EHAS
ENLACE HISPANO AMERICANO DE SALUD

APLICACIÓN DE TELEMICROSCOPÍA

MANUAL TÉCNICO

Proyecto AECID 2012
"Nuevos procedimientos para el diagnóstico de enfermedades olvidadas utilizando tele-microscopía de bajo coste".

Índice

1 Introducción

Este manual recoge los aspectos técnicos de la aplicación AppTm4l. En el capítulo 2 se explican los componentes que forman la aplicación y que intervienen desde que se recoge el vídeo en el microscopio hasta que el usuario remoto puede visualizarlo en su navegador web. Todos estos componentes ya se encuentran instalados en la imagen de disco disponible para el Odroid-U3. No obstante, también se explican los pasos que habría que dar para poder ejecutar la aplicación en una máquina que tenga una distribución de Linux (preferiblemente Ubuntu/Debian) como sistema operativo. En el capítulo 3 se explica la manera de ejecutar la aplicación. El capítulo 4 describe de manera más detallada cada uno de los componentes de que forman AppTm4l. Finalmente el capítulo 5 trata el tema de la licencia de AppTm4l y cada uno de sus componentes.

2 Componentes e instalación

El esquema de la Figura 1 muestra los componentes que forman parte de la aplicación. Las líneas continuas representan flujo de vídeo (imagen del microscopio) o de datos (página web del servidor), mientras que las líneas discontinuas representan órdenes de control (inicio o parada del componente, envío de parámetros, configuración, etc).

La función que cumple cada componente es:

- AppTm4l: es la parte central de la aplicación. Se encarga de controlar la ejecución del resto de procesos, de actuar de servidor y de proporcionar herramientas básicas de marcado de imágenes (ImageJ).
- Video4linux2: librería para acceder a las cámaras en Linux. Permite además controlar parámetros de las mismas, como brillo, saturación, etc.
- v4l4j: paquete Java que permite acceder la API de video4linux2 desde Java.
- FFmpeg: herramienta para la conversión y codificación de vídeo.
- Crtmpserver: servidor de contenido multimedia.
- Mutt (opcional): cliente de correo electrónico para enviar logs automáticamente.

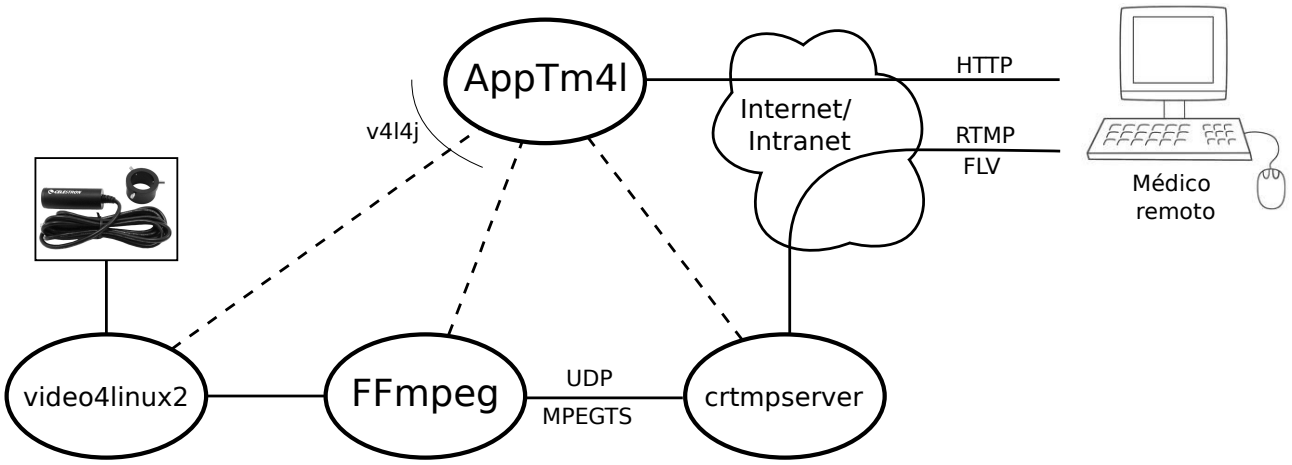

Figura 1: Esquema general de AppTm4l

Todos estos componentes ya se encuentran instalados en la imagen de disco disponible para Odroid-U3. Para copiarla a una tarjeta microSD de, al menos, 8GB de capacidad hay que hacer lo siguiente (instrucciones para Linux; en Windows o iOS serán diferentes):

1. Insertar la tarjeta microSD en un ordenador e identificar el nombre que se le asigna. Para ello ejecutar el comando “sudo fdisk -l” y buscar la tarjeta microSD, la cual suele tener un nombre como “/dev/mmcblk0”,“/dev/sdb” o “/dev/sdc”.
2. Copiar la imagen utilizando el comando “dd” (puede tardar unos 10-20 minutos):

```
# Suponiendo que la tarjeta tenga el nombre /dev/mmcblk0
$ sudo dd if=nombre_de_la_imagen.img of=/dev/mmcblk0 bs=4M
$ sync
```

3. Desmontar la tarjeta microSD y retirar del ordenador.

Si se quiere instalar manualmente los componentes necesarios para ejecutar la aplicación, lo cual es necesario cuando se utilice una máquina diferente al Odroid-U3, habrá que hacer las siguientes instalaciones:

- Java: puede comprobarse que está instalado y es accesible desde el PATH ejecutando “java -version” en la línea de comandos.
- v4l4j: Su instalación está explicada en el apartado Video4Linux4Java.
- Video4linux2: En las distribuciones de Linux más habituales viene instalado por defecto, aunque puede ser conveniente (en el caso del Odroid-U3) instalarlo manualmente. Su instalación está explicada en el apartado Video4linux2.
- FFmpeg: Su instalación está explicada en el apartado Instalación de FFmpeg.
- Crtmpserver: Su instalación está explicada en el apartado Instalación de crtmpserver.

