COMP9336
Mobile Data Networking

Device-to-device Communication over Audio

Report

Z5004850
Tengyu Ma

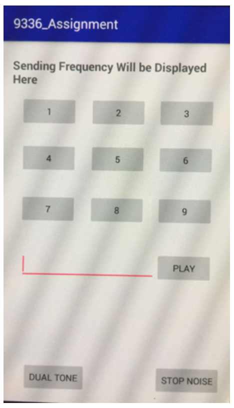
Over View

Communication between different devices basically depends on sending and receiving waves. Currently, our communication methods are mostly established on Electromagnetic waves with different frequency. As the incensement of telecommunication requirement, the electromagnetic spectrum is becoming more and more crowed. Some alternative method should be introduced into low range communication in order to relieve current spectrum crowding problem. Based on the fact that most of mobile devices (Phones and wearable devices) have built-in speakers and microphones. This assignment will try to implement a Device-to-device communication over audio.

Transmitting device

Single Tone generator

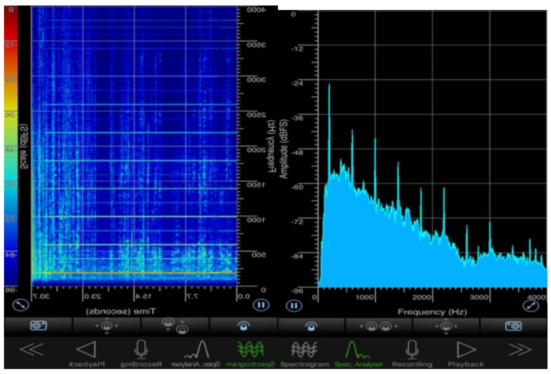
Android provides an API called AudioTracker which manages and plays a single audio resource for Java applications[1]. By writing a 16-bit binary array into it, we could simply implement a single Tone generator of any frequency.



The picture shows the interface of my tone generator. There are two things we should take into consideration. First is sampling rate, according to Nyquist's Law, fs.max>=2fmax, which means if we wish to generate the audio wave we want, the sampling rate should be at least twice as the audio frequency. For this assignment, both transmitter and receiver's sampling

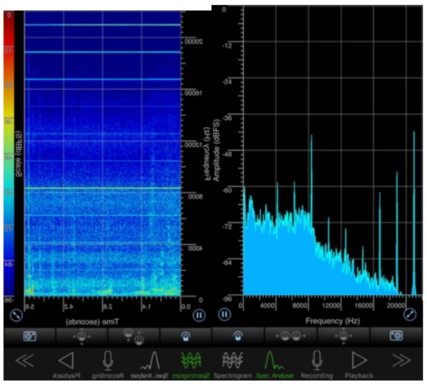
[1] Audio Track https://developer.android.com/reference/android/media/AudioTrack.html

rate will be set to 44100. The transmission frequency will be at most 22000Hz. Second one is how to generate 16-bit PCM sound array. It basically depends on a formula "Sin(2*PI*Frequency*i/SamplingRate)". This formula will calculate the numeric value of the Sin-wave at each sampling point. When we writing the audio track message to 16-bit PCM array, we should first scale the message to the maximum amplitude, then write binary code into the array. In this way, we could generate accurate audio track for playing.



picture1-1 200Hz

These two pictures(picture1-1) show the test result of the transmitter on 200Hz. We can see in both frequency and time domain, the audio track we created was not strict a single frequency sound. That was because the sound in the natural world is a combination of audio tracks of different frequency. But the graph also shows that the base frequency, which is 200 Hz in this experiment, has the largest energy. This provides the most important idea for the receiver part, and it will be explained in receiver part of this report.



