Sefa Kayraklık
2015401066

Project 3

EE 479
Fall 2020

I. Introduction

The Dual-Tone Multi-Frequency (DTMF) signalling is a telecommunication signaling system using the voice-frequency band over telephone lines between telephone equipment and other communications devices and switching centers. DTMF system consists of encoding and decoding 16 different symbols (10 digits, 4 letters and 2 signs) with 8 distinct sinusoidal signals (4 low-frequencies and 4 high-frequencies). Each symbol is specified with a composition of one low-frequency and one high-frequency sinusoidal signal. The table I shows the symbol-to-dual mapping of DTMF.

	1209 Hz	**1336 Hz**	**1477 Hz**	**1633 Hz**
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

TABLE I
THE SYMBOL-TO-DUAL TONE MAPPING OF DTMF

The aim of the project is to design an arduino setup which receives the transmitted DTMF signals by the computer and to detect the symbols via using digital filtering in MATLAB. Thus, the project contains three different parts such as generating the DTMF signals, receiving the signals through the arduino's analog input, and decoding and displaying the symbols. The details of the parts are provided in the following section.

II. System Design

A. Transmitting of DTMF Signals

The DTMF signals are generated by a website, http://onlinetonegenerator.com/dtmf.html, and transmitted to the arduino's analog input via an aux cable from the computer's sound output.

B. Receiving DTMF Signals

As discussed in the previous projects, a DC bias should be added the signal before receiving; therefore, the same circuit setup as before is used to add a DC bias of 2.5V. Hence, the Fig. 1 shows the arduino setup which receives the signal to process further. The setup is composed of three distinct pieces, such as transmission via aux cable, DC biasing through the circuit which is used previous projects, and reception via the analog input of the arduino.

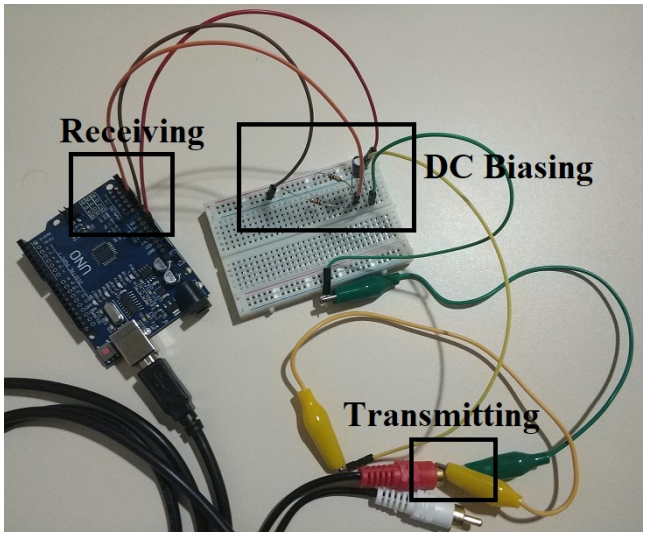

Fig. 1. The arduino setup

The following arduino code is used to read the DTMF signal from one of the analog pins of arduino board and write it to the serial interface.

```
int sensorValue = 0 ;
void setup ( ) {
Serial.begin(115200); } // serial communication
    hizini belirleme
void loop ( ) {
sensorValue = analogRead(A0) ; // A0 portundan
    sample alma
sensorValue = sensorValue / 4 ; // 10 bitlik
    sample iprecision kaybi yaparak 8 bit e
    cevirimis oluyoruz
Serial.write(sensorValue); } // serial port dan
    PC ye yollama
```

C. Decoding and Displaying DTMF Symbols

The received DTMF signals are processed in MATLAB to decode and to display the sent symbols. The MATLAB code consists of one main script and three functions, *receive_decode_display_DTMF*, *decode_DTMF* and *print_output*.

The main script calls the first function with the user defined parameters of *port name* of the arduino, *buffer size* of the received data, *sampling rate* of the arduino [1], and the *duration* of running. The following MATLAB code shows the main script.

```
port_name = 'COM3';
buffer_size = 1000;
Fs = 8872;
duration = 100; % how many seconds to run
receive_decode_display_DTMF(port_name,buffer_size
    , Fs, duration);
```

[1]It is set to the same value as in the previous project

The MATLAB function of *receive_decode_display_DTMF* takes the user defined parameters and runs the amount of seconds which the user gives as *duration*. The DTMF signals are received as stacks of *buffer size* of bits. After receiving the signals, it passes the signals, while removing the DC biasing, and the *sampling rate* to the second function to decode the DTMF symbols. After obtaining the symbols, it gives the current symbol and the previous symbol to the third function to display the result on command window. The following MATLAB code shows the function.

```
function [] = receive_decode_display_DTMF(
    port_name,buffer_size, Fs, duration)
%receive_decode_DTMF Summary of this function
    goes here
%  Detailed explanation goes here

priorports=instrfind; % halihazirdaki acik
    portlari bulma
delete(priorports); % bu acik port lari kapama (
    yoksa hata verir)
s = serial(port_name); % bilgisayarinizda hangi
    port olarak define edildiyse,
% o portu girin . . COM1, COM2, vs . .

s.InputBufferSize = buffer_size; % serial
    protokolunden oturu , datayi bloklar halinde
% almaniz gerekiyor. kacar bytelik bloklar
    halinde almak istiyorsaniz, onu girin
set(s, 'BaudRate', 115200) ; % arduino da set
    ettigimiz hiz ile ayni olmali
fopen(s) ; % COM portunu acma


output = nan;
time0 = tic;
while toc(time0) < duration
   temp_output = output;
   data = fread(s);
   output = decode_DTMF(data-125,Fs);
   print_output(temp_output, output);
end
fclose(s); % Serial port u kapatmak
fprintf('\n Time of %d seconds is up. \n',
    duration);
end
```