3 Ejecución

Para la ejecución del programa es necesario disponer de un usuario de Linux que tenga permisos de superusuario (pueda ejecutar el comando “sudo”). La razón de esto es que el programa AppTm4l incluye un servidor que, por defecto, escucha en el puerto 80 (los puertos por debajo 1024 son los puertos privilegiados).

Para ejecutar el programa hay que abrir una terminal, situarse en el interior de la carpeta “AppTm4l” y ejecutar:

```
:~/AppTm4l$ sudo java -jar ij.jar -macro AppTm4l.ijm
```

También se puede ejecutar el script “AppTm4l.sh”, el cual contiene la línea anterior (hay que ejecutarlo con sudo):

```
$ sudo sh directorio_de_la_aplicacion/AppTm4l/AppTm4l.sh
```

También se puede crear en el escritorio un lanzador que ejecute la sentencia anterior, de modo que con un doble click se pueda acceder directamente a la aplicación.

4 Descripción detallada de los componentes

4.1 Interacción con la cámara

4.1.1 Video4linux2

A través de la API video4linux2 (v4l2) se puede tener acceso al control de la cámara. Normalmente v4l2 está instalado por defecto en las distribuciones habituales de Linux, aunque es necesario instalarlo manualmente en sistemas como Odroid-U3 para poder instalar luego el paquete java Video4Linux4Java (v4l4j). Esta publicado bajo la licencia GPL.

Para instalar libv4l hay que ejecutar lo siguiente desde una terminal.

```
# Descargamos los archivos necesarios
$ git clone git://git.debian.org/git/collab-maint/libv4l.git

# Nos movemos al directorio y lo compilamos
$ cd libv4l
$ ./configure
$ make
$ sudo make install
```

4.1.2 Video4Linux4Java

V4l4j es un paquete Java que permite controlar desde el propio código Java los dispositivos de vídeo que soporten video4linux2. Está publicado bajo la licencia GPL v3.

Para instalar v4l4j en el Odroid-U3 hay que seguir las instrucciones para ARM de la página oficial[1]. Las instrucciones para Raspberry Pi son válidas (hay que tener en cuenta que la arquitectura de Odroid-U3 es “armhf”).

```
# Instalamos la versión 7 de Java y ant
$ sudo apt-get install openjdk-7-jdk ant libjpeg8-dev libv4l-dev subversion
$ sudo apt-get remove openjdk-6-*

# Obtenemos los archivos
$ svn co http://v4l4j.googlecode.com/svn/v4l4j/trunk v4l4j-trunk
$ cd v4l4j-trunk

# Construimos e instalamos
$ export JDK_HOME=/usr/lib/jvm/java-7-openjdk-armhf/
$ ant clean all
$ sudo ant install
```

4.2 AppTm4l

La carpeta con el nombre AppTm4l es la carpeta principal de la aplicación. En ella se encuentra todo lo necesario para la ejecución de la aplicación (a excepción de programas o librerías externas

1 https://code.google.com/p/v4l4j/wiki/GettingStartedOnRPi

como FFmpeg, video4linux2 y video4linux4java, que deben estar instaladas en el sistema).

La estructura de la carpeta AppTm4l, así como el funcionamiento de la propia aplicación, están basados en el programa ImageJ. Tanto ImageJ como su código fuente Java son de dominio público y no necesitan ninguna licencia. Puede descargarse desde la sección de descargas de su página oficial[2].

La carpeta AppTm4l tiene el siguiente contenido:

- ij.jar: ejecutable java modificado de ImageJ (ver sección 4.2.2 para consultar modificaciones).
- plugins: carpeta propia de ImageJ. Contiene los plugins.
- macros: carpeta propia de ImageJ. Contiene los macros.
- web: carpeta que contiene los ficheros de los reproductores web.
- log: carpeta con logs de la aplicación.
- AppTm4l.properties: archivo de configuración de la aplicación.
- crtmpserver.lua: archivo de configuración de Crtmpserver.
- AppTm4l.sh: ejecutable de la aplicación.

4.2.1 Flujo de ejecución

La ejecución de la aplicación AppTm4l está basada completamente en la ejecución de ImageJ. ImageJ, a su vez, basa su funcionamiento en el uso de macros y plugins:

- Macros: son programas simples que automatizan una serie de comandos de ImageJ. Por lo general se encuentran bajo la carpeta “macros”.
- Plugins: son pequeños programas escritos en Java que expanden la funcionalidad de ImageJ. Los plugins que se encuentran bajo la carpeta “plugins” son reconocidos e instalados inmediatamente por el sistema. Una característica importante es que el nombre del plugin debe contener, al menos, una barra baja. Esta barra baja se sustituye por un espacio en blanco cuando se esté haciendo una llamada a ese plugin (más adelante se verá con ejemplos).

(Para profundizar un poco más en el uso y definición de macros y plugins para ImageJ se recomienda consultar la documentación oficial de la aplicación[3][4][5])

La aplicación AppTm4l consiste, fundamentalmente, en una aplicación ImageJ a la que se le han añadido una serie de fichero adicionales:

- Un fichero de configuración: “Sistema_Telemicroscopia.txt”. Se encuentra dentro de la carpeta “plugins/ActionBar”. Es el fichero de configuración de la ventana principal de AppTm4l.
- Un plugin: “Plugin_Telemicroscopia.jar”. Se encuentra dentro de la carpeta “plugins”. Es el archivo principal y contiene la mayor parte de las funciones utilizadas en AppTm4l.

