characterize audio signals. A common audio input is the dual-tone multi-frequency (DTMF) signal, which generates a different tone for the different buttons pressed on a receiver. In this lab, we explored ways to integrate these tones into the texting system developed in Lab 3.

GOALS

The goal of this lab was to develop a system that would take a DTMF input from a microphone amplifier, convert the signal from analog to digital using an ADC converter, and then decode it into characters that would be displayed on the OLED screen. This task would involve using many concepts that were developed in Lab 3, such as polling, interrupts, and timers. In addition, we were to develop new skills including interfacing the SPI protocol to communicate with an external ADC device, and decoding a signal received by a microphone amplifier.

METHODS

The first step in implementing this system was to set up all the connections between the external components. The OLED was connected in the same way it was in Lab 3, as its function remained the same. The new components we needed to connect in this lab was the microphone, ADC and voltage regulator.

The microphone we used had a very delicate output signal that was easily corrupted by noise on the power line. To combat this issue, we needed to attach the microphone to the cleanest possible voltage signal. So we connected the microphone directly to the 3.3V source on the board and moved the OLED power source to the 5V pin. However, the OLED requires only 3.3V, so we had to attach a voltage regulator. This brought the 5V supply down to the necessary 3.3V for powering the OLED. The OLED and the voltage regulator were grounded to one Launchpad ground pin, and the microphone and ADC were connected to a different ground pin. This again, was to help mitigate corruption to the microphone.

Connecting the rest of the microphone's pins was relatively simple. We wanted the output gain to be 50dB to reduce noise on the output signal, so we tied the gain pin to ground. The A/R pin (Attack and Release Ratio) is used to control the ratio between the time it takes the microphone to turn on (fading in) and the time it takes to turn off (fading out). A release time of 120 ms is generally the best option for a DTMF system, so to ensure this ratio, we tied the A/R pin to ground.

The final step of the hardware setup for this project was connecting the analog/digital converter. The ADC would take an input from the output of the microphone, convert the signal from analog to digital and then send the digital output to a GPIO pin for decoding. This GPIO pin was chosen to be separate from the GPIO used by the OLED so that the two devices could be accessed separately.

After the hardware setup was complete, the next step was to implement the software. In Lab 3 we implemented two separate header and source files one was designed to implement the IR remote typing and the other was designed for the UI and message communication. This modularization of our code reduced the amount of code needed for this Lab we simply needed to implement a new typing method and link it to the previously implemented text messenger code.

In order to implement the typing portion we needed to understand how DTMF keypads work. There are two groups of frequencies (697 Hz, 770 Hz, 852 Hz, 941 Hz)

corresponding to a row and (1209 Hz, 1336 Hz, 1477 Hz, 1633 Hz) corresponding to a column on the keypad. Each key then makes two frequencies when pressed. By determining which dominating frequencies are present we can lookup the appropriate number that was pressed.

Using the ADC we use an interrupt to sample the microphone at 16 kHz. We setup a timer that called a sampling function which simply took in an ADC value. We kept this in a buffer with a length of 410 samples. To do this we used SPI to request data from the ADC. The ADC is a 10 bit ADC. This meant we needed to read from the SPI twice since the SPI was setup to read only 8 bits.

After sampling the buffer was processed using the Goertzel algorithm to detect the if one of the row frequencies was present. The Goertzel algorithm was implemented according to the a source found on embedded.com[1]. We precalculated coefficients for each frequency and kept them in an array called freqCoeffTable. This reduced the actual calculations that we needed.

Upon completion of sampling, timer interrupts for sampling were disabled. The goertzel was run on the buffer for each row frequency. To determine whether a frequency was in the sample we used a simple threshold test based on each desired frequency. If the frequency was present we continued to determine if a column frequency was present. This reduced calculation time. We then followed the same procedure for Lab 3 by creating a timer that determined if the button was pressed multiple times during a certain time interval to change the letter. The characters were then placed in a string in the TextManager with enter being used for the sendmessage(). We used the '*' character for send and '#' character for delete.

DISCUSSION

We ran into a problem wiring the ADC correctly, it turned out that the ADC was fried and upon receiving a new ADC the microphone worked. We then ran into problems with reading the ADC. After reading the datasheets and looking at the section 5 we realized we needed to do bit shift operators on the input data since some of the bits were garbage bits.

The last hurdle was implementing the Goertzel function. We used the Matlab implementation of the Goertzel function to understand how to code our own Goertzel function. We then compared the data from the Matlab implementation and ours to verify that it was working properly. This then needed to be converted into C code which was easier than expected.

CONCLUSION

This lab was an interesting and challenging continuation of all the skills learned in Lab 3.

In addition we learned a lot of new skills through decoding audio signals to represent the buttons pressed, and then using this information to send messages between the two Launchpads. In the end we successfully managed to complete our goal to implement a system that would send text messages based on the DTMF input from a microphone.

---

1 https://www.embedded.com/design/configurable-systems/4024443/The-Goertzel-Algorithm