USER GUIDE FOR EQUALIZATION BASED MUTIL-CHANNEL COMPRESSION ON ANDROID PLATFORM FOR HEARING AID APPLICATIONS

Using

Superpowered SDK to Complete Implementation on Android Google Pixel

Yiya Hao, Ziyan Zou, Issa Panahi
STATISTICAL SIGNAL PROCESSING LABORATORY (SSPRL)
UNIVERSITY OF TEXAS AT DALLAS

APRIL 2018

This work was supported by the National Institute of the Deafness and Other Communication Disorders (NIDCD) of the National Institutes of Health (NIH) under the award number 1R01DC015430-01. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

5/2/2017

Table of Contents

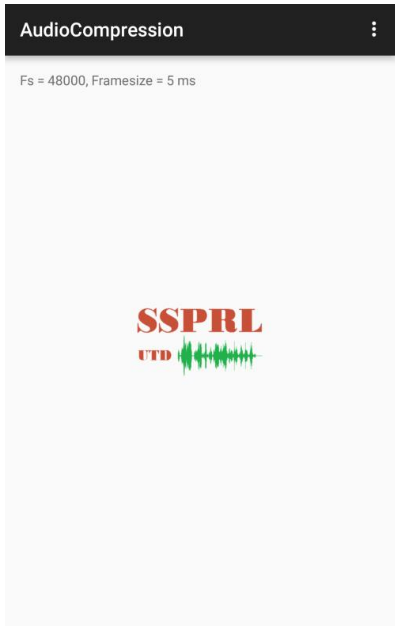
1. INTRODUCTION

The Mutil-Channel Audio Compression app is designed for multi-channel compression hearing aids in real time. The contents of this user guide gives you the steps to implement the Mutil-Channel Audio Compression algorithm on Android devices (Android smartphones and Android Pads) and the steps to be followed after installing the app on the smartphone. This app will be an open source and portable research platform for hearing improvement studies.

This user guide covers the software tools required for implementing the algorithm, how to run C codes on Android devices and usage of other tools that are quite helpful in creating audio apps for audio playback in real time.

The C codes used for the Mutil-Channel Audio Compression algorithm are made available publicly on the following website:
http://www.utdallas.edu/ssprl/
The codes can be accessed and used with proper consent of the author for further improvements in research activities related to hearing aids.

The screenshot of the first look of our app is as shown in Figure 1 below,



Fs = 48000, Framesize = 5 ms

Figure 1

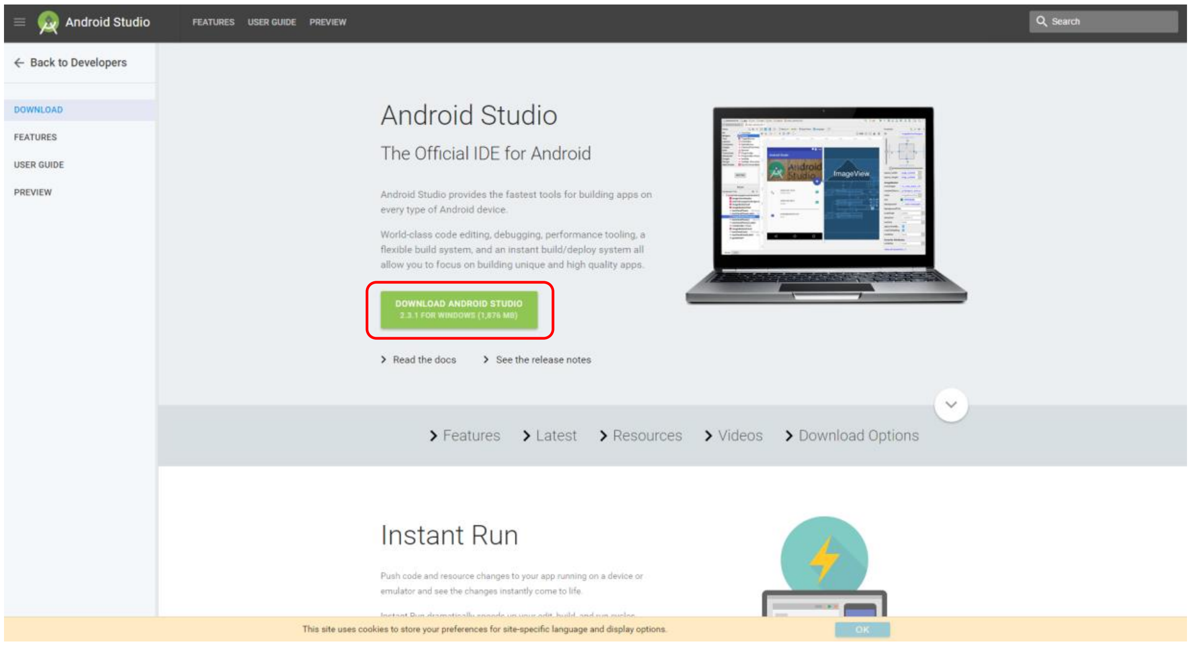
2. SOFTWARE TOOLS

Android is an open-source operating system developed by Google for mobile phones and tablets. The Android apps are normally coded in Java. In this section, it is shown how to set up the Android Studio IDE (Integrated Development Environment) for developing Android apps.

