UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

INGENIERO DE TELECOMUNICACIÓN

DEPARTAMENTO DE INFORMÁTICA

PROYECTO FIN DE CARRERA Nº 3123121

TELEVISION REMOTE CONTROL APPLICATION FOR A TERMINAL TYPE ANDROID

LIDIA VALBUENA GARCIA
MARZO 2012

ÁNGEL NEIRA ÁLVAREZ

RESUMEN DEL PROYECTO

El objetivo del proyecto es diseñar y desarrollar una aplicación software descargable desde un sitio web.

El usuario de un terminal tipo "Android" conectado a internet puede identificarse en este sitio y descargar la aplicación. Esta aplicación puede comunicarse con un decodificador digital instalado con el sistema operativo Media-OS a través de internet. El usuario puede entonces acceder al menú del decodificador digital para escoger el contenido que desea ver: sus programas favoritos, fotos, videos…

El proyecto ha sido llevado a cabo en una empresa llamada EM-SYS EMBEDDED MULTIMEDIA SYSTEMS. Esta compañía proporciona un innovador software para dispositivos multimedia digitales, mejorando la visualización de contenidos multimedia desde la red doméstica. El proyecto ha sido supervisado por Abdelhak Radouani, director ejecutivo de EM-SYS.

La aplicación fue desarrollada usando Eclipse para Java y añadiendo diferentes plugins para integrar características específicas del sistema operativo Android. Para ello se utilizó el SDK de Android que integra varios simuladores. Para extender las posibilidades de la aplicación se añadió otra herramienta: el NDK de Android. Esta herramienta permite añadir trozos de código nativo. Proporciona encabezados y librerías para la programación con C o C++.

Este proyecto fue supervisado además por Franck Rousseau, tutor de la escuela nacional superior de informática y de matemáticas aplicadas de Grenoble (École Nationale Supérieure d'Informatique et de Mathématiques Appliquées - Grenoble INP – Ensimag).

La televisión digital tiene muchas ventajas sobre la televisión analógica. Lo más significativo es que los canales digitales toman menos ancho de banda. Esto significa que los operadores pueden proveer más canales en el mismo espacio. Este hecho facilita la retransmisión de televisión digital de alta definición u otros servicios multimedia o interactivos.

La televisión digital también permite servicios especiales como el multiplexing (más de un programa en el mismo canal), guías de televisión digitales e idiomas adicionales (tanto hablados como subtitulados).

Lo que el cliente pide a día de hoy es disfrutar de los canales y el contenido multimedia que tienen en sus casas cuando están fuera (place-shifting). El principio es que los consumidores pagan una sola vez a su proveedor de contenidos digitales y pueden acceder a esos contenidos desde cualquier otro lugar. Por lo tanto, se convierte en imprescindible que los operadores de televisión de pago puedan ofrecer sus servicios en cualquier tipo de dispositivo multimedia.

Uno de los sistemas operativos más populares en la actualidad es Android. Es un sistema operativo basado en Linux enfocado a dispositivos móviles como smartphones y tablets. Ha sido desarrollado por Google.

Ahora mismo hay más de 450.000 aplicaciones disponibles para Android y se estima que el número de aplicaciones que se podrán descargar desde el Android Market en 2014 sean de más de 1 millón.

Android fue la plataforma de smarthphones más vendida en todo el mundo con más de 300 millones de dispositivos en uso.

Éstas son las razones por las que el tema de este proyecto es tan interesante.

Por lo tanto, lo que se plantea es que el dispositivo Android reciba un stream, enviado por el decodificador situado en nuestras casas, bajo el protocolo rtsp o http. En este flujo de información podremos encontrar los diferentes canales de video, audio, subtítulos o información. El usuario será capaz de elegir cuál de ellos reproducir.

Será importante tener en cuenta, que al ser servicios de pago, posiblemente la información deberá ser codificada por el decodificador digital y decodificada por el dispositivo móvil.

También habrá que tener en cuenta la codificación de la información. Será necesario llegar a un acuerdo para que la información a enviar al dispositivo no sea demasiado pesada pero manteniendo la calidad de la información. Ésta codificación tendrá que ser compatible tanto para el sistema operativo del decodificador como para Android.

El proyecto está planeado para ser realizado en cinco pasos:

- Documentación e instalación del entorno de trabajo
- Desarrollo de un reproductor de video simple
- Desarrollo de un cliente de streaming basado en un protocolo rtsp o http
- Desarrollo de un demultiplexor para extraer los diferentes canales
- Fusión de los desarrollos previos

CONTENT

1. CONTACT

Student
Lidia Valbuena García
3 Place de Gordes 38000 Grenoble
T. +34 600 38 25 75
Lidia.valbuena.garcia@gmail.com

Responsible in the company
Abdelhak Radouani
27 Rue du Tour de L'Eau 38000 Grenoble (France)
T. +33 665 97 87 18
aradouan@em-sys.fr

Tutor of the Grenoble INP - Ensimag
Franck Rousseau
Bâtiment de l'Ensimag, bureau D313, Grenoble (France)
Franck.Rousseau@imag.fr

Tutor of the University of Oviedo
Ángel Neira Álvarez
Desp. 1.b.8, Departamento de Informática,
Campus de Viesques S/N, Xixón, Asturies (Spain)
T. +34 985 18 24 81
neira@uniovi.es

2. SUMMARY

The objective of the project is designing and developing a software application downloadable from a website.