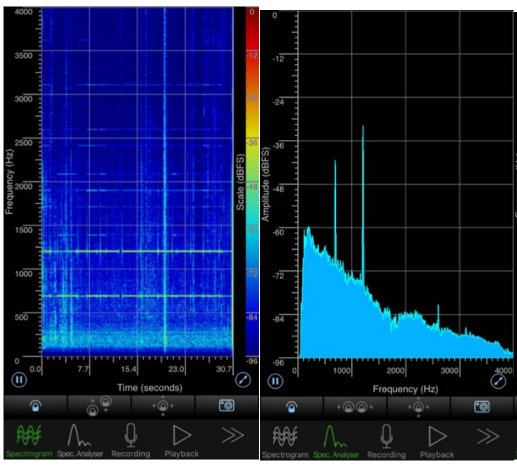
picture1-2 21000Hz

As for ultrasonic wave, although most of the device could generate semi-ultrasonic wave(picture1-2), they could hardly generate any real high frequency ultrasonic wave (above 22000Hz). Due to the limitation of our devices (None of the commercial phones has an ultrasonic wave speaker), the ultrasonic we generate brings very low energy and very hard to detect. For reasons above, we just use semi-ultrasonic represents ultrasonic part.

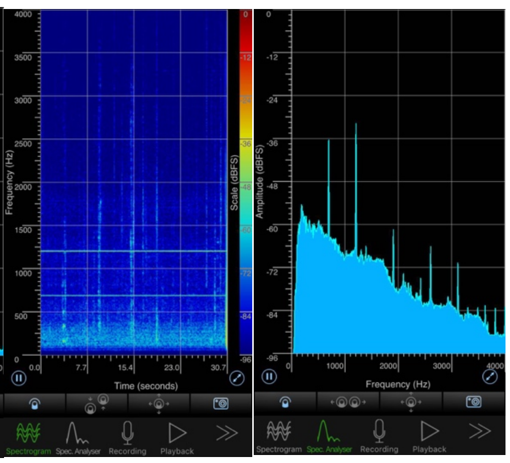
The other point found in the experiment was the volume of speaker influences experiments a lot. Most of mobile phones have plastic covering, which means they may have resonance during speaker's working at low frequency.

Dual Tone generator

In this Assignment, both single tone and dual tone uses same interface. The difference between them was just different PCM arrays. For dual tone part, when calculating audio track message, I simply make two wave superimposed together and the speaker will generate the dual tone on two frequencies. I also implement a build in standard dual tone function as reference. Both experiment data are showed in picture 1-3 and 1-4.



picture1-3 Standard dual tone 1

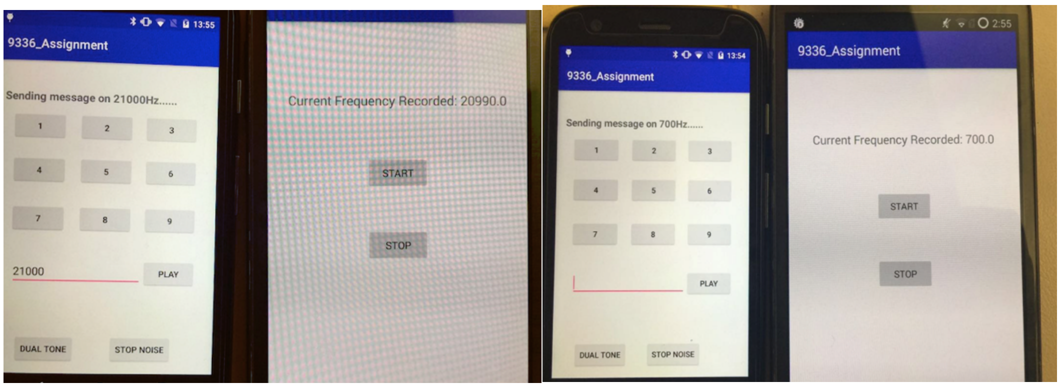


picture1-4 Self-generated dual tone(1209Hz*697Hz)

Receiving Device

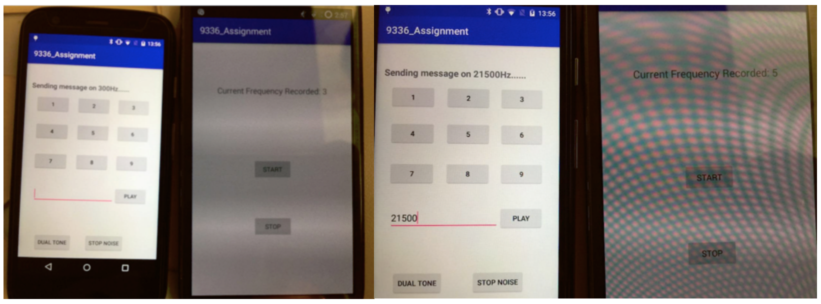
Android provides an API called AudioRecord. This class manages the audio resources for Java applications to record audio from the audio input hardware of the platform[2]. We could simply read audio data from the audio hardware for recording into a direct buffer and convert it into decimal value. Then we alternatively use Goertzel algorithm[3] to analysis the recorded data's energy in different frequency segments, and find which frequency have received by our devices. Goertzel algorithm code was modified from the website announced in reference.