The MATLAB function of *decode_DTMF* takes the received signals and sampling rate, applies digital filters to detect the sent symbols, and returns the symbols as an output.

The function uses an infinite impulse response (IIR) type band pass filter which is designed by using butterworth method. An IIR filter is selected in the project since it can provide the same performance with a much lower filter order in comparison to a finite impulse response filter. Thus, the required time to apply filtering gets much shorter. If the order of the band pass filters is set to higher values, the process time of the code will increase; so, the order is set to a fairly small number, 6.

The cut-off frequencies of the band pass filters are determined as the middle points of the given frequencies, which are stored in the variables of *low_freq* and *high_freq*.

In order to decide whether there exists a signal in the given band of frequencies, the function compares the average of summation of absolute values of the filtered signal. The threshold for the comparison is chosen by inspecting the results. Therefore, for the low frequencies, the threshold is selected 1, and for the high frequencies, it is selected 2.

In order to illustrate the designed digital band pass filter, the frequency domain representation of the filter which is used for the detection of the signal whose frequency is $1336Hz$ is shown in the Fig. 2.

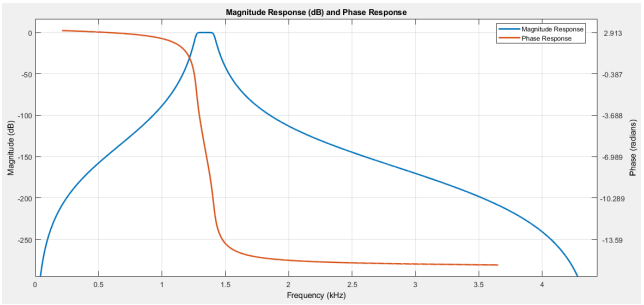

Fig. 2. The magnitude and phase response of the filter

The function sets the output as NaN for a default value. Then, it first decides the rows of the symbols by only filtering the low frequencies of the signals. After determining the row, it detects the symbol by filtering the high frequencies.

The following MATLAB code shows the function.

```
function [output] = decode_DTMF(data,Fs)
%decode_DTMF Summary of this function goes here
%  Detailed explanation goes here
output = nan;

bs = length(data);
low_freq = [650 730 810 900 980];
high_freq = [1130 1270 1410 1550 1690];
temp_l = ['1','2','3','A';'4','5','6','B';'7','8'
    ,'9','C';'*','0','#','D'];

for i=1:4
   [b, a] = butter(6, low_freq(i:i+1)/(Fs/2));
   if sum(abs(filter(b, a, data)))/bs>1
      temp_h = temp_l(i,:);
      break
   end
end

for i=1:4
   [b, a] = butter(6, high_freq(i:i+1)/(Fs/2));
   if sum(abs(filter(b, a, data)))/bs>2
      output = temp_h(i);
      break
   end
end

end
```

The MATLAB function of *print_output* takes the previous and current values of the DTMF symbol and prompt the current symbol on the command window. It first checks whether the symbol is NaN or not; if the symbol is set to the default value, it does not display. If not, then it checks whether the symbol is the same as the previous value within a period of time; if they are different from each other, then it prompts the current symbol. The second control is done to avoid prompting multiple times for a long press.

The following MATLAB code shows the function.

```
function [] = print_output(temp_output, output)
%print_output Summary of this function goes here
%  Detailed explanation goes here

if ~isnan(output)
   if ~isequal(temp_output,output)
      fprintf('%c', output)
   end
end

end
```

III. Results and Conclusion

A DTMF system is designed by using an online tool to generate the signal, an arduino to receive the generated signals and MATLAB scripts to decode and display the sent DTMF symbols. In the decoding process, band pass digital filters are employed.

The constructed DTMF system can comply the performance requirements which are stated in the project description. Namely, the system give the correct symbols with 100% accuracy rate within an acceptable time frame. However, the speed of the algorithm can be improved by lowering the order of the filters yet this can cause a decrease in accuracy.

In the first attempt, the decoding algorithm was built with the low pass filters by increasing its cut-off frequency and checked whether there exists a signal in certain frequencies. However, this approach did not work as expected. Hence, the low pass filtering method is changed to the band pass filtering and this solves the problem.

Appendix

The MATLAB codes which are used in this project is in the *EE479Project3(Sefa Kayraklık).zip* file. The content of file is following m, files and the reports:

- Main.m, receive_decode_display_DTMF.m, decode_DTMF.m, and print_output.m for the MATLAB part
- sketch_nov11a.ino for the arduino part
- EE479Project3.pdf for the reports