International Journal of Computer and Communication Technology

Volume 6 | Issue 4 Article 13

October 2015

DESIGN AND IMPLEMENTATION OF AUDIO/VIDEO CODEC BASED ON ANDROID PLATFORM

MUNEEB AHMED QURESHI
Embedded System Department, Aurora Technological & Research Institute, Hyderabad,
muneebmaq19@gmail.com

M.MADAN GOPAL
Electronics & Communication Engineering, Aurora Technological & Research Institute, Hyderabad,
mekala.madan@gmail.com

MOHAMMED SADIQ
Embedded System Department, Aurora Technological & Research Institute, Hyderabad,
snathick.sadiq@gmail.com

Recommended Citation
AHMED QURESHI, MUNEEB; GOPAL, M.MADAN; and SADIQ, MOHAMMED (2015) "DESIGN AND IMPLEMENTATION OF AUDIO/VIDEO CODEC BASED ON ANDROID PLATFORM," *International Journal of Computer and Communication Technology*: Vol. 6 : Iss. 4 , Article 13.
DOI: 10.47893/IJCCT.2015.1320
Available at: https://www.interscience.in/ijcct/vol6/iss4/13

DESIGN AND IMPLEMENTATION OF AUDIO/VIDEO CODEC BASED ON ANDROID PLATFORM

MUNEEB AHMED QURESHI[1], M.MADAN GOPAL[2] & MOHAMMED SADIQ[3]

[1,3]Embedded System Department, Aurora Technological & Research Institute, Hyderabad
[2]Electronics & Communication Engineering, Aurora Technological & Research Institute, Hyderabad
E-mail:muneebmaq19@gmail.com, mekala.madan@gmail.com, snathick.sadiq@gmail.com

Abstract—The Open Source Environment and API's Android not only created a boom in the market but also attracted large number of people to turn in to application developers. Android not only changed the technology but also help people how to get better with the means of technology This paper mainly showcase kernel level development with the help of java native interface technology; and FFmpeg (open source codec project) tool and associated libraries and at the end android based codec application is designed and implemented with the help of c/c++ and other languages

Keywords—*Android; Application development; Audio/Video Codec; JNI; Ffmpeg.*

I. INTRODUCTION

Google lunched a new mobile platform called Android which was designed especially for many purposes one is mobile devices in November 2007[1] Android is a free source mobile platform android is a Linux based multithreadeds multi process operating system Android is not a device or a product it is not even limited to cell phones . Android used a virtual machine which is mainly optimized for mobile devices named as Dalvik Virtual machine The Development environment includes a device Emulator, tools for debugging, memory and performance profiling and plug-in for the Eclipse IDE

Google Android released NDK, which stands as Native Development Kit, which was released on June 26, 2008,a component for SDK[2], The main function of NDK is to allow to compile c,c++ code to native machine code and to embed it with the respective application designed by the developer therefore large number of C,C++ open source projects can be transplanted to android platform to enhance functionality of the platform. This paper Basically concentrate on Audio/video Codec project FFmpeg's core libraries to Android platform

The sections of this paper are as follows: Section II Introduction to Android Architecture. Section III discusses the approach to kernel level component. Section IV Descriptions of FFmpeg in Android. Section V Android application development. Section VI Implementation of audio/video application based on android and the result in detail. Section VII Conclude the paper

II. ARCHITECTURE OF ANDROID

The Architecture of Android and Components are as follows which are shown in the Fig.1 as follows [3] [4].

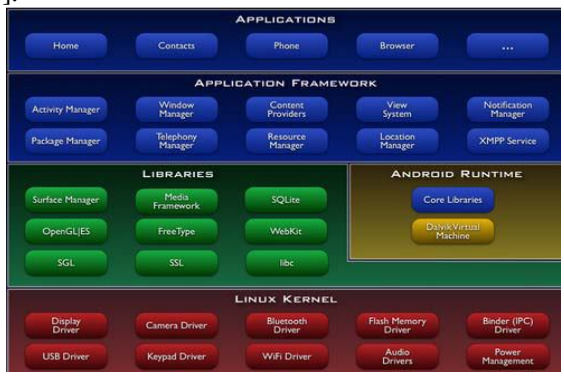


Figure 1. Android Architecture

A. Applications

All Compiled Java code and resources are bundled into an .apk file to form an application each .apk file is considered an application by default, every application runs in its own Linux process with its own JVM so code runs in isolation from other applications each application is assigned a unique Linux user ID with security set so other user IDs can't read the application's files. It's possible for two applications to share a single Linux user id.