Android Studio IDE requires either Windows Operating System or Apple Operating System. Android Studio IDE can directly build and upload source codes into Android smartphone or generate a APK file which can be downloaded and installed on the Android smartphone.

To download the latest version of Xcode

1. Open the Android Studio website to download. (https://developer.android.com/studio/index.html)
2. Click Download Android Studio Button (Figure 2).
3. Install the execution file after download.



Android Studio

The Official IDE for Android

Android Studio provides the fastest tools for building apps on every type of Android device.

World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow you to focus on building unique and high quality apps.

Instant Run

Push code and resource changes to your app running on a device or emulator and see the changes instantly come to life.

This site uses cookies to store your preferences for site-specific language and display options.

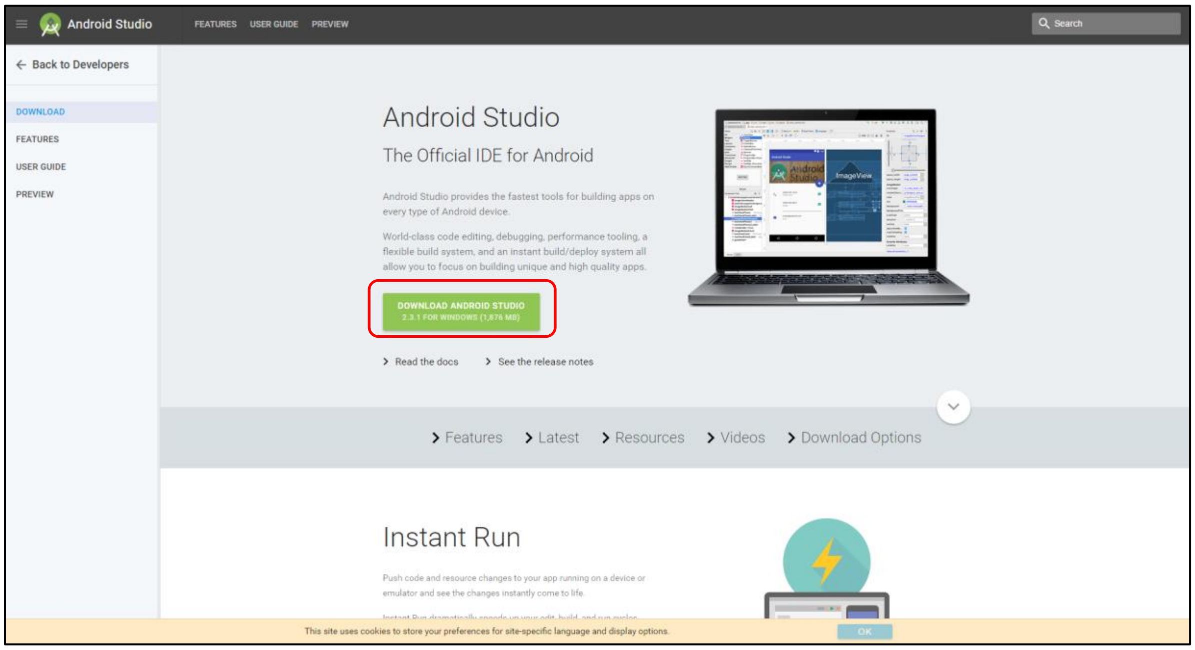
Figure 2

Android is an open-source operating system developed by Google for mobile phones and tablets. The Android apps are usually coded in Java. In this section, it is shown how to set up the Android Studio IDE (Integrated Development Environment) for developing Android apps.

Android Studio IDE requires either Windows Operating System or Apple Operating System. Android Studio IDE can directly build and upload source codes into Android smartphone or generate a APK file which can be downloaded and installed on the Android smartphone.

To download the latest version of Android Studio

4. Open the Android Studio website to download. (https://developer.android.com/studio/index.html)
5. Click Download Android Studio Button (Figure 2).
6. Install the execution file after download.



