EE 479

COMMUNICATION LABORATORY

PROJECT REPORT 1

DUAL-TONE MULTI-FREQUENCY SIGNAL DETECTION

ZELİHA ASENA KIRIK

2015401033

Table of Contents

1.1) Introduction to Project

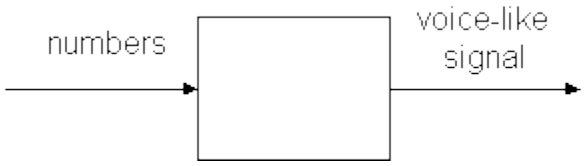
In this project, we are expected to use Arduino Uno to receive and detect Dual-Tone Multi-Frequency (DTMF) signals which are used in classic telephone signaling. Arduino Uno is used to digitize audio signal and in order to process this digitized signal, I used MATLAB. The aim of the project is to detect the frequencies of DTMF signal with the filter implementation and display numbers or characters which corresponds to input data.

In order the code to work as expected, user needs to give sound input, in some seconds after start of running, code will start to process data(in the beginning there is part for implementing filters and it takes some seconds) and display the DTMF symbol next to previous symbol which is given. If user continuously pushes a button without releasing, program will only write one symbol, if on the other hand user pushes the same button with releasing in between pushes, it will write the same symbol consecutively. Program do not detect any other signal than DTMF signals; so if user sends another input value let's say 770 Hz cosine signal (one component of the DTMF frequency), program will not write any symbol. In order to generate dual tone signals as it can be seen in below, user can use some websites for instance http://onlinetonegenerator.com/dtmf.html) and simply pushing buttons these websites produce sound with corresponding two frequencies.

1.2) Dual-Tone Multi-Frequency Signaling

DTMF signaling was developed to signal the destination telephone number of calls without requiring a telephone operator. It was standardized by the International Telecommunication Union (ITU) Telecommunication Standardization Sector recommendation Q.23.[1]

DTMF tones are also used by cable television broadcasters to indicate the start and stop times of commercial insertion points during station breaks for cable company benefit. The frequencies used prevent harmonics from being incorrectly detected by receivers as other DTMF frequencies. Another application to this is that you may connect to some service that asks you to enter your credit card number or account number, or asks you to respond to certain questions by pressing buttons on your telephone keypad. This process will have this kind of structure below:



It converts sequences of numerical digits into signals that can be easily traverse circuits designed for voice. The internationally standardized way to do this is DTMF signaling, or dual-tone, multi-frequency. DTMF signaling converts decimal digits (and the symbols '*' and '#') into sounds that share enough essential characteristics with voice.

The first DTMF tone producing telephone was introduced in November of 1963 for the Bell System. More user friendly than the rotation of a rotary dial, touch tone phones quickly supplanted the rotary phone. Dialing speed increased, stress on the network decreased, and users flocked to the new technology. With DTMF, each key you press on your phone generates two tones of specific frequencies. So that a voice can't imitate the tones, one tone is generated from a high-frequency group of tones and the other from a low frequency group.

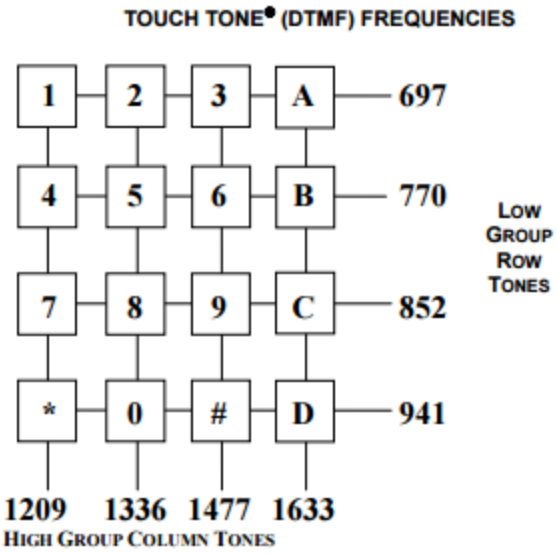


Figure 1:DTMF frequencies

The DTMF system uses a set of eight audio frequencies transmitted in pairs to represent 16 signals, as shown in the figure on the right[2]. For example, to represent the number “2”, two tones with frequencies of 697 Hz and 1336 Hz are generated at the same time.

