Assignment for COMP4336/9336 Mobile Data Networking
Semester 2, 2019 (Individual Assignment)

Due: 11:59pm Thursday 01 August 2019

Weighting: 20%

Title: Device-to-device Communication over Audio

Background and Motivation

Most mobile devices include a speaker and a microphone. This creates an opportunity to enable device-to-device (D2D) communication over audio that would work for almost 100% of the available devices in the market. Audio-based D2D would also enable any other devices with a speaker, such as a TV, FM radio, musical instruments, alarm systems, and so on to communicate directly with a personal mobile device without having to configure WiFi or Bluetooth connections. Audio-based D2D can also provide a more efficient and ubiquitous alternative to NFC-based communication used for a growing volume of mobile payment transactions and other emerging services. Tech giants have already started exploring the potentials of audio-based D2D giving rise to new developments such as Google Tone, an extension of Chrome that lets nearby devices to share URLs over inaudible sound frequencies.

Learning objectives

Upon completing this assignment, students will:

1. Gain insight to audio-based data communication applicable to current mobile devices,
2. Master the access to speaker and microphone in mobile devices, and
3. Learn how to design and implement a basic audio-based data transmitter and receiver for personal mobile devices

Assignment Tasks

Task 1 Single Tone Detection [3 marks]

In this task, the transmitting device, such as a laptop, will continuously transmit a single sinusoidal tone with a given frequency using the built-in speaker and a receiving device, such as a mobile phone, will detect the frequency. Upon tone detection, the receiver may do some simple tasks, such as displaying the value of the frequency detected, to confirm that it is working correctly. The transmitting device may implement a very simple interface for the user to generate a single tone with a given frequency, such as a text input for the user to specify the frequency (tone) to be transmitted. This should work for both lower frequencies (audible) and higher frequencies (less audible or even inaudible).

Knowledge of conventional digital signal processing (DSP), which employs Fast Fourier Transform to analyse all frequencies in the entire spectrum, is NOT required to complete this task. You are instead encouraged to implement a very simple method, the Goertzel algorithm, which can detect the presence of a single tone with only a few lines of code. There are many sites on the Internet offering tutorial and even codes for Goertzel algorithm (you need to do some search). You are free to use any such codes to complete your task, but you need to acknowledge the source of the code and any modifications you have done to the original code to make it work for your case.

[*Note that Goertzel algorithm is not explained in the lectures, but its study and implementation on your own is an essential part of this assignment and learning. Your tutor may provide some basic help regarding tone generation etc., you are also allowed to discuss these topics on the Moodle forum.*]

Task 2 Extension of Single Tone Detection [3 marks]

Extend your transmitter code so it allows the user to input any digit between 1-9 and let the receiver display the digit. For this, you will need to hardcode different frequencies to different digits. You may have to play around with different frequencies until it works properly (Hint: if the frequencies are too close to each other, Goertzel algorithm may not work properly).

You should demonstrate two versions, *audible* and *inaudible* (*less audible)*. For the inaudible version, finding 9 high frequencies for your speaker/microphone that are completely inaudible may be difficult. You are, therefore, allowed to go down the frequency range at the price of making them slightly audible. Try to be as much inaudible as possible and report the maximum frequency range that works for your hardware.

Task 3 Dual Tone Detection [4 marks]

In the previous tasks, you generated a single tone using a single sinusoidal frequency. Find out how you can combine two distinct frequencies in the same sine wave, so each signal actually carries two frequencies. Addition of multiple frequencies in the same signal makes the communication more robust to noise. Do the following

(a) Extend your code from Task 2 to implement the traditional Dual Tone Multiple Frequency (DTMF) still used in many touch-tone handsets. Select the standard frequencies of DTMF system (see Table 1), so it sounds like touch tone handsets.

(b) Then explore the possibility of implementing DTMF in the *inaudible* band and report the capability of your hardware in terms of the frequency range that works. Note that all you have to do for this is to try and observe different combinations of high frequencies (as shown in Table 1, standard DTMF uses audible frequencies).

Table 1 DTMF frequencies (in Hz) for digits 1-9. For example, you need to transmit frequencies 697 Hz along with frequency 1209 Hz simultaneously to represent the digit 1.

	1209	**1336**	**1477**
697	**1**	**2**	**3**
770	**4**	**5**	**6**
852	**7**	**8**	**9**

Task 4 Packetized Data Communication with Audio Tones [6 marks]

In Tasks 1-3, you have implemented and demonstrated the capability of a commercial mobile device to detect audio tones using its built-in microphone. In this task, you will develop a basic data communication system that can use this capability to transmit a small packet of few bits, which may represent a short URL, a short text message, a credit card number, and so on. To implement packetized communication, you will need to refresh your basic networking prerequisite knowledge of a number of concepts including preamble detection and modulation.

Your implemented system will work as follows. The transmitter will allow the user to input a single word or a multi-digit number. It will then transmit a series of bits representing the letters or digits of the word/number (you could consider the ASCII standard) and the receiver will be required to detect all those bits received in sequence, recognise the corresponding letters/digits, and display the transmitted word/number.

While it appears a simple extension from previous tasks, the main challenge lies in synchronizing the receiver with the transmitter so each bit interval is detected correctly. That’s where the preamble comes in. There are many different choices for designing the preamble of a packet. You are free to design your own preamble. The only objective is to make it work for your audio-based communication.

For modulating a ‘1’ and a ‘0’ bit on the audio tone (carrier signal), there are many options. You are required to implement only the most basic and simplest modulations, such as either binary amplitude shift keying (BASK) or binary frequency shift keying (BFSK). For BASK, you can simply use a zero amplitude, i.e., no transmission, and a positive amplitude, i.e., a transmission, also known as ON-OFF keying (OOK). For BFSK, you can choose to use two different frequencies for data transmission, one representing a ‘1’ and the other a ‘0’. You are free to choose any other modulation techniques that you feel comfortable with.

Task 5 Error correction [4 marks]

Audio communication may be affected by noise corrupting your data transmission. In this task, you will implement some basic error correction, such as Hamming code, Parity code, etc. that will allow the receiver to detect whether the received data is the same as the transmitted data. You may then decide to discard any incorrectly received packet and only display correctly received packet at the receiver.

Some Interesting Links

Android microphone API
https://developer.android.com/reference/android/media/MediaRecorder.AudioSource

Hidden acoustic communication with smartphones
https://cse.buffalo.edu/~lusu/papers/MobiCom2016.pdf

Google's ultrasonic networking
http://smus.com/ultrasonic-networking/
http://www.theverge.com/2014/6/26/5846726/chromecast-will-use-ultrasonic-sounds-to-connect-nearby-devices

Assignment Submission and Marking

For this assignment, you will NOT have to submit your code, but you will upload in Moodle a pdf report explaining how you solved your tasks, what design choices you have made and why, by the assignment due date (11:59pm Thursday 01 August 2019) and demonstrate your working project in Week 11. Demo slots and venue will be posted on Moodle later.

In the report, you have to provide the **main code fragment** (up to 30 lines) of the **last task** you complete. For example, you should provide the main code of task 3 if you finish the first three tasks. Based on that, we only assess tasks 1-3 during the demo, even though you complete more tasks after the due. Each task will be assessed based on both the report and the demo, which contribute 20% and 80% respectively.

Late penalty at the rate of 10% per day late will apply if the report is submitted after 11:59pm Thursday 01 August 2019.

The End (We hope you enjoy doing this assignment)