Android Studio

The Official IDE for Android

Android Studio provides the fastest tools for building apps on every type of Android device.

World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow you to focus on building unique and high quality apps.

Instant Run

Push code and resource changes to your app running on a device or emulator and see the changes instantly come to life.

This site uses cookies to store your preferences for site-specific language and display options.

Figure 2

3. BUILD AN ANDROID APP

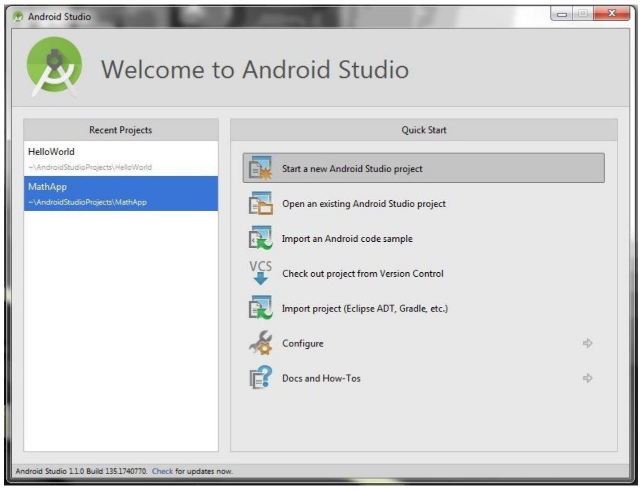
3.1 Programming Language

For creating Android apps, Java is used to create the required shell. The Java Development Kit (JDK) needs to be firstly installed on your computer. This link contains the latest version of JDK: http://www.oracle.com/technetwork/java/javase/downloads/index.html

3.2 Creating Android Apps

After installations of Android Studio and JDK completed, android apps can be created using Android Studio.

1. Open Android Studio.
2. Under the "Quick Start" menu, select "Start a new Android Studio project." (Figure 3)
3. On the "Create New Project" window that opens, name your project "HelloWorld".(Figure 4)
4. If you choose to, set the company name as desired*.
5. Note where the project file location is and change it if desired.
6. Click "Next."
7. Make sure on that "Phone and Tablet" is the only box that is checked. (Figure 4)
8. If you are planning to test the app on your phone, make sure the minimum SDK is below your phone's operating system level.
9. Click "Next."
10. Select "Blank Activity." (Figure 4)
11. Click "Next."
12. Leave all of the Activity name fields as they are. (Figure 4)
13. Click "Finish."