2 http://rsb.info.nih.gov/ij/download.html
3 http://rsb.info.nih.gov/ij/developer/macro/macros.html
4 http://rsb.info.nih.gov/ij/docs/guide/146-14.html#toc-Section-14
5 http://rsb.info.nih.gov/ij/docs/guide/146-16.html#sub:Plugins

- Un macro: “AppTm4l.ijm”. Se encuentra dentro de la carpeta “macros”. Se ejecuta al inicio de la aplicación y sirve para mostrar la ventana principal de AppTm4l en vez de la de ImageJ.

Como se vio en el apartado Ejecución, la aplicación AppTm4l se ejecuta con la siguiente línea:

```
:~/AppTm4l$ sudo java -jar ij.jar -macro AppTm4l.ijm
```

Por tanto, la aplicación comienza ejecutando el archivo ejecutable de ImageJ al que se le pasa como parámetro el macro “AppTm4l.ijm”. El contenido de este macro es:

```
run("Menu Customized","/plugins/ActionBar/Sistema_Telemicroscopia.txt");
exit();
```

El comando “run()” del lenguaje macro de ImageJ ejecuta comandos definidos internamente en ImageJ o definidos mediante plugins. En este caso, “Menu Customized”[6] se trata de un plugin definido en el archivo “Plugin_Telemicroscopia.jar”. A grandes rasgos, “Menu Customized” lee el contenido del archivo que se le pasa como parámetro y con esa información genera una ventana principal.

El archivo de configuración “Sistema_Telemicroscopia.txt” contiene la información de configuración de la ventana principal de ImageJ. Está formado por una serie de entradas que definen cada elemento del menú. Como ejemplo, el siguiente código muestra la entrada para un botón de la ventana.

```
<button>
label=Abrir
icon=MicroscopyToolBar/open.gif
arg=run("Open ");
```

La primera línea indica que se trata de un botón. La siguiente indica el nombre que tendrá el botón. La siguiente el icono que se muestra en el botón. Y, finalmente, la última línea define lo que se tiene que hacer cuando se presiona el botón. En este caso, al presionar el botón se ejecuta la función “run()” de ImageJ con el parámetro “Open “. Es decir, se está llamando al plugin “Open_”.

Con esto ha finalizado la ejecución de inicio de la aplicación AppTm4l. La Figura 2 muestra un resumen del flujo de ejecución inicial de la aplicación.

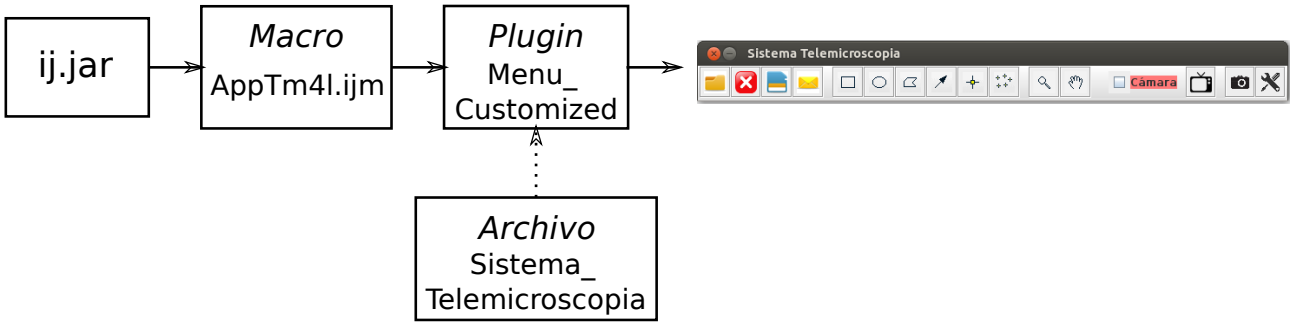

Figura 2: Ejecución inicial de AppTm4l.

6 Hay que recordar que cuando se hace una llamada al plugin, las barras bajas contenidas en el nombre del plugin se sustituyen por espacios en blanco. En este caso, para llamar al plugin se utiliza “Menu Customized”, aunque el nombre real del archivo java que define el plugin es “Menu_Customized.class”.

Cada uno de los botones de la ventana principal de la aplicación hace una llamada a un plugin definido en ImageJ, a un plugin definido en "Plugin_Telemicroscopia.jar" (ver sección 4.2.3) o a una herramienta definida en ImageJ.

4.2.2 ImageJ

La versión de ImageJ utilizada en el desarrollo de la aplicación ha sido la versión 1.47q. La aplicación puede descargarse directamente desde el área de descargas de la página oficial de ImageJ. Al descargar el fichero .zip de la versión para Linux de ImageJ, se puede observar que el contenido de la carpeta es muy similar al contenido de AppTm4l.

Se ha intentado utilizar el programa ImageJ sin realizar ninguna modificación en su código con el objetivo de facilitar la actualización de nuevas versiones de ImageJ. No obstante, la apertura de ficheros los ficheros .zip personalizados que se utilizan en AppTm4l para guardar las imágenes junto con las regiones de interés hizo que fuese necesario realizar pequeñas modificaciones en el código de ImageJ.