The user of a terminal type "Android" connected to the internet can log onto this site and download the application. This application can communicate with the digital decoder installed with the Media-OS through the Internet. The user can then accede to the menu of the digital decoder to choose the content he wants to watch: his favorite show, photos, videos, etc…

The project will be carried out in a company called EM-SYS. This company provides an innovative software framework for Digital Multimedia Devices, enhancing multimedia video streaming over home networks. The project will be supervised there for Abdelhak Radouani, chief executive officer.

The application will be developed using Eclipse for Java adding a plugin to integrate Android's specific characteristics and which also integrates several emulators. This tool is the android SDK. To extend the possibilities of the application it will be added another tool: the Android NDK. It lets build portions in native code. It provides headers and libraries when programming in C or C++.

Keywords: Android, Media-OS, streaming.

3. INTRODUCTION

DTV (Digital TV) has several advantages over analog TV. The most significant is that digital channels take up less bandwidth, and this means that digital broadcasters can provide more digital channels in the same space, provide high-definition television service, or provide other non-television services such as multimedia or interactivity. DTV also permits special services such as multiplexing (more than one program on the same channel), electronic program guides and additional languages (spoken or subtitled).

What clients demand nowadays is enjoying their linear channels and content from home when they are staying away (place-shifting). The principle is that customers pay once for their service provider content and can access it anywhere. Therefore it will become imperative that Pay TV operators can deliver their paid services to all popular devices.

One of the most popular devices is Android. It is a Linux-based operating system for mobile devices such as smartphones and tablet computers and it is developed by Google.

Nowadays there are more than 450,000 applications available for Android, and the estimated number of applications downloaded from the Android Market as of December 2011 exceeded 10 billion.

Android was listed as the best-selling smartphone platform worldwide with over 300 million Android devices in use.

These are the reason why the topic of this project is so interesting.

Therefore, the device Android will receive a stream, sent by the decoder placed in our house, under a protocol rtsp or http. In this flow of information we could find the different channels of video, audio, subtitles or information. The user will be able to choose which of them to reproduce.

It will be important to note that, due to these services are surcharge, maybe the information must be encoded by the digital decoder and decoded by the mobile device.

It is also necessary to take into account the information encoding. It will be necessary to reach an agreement so that the information to be sent to the device is not too heavy but maintaining the quality of the information. This encoding must be compatible for both the decoder operating system and Android.

4. EM-SYS EMBEDDED MULTIMEDIA SYSTEMS

Em-Sys provides a comprehensive range of support services centered around digital multimedia software including high level consulting to define the best architecture or select the appropriate software solutions to respond to the rapid evolution of the market.

This company develops any piece of software or full application for any type of devices like Satellite, Cable Terrestrial TV as well as IP protocol with standard or High Definition TV. They can easily and quickly develop around Multimedia standards like: MPEG2, MPEG4, DVB-S/S2, DVB-T/T2, DVB-H, DVB over IPDVB SI/PSI, DVB Teletext, DVB Subtitling, DVB-Datacarousel, etc.

The world of Digital Multimedia is in permanent evolution and consumers want to watch TV anywhere at any time. Existing solutions are limited as they are not planned for sharing video in general. Em-Sys has anticipated the needs for more flexibility and higher performances.

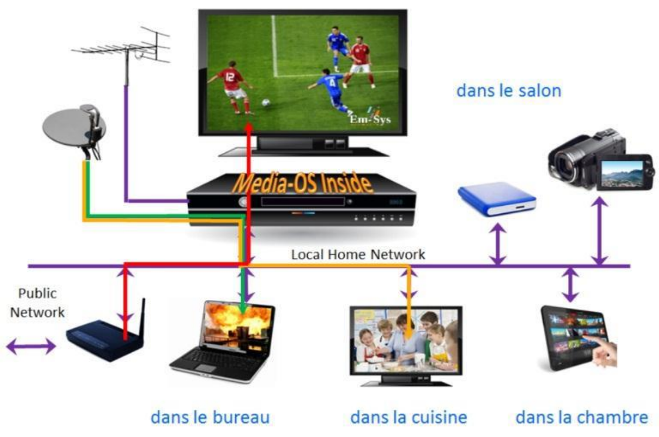


Figure 1.- The Media OS product

The new architecture developed by Em-Sys significantly enhances the performance for video streaming using current chipsets, allowing delivery of High Definition content over home networks with current and future chipsets available on the market.

The idea is that this new digital decoder expands the range of services from operators to its customers. EM-SYS proposes to create something like a home network in which devices are all connected. Some of them will provide content, such as the antenna or a camera, while others will play it, such as televisions or smartphones.

5. WORK PLAN

The project is planned to be realized in five steps:

- The first one of documentation and accustoming to the environment of work. Also knowing well the tools of work and being able to install applications in a tablet Android where the tests will be realized.
  A general study of the Android operating system will be done to explore possibilities.

- In the second one there will be programmed a video player from the local source of the terminal putting special interest to reproduce those of extension .ts since it will be the extension that it will be received finally.
  For this purpose it will be necessary to explore several libraries to decode the information. Open-source libraries will be preferred over the payment ones.