[2] Audio Record https://developer.android.com/reference/android/media/AudioRecord.html
3 Result from processing audio signal with Goertzel algorithm http://stackoverflow.com/questions/7339763/result-from-processing-audio-signal-with-goertzel-algorithm


9336_Assignment
Sending message on 21000Hz......
21000
PLAY
DUAL TONE
STOP NOISE
Current Frequency Recorded: 20990.0
START
STOP
Sending message on 700Hz......
Current Frequency Recorded: 700.0

picture 2-1

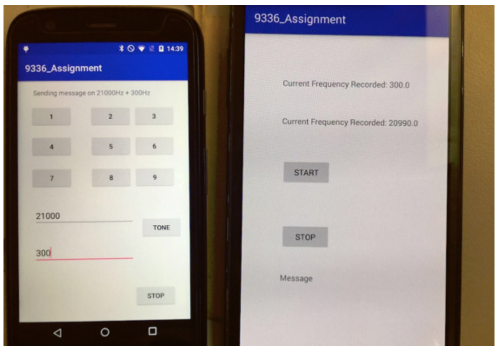
Picture 2-1 shows the receiver designed in this assignment works in both low and ultrasonic frequency. Basically, the way to design the receiver is scanning all frequency from 1 to 22000Hz with an internal of 10Hz. As we could see in picture2-1, there might be some differences between sending and receiving frequency, that was because some of the devices could not accurately generate some special frequency. As it is impossible to test all frequency from 1-22000Hz, there is no results to show which frequency is accurate or not. So the internal I set for different message is 100Hz. For every 1000Hz, I set 1-9 for each hundred value. Picture shows recorder could accurately recognize the numeric message generated by transmitting device.


9336_Assignment
Sending message on 300Hz......
Current Frequency Recorded: 3
Sending message on 21500Hz......
21500
PLAY
DUAL TONE
STOP NOISE
Current Frequency Recorded: 5
START
STOP

picture 2-2

The trickiest part of the recorder was how to choose the energy threshold of different frequency segment, a bad threshold choice will lead to miss-detecting of recorder. For this assignment, every increment of 5000Hz, the threshold will multiply by 10^-1.

Same strategy works well on dual tone detection. The only difference was the recorder will keep scanning on two different frequency segments. Picture 2-3 shows the result for dual tone detecting result.



Sending message on 21000Hz + 300Hz

21000

300

Current Frequency Recorded: 300.0

Current Frequency Recorded: 20990.0

picture 2-3

Data Transmitting

Previous tasks are the basic components of a data transmitter. We had already implement the sending and receiving part of a communication system. Also we finished the dual tone generator.

Dual Tone Multi-frequency was first used in 1963 for dialling system and becoming a industry standard for communication services. In this assignment, this technology will be used in data transmission part. It has two benefits. Firstly, compared to single tone, dual tone has a better reorganization features. This could filter out the influence of noise. Secondly, assume we try to do non-binary transmission, combination of different frequency will reduce the occupancy of frequency. For example, if we want to transfer 256 ASCII code throw audio network, single tone should use at least 256 distinct frequencies to represent all codes in the standard table, but if we use dual tone, only 16 frequencies' combination will be enough for ASCII code. Considering binary transmission is not effective enough, dual tone multi-frequency will be a better choice in audio data transmitting.

As it is mentioned above, binary transmitting is not effective enough, the data design will be designed as follow

	21900	21600	21300	21000
19900	0	1	2	3
19600	4	5	6	7
19300	8	9	A	B
19000	C	D	E	F

Any two frequency represent a hex number; any two hex number represents a value in standard ASCII table.

For synchronization problem, at the beginning of each message, the transmitter will first generate serval bits starting code. Once the receiver gets these code, it will start to save the message. At the end of message, transmitter will generate ending code either.

As the transmission is not stable, there is no graph in this part, further explanation will be established in Demo.