Android Studio

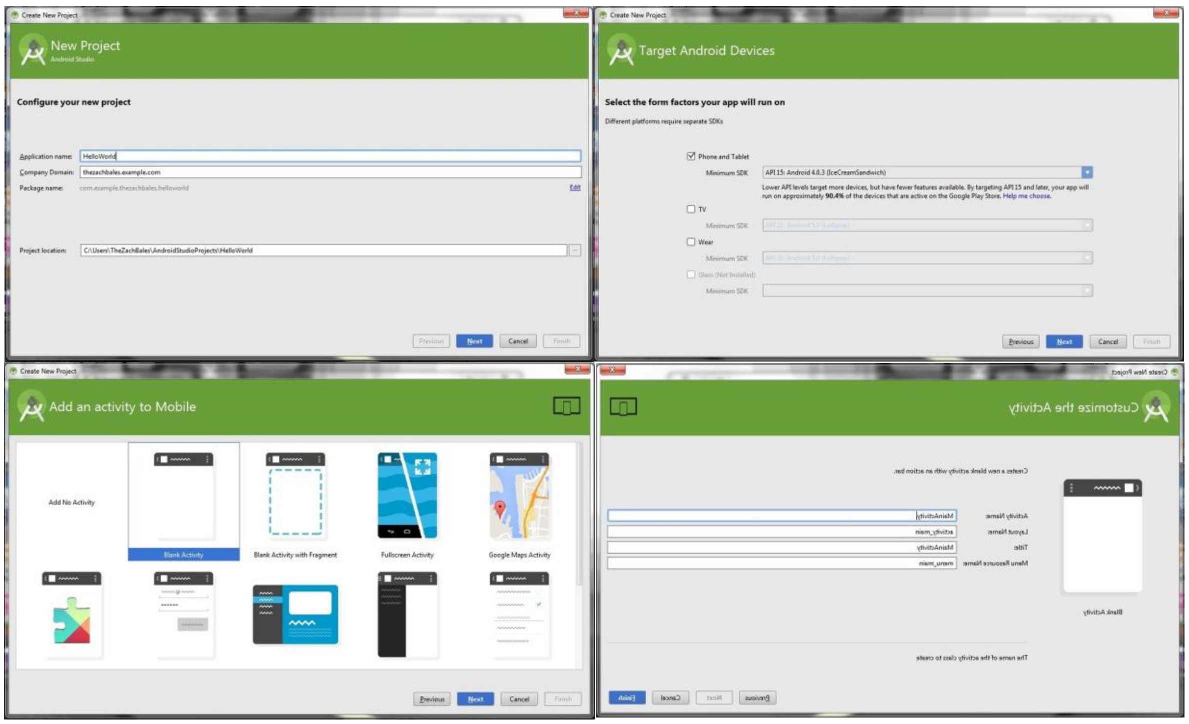
Welcome to Android Studio

Recent Projects
- HelloWorld ~\AndroidStudioProjects\HelloWorld
- MathApp ~\AndroidStudioProjects\MathApp

Quick Start
- Start a new Android Studio project
- Open an existing Android Studio project
- Import an Android code sample
- Check out project from Version Control
- Import project (Eclipse ADT, Gradle, etc.)
- Configure
- Docs and How-Tos

Android Studio 1.1.0 Build 135.1740770. Check for updates now.

Figure 3


Create New Project

New Project
Android Studio

Configure your new project

Application name: HelloWorld
Company Domain: thezachbales.example.com
Package name: com.example.thezachbales.helloworld Edit

Project location: C:\Users\TheZachBales\AndroidStudioProjects\HelloWorld

Previous Next Cancel Finish

Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms require separate SDKs

[✓] Phone and Tablet
Minimum SDK API 15: Android 4.0.3 (IceCreamSandwich)
Lower API levels target more devices, but have fewer features available. By targeting API 15 and later, your app will run on approximately **90.4%** of the devices that are active on the Google Play Store. Help me choose.
[ ] TV
Minimum SDK API 21: Android 5.0 (Lollipop)
[ ] Wear
Minimum SDK API 21: Android 5.0 (Lollipop)
[ ] Glass (Not Installed)
Minimum SDK

Previous Next Cancel Finish

Create New Project

Add an activity to Mobile

Previous Next Cancel Finish

Create New Project

Customize the Activity

Creates a new blank activity with an action bar.

Activity Name: MainActivity
Layout Name: activity_main
Title: MainActivity
Menu Resource Name: menu_main