2) Materials

- Arduino Uno
- 2 x 10k resistor
- 1x 1 uF capacitor
- Breadboard
- Jumper cables
- 2x crocodile cable
- 3.5 mm aux cable

3) Methods & Design

In this part of the report, I will explain my methodology. I started doing project first connecting stereo aux cable and Arduino (Arduino Uno). Sound signal cannot directly be given to Arduino as input because Arduino can sample and digitize signals in 0-5 V voltage interval and sound signal oscillates -+300mV around 0V. In order to

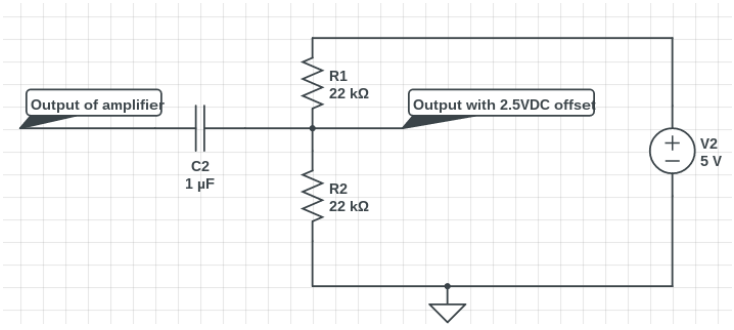


Figure 2:Voltage Offset

solve this problem, we can use the simple circuit above Figure 2[3] and it adds +2.5 V DC, so that sound signal oscillates +-300mV around 2.5V.

In Arduino software, I used the first alternative in homework assignment to read the data.

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

I prefer this to other because in this method for one sample 1 byte is used, in the other method, we use 2 bytes for one sample which makes the transmission slower. The disadvantage of first method is precision loss but for my task, losing precision did not jeopardize the performance so I decided to go on with the first method.

After digitizing data, next task is processing data in MATLAB. After some coding, MATLAB can detect Arduino, port number can be detected in port section in Arduino software. The value of data taken by MATLAB is in the range of 0 to 255(2^8=256)

Without knowing the sampling rate of Arduino, we cannot go any further in filtering functions in MATLAB so we need to first determine sampling rate. In order to do this, I sent 2 kHz cosine signal. In the fft of the signal it is 160 points away from its DC component and there are 357 points wide which corresponds to half of sampling frequency. In frequency domain, we can think if 2 kHz is in the position of 160, which frequency corresponds to the position 357x2. I got the result 8925 Hz.

After determining sampling rate, I determined buffer size too, in industry for one process to happen, time required is about 40ms. So the number of samples in one loop time can be determined by

(0.040msecond x 8910 sample/second) =~ 356 sample

So in every reading process, I get a data vector whose size is 356. Since DTMF signals have 8 possible frequency values, I designed 8 band pass filters whose bandwidths are all equal to each other (70 Hz); for range of frequencies it is important that they do not coincide. Other than this I assigned the middle frequency of filter’s pass band to specific DTMF desired frequency values. I

choose this value because approximately between two different frequencies, frequency difference is at least 70 Hz. I used IIR band-pass filters because IIR filters usually require fewer coefficients than FIR filters to execute similar filtering operations, that IIR filters work faster, and require less memory space.

The disadvantage of IIR filters is the nonlinear phase response but IIR filters are well suited for applications that require no phase information, for example, for monitoring the signal amplitudes. After implementing all, I decide to use IIR Filter because there was no problem in my results.

After filtering signals with these filters, I calculate energy of the filtered signals in different frequency ranges. (665-735, 735-805, 815-885, 905-975) (1175 – 1245, 1305-1375, 1445 – 1515, 1595-1665). There are two categories of signals: high (1209 1336 1477 1633) and low frequency components (697 770 852 941). So I find the maximum of both high frequency and low frequency components.

Even though there is no button pushed, because of the noise there is always some amount of energy. By setting a threshold value for energy, noise can be ignored and only when there is enough energy which shows button is actually pressed, then symbol will appear on screen (My threshold value is 1000). After finding 2 frequencies only thing left is to choose the symbol which corresponds to those 2 frequencies in DTMF matrix.

Code guarantees that if the sound signal is not one of these dual tones, it is not going to show any symbol because even though it can find one frequency it cannot find other frequency so it has not enough argument to call the DTMF matrix. Another point is for pushing the button continuously in my code I defined loop variable and if energy is enough for the first time (button is pushed), it will write the symbol and set loop to 3, if button is released, this value will become one less and after 3 loops it will become again available to write new symbols. On the other hand after writing one symbol if the next one has also enough energy it will mean that button is not released so it will not write it because count is not zero. To sum up in order next symbol to be written there should be at least three non-energetic stages. After correcting this problem I came to end in my project.

REFERENCES

[1] Techopedia.com. (2019). *What is Dual-Tone Multifrequency (DTMF) ? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/26023/dual-tone-multifrequency-dtmf [Accessed 15 Nov. 2019].

[2]Sigidwiki.com. (2019). *Dual Tone Multi Frequency (DTMF) - Signal Identification Wiki*. [online] Available at: https://www.sigidwiki.com/wiki/Dual_Tone_Multi_Frequency_(DTMF) [Accessed 15 Nov. 2019].

[3] cable, A. and Jeet, P. (2019). *Arduino Audio Input from AUX cable*. [online] Electrical Engineering Stack Exchange. Available at: https://electronics.stackexchange.com/questions/211904/arduino-audio-input-from-aux-cable [Accessed 15 Nov. 2019].