B. Application Framework

Underlying all applications is a set of services and systems, including:
A rich and extensible set of views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data

A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
A Notification Manager that enables all applications to display custom alerts in the status bar
An Activity Manager that manages the lifecycle of applications and provides a common navigation back stack

C. Libraries

System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
Media Libraries - based on Packet Video's Open CORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG Surface Manager - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications LibWebCore - a modern web browser engine which powers both the Android browser and an embeddable web view.

D. Android Runtime

Dalvik VM Uses the Java language syntax, but does not provide the full-class libraries and APIs bundled with Java SE or ME. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model

E. Linux Kernel

Android which relies on Linux for core System services such as memory management, process management, network stack, and security.The kernel also acts as an abstraction layer between the hardware and the rest of the software stack. It can be difficult to maintain applications working on different versions of Android, because of various compatibility issues between versions 1.5 and 1.6, specifically concerning the different resolution ratios of the various Android phones.

III. KERNEL LEVEL DEVELOPMENT

A. JNI functioning in Android

JNI can help to write the programs in other languages in java languages refers to the ability to work together with other languages in the form of libraries(DLL,So) or executable files such as Assembly, C and C++. JNI [5] comes in action when

1) Programs or readymade class libraries written in other languages java programs can reuse them

2) All applications need to use system related functions while Java does not support or difficult to implement.

3) C and C++ languages are used for the implement of some features for higher performance and requirement

Android which supports JNI. Its library is written by C/C++ functions interacted by hardware and achieved through JNI. The project is concentrated on JNI for the reuse of C and FFmpeg Fig shows the position of JNI and other modules

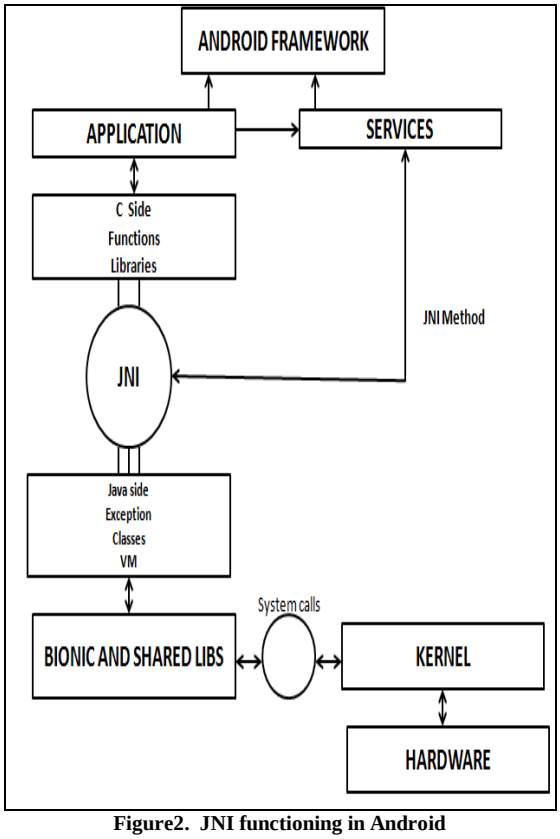


Figure2. JNI functioning in Android

B. The Steps of Kernel Level Development

The steps of kernel level development are as follows:

1) According to Android's Linux 2.6 kernel code kernel-level module and set aside interface for JNI functions.

2) Following the JNI rules Code JNI functions.

3) To compile in the form of libraries (*.So) use NDK compiler and load the libraries in to java application package.

4) Program UI and user handling procedures by java language under the rules of SDK.

.

IV. FFMPEG ALLOCATION

A. Introduction of FFmpeg

The FFmpeg tool and associated libraries (as of Natty, Ubuntu has switched from FFmpeg [6] to the libav fork) is the premier video decoding and encoding system on Linux (and in computing in general).

The FFmpeg tool is a command line program that can be used to encode from one of many dozen codec/formats into a similar number of other formats. The libraries from the project are available for developers to use in their own programs to provide video codecs, formats, devices, filters, scaling, and post-processing.