Blank Activity

The name of the activity class to create

Previous Next Cancel Finish

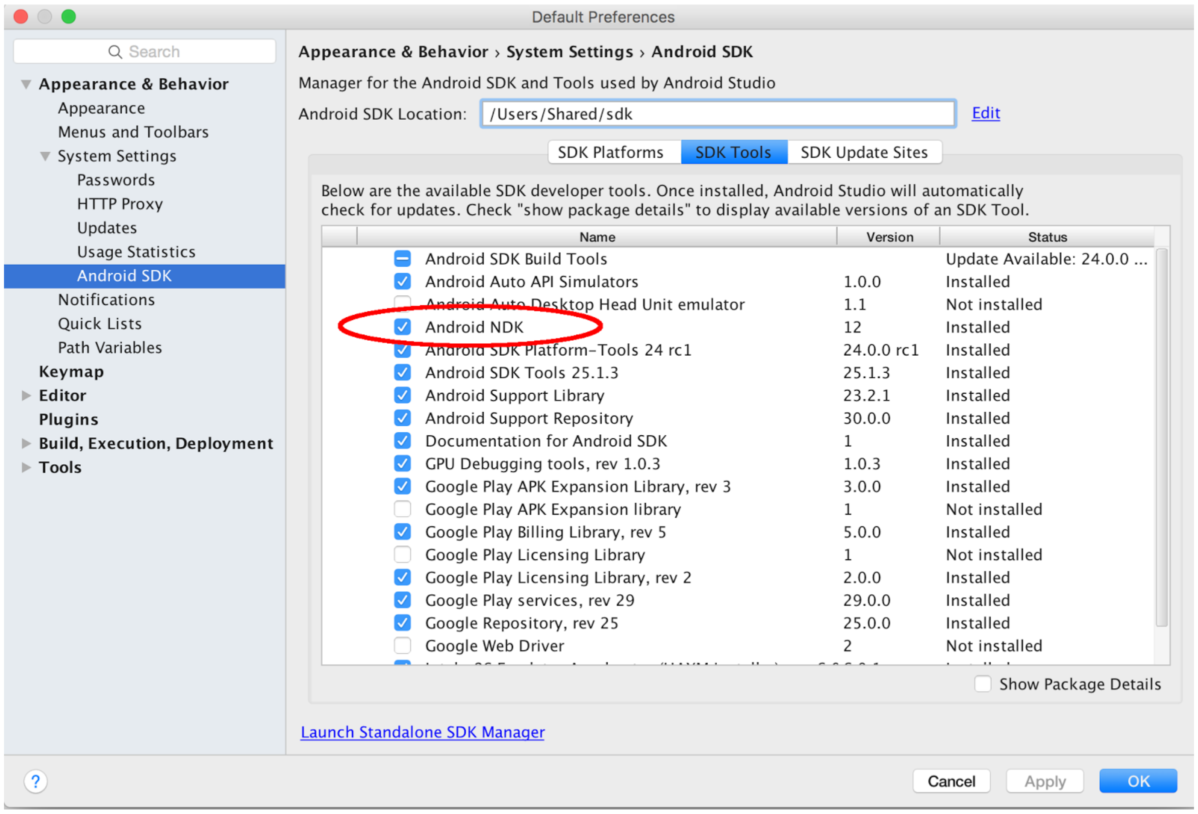
Figure 4

3.3 Adding C File

Android apps are typically written in Java, with its elegant object-oriented design. However, at times, you need to overcome the limitations of Java, such as memory management and performance, by programming directly into Android native interface. Android provides Native Development Kit (NDK) to support native development in C/C++, besides the Android Software Development Kit (Android SDK) which supports Java.

i. Installing the Native Development Kit (NDK)

1. Menu "Tools" > "Android" > "SDK Manager" (Figure 5)
2. Select tab "SDK Tools"
3. Check "Android NDK"[or "NDK"] if it is not checked
4. Sync and re-build the project.



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/Shared/sdk

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

	Name	Version	Status
–	Android SDK Build Tools		Update Available: 24.0.0 ...
✓	Android Auto API Simulators	1.0.0	Installed
	Android Auto Desktop Head Unit emulator	1.1	Not installed
✓	Android NDK	12	Installed
✓	Android SDK Platform-Tools 24 rc1	24.0.0 rc1	Installed
✓	Android SDK Tools 25.1.3	25.1.3	Installed
✓	Android Support Library	23.2.1	Installed
✓	Android Support Repository	30.0.0	Installed
✓	Documentation for Android SDK	1	Installed
✓	GPU Debugging tools, rev 1.0.3	1.0.3	Installed
✓	Google Play APK Expansion Library, rev 3	3.0.0	Installed
	Google Play APK Expansion library	1	Not installed
✓	Google Play Billing Library, rev 5	5.0.0	Installed
	Google Play Licensing Library	1	Not installed
✓	Google Play Licensing Library, rev 2	2.0.0	Installed
✓	Google Play services, rev 29	29.0.0	Installed
✓	Google Repository, rev 25	25.0.0	Installed
	Google Web Driver	2	Not installed

Figure 5

ii. Writing a Hello-World Android NDK Program

Creating a new project with support for native code is similar to creating any other Android Studio project, but there are a few additional steps:

1. In the Configure your new project section of the wizard, check the Include C++ Support checkbox.
2. Click Next.
3. Complete all other fields and the next few sections of the wizard as normal.
4. In the Customize C++ Support section of the wizard, you can customize your project with the following options:
   - C++ Standard: use the drop-down list to select which standardization of C++ you want to use. Selecting Toolchain Default uses the default CMake setting.
   - Exceptions Support: check this box if you want to enable support for C++ exception handling. If enabled, Android Studio adds the -fexceptionsflag to cppFlags in your module-level build.gradle file, which Gradle passes to CMake.
   - Runtime Type Information Support: check this box if you want support for RTTI. If enabled, Android Studio adds the -frtti flag to cppFlags in your module-level build.gradle file, which Gradle passes to CMake.
5. Click Finish.

After Android Studio finishes creating your new project, open the Project pane from the left side of the IDE and select the Android view. As shown in figure 6, Android Studio adds the cpp and External Build Files groups:

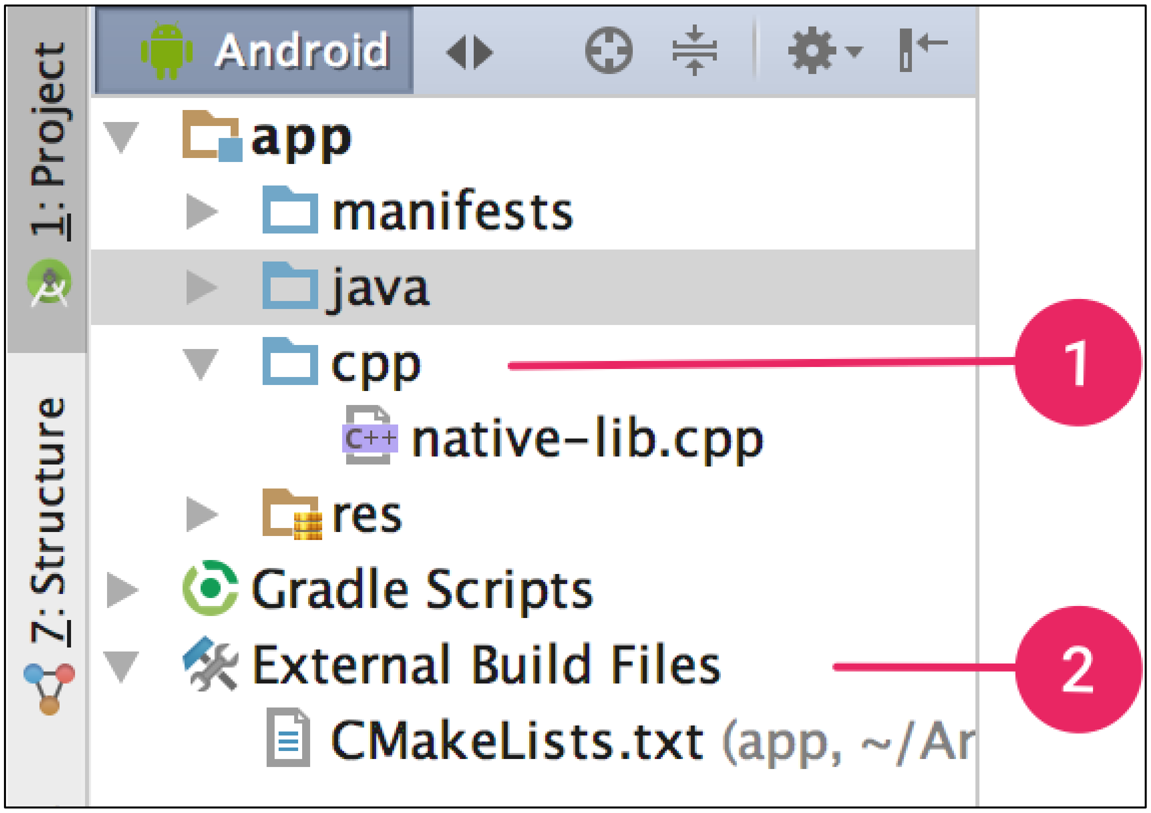


Figure 6

1. The cpp group is where you can find all the native source files, headers, and prebuilt libraries that are a part of your project. For new projects, Android Studio creates a sample C++ source file, native-lib.cpp, and places it in the src/main/cpp/ directory of your app module. This sample code provides a simple C++ function, stringFromJNI(), that returns the string "Hello from C++".

2. The External Build Files group is where you can find build scripts for CMake or ndk-build. Similar to how build.gradle files tell Gradle how to build your app, CMake and ndk-build require a build script to know how to build your native library. For new projects, Android Studio creates a CMake build script, CMakeLists.txt, and places it in your module's root directory.

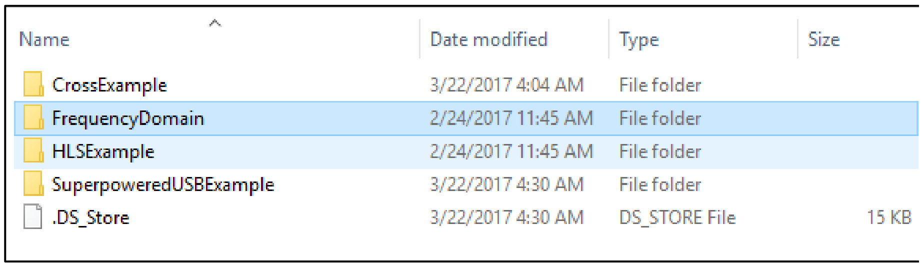
4. SUPERPOWERED SDK

- This is a low latency audio SDK for iOS and Android.
- Superpowered accomplishes this using patent-pending DSP optimization technology to achieve desktop grade performance on mobile devices.
- The Superpowered Audio SDK empowers developers to remove CPU resource limitations, and develop cross-platform audio for iOS, Android and wearable devices. It includes:

1. Example apps/projects for iOS, OSX and Android
2. Static library files
3. Decoder for MP3, AAC, WAV, AIFF and STEMS
4. Advanced Audio Player (including time stretching, pitch shifting, resampling, looping, scratching, etc.)
5. HTTP Live Streaming
6. Effects: echo, flanger, gate, reverb, rool, whoosh, 3 band equalizers, biquad IIR filters (low-pass, high-pass, bandpass, high-shelf, low-shelf, parametric, notch)
7. Dynamics: compressor, limiter, clipper.
8. Time Stretching and Pitch Shifting, resampler
9. Polar and Complex FFT
10. Recorder
11. Open-source audio system input/output classes
12. Time-domain to frequency domain class (including inverse)
13. Bandpass filterbank for time-domain frequency analysis
14. Audio analyzer: key detection, bpm detection, beatgrid detection, waveform generation, loudness/peak analysis
15. Stereo and mono mixers

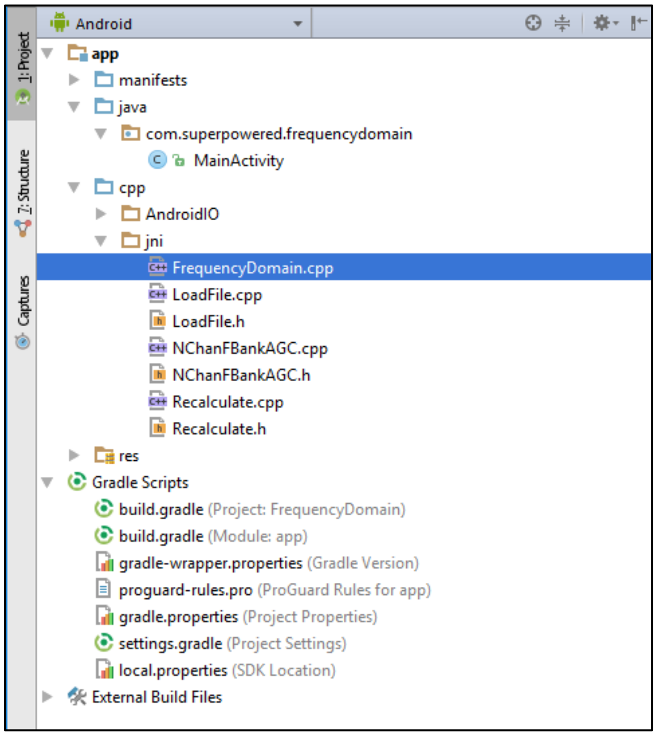
16. Simple audio functions (volume, volume ramp, peak, float-short conversion, interleaving and de-interleaving)

- The code is made as an open source and can be downloaded by going to the following link below,
  http://superpowered.com/
- Once it is downloaded you can go to examples_android folder as shown in Figure 7, to find various examples provided by superpowered that can be used and modified based on the requirements for our applications.
- When you click on the SuperpoweredFrequencyDomain as shown in Figure 6, the code can be opened on Android Studio directly.
- All modifications can be done in “FrequencyDomain.cpp” file present inside as shown in Figure 8, the data in time domain can be processed by us. Other cpp files can be added in jni folder.



Name	Date modified	Type	Size
CrossExample	3/22/2017 4:04 AM	File folder	
FrequencyDomain	2/24/2017 11:45 AM	File folder	
HLSExample	2/24/2017 11:45 AM	File folder	
SuperpoweredUSBExample	3/22/2017 4:30 AM	File folder	
.DS_Store	3/22/2017 4:30 AM	DS_STORE File	15 KB

Figure 7



```
app
  manifests
  java
    com.superpowered.frequencydomain
      MainActivity
  cpp
    AndroidIO
    jni
      FrequencyDomain.cpp
      LoadFile.cpp
      LoadFile.h
      NChanFBankAGC.cpp
      NChanFBankAGC.h
      Recalculate.cpp
      Recalculate.h
  res
Gradle Scripts
  build.gradle (Project: FrequencyDomain)
  build.gradle (Module: app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
External Build Files
```

Figure 8

- The input will be considered to be in time domain, then the input will be processed by our compression algorithm. The output in time domain will send back to the Superperered SDK.
- Refer the block diagram shown in Figure 9 for better understanding,