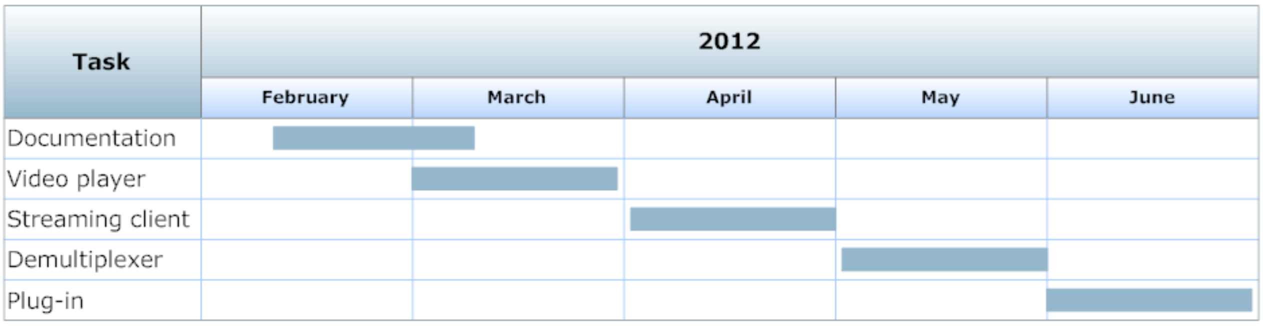
- In the third step there will be programmed a streaming client which will receive the information by rtsp or http protocol from the digital decoder. This means that the application obtained in the previous step will be modified to play over the network instead of reading from a file saved in the memory of the tablet.
  At this point it will be necessary to see the capacity of the device to decode and play content on the fly. It will be also necessary to decide which will be the highest quality that the device is capable to support. This means how much definition the device will be able to play without stopping the playing of the stream.
  At first, the content encoding will not be treated. It will be a topic to be discussed later.

- In the fourth one a demultiplexer will be realized to extract the different tracks of audio, video and information contained in the signal based on the

DVB standard. This way we will be able for example to choose the different available languages for the audio or the subtitles.
We are not sure this is the optimal solution. It will be necessary to study other possibilities. Maybe the decoder will be able to send only the content requested by the device. By this way, the performance in the Android system will be better. The decoder is supposed to be more powerful.

- In the last step everything will be joined and the application will be transformed into a plug-in for the Android navigator to facilitate its utilization to the users. The plugin will identify the address of our digital decoder when it will be introduced in the Android navigator and the information will be played in the window.

Table 1.- Work plan



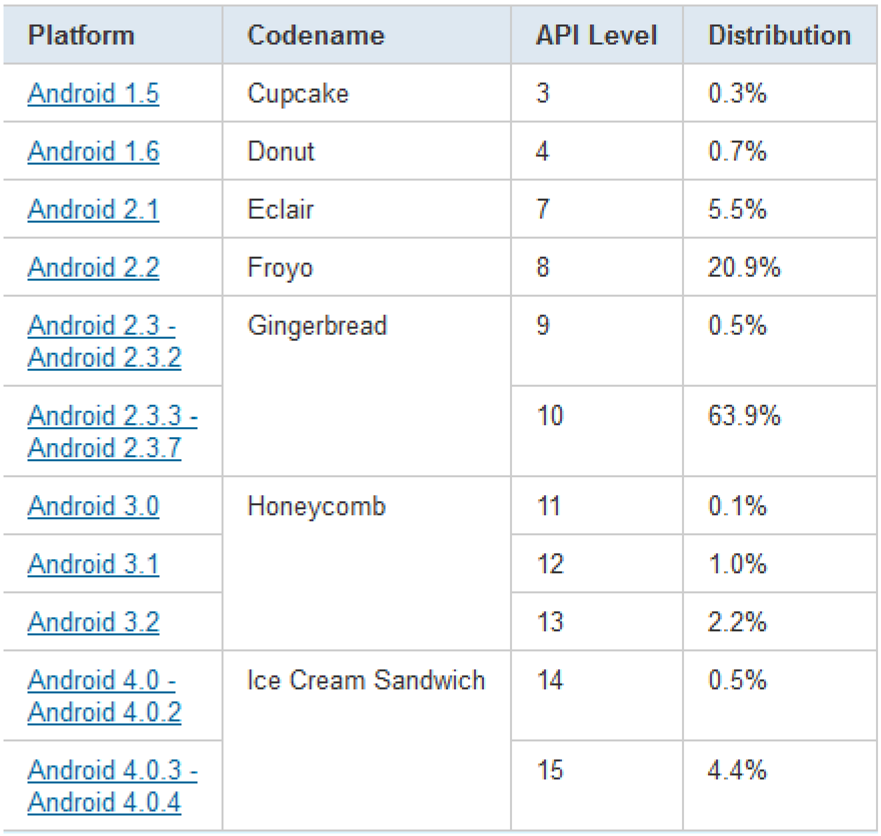
Task	2012				
	February	March	April	May	June
Documentation	▨	▨			
Video player		▨			
Streaming client			▨		
Demultiplexer				▨	
Plug-in					▨

6. GOOGLE’S ANDROID OPERATING SYSTEM

Android is an operating system specially designed for mobile devices which has rapidly become the fastest-growing mobile OS. It is based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets.

Android’s source code is released by Google under open source licenses. Android is popular with technology companies that require a low-cost and customizable operating system. Table 2 reviews Android’s version history.

Table 2.- Android’s version history



Platform	Codename	API Level	Distribution
Android 1.5	Cupcake	3	0.3%
Android 1.6	Donut	4	0.7%
Android 2.1	Eclair	7	5.5%
Android 2.2	Froyo	8	20.9%
Android 2.3 - Android 2.3.2	Gingerbread	9	0.5%
Android 2.3.3 - Android 2.3.7		10	63.9%
Android 3.0	Honeycomb	11	0.1%
Android 3.1		12	1.0%
Android 3.2		13	2.2%
Android 4.0 - Android 4.0.2	Ice Cream Sandwich	14	0.5%
Android 4.0.3 - Android 4.0.4		15	4.4%

Applications (“apps”), which extend the functionality of devices, are written using the Android software development kit (SDK) and, often, the Java programming language that has complete access to the Android APIs. The SDK includes a comprehensive set of development tools, including a debugger, software libraries, an emulator, documentation, sample code and tutorials.

