EE479 Project 1

Aydın Uzun
2015401210

I. Report

DTMF keypads are laid out on a 4x4 matrix, in which each row represents low frequency and each column represents high frequency. With DTMF, each key pressed on a phone generates two tones of specific frequencies. One tone is generated from a high-frequency group of tones, while the other is from a low-frequency group. DTMF signalling was first developed and widely used for telecommunication signalling between telephone handsets and switching centers over analog telephone lines.

The DTMF signal I generated with a *.wav* file gained in the beginning a DC-offset on breadboard through a simple circuit consisting of two resistors and a capacitor. After that the signal is transferred to *MATLAB* via *analogread* and *serial port*. In *MATLAB* I did the neccessary processing. First issue was to determine sampling rate of the signal I received on *MATLAB*. Because my filter design will depend on this parameter highly. As already mentioned by you one should expect a sampling rate smaller than 10 kHz. Because the other commands in the loop function reduces the overall sampling rate of the signal. I sent a 10 Hz sinusoidal signal to my device and investigated the signal on Matlab inside *data* array. Doing this I calculated my sampling rate as 8815 Hz.

The next step was to determine which type of filter I am going to use in my design. I choosed Butterworth bandpass filters, because the frequency response of the butterworth filter has no ripples in the passband and it's a maximally flat filter. So it's very good at simulating the passband of an ideal filter. I created the filters on *MATLAB* with the help of this reference [1].

The next challenging thing was to determine the order and the bandpass length of the butterworth filter. It's known that the higher order the Butterworth filter is designed and implemented, the flatter frequency response we get inside bandpass interval. I choosed empirically 60th order bandpass Butterworth filter. Now I should decide the cutoff frequencies of the filters. All of the filters should have the same bandpass length, because later I am going to compare their energies inside this decided banpass. If the bandpass lengths of different filters will be different, my comparision is not valid. To determine the right bandpass length one should examine the encoding frequency values of the different characters. The minimum difference is between the frequencies 697 Hz and 770 Hz and it is 73 Hz. So one should expect that the maximum bandpass length should be definitely smaller than 73 to prevent overlapping of working frequencies of filters. I choosed empirically 43 Hz bandpass length.

After the filters are ready, using the given code and choosing buffer size 1000, the signals coming from Arduino are read into an array called *data()*. This data array has a DC component. Because the input signal is fed to a voltage divider circuit and gained 2.5 V DC component. Before the data array filtered, the DC component should be removed, I did this easily subtracting the mean of the data array from itself. Using the designed 8 bandpass Butterworth filters to detect distinct frequencies the data array is filtered and outputs are now to investigate. The energy values of each filters are calculated and maximum output value of the filters, whose center frequencies are 1209, 1336, 1477 and 1633 Hz , is assigned to a variable. The same has done to filters with center frequencies 697 , 770 , 852 and 941. One should check this maximum output values to determine if the given signal is a noise or not. If it's a noise nothing should be printed onto screen. If it's not a noise, in other words the maximum values are bigger than 0.05, these maximum values should be investigated. For example, it these two maximum values of outputs of the filters (one is from filters with center frequencies 1209 or 1336 or 1477 or 1633 Hz, the other is from filters with center frequencies 697 or 770 or 852 or 941 Hz) belongs to the filters with center frequency 1209 Hz and 697 Hz, a *1* should be printed on the command window. All the possibilities should be checked and corresponding caharacters and numbers should be printed onto command window. In addition to that, when printing onto command window it should also be checked whether it's a repeated signal or not. There should be a blank space in other words noise between repeated signals to print both of them. This is all the algorithm. In the end the port should be closed inherently.

II. References

[1] "fdesign," Reconstructing an Image from Projection Data - MATLAB & Simulink Example. [Online]. Available: https://www.mathworks.com/help/dsp/ref/fdesign.bandpass.html. [Accessed: 22-Nov-2018].