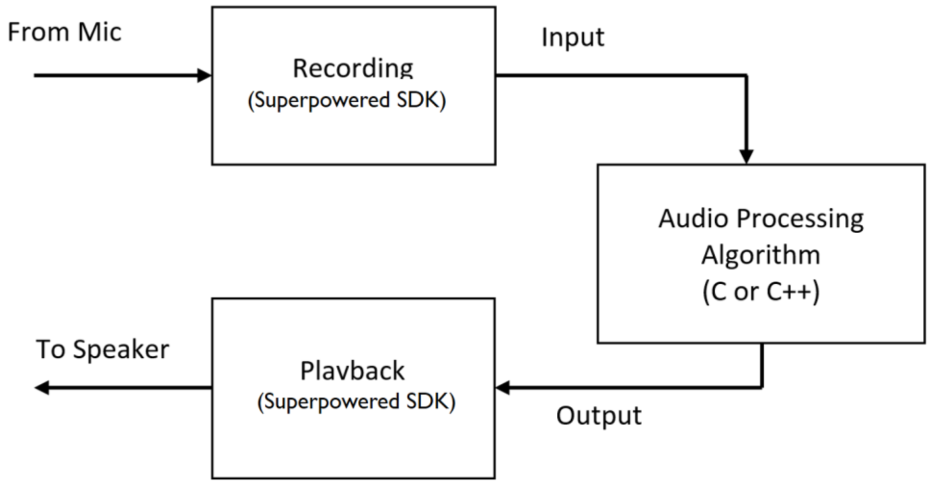
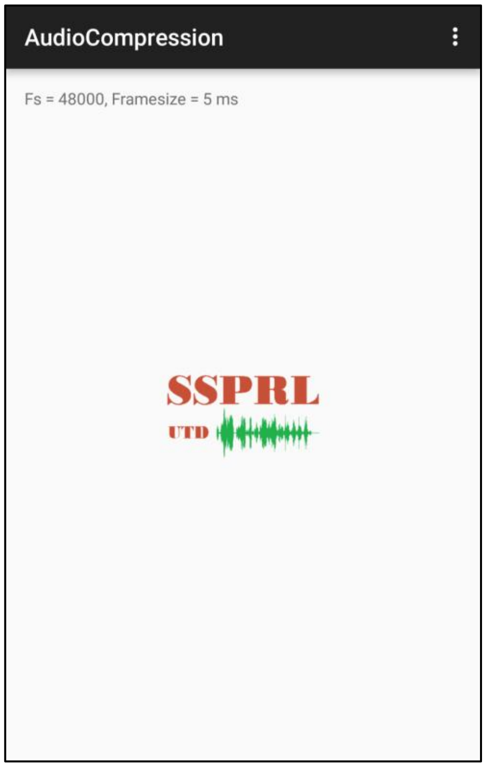


Figure 9

5. NINE-CHANNEL EQUALIZATION BASED DYNAMIC-RANGE COMPRESSION APPLICATION

- The algorithm is developed for nine-channel compression (Figure 11).
- Now the 1st version of this app is completed.
- In this version, nine channels filters banks are using for dynamic range compression.
- The default settings of this app is based on average subjective test results of hearing impaired subjects.
- The sampling frequency is fixed at 48K, the frame size is fixed at 10 ms.
- The steps to be followed once the app is installed on your iOS device are as follows,
    1. Make sure you have the hearing aid device paired to your iPhone or iPad.
    2. Click on the icon by the name AudioCompression, present amongst the apps.
    3. You will see a display as shown in Figure 10.



Fs = 48000, Framesize = 5 ms

Figure 10

4. Once you open the app, the real time audio compression starts.

- If you want to know more about the algorithm, do refer our paper, which is under review. All the codes will be made available in our website.

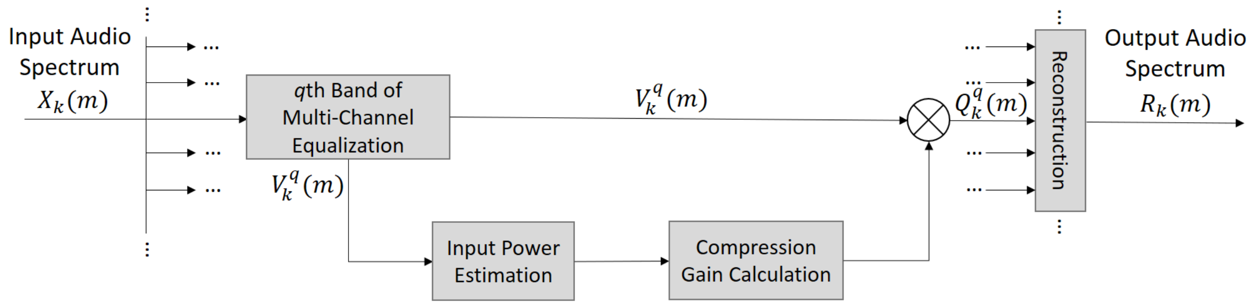

Input Audio Spectrum $X_k(m)$
qth Band of Multi-Channel Equalization
$V_k^q(m)$
$V_k^q(m)$
Input Power Estimation
Compression Gain Calculation
$Q_k^q(m)$
Reconstruction
Output Audio Spectrum $R_k(m)$

Figure 11