Applications are usually released via the Android Market, Google’s official online store. Publication of the applications is not restricted, allowing installation from any other source.

Android has an active community of developers and enthusiasts to develop and distribute their own modified versions of the operating system and applications.

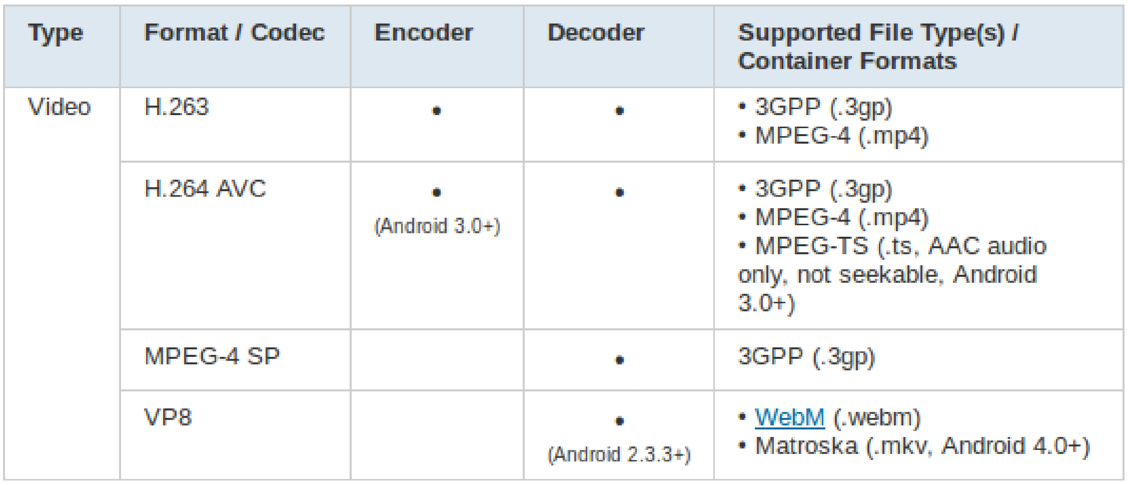
To date, there are more than 200 million activated Android devices, and every day more than 550,000 new devices are activated in more than 137 countries and regions. Android gives you access to the latest technologies and innovations across a multitude of devices.

6.1. Android Supported Media Formats

The table below describes the media format support built into the Android platform. Note that any given mobile device may provide support for additional formats or file types not listed in the table.

It is essential to achieve that the Android application plays the same media formats as our decoder.

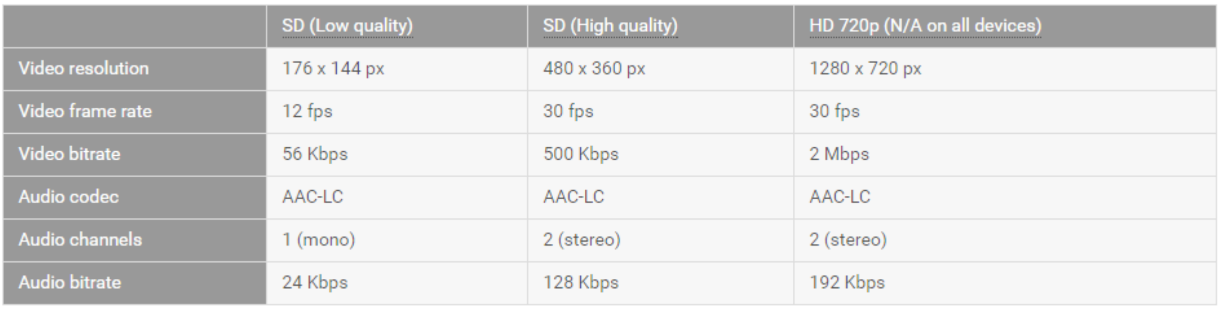
Table 3.- Android Supported Media Formats



Type	Format / Codec	Encoder	Decoder	Supported File Type(s) / Container Formats
Video	H.263	•	•	• 3GPP (.3gp) • MPEG-4 (.mp4)
	H.264 AVC	• (Android 3.0+)	•	• 3GPP (.3gp) • MPEG-4 (.mp4) • MPEG-TS (.ts, AAC audio only, not seekable, Android 3.0+)
	MPEG-4 SP		•	3GPP (.3gp)
	VP8		• (Android 2.3.3+)	• WebM (.webm) • Matroska (.mkv, Android 4.0+)

Below, lists examples of video encoding profiles and parameters that the Android media framework supports for playback in the H.264 Baseline Profile codec.

Table 4.- Video Encoding Recommendations



	SD (Low quality)	SD (High quality)	HD 720p (N/A on all devices)
Video resolution	176 x 144 px	480 x 360 px	1280 x 720 px
Video frame rate	12 fps	30 fps	30 fps
Video bitrate	56 Kbps	500 Kbps	2 Mbps
Audio codec	AAC-LC	AAC-LC	AAC-LC
Audio channels	1 (mono)	2 (stereo)	2 (stereo)
Audio bitrate	24 Kbps	128 Kbps	192 Kbps

6.2. Network Protocols

The following network protocols are supported for audio and video playback:

- RTSP (RTP, SDP)
- HTTP/HTTPS progressive streaming
- HTTP/HTTPS live streaming draft protocol:
    - MPEG-2 TS media files only (Appendix A)
    - Protocol version 3 (Android 4.0 and above)

- Protocol version 2 (Android 3.x)
- Not supported before Android 3.0

Note: HTTPS is not supported before Android 3.1.

