Helpful Guide for Manipulating Speakers and Microphones

1- How to generate a single tone signal?

Assume s(t) is a single tone signal:

$$s(t) = A \sin(2\pi f t + \varphi)$$

Where A is signal amplitude, f is signal frequency and φ is phase. All these 3 parameters can be used to modulate data over this signal. For example assume we are using frequency to send data. A frequency shift keying (FSK) is a frequency modulation scheme in which digital information is transmitted through discrete frequency changes of a carrier signal. In a binary case (BFSK), we can use only 2 frequencies, f_1 and f_2, to send single bit 0 and 1 respectively.

$s_1(t) = A \sin(2\pi f_1 t + \varphi)$ sending bit 0
$s_2(t) = A \sin(2\pi f_2 t + \varphi)$ sending bit 1

In audio communication, you need to send this signals by audio ton. So you just need to play one of the signal as audio to speakers. In basic FSK modulation, A should be constant and depend on the hardware you are using, It is recommended to chose maximum possible value (usually 1). φ also is not important in FSK modulation and can be zero.

2- How to generate a multiple ton signal?
A multi tone signal is the summation of several sine waves or tones. So, to have a signal with two frequencies (Dual Tone):

$$s(t) = s_1(t) + s_2(t) = A_1 \sin(2\pi f_1 t + \varphi_1) + A_2 \sin(2\pi f_2 t + \varphi_2)$$

For basic FSK, $A_2 = A_1$ and $\varphi_1 = \varphi_2 = 0$.

Important note: The above examples are for FSK modulation. For other modulations such as BASK, the amplitude and phase of signal may be used to embed data.

3- How to transmit signal?
You can use a laptop or mobile device to play the signal as sound. MATLAB is a good tools to generate and play signals.

Here is the MATLAB code to generate and play a dual tone signal:

```
Fs = 44100;               % signal sample rate
f1 = 7000;                % first Tone frequency
f2 = 12000;               % Second Tone frequency
t = 0:1/Fs:10;            % Time vector, 10 second, the step is...
                          %... exactly related to Fs
s1 = sin(2*pi*f1*t);      % First Tone
s2 = sin(2*pi*f2*t);      % Second Tone
signal = s1 + s2;         % Summation of signals
soundsc(signal,Fs)        % playing signal on speakers
```

You can also use android to create and play Tones from mobile device:
http://stackoverflow.com/questions/2413426/playing-an-arbitrary-tone-with-android

Note: The highest audio frequency you can send depends on the device speakers. Also your device may play frequency above 18 kHz but it is hard to be heard.

4- How receive audio in receiver?
It will be received by microphone. In android, it is recommended to use android.media.AudioRecord;

```
AudioRecord recorder = new AudioRecord(MediaRecorder.AudioSource.MIC,
SAMPLE_RATE, AudioFormat.CHANNEL_IN_MONO,
AudioFormat.ENCODING_PCM_16BIT, bufferSize);
```

Choose higher sample rate (recommended 44100);

5- How to detect tones (frequencies) in received audio?
Goertzel algorithm is designed to detect whether a tone exist in a specific window in frequency domain. To learn about Goertzel you can find online documents where a useful one is :
http://www.embedded.com/design/configurable-systems/4024443/The-Goertzel-Algorithm

You can use following class for Goertzel in android. You need to create a class in your project and copy this code there. To use this class in main android activity you should initialize, process and reset it each time.

```java
public class Goertzel {

    private float samplingRate;
    private float targetFrequency;
    private long n;

    private double coeff, Q1, Q2;
    private double sine, cosine;

    public Goertzel(float samplingRate, float targetFrequency, long inN) {
        this.samplingRate = samplingRate;
        this.targetFrequency = targetFrequency;
        n = inN;

        sine = Math.sin(2 * Math.PI * (targetFrequency / samplingRate));
        cosine = Math.cos(2 * Math.PI * (targetFrequency / samplingRate));
        coeff = 2 * cosine;
    }

    public void resetGoertzel() {
        Q1 = 0;
        Q2 = 0;
    }

    public void initGoertzel() {
        int k;
        float floatN;
        double omega;

        floatN = (float) n;
        k = (int) (0.5 + ((floatN * targetFrequency) / samplingRate));
        omega = (2.0 * Math.PI * k) / floatN;
        sine = Math.sin(omega);
        cosine = Math.cos(omega);
        coeff = 2.0 * cosine;

        resetGoertzel();
    }

    public void processSample(double sample) {
        double Q0;

        Q0 = coeff * Q1 - Q2 + sample;
        Q2 = Q1;
        Q1 = Q0;
    }

    public double[] getRealImag(double[] parts) {
        parts[0] = (Q1 - Q2 * cosine);
        parts[1] = (Q2 * sine);

        return parts;
    }

    public double getMagnitudeSquared() {
        return (Q1 * Q1 + Q2 * Q2 - Q1 * Q2 * coeff);
    }
}
```