The first two things you can do with FFmpeg are to list out the formats and codecs that this copy supports. This may change based on what is installed on your computer, so it is best to check these before running a command, to make sure you have the correct support available. List all container formats: ffmpeg -formats List all codecs: ffmpeg -codecs
To convert a video, simply run the command "ffmpeg" with four additional parts:

```
ffmpeg [input] [video options] [audio options] [output]
```

The input part is composed of a "-i" and the name of the video you have that you want to convert to something else. You could have more than one of these input files (each one gets its own "-i") if for example you have a video with an audio track in a separate file.

```
ffmpeg -i InputVideo.mpg ...[video options] [audio options] [output]
ffmpeg -i InputVideoTrack.mpg -i InputAudioTrack.wav ...[video options] [audio options] [output]
```

The video options are where you specify the codec (with the "-vcodec" option) and bit-rate (with the "-b" option). In addition you can specify a video preset ("-vpre") which is essential for x264 encoding, and you can specify a size ("-s") with either a standard size reference or the format WIDTHxHEIGTH.

```
ffmpeg [input] -vcodec mpeg4 -b 3000k ...[audio options] [output]
ffmpeg [input] -vcodec libtheora -b 3000k ...[audio options] [output]
ffmpeg [input] -vcodec libx264 -vpre medium -b 3000k ...[audio options] [output]
ffmpeg [input] -vcodec mpeg4 -b 3000k -s hd480 ...[audio options] [output]
ffmpeg [input] -vcodec mpeg4 -b 3000k -s vga ...[audio options] [output]
ffmpeg [input] -vcodec mpeg4 -b 3000k -s 1280x720 ...[audio options] [output]
```

The audio options are where you specify the audio codec ("-acodec") and bit-rate ("-ab").

```
ffmpeg [input] [video options] -acodec flac ...[output]
ffmpeg [input] [video options] -acodec libmp3lame -ab 256k ...[output]
ffmpeg [input] [video options] -acodec libvorbis -ab 192k ...[output]
ffmpeg [input] [video options] -acodec -libfaac -ab 192k ...[output]
```

The output is where you specify the filename that the converted video will go into. Usually the extension of the filename (.mkv, .mp4, .avi, etc) will allow the program to determine what format the file will be written as, however if it is ambiguious or you want to use format that isn't tied to that file extension, you can use "-f" and the format name.

```
ffmpeg [input] [video options] [audio options] OutputVideo.mkv
ffmpeg [input] [video options] [audio options] OutputVideo.mp4
ffmpeg [input] [video options] [audio options] -f dvd OutputVideo.mpg
ffmpeg [input] [video options] [audio options] -f matroska OutputVideo.vid
ffmpeg [input] [video options] [audio options] -f mp4 OutputVideo.vid
ffmpeg [input] [video options] [audio options] -f avi OutputVideo.vid
```

The result could look like:

```
ffmpeg -i InputVideo.mpg -vcodec mpeg4 -b 3000k -s hd480 -acodec flac OutputVideo.mkv
```

B. FFMPEG Cross- compiling

This paper transplant avcodec, avformat core modules of FFmpeg. All the user needs to download the FFmpeg source file. The source code is cross compiled in order to use in ARM Linux. Which is provided by NDK? Modify the files as follows:

```
#vi configure
prefix="/home/arm/video/libffmpeg"
cross_prefix="/usr/local/android-ndk-1.5_r1/build/prebuilt/linux-x86/arm-eabi-4.2.1/bin"
cc="arm-linux-gcc"
ar="arm-linux-ar"
```

Then run the command

```
./configure--cpu=armv5te--enable-static--enable-version3--enable-encoder=amr_nb--enable-decoder=amr_nb
```

After configuration programmer commands make commands make install. For next development system will create the static libraries the static libraries are avutil, avcodec.a and avformat.a.

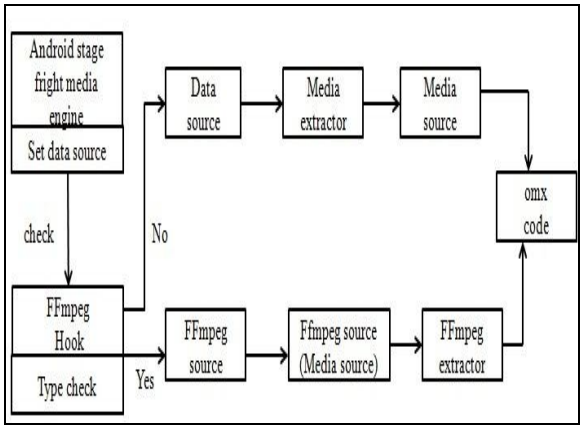


Figure3. FFmpeg on Android Flowchart

V. ANDROID APPLICATION DEVELOPMENT

The application which is developed by the developer is done with the help of the code which is written in eclipse which is only possible if and only of android sdk files are install in the eclipse software the sdk files can be downloaded according to the background if it is Linux then the Android website allows to download Linux android sdk format and same applies to windows the result can be seen on emulator once the application is developed which is error free the output of the file is .apk which can be installed in to the mobile devices and registered in to the Android market store for use of all android users as paid or free application.