6.3. Developing

The first step in developing for Android is downloading the SDK (Software Development Kit), which has the tools you need to build and publish apps. It includes Android platform versions, add-ons, tools, samples, and documentation.

It is also needed to install some drivers for the tablet. The Motorola Xoom (Android version 3.2) is the tablet to be used to install and prove the applications.

To verify that everything is installed correctly I created a simple first application that consists of a menu with buttons.

The first big problem to solve is that Android does not support some of the media formats that the decoder will send. Due to this fact it is needed to include an external library. Libraries written in C and other languages can be compiled using the Android NDK (Native Development Kit).

The Android provides headers and libraries that allow you to build activities, handle user input, use hardware sensors, access application resources, and more, when programming in C or C++. Writing native code does not result in an automatic performance increase, but always increases application complexity.

Two possible solutions are FFmpeg (Appendix B) and MEncoder. Both are libraries to record, convert and stream audio and video.
FFmpeg is chosen as the solution because it includes a variety of codecs wider and in addition it has a license for developers. But as disadvantages it presents a very lim-

ited documentation and that there is not a version for Android. It is necessary to programme some glue code.

7. DESIGN AND IMPLEMENTATION

In this paragraph I am going to present the three solutions I have been working with and the problems or disadvantages of each one of them. The three solutions use FFmpeg libraries to be able to decode the information but in different ways.

7.1. First solution

The first solution is to use the Android classes to play the video and the audio. So, an information flow will be sent from the digital decoder, placed in our house, over the network (rtsp or http protocol). This flow will be received and process by the FFmpeg library.

These are the steps of the FFmpeg code:

- the information is received over the network
- the different streams are identified and separated
- the user chooses the streams which want to play
- FFmpeg chooses the best decoders for each stream
- each selected stream is decoded independently
- the decoded video stream is sent to an Android class called Bitmap
- the decoded audio stream is sent to an Android class called AudioTrack

Due to the fact that FFmpeg uses C code and Android uses Java code, it is needed to utilize the Android NDK between these two parts of code in order that they understand each other.

When the Bitmap class receives the decoded video stream into a buffer, the Java code starts to copy each pixel information from the buffer to the screen. The AudioTrack class also receives a buffer with the information and it sends it to the speakers.

The figure below shows a scheme of how it works.

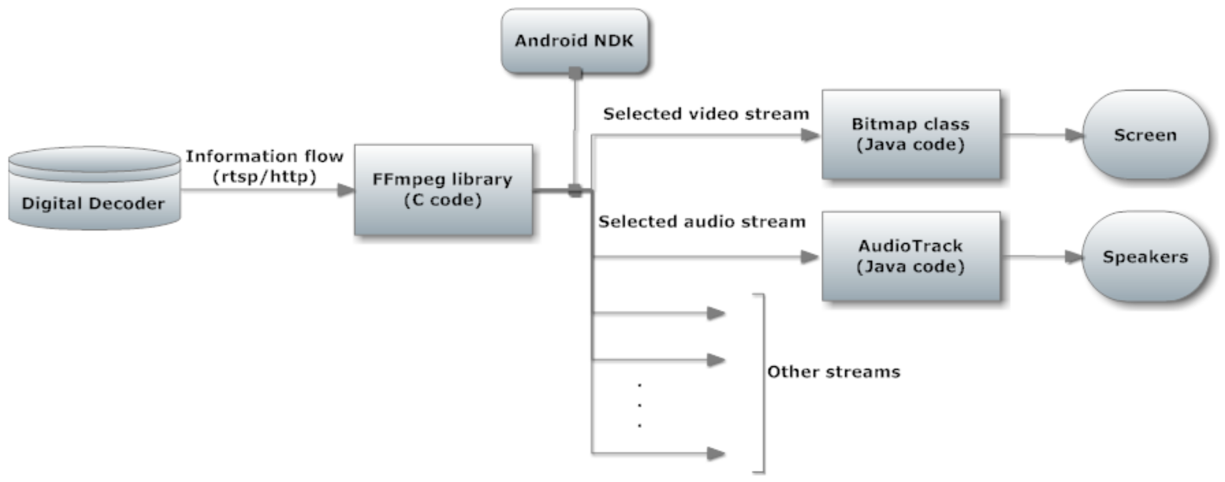


Figure 2.- Scheme of the first solution

Though at first it looks like a good solution there are some problems that I was not able to correct.

First of all, I could not find a good configuration of the FFmpeg library to receive a stream over the network. The documentation of FFmpeg is really wide but it is almost all referred to the version created for Linux. There is not a version for Android and therefore there is not documentation for this topic. I have done several compilations changing all the options I supposed that they were necessary. None of these changes gave the expected result.

On the other hand, FFmpeg is able to read a file saved in the memory of the tablet. Playing this file I encountered two more problems: one for the video and another one for the audio.

As I explained previously, the Bitmap class copies each pixel from the buffer to the screen. The code is executed too slowly while obtaining the information and this gives as result that the video is not fluently played even though the quality of the video is good.

As for the audio, it was impossible to find the correct way to read from the buffer of decoded information using AudioTrack. The steps executed by FFmpeg looks to be correct. But everything what was obtained was noise. I am not totally sure of the reason of this problem. I decline myself to thinking that I am not using correctly AudioTrack class though it is also possible that FFmpeg does not decode properly the audio stream.

7.2. Second solution

The second solution is to replace the Android classes to play the video and the audio with a new library called SDL (Appendix C). The FFmpeg version for Linux uses this library to play audio and video after being decoded. As FFmpeg, SDL is an open source library with lots of possibilities. This is the reason why I thought it would not be a bad solution. The main problem is that it does not have a version for Android so it was necessary to add some new code to compile it.