El código fuente de ImageJ se encuentra en la carpeta "ImageJ_source". Para modificar el contenido es recomendable abrir la carpeta como un proyecto de Eclipse[7]. Las modificaciones que se han hecho sobre el código original de ImageJ (el cual puede descargase también desde el área de descargas de la página oficial) son las siguientes:

- Clase ij.io.OpenDialog.java:

Añadir esta línea en la declaración inicial de variables.

```
private ArrayList<String> nameArray = new ArrayList<String>();
```

En el método "jOpenInvokAndWait()" añadir las siguientes líneas sobrescribiendo parcialmente las que ya existen. Esto está situado dentro del Runnable(), justo al final:

```
File[] file = fc.getSelectedFiles();
if (file==null)
      {Macro.abort(); return;}
name = file[0].getName();
dir = fc.getCurrentDirectory().getPath()+File.separator;
for (int i=0;i<file.length;i++){
      nameArray.add(file[i].getName());
}
```

Dentro del mismo método, añadir la línea que está en negrita justo encima de la que ya existe.

```
fc.setMultiSelectionEnabled(true);
fc.setDialogTitle(title);
```

También dentro de la misma clase, añadir el siguiente método.

7 En Elicpse: File → Import... → General → Existing projects into Workspace → Select root directory. Selecionamos la carpeta "ImageJ_source".

```
/** Returns the selected file name. */
public ArrayList<String> getFileNames() {
      lastName = nameArray.get(nameArray.size()-1);
      return nameArray;
}
```

- Clase ij.gui.ImageWindow.java:

Esta modificación busca gestionar de una manera más controlada el cierre de las ventanas cuando se está trabajando con los gestores de imágenes y de regiones de interés. Hay que modificar el método "windowClosing()" de la siguiente manera:

```
public void windowClosing(WindowEvent e) {
            new MacroRunner("run(\"Close \")\n");
}
```

4.2.2.1 Construcción

La construcción de "ImageJ_source" se realiza mediante Ant[8]. Dentro del proyecto "ImageJ_source" existe un archivo llamado "build.xml" que define todo el proceso de construcción. Los puntos más importantes a tener en cuenta son:

- Es necesario crear un ejecutable .jar llamado ij.jar. Este será el ejecutable principal de la aplicación AppTm4l.
- El Manifest de dicho .jar se encuentra en el archivo "MANIFEST.MF". En él se define la clase principal y el classpath.
- Es necesario copiar el archivo generado ij.jar a dos destinos:
  - A la carpeta "AppTm4l", ya que ij.jar es el ejecutable principal de la aplicación.
  - A la carpeta "Plugin_Telemicroscopia/lib", ya que hay clases de ese plugin que utilizan funciones de ImageJ y es necesario que estén presentes en el momento de la compilación.

4.2.3 Plugin_Telemicroscopia

Dentro de la carpeta Plugin_Telemicroscopia se encuentra el plugin (aunque en realidad es un conjunto de plugins) que proporciona la mayor parte de la funcionalidad añadida a ImageJ. La carpeta tiene estructura de proyecto de Eclipse[9].

Las clases que componen el proyecto "Plugin_Telemicroscopia" son, fundamentalmente, de dos tipos:

1. Las que no forman parte de ningún paquete: estás clases se encuentran directamente dentro la carpeta "src" (la cual queda representada por "(default package)" en Eclipse). Todas estas clases tienen dos características principales:

8 Ant es una librería Java cuya función principal es dirigir el proceso de construcción de aplicaciones Java. Está publicado bajo la licencia Apache v2.

9 Para importarlo en Elicpse: File → Import... → General → Existing projects into Workspace → Select root directory. Selecionamos la carpeta "Plugin_Telemicroscopia".

- Su nombre contiene, al menos, una barra baja.
- Extienden la clase “ij.plugin.PlugIn” de ImageJ, la cual obliga a tener un método “run()”.

Estas clases son plugins de ImageJ y, por tanto, se ejecutan de una manera especial. Las dos maneras principales son:

- Cuando se pulsa algún botón: anteriormente se vio la manera de definir botones en la ventana principal de AppTm4l. Como ejemplo, se vio que el botón “Abrir” estaba asociado al comando “run('Open ')”. Esto quiere decir que al pulsar sobre ese botón se ejecutará el plugin que tenga el nombre “Open_” (la ejecución de los plugins se inicia ejecutando el método “run()”).
- Directamente desde código Java: ImageJ tiene una clase (MacroRunner) que permite ejecutar plugins de este modo. El ejemplo anterior quedaría así:

```
new MacroRunner("run(\"Open \")\n");
```

2. Las que sí forman parte de algún paquete: estas clases no son plugins en el sentido de que no puede ejecutarse directamente desde ImageJ.

4.2.3.1 Construcción

La construcción de “Plugin_Telemicroscopia” se realiza mediante Ant. Dentro del proyecto “Plugin_Telemicroscopia” existe un archivo llamado “build.xml” que define todo el proceso de construcción. Los puntos más importantes a tener en cuenta son:

- Para la compilación, es necesario que el classpath contenga, además de los archivos de la carpeta “Plugin_Telemicroscopia/lib”, la referencia a la librería video4linux4java (habitualmente estará en “/usr/share/java/v4l4j.jar”).
- Es necesario copiar el archivo generado “Plugin_Telemicroscopia.jar” a la carpeta de plugins de la aplicación, es decir, a “AppTm4l/plugins”. También hay que copiar a esa misma carpeta las librerías utilizadas por “Plugin_Telemicroscopia” excepto dos: “ij.jar”, porque es el ejecutable principal de la aplicación y, por tanto, ya está presente en el momento de la ejecución; “v4l4j.jar”, porque se encuentra instalada en el sistema y, por tanto, la aplicación AppTm4l puede acceder a ella directamente.