The Block diagram below describe the application development

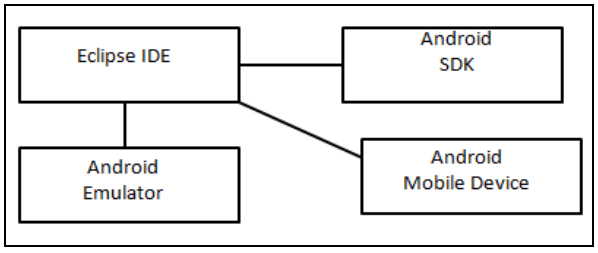


Figure4. Blocks of Android application

The Android Manifest and resource XML are generated are generated classes can be taken through java source and android libraries as shown in figure5 the generated classes are given to java compiler which is complied as .Dex file the operation is performed under Dalvik virtual machine.

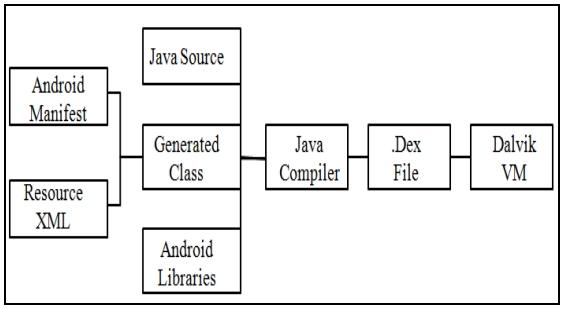


Figure5. Android Code Development

VI. ANDROID BASED AUDIO/VIDEO CODEC DEVELOPMENT

A. Audio/Video Codec Programming

The approach of audio/video codec programming is described by means of analyzing and encoding the code as follows.

a) Encode Flowchart

The main flow of encode process is shown in Fig. 6.

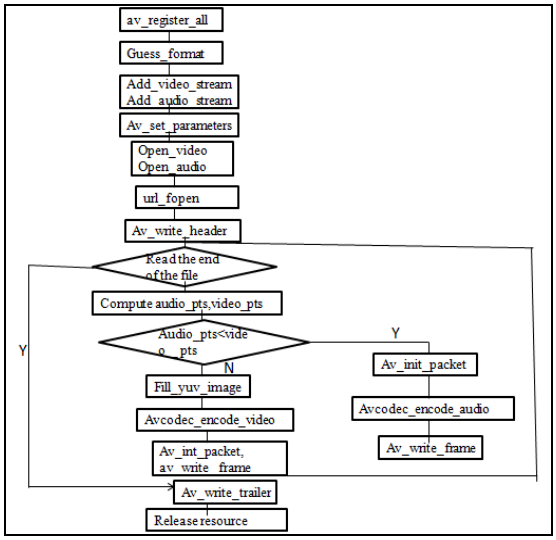


Figure6. Encoding flowchart

b) Main data structures

The following structures are mainly used in encoding process:

```
AVFormatContext *oc;
AVOutputFormat *fmt;
AVStream *audio_st, *video_st;
AVFrame *picture  ;
AVCodecContext *c;
AVCodec *codec; //both video and audio
AVPacket pkt;
int16_t *samples;
```

c) Registration

```
//Register all the available codecs
av_register_all();
//Get the format information of the output file
fmt=guess format(NULL, ofilename, NULL);
```

d) Initialize the AVFormatContext

```
oc = av_alloc_format_context();
oc->oformat = fmt;
```

e) Set basic parameters

```
Initialize audio/video stream,set basic parameters.
video_st=add_video_stream(oc,fmt->video_codec);
audio_st=add_audio_stream(oc,fmt->audio_codec);
Set parameters of AVFormatContext.
av_set_parameters(oc, NULL);
```

f) Set encoder

```
codec = avcodec_find_encoder(c->codec_id);
avcodec_open(c, codec);
```

g) Resource allocaton

```
Video buffer
picture=alloc_picture(c->pix_fmt,c->width,c->height
);
Audio buffer
samples=av_malloc(audio_input_frame_size*c->cha
nnels)
```

h) Begin encoding

```
av_init_packet(&pkt);
fill_yuv_image(picture,video_frame_count,fp2,c->wi
dth, c->height);
avcodec_encode_video(c,video_outbuf,video_outbuf
_size, picture);
```

i) End of encoding

```
av_write_trailer(oc);
for(i = 0; i < oc->nb_streams; i++) {
av_freep(&oc->streams[i]->codec);
av_freep(&oc->streams[i]);
av_free(oc);
```