The FFmpeg version for Linux provides several tools. The most interesting for this project is ffplay that allows playing a file using the FFmpeg library for the decoding and the library SDL for the playback. Taking this tool as an example, the code was modified to use the SDL library in the place of the Android Bitmap and AudioTrack classes.

The first steps of the solution are exactly the same. The information flow will be sent from the digital decoder over the network (rtsp or http protocol) and his flow will be received and process by the FFmpeg library.

FFmpeg code will receive the information over the network, it will identify the different streams, it will let the user to choose the streams to decode, it will choose the best decoders for each stream and it will decode each stream independently. From here it is when the solution changes.

The selected streams are sent to the SDL library to play them.

In this case the whole code of this application is written in code C. Nevertheless it is necessary to programme the initialization of the application in code Java. For this reason, it is also needed to use the Android NDK for a correct working.

The figure below shows a scheme of how it works.

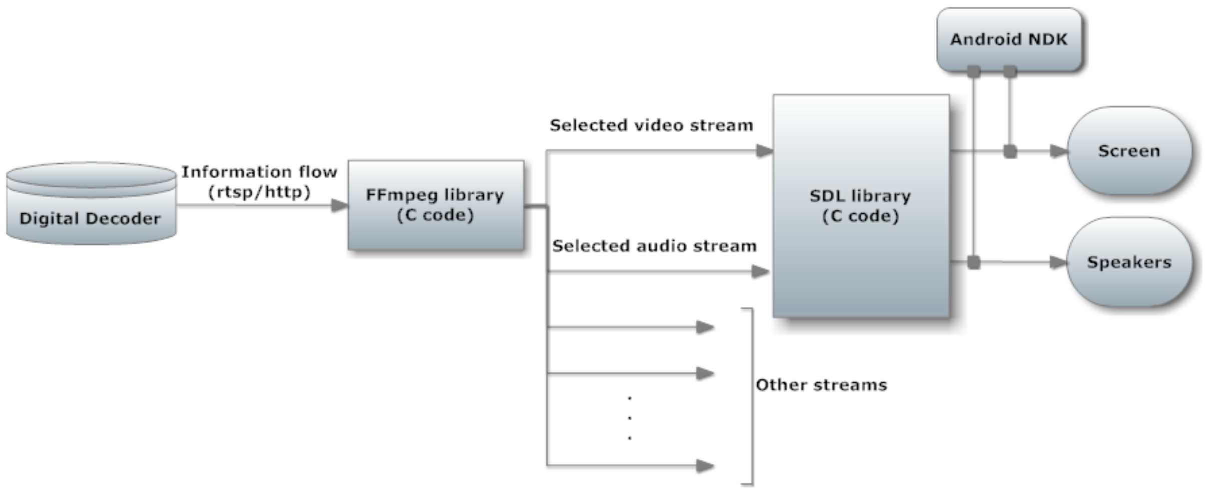


Figure 3.- Scheme of the second solution

As in the previous solution, I could not find a good configuration of the FFmpeg library to receive a stream over the network. Playing a file saved in the memory of the tablet I encountered another problem with the audio.

In this case, unlike the previous one, I was not able to play any sound, not even noise. I believe that the problem is the configuration of the SDL library. As FFmpeg, SDL is an open source library under development. There is a huge amount of information for the version used by Linux. But there is not a version for Android and therefore there is not documentation. As I did before with FFmpeg library, I make several compilations of the code changing the options but I did not find the correct configuration. According to some opinions of developers with the same problem it is possible that the reason is that the SDL library is not able to accede to the speakers of the tablet due to some security protocols of the hardware of the device. From my point of view, I believe that it is more probably that it is an erroneous configuration of the SDL library.

Playing the video, SDL works perfectly improving the problems obtained with the Android Bitmap class. It plays the video fluently and with a good quality.

7.3. Third solution

This third solution is the simplest one and the only one of the three solutions that works properly, playing audio and video with good quality and perfectly synchronized.

This solution consists of using a tool called Vitamio.

Figure 4.- Vitamio logo

Vitamio is a multimedia framework for all Android devices. Vitamio works like the Android's default MediaPlayer except that it includes much more powerful features.

This tool is used as a plug-in in the applications. Replacing Android's classes with the supplied ones by Vitamio, it is possible to play almost all the formats (also streaming). FFmpeg provides the software codecs and demuxers.

The following network protocols are supported for audio and video playback:

- MMS
- RTSP (RTP, SDP)
- HTTP progressive streaming

- HTTP live streaming (M3U8), for Android 2.1+

Many audio and video codecs are packed into Vitamio beside the default media format built in Android platform: divx/xvid, wmv, flv, ts, rmvb, mkv, mov, m4v, avi, mp4, 3gp…

In short, Vitamio is an already programmed tool and on that is available for developers who need to use FFmpeg but they do not want to create their own code. Vitamio provides the whole necessary code. With a basic program of streaming playback in Java (without including native code in C) the wished application is obtained.

A scheme of how it works is shown in the figure below.

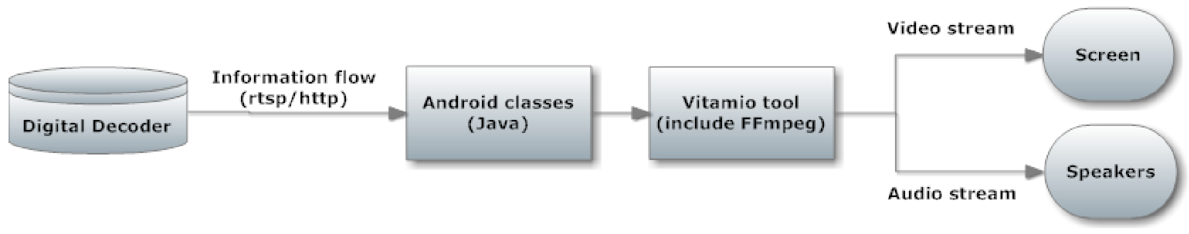


Figure 5.- Scheme of the solution using Vitamio tool

From the digital decoder placed in our house and across the Internet (over rtsp or http protocol) a flow of information is received. With the Android's classes (code in Java) this flow is received and an audio and video player is created.