4.3 FFmpeg

FFmpeg es una herramienta de software libre que proporciona librerías y programas ejecutables para el manejo de contenido multimedia. FFmpeg está publicado bajo la licencia LGPL[10] o GPL[11], dependiendo de las opciones que estén habilitadas. En este caso se hará uso de la librería x264, la cual está publicada bajo la licencia GPL, por lo que la versión de FFmpeg que se utilizará tendrá que estar también bajo la licencia GPL.

FFmpeg está compuesto por librerías y por programas de línea de comandos. Los cuatro programas que forman FFmpeg son:

- ffmpeg: es la herramienta utilizada para realizar las conversiones de vídeo. También puede

10 http://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License
11 http://en.wikipedia.org/wiki/GNU_General_Public_License

capturar el vídeo de dispositivos en tiempo real como cámaras web.

- ffserver: es un servidor de streaming para difusión de vídeo en tiempo real. Puede utilizar los protocolos HTTP y RTSP.
- ffplay: es un reproductor de contenido multimedia.
- ffprobe: es un analizador de contenido multimedia. Proporciona información sobre el contenido, como formato contenedor, códecs de vídeo y audio, duración, resolución, etc.

4.3.1 Instalación

Antes de explicar la instalación de FFmpeg es necesario hacer una aclaración:

Con las versiones más recientes de Debian y Ubuntu habitualmente viene instalado por defecto el paquete Libav. Este paquete nació en 2011 como una división del proyecto FFmpeg, y tienen una funcionalidad muy similar. Los comandos para ejecutar Libav en línea de comandos son avconv, avserver, avplay y avprobe, aunque también responde a los comandos propios de FFmepg (ffmpeg, ffserver, ffplay y ffprobe) pero mostrando un mensaje de aviso de que FFmpeg está obsoleto. Para descubrir si al ejecutar los comandos ffmpeg, ffserver, ffplay o ffprobe se está ejecutando FFmpeg o Libav, hay que fijarse en la primera línea que aparece. Si esta línea termina con “the FFmpeg developers” entonces se está ejecutando FFmpeg; si termina con “the Libav developers”, entonces Libav.

Aunque las dos herramientas tienen una funcionalidad muy parecida y la sintaxis es prácticamente igual, en este proyecto se ha utilizado FFmpeg por parecer más robusto que Libav.

Para instalar FFmpeg hay que seguir los pasos descritos en la página oficial[12], aunque no es necesario completarlos todos. A continuación se ponen los comandos para su instalación, aunque habría que consultar la página para obtener la última versión de cada módulo.

12 https://trac.ffmpeg.org/wiki/UbuntuCompilationGuide

```
Yasm
cd ~/ffmpeg_sources
wget http://www.tortall.net/projects/yasm/releases/yasm-1.2.0.tar.gz
tar xzvf yasm-1.2.0.tar.gz
cd yasm-1.2.0
./configure
make
sudo make install
make distclean
. ~/.profile

x264
cd ~/ffmpeg_sources
git clone --depth 1 git://git.videolan.org/x264.git
cd x264
./configure --enable-static
make
sudo make install
make distclean

FFmpeg
cd ~/ffmpeg_sources
git clone --depth 1 git://source.ffmpeg.org/ffmpeg
cd ffmpeg
./configure --extra-libs="-ldl" --enable-gpl --enable-libass --enable-libtheora
--enable-libvorbis --enable-libx264 --enable-nonfree --enable-x11grab
make
sudo make install
make distclean
hash -r
. ~/.profile
```

Con esto al ejecutar ffmpeg en la línea de comandos deberá aparecer “the FFmpeg developers” en la primera línea.

La comunicación con la cámara se realiza mediante la librería video4linux2. Esta librería está instalada por defecto en las distribuciones habituales de Linux, aunque puede ser conveniente instalarla manualmente (ver apartado 4.1.1).

4.3.2 Ejecución

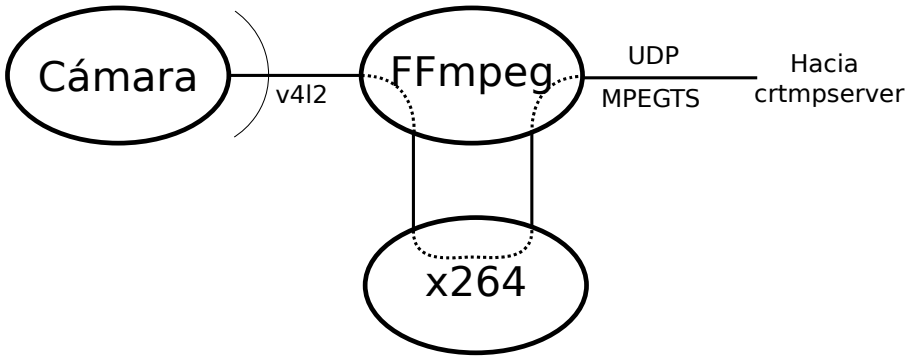

Figura 3: Flujo de vídeo en FFmpeg.