j) Called when activity is created

```
private static final String TAG = "MediaPlayer";
private static final int SELECT_MUSIC = 1;
private MediaPlayer mp;
private SurfaceView mPreview;
private SurfaceHolder holder;
private Button btnrestart;
private Button btnStop;
private TextView tv;
private SeekBar seekBarProgress;
private TextView tvtime;

private Timer updateTimer;
private String mPath=null;

@Override
public boolean onCreateOptionsMenu(Menu menu)
{
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.file_menu, menu);
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem
item) {
// Handle item selection
switch (item.getItemId()) {
case R.id.file_s:

Intent intent = new Intent();//intent for launching
action for pick up video file
intent.setType("video/*");
intent.setAction(Intent.ACTION_GET_CONTENT);
startActivityForResult(Intent.createChooser(intent,
"Select Music"), SELECT_MUSIC);
Log.i("fila name","selectedMusic");
```

k) Set the surface for the video output

```
mp.setDisplay(mPreview.getHolder());
```

l) Change progress of mediaController

```
private void updateMediaProgress()
updateTimer = new Timer("progress Updater");
updateTimer.schedule(new TimerTask()

@Override
public void run()
runOnUiThread(new Runnable()

@Override
public void run()
seekBarProgress.setProgress(mp.getCurrentPosition(
)/1000) ;
```

After complrting the audio video codec JNI methods will call the codec modules to encode and decode on Android platform.

B. Writing JNI Functions

Two native methods are declared in java layer

```
public native String encode();
public native String decode();
```

These native methods are implemented with the help of C language

C. Compiling JNI Library

After finishing the native code, we will write the Android.mk file of NDK. The Android.mk must include those three static libraries which have been compiled before, compile the code and libs to a dynamic library named “libendec.so”,and then put it in the folder named “libs” in the application.

D. After Finishing Java application and loading JNI libs

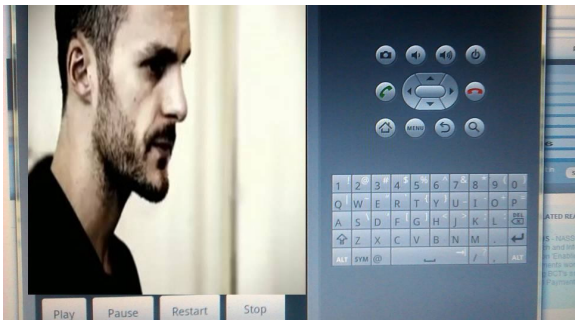
Program the front UI and user handling procedures modules according to Android SDK, use method loadLibrary(“endec”) to load the compiled dynamic library, then java application can call the native methods as normal method.

So far the paper has shown the application availability and coding work.

E. Results

Program interface is shown in Fig. 7. There are four buttons in the UI, play pause restart stop and a seek bar to play both raw audio and video file the application is also interfaced with the arm 11 board the figure can show the arm 11 board as a result the program can play all the video files in the android OS specific boards.

The interface file is shown in the figure 7.



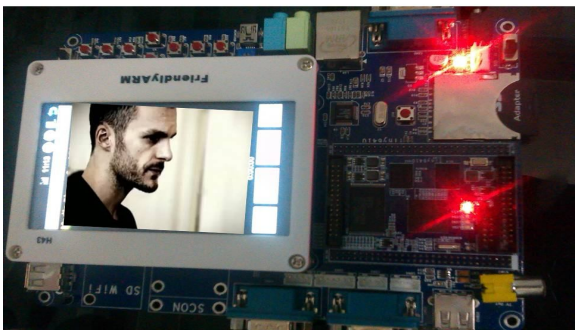


Figure 7. Play Interface of the File on emulator and arm11s3c6410 board

VII. CONCLUSION

As shown in the Figure &, the play effect that indicate the integration of ffmpeg code libraries java code application framework does well. Thus the paper indicates the feasibility of allocating open source project to android platform. The future work is to design a HD recording video surveillance system based on android platform.

REFERENCES

[1] OpenHansetAlliance, http://www.openhandsetalliance.com/.

[2] N. Gramlich, Android Programming, PDF Electronic Book, 2008. Available from: http://androidos.cc/dev/index.php.

[3] Android Developers, http://www.androidin.com/.

[4] Zhifeng Jiang. The quick codec development method of FFmpeg.Microcontrollers & Embedded Systems, 2008, pp. 169–71.

[5] Android-An Open Handset Alliance Project, http://code.google.com-/intl/zh-CN/android/

[6] C. Haseman, Android Essentials, PDF Electronic Book, 2008. Available from: http://androidos.cc/dev/index.php.

◈◈◈