When this code Java tries to play the information, it is detected that Vitamio is installed in the device and this tool takes charge of separating the different streams. Once this is done, the video stream selected is played on the screen and the selected audio stream in the speakers.

It is possible to observe that this solution is very simple since it is only necessary to programme a player over the network in code Java. The disadvantage of using this tool is that we are not who handle the code of the FFmpeg library and therefore our freedom diminishes at the moment of processing the information received of the streaming server. For example we cannot choose the streams we want to play. The code FFmpeg chose the streams with best quality automatically.

8. FINAL WORK PLAN

In the tables below they are presented the initial work plan and the work plan that I have finally realized.

Table 5.- Initial work plan

Task	2012				
	February	March	April	May	June
Documentation	🟦	🟦			
Video player		🟦			
Streaming client			🟦		
Demultiplexer				🟦	
Plug-in					🟦

Table 6.- Final work plan

Task	Completed	2012				
		February	March	April	May	June
Documentation	100%	🟦	🟦			
Video player (including demultiplexer)	Video player / Audio and video player with Vitamio		🟦	🟦	🟦	
Streaming client	Only with Vitamio				🟦	🟦
Plug-in	0%					🟦

The first phase of documentation and accustoming to the environment of work was carried out in the foreseen time without any problem.

In the second step and due to the fact that creating the video player, Android allows to realize the part of demultiplexing, choosing the stream to process, the second and the fourth step were joined into only one. Therefore this second step would have supposed the double of time (March and April) displacing next step to May. But due to the problems that I found I decided to dedicate also the half of May.

At the end of this step I was able to play only video using the FFmpeg and SDL libraries, and a complete player, with audio and video, with Vitamio tool.

As said before the step of the streaming client was displaced to the second half of May. I was not able to configure correctly the FFmpeg library to receive a stream over rtsp or http protocols. On the other hand, I could implement it with Vitamio tool. It is in this step where I am now working. I will continue doing several compilations of the FFmpeg library and SDL library for Android changing the compilation options to find the correct configuration.

As for the last step, I have not developed even anything due to the fact that the previous ones occupied me more time that I expected at first.

9. CONCLUSION

This project has been very interesting due to the fact that it has allowed me to approach a topic which is growing nowadays: Android. I have learned really much of the Android world and the process for creating an application and how to publish it in the Android market.

It has also helped me to learn more Java and C programming and to know the great amount of open source projects, like FFmpeg or SDL, which are now under development. These libraries are habitually used in many applications and we don’t even realize.

Developing this project in a French company has helped me to enhance my experience as an Erasmus student and also to improve my French and my English. It has been motivating to be employed in a company that is developing innovating software framework for digital multimedia devices and it has allowed me to know the environment of work.

I would have liked to be able to go further in this project and to finish the application with all its functions because I believe that the possibilities in this topic are really wide.

10. BIBLIOGRAPHY

- **« Cours fondamental de télévision. Emision-réception-péritélévision »** R.Besson. Editions radio
- **« La télévision numérique. mpeg-1, mpeg-2, système européen DVB »** Hervé Benoit. Editions Dunod
- http://www.v-net.tv/multi-screen-tv-place-shifting-has-a-role-to-play/
- http:/www.slingmedia.com
- http:/webinars.telecoms.com/webinar/tv-everywhere/
- http:/www.degroupnews.com
- **Android developers' site**. http://developer.android.com
- **FFmpeg libraries**. http://www.ffmpeg.org
- **FFmpeg and SDL Tutorial**. http://dranger.com/ffmpeg/
- **Question and answer site for programmers.** http://stackoverflow.com

APPENDIX A - MPEG-TS

MPEG transport stream (MPEG-TS, MTS or TS) is a standard format for transmission and storage of audio, video, and data. It is used in broadcast systems such as DVB, ATSC and IPTV.

Transport stream specifies a container format encapsulating packetized elementary streams, with error correction and stream synchronization features for maintaining transmission integrity when the signal is degraded.

Important elements of a transport stream

1. Packet

A packet is the basic unit of data in a transport stream.

Packets are 188 bytes in length, but the communication medium may add some error correction bytes to the packet. ISDB-T and DVB-T/C/S uses 204 bytes and ATSC 8-VSB, 208 bytes as the size of emission packets (transport stream packet + FEC data). ATSC transmission adds 20 bytes of Reed-Solomon forward error correction to create a packet that is 208 bytes long. The 188-byte packet size was originally chosen for compatibility with ATM systems.

2. PID

Each table or elementary stream in a transport stream is identified by a 13-bit packet ID (PID). A demultiplexer extracts elementary streams from the transport stream in part by looking for packets identified by the same PID. In most applications, Time-division multiplexing will be used to decide how often a particular PID appears in the transport stream.

3. Programs

Transport stream has a concept of programs. Each single program is described by a

Program Map Table (PMT) which has a unique PID, and the elementary streams associated with that program have PIDs listed in the PMT. For instance, a transport stream used in digital television might contain three programs, to represent three television channels. Suppose each channel consists of one video stream, one or two audio streams, and any necessary metadata. A receiver wishing to decode a particular "channel" merely has to decode the payloads of each PID associated with its program. It can discard the contents of all other PIDs. A transport stream with more than one program is referred to as MPTS - Multi Program Transport Stream. A single program transport stream is referred to as SPTS - Single Program Transport Stream.