La Figura 3 representa el flujo que sigue el vídeo en la ejecución de FFmpeg. La línea que se toma como base para la ejecución de FFmpeg es la siguiente (las palabras en negrita son parámetros a los

que habrá que asignar algún valor concreto).

```
ffmpeg -re -f video4linux2 -video_size widthxheight -framerate framerate -i
device -vcodec libx264 -r framerate -pix_fmt yuv420p -profile:v high -tune
zerolatency -s  widthxheight -g gop -b:v bitrate -maxrate maxrate -bufsize
bufsize -me_method zero -acodec none -f mpegts udp://127.0.0.1:10000
```

Explicación de la línea:

- La primera parte de la línea hace referencia a la entrada de vídeo:
  - video4linux2: es el formato de la entrada de vídeo.
  - widthxheigth: tamaño de la imagen (típicamente 640x480).
  - framerate: tasa de imágenes por segundo.
  - device: dispositivo de captura (típicamente /dev/video0).
- La segunda parte hace referencia a la salida del vídeo:
  - libx264: codec de salida.
  - framerate: tasa de imágenes por segundo.
  - high: perfil de H264.
  - zerolatency: “tune” de H264. Un “tune” es un conjunto de opciones para H264 que adaptan el vídeo a una situación determinada. En concreto, “zerolatency” busca que el retardo de codificación sea muy bajo para favorecer la codificación en tiempo real.
  - widthxheigth: tamaño de la imagen (típicamente 640x480).
  - gop: group of pictures. Es el número de imágenes que puede haber como máximo entre dos imágenes I (por ejemplo 60)
  - bitrate: tasa de bit que se marca como objetivo de la codificación (por ejemplo 256k).
  - maxrate: tasa máxima de bit (por ejemplo 256k).
  - bufsize: valor del buffer para almacenar excesos en la tasa de bit (por ejemplo 256k).
  - mpegts: formato de salida de vídeo (MPEG transport stream es un formato para la transmisión y almacenamiento de contenido multimedia).
- La última parte hace referencia a cómo se envía o almacena el vídeo.
  - udp: protocolo de envío.
  - 127.0.0.1:10000: dirección IP y puerto de envío.

4.4 Crtmpserver

Crtmpserver es un servidor de streaming de alto rendimiento capaz de distribuir contenido tanto pre-grabado como en tiempo real. Permite publicar contenido en Flash (RTMP, RTMPE, RTMPS, RTMPT, RTMPTE). Está publicado bajo la licencia GPL v3.

4.4.1 Instalación

Siguiendo los pasos descritos en la wiki de crtmpserver[13], para instalar el servidor en Ubuntu hay que ejecutar:

```
#install needed stuff
sudo apt-get install g++ subversion cmake make libssl-dev

#fetch the latest repo version
cd ~
svn co --username anonymous --password "" https://svn.rtmpd.com/crtmpserver/trunk
crtmpserver
cd crtmpserver/builders/cmake/
COMPILE_STATIC=1 cmake -DCMAKE_BUILD_TYPE=Release .
make
```

Al construir la aplicación es posible que aparezcan errores relacionados con que no se pueden localizar algunas librerías. Este error ha sido comentado en la lista de usuarios, y una solución sugerida es realizar los siguientes cambios en los ficheros de configuración:

- Fichero "crtmpserver/builders/cmake/cmake_find_modules/Find_openssl.cmake".
  - Hay que eliminar la línea "NO_DEFAULT_PATH" de las secciones "FIND_LIBRARY(OPENSSL_LIBRARY_PATH", "FIND_LIBRARY(CRYPTO_LIBRARY_PATH" y "FIND_LIBRARY(Z_LIBRARY_PATH".
- Fichero "crtmpserver/builders/cmake/cmake_find_modules/Find_dl.cmake".
  - Hay que eliminar la línea "NO_DEFAULT_PATH" de la sección "FIND_LIBRARY(DL_LIBRARY_PATH".

Con ello, el servidor debería compilarse sin problemas.

Por último queda situar el ejecutable de crtmpserver dentro del path del sistema para que pueda ser utilizado por la aplicación. Un lugar común para situarlo es "/usr/local/bin".

```
sudo crtmpserver/builders/cmake/crtmpserver/crtmpserver /usr/local/bin
```

Para comprobar que crtmpserver está en el lugar correcto puede ejecutarse el siguiente comando, tomando como fichero de configuración el fichero crtmpserver.lua presente dentro de la carpeta "AppTm4l":

```
:~/AppTm4l$ crtmpserver crtmpserver.lua
```

Si el programa se ejecuta sin errores, entonces crtmpserver está instalado correctamente.

4.4.2 Configuración

Crtmpserver basa su configuración en un fichero principal llamado "crtmpserver.lua". En él se establecen y configuran las aplicaciones que van a estar corriendo dentro de crtmpserver. La aplicación relevante para este proyecto es "flvplayback".

13 http://wiki.rtmpd.com/quickbuild

```
configuration=
{
	daemon=false,
	pathSeparator="/",
	logAppenders=
	{
		{
			name="console appender",
			type="coloredConsole",
			level=6
		}
	},
	applications=
	{
		rootDirectory="applications",
		{
			description="FLV Playback Sample",
			name="flvplayback",
			protocol="dynamiclinklibrary",
			acceptors =
			{
				{
					ip="0.0.0.0",
					port=1935,
					protocol="inboundRtmp"
				},
				{
					ip="0.0.0.0",
					port=10000,
					localStreamName="AppTm4l",
					targetStreamName="AppTm4l",
					protocol="inboundUdpTs"
				},
			},
			validateHandshake=false,
			clientSideBuffer=0,
			keyframeSeek=false,
		}
	}
}
```

La configuración de la aplicación “flvplayback” es muy sencilla. Solamente hay que añadir dos “acceptors”:

- El primero escucha y sirve las peticiones RTMP por el puerto 1935 (comunicación con el exterior).
- El segundo escucha en el puerto 10000 y espera un flujo de vídeo en formato TS y protocolo UDP (comunicación con FFmpeg). Además le asigna la etiqueta AppTm4l a ese flujo de vídeo.