4. Program Specific Information (PSI)

There are 4 PSI tables: Program Association (PAT), Program Map (PMT), Conditional Access (CAT), and Network Information (NIT). The MPEG-2 specification does not specify the format of the CAT and NIT.

5. PAT

PAT stands for Program Association Table. It lists all programs available in the transport stream. Each of the listed programs is identified by a 16-bit value called program_number. Each of the programs listed in PAT has an associated value of PID for its Program Map Table (PMT).

The value 0x0000 of program_number is reserved to specify the PID where to look for Network Information Table (NIT). If such a program is not present in PAT the default PID value (0x0010) shall be used for NIT.

TS Packets containing PAT information always have PID 0x0000.

6. PMT

Program Map Tables (PMTs) contain information about programs. For each program, there is one PMT. While the MPEG-2 standard permits more than one PMT section to be transmitted on a single PID, most MPEG-2 "users" such as ATSC and SCTE require each PMT to be transmitted on a separate PID that is not used for any other packets. The PMTs provide information on each program present in the transport stream, including the program_number, and list the elementary streams that comprise the described MPEG-2 program. There are also locations for optional descriptors that describe the entire MPEG-2 program, as well as an optional descriptor for each elementary stream. Each elementary stream is labeled with a stream_type value.

7. PCR

To enable a decoder to present synchronized content, such as audio tracks matching the associated video, at least once each 100 ms a Program Clock Reference, or PCR is transmitted in the adaptation field of an MPEG-2 transport stream packet. The PID with the PCR for an MPEG-2 program is identified by the pcr_pid value in the associated Program Map Table. The value of the PCR, when properly used, is employed to generate a system_timing_clock in the decoder. The STC decoder, when properly implemented, provides a highly accurate time base that is used to synchronize audio and video elementary streams. Timing in MPEG2 references this clock, for example the presentation time stamp (PTS) is intended to be relative to the PCR. The first 33 bits are based on a 90 kHz clock. The last 9 are based on a 27 MHz clock. The maximum jitter permitted for the PCR is +/- 500 ns.

APPENDIX B – FFMPEG

FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video.

FFmpeg is free software licensed under the LGPL or GPL depending on the choice of configuration options.

FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play almost anything.

It contains libavcodec, libavutil, libavformat, libavdevice, libswscale and libswresample which can be used by applications. As well as ffmpeg, ffserver, ffplay and ffprobe which can be used by end users for transcoding, streaming and playing.

The FFmpeg project tries to provide the best technically possible solution for developers of applications and end users alike. To achieve this, the best free software options available are combined.

1. Tools

- ffmpeg is a command line tool to convert multimedia files between formats.
- ffserver is a multimedia streaming server for live broadcasts.
- ffplay is a simple media player based on SDL and the FFmpeg libraries.
- ffprobe is a simple multimedia stream analyzer.

2. Developers libraries

- libavutil is a library containing functions for simplifying programming, including random number generators, data structures, mathematics routines, core multimedia utilities, and much more.

- libavcodec is a library containing decoders and encoders for audio/video codecs.
- libavformat is a library containing demuxers and muxers for multimedia container formats.
- libavdevice is a library containing input and output devices for grabbing from and rendering to many common multimedia input/output software frameworks, including Video4Linux, Video4Linux2, VfW, and ALSA.
- libavfilter is a library containing media filters.
- libswscale is a library performing highly optimized image scaling and color space/pixel format conversion operations.
- libswresample is a library performing highly optimized audio resampling, rematrixing and sample format conversion operations.

APPENDIX C – SDL

Simple DirectMedia Layer is a cross-platform multimedia library designed to provide low level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video framebuffer.

It is used by MPEG playback software, emulators, and many popular games.

SDL supports Linux, Windows, Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. The code contains support for AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS, and OS/2, but these are not officially supported.

SDL is written in C, but works with C++ natively, and has bindings to several other languages, including Ada, C#, D, Eiffel, Erlang, Euphoria, Go, Guile, Haskell, Java, Lisp, Lua, ML, Objective C, Pascal, Perl, PHP, Pike, Pliant, Python, Ruby, Smalltalk, and Tcl.

SDL is distributed under GNU LGPL version 2.

RÉSUMÉ

L'objectif du stage est concevoir, développer et mettre au point une application logicielle téléchargeable à partir d’un site internet. L’utilisateur d’un terminal du type « Android » connecté à internet pourra alors se connecter sur ce site et télécharger l’application. Cette application une fois installée sur le terminal permet de communiquer avec le décodeur numérique installé avec le Media-OS à travers le réseau internet local. L’utilisateur pourra donc ensuite à partir du menu du décodeur numérique choisir le contenu qu’il veut regarder, son émission préférée, ses photos, ses vidéos. Etc.

Le projet sera effectué dans l’entreprise EM-SYS et le tuteur de la structure d’accueil sera Abdelhak Radouani.

L'application sera développée en utilisant Eclipse et en ajoutant un plugin pour intégrer les caractéristiques spécifiques du Android et aussi plusieurs émulateurs. Ces outils sont l'Android SDK (pour le code Java) et l’Android NDK (pour le code C/C++).

Mais il y a des formats qui ne sont pas directement supportés par Android donc c'est nécessaire d'inclure des bibliothèques externes. La librairie FFmpeg est choisie grâce à la variété de codecs qu'elle inclut et la librairie SDL pour l'affichage.