4.4.3 Ejecución

Para arrancar el servidor basta con ejecutar la línea de comando anterior:

```
:~/AppTm4l$ crtmpserver crtmpserver.lua
```

En pantalla aparecerá el log de todos los eventos que van ocurriendo en crtmpserver. Los dos más importantes son:

- Entrada de un flujo de streaming. Esto ocurre cuando se ejecuta FFmpeg para servir vídeo al servidor (ver apartado Ejecución de FFmpeg).
- Conexión/desconexión de un reproductor de vídeo (ver Visualizadores de vídeo).

La Figura 4 muestra el flujo de vídeo en crtmpserver.

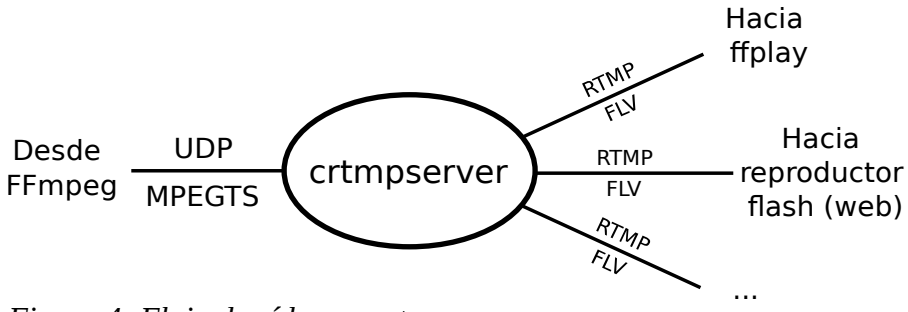

Figura 4: Flujo de vídeo en crtmpserver.

4.5 Visualizadores de vídeo

4.5.1 Reproductor flash (web)

Aquí se presentan dos alternativas probadas y muy extendidas de reproductor flash para la web con soporte para reproducción de vídeo con protocolo RTMP.

4.5.1.1 Video.js

Video.js es un reproductor web de vídeo que puede reproducir contenido de tipo vídeo HTML5 o Flash. Es de código abierto y su apariencia es personalizable. Puede descargarse desde desde su página oficial[14]. Existe una amplia documentación sobre el reproductor accesible directamente desde su página web.

Los archivos más importantes de video.js, y que deberán ser accesibles por el servidor, son:

- video.js: javascript que proporciona la funcionalidad al reproductor.
- video-js.min.css o video-js.css: archivos .css que determinan la apariencia del reproductor: situación de los botones, tamaño, color, comportamiento, etc. Aquí se pueden ocultar botones innecesarios, como el del volumen.
- video-js.swf: es el reproductor en sí mismo.
- font/vjs.woff y font/vjs.ttf: son las fuentes de los elementos de video.js: cursores, botones, iconos, etc.

Para incrustar el reproductor video.js en la página web hay que hacer lo siguiente:

14 http://www.videojs.com/downloads/video-js-4.3.0.zip.

```
<!-- Incluir en la cabecera las ficheros necesarios -->
<head>
      <link href="video-js.min.css" rel="stylesheet">
      <script type="text/javascript" src="video.js"></script>
</head>

<!-- Incrustar en el cuerpo el vídeo -->
<body>
      <video class='video-js vjs-default-skin'
            id='video'
            width="100%"
            height="100%"
            data-setup='{
                  "controls": true,
                  "autoplay":true,
                  "preload":"auto"}'>
            <source src="rtmp://ip_del_servidor&AppTm4l" type="rtmp/flv">
      </video>
</body>
```

Dado que la dirección ip del servidor será diferente dependiendo del punto en el que se conecte o de la red a la que pertenezca, la solución que se ha escogido ha sido incluir una línea de javascript que genere de manera dinámica la línea <source>. El código anterior quedaría de la siguiente manera:

```
<!-- Incluir en la cabecera las ficheros necesarios -->
<head>
      <link href="video-js.min.css" rel="stylesheet">
      <script type="text/javascript" src="video.js"></script>
</head>

<!-- Incrustar en el cuerpo el vídeo -->
<body>
      <video class='video-js vjs-default-skin'
            id='video'
            width="100%"
            height="100%"
            data-setup='{
                  "controls": true,
                  "autoplay":true,
                  "preload":"auto"}'>
            <script> var serverip = '<source src="rtmp://' +
                  location.host.split(':')[0] + '&AppTm4l" type="rtmp/flv">';
            document.write(serverip);
            </script>
      </video>
</body>
```

Video.js hace una comprobación sobre la versión que se está utilizando y, en ocasiones, puede decidir que necesita descargarse la versión del reproductor desde la web. Si quiere evitarse esa petición para que el reproductor pueda funcionar sin problema cuando no dispone de conexión a Internet, hay que hacer la siguiente modificación en el código del fichero “video.js”:

- Buscar la línea
  - v.options.flash.swf=u.Fc+"vjs.zencdn.net/"+u.Tb+"/video-js.swf"

- y sustituirla por
  - v.options.flash.swf="video-js.swf"

De este modo se le exige que tome el fichero local “video-js.swf” como reproductor.

Por último, si se quiere modificar la apariencia del reproductor, eso puede hacerse modificando el fichero .css que se haya escogido. En el ejemplo se ha tomado el fichero “video-js.min.css”. La mayor parte de los campos son bastante intuitivos y existe una buena documentación sobre este fichero en la página oficial de video.js y en sus foros.

4.5.1.2 Flowplayer

Flowplayer es un reproductor de vídeo HTML5 y flash para la web. Está liberado bajo la licencia GNU GPL v3, aunque su versión gratuita exige que se mantenga el logo de Flowplayer en el reproductor. La versión Flowplayer Flash puede descargarse en la sección de descargas de su página oficial[15]. También es necesario descargar el plugin para reproducir vídeo RTMP[16].

Los ficheros más importantes de Flowplayer, y que deberán ser accesibles por el servidor, son (la versión de Flowplayer puede variar):

- flowplayer-3.2.13.min.js: javascript que proporciona la funcionalidad al reproductor.
- flowplayer-3.2.17.swf: es el reproductor en sí mismo.
- flowplayer.rtmp-3.2.13.swf: plugin para reproducir contenido con protocolo RTMP.

Para incrustar el reproductor Flowplayer un la página web hay que hacer lo siguiente:

15 http://releases.flowplayer.org/flowplayer/flowplayer-3.2.18.zip.
16 http://flash.flowplayer.org/plugins/streaming/rtmp.html. En la parte inferior, sección “Downloads”.

```
<!-- Incluir en la cabecera las ficheros necesarios -->
<head>
      <script type="text/javascript" src="flowplayer-3.2.13.min.js">
      </script>
</head>

<!-- Incrustar en el cuerpo el vídeo →
<body>
      <object id='player'>
      </object>
      <script>
            flowplayer("player", "flowplayer-3.2.17.swf", {
                  buffering: false,
                  clip: {
                        url: 'AppTm4l',
                        autoPlay: true,
                        live: true,
                        bufferLength: 1,
                        controls: null,
                        provider: 'crtmpserver',
                        scaling: 'fit'
                  },
                  plugins: {
                        crtmpserver: {
                              url: 'flowplayer.rtmp-3.2.13.swf',
                              netConnectionUrl: 'rtmp://ip_del_servidor/'
                        },
                        controls: null
                        }
                  }
            );
      </script>
</body>
```

Dado que la dirección ip del servidor será diferente dependiendo del punto en el que se conecte o de la red a la que pertenezca, la solución que se ha escogido ha sido incluir una línea de javascript que genere de manera dinámica la línea “netConnectionUrl”. El código anterior quedaría de la siguiente manera:

```
<!-- Incluir en la cabecera las ficheros necesarios -->
<head>
      <script type="text/javascript" src="flowplayer-3.2.13.min.js">
      </script>
</head>

<!-- Incrustar en el cuerpo el vídeo →
<body>
      <object id='player'>
      </object>
      <script>
            var serverip = 'rtmp://' + location.host.split(':')[0] + '/';
            flowplayer("player", "flowplayer-3.2.17.swf", {
                  buffering: false,
                  clip: {
                        url: 'AppTm4l',
                        autoPlay: true,
                        live: true,
                        bufferLength: 1,
                        controls: null,
                        provider: 'crtmpserver',
                        scaling: 'fit'
                  },
                  plugins: {
                        crtmpserver: {
                              url: 'flowplayer.rtmp-3.2.13.swf',
                              netConnectionUrl: serverip
                        },
                        controls: null
                        }
                  }
            );
      </script>
</body>
```

Por último, hay que tener en cuenta los requisitos que debe cumplir el navegador web para el código javascript anterior funcione correctamente. Flowplayer necesita que el navegador web disponga de la versión 1.5 de javascript, la cual está soportada por los principales navegadores: Internet Explorer 6.0 y siguientes, Firefox FF 2 y siguientes, Safari 2.0 y siguientes, Opera 9.0 y siguientes, y Chrome.

4.5.2 ffplay

Para reproducir el streaming de vídeo con ffplay hay que ejecutar la siguiente línea de comandos:

```
$ ffplay -probesize 3000 rtmp://ip_de_crtmpserver/AppTm4l
```

Explicación de la línea:

- probesize: es el tamaño de la muestra. Un valor bajo reduce el retardo con el que se inicia el vídeo.
- rtmp: protocolo de aplicación. Por defecto realiza las peticiones al puerto 1935.
- ip_de_crtmpserver: sirve “localhost” si se visualiza en el mismo equipo.

- AppTm4l: etiqueta del vídeo en AppTm4l.

4.6 Correo electrónico (Opcional): Mutt

Mutt es un cliente de correo electrónico que puede ser utilizado por la aplicación para el envío de logs automáticos. Mutt se encuentra en los repositorios de las distribuciones Debian/Ubuntu, y puede ser instalado fácilmente desde los mismos.

```
$ sudo apt-get install mutt
```

Durante el proceso de instalación se lanzará un diálogo solicitando qué tipo de configuración Postfix se desea. Seleccionar “Internet Site” e introducir un nombre que sea identificativo.

Para activar el envío automático de logs hay que modificar el archivo de configuración AppTm4l.properties.

```
logsynchronization=true  (“true” activa el envío automático de logs)

logname=ehas_teleco      (nombre que identique al equipo)

logemail=victor.garcia@ehas.org      (dirección de correo de destino)
```

5 Licencia

La aplicación se publica bajo la licencia GNU General Public License v3 (GPL v3). A continuación se muestra un resumen de licencias de las aplicaciones que se han utilizado, explicando su compatibilidad con GPL v3.

- **Ant:** Apache License v2.0. Es compatible con GPL v3 siempre que el trabajo resultante sea publicado bajo la lincecia GPL v3.
- **Video4linux2:** GNU General Public License (GPL).
- **Video4linux4java:** GNU General Public License v3.
- **slf4j:** MIT License (Massachusetts Institute of Technology). Compatible con GNU GPL v3.
- **ImageJ:** dominio público. No se requiere ninguna licencia.
- **FFmpeg:** GNU Lesser General Public License (LGPL) v2.1. Compatible con GNU GPL v3.
- **x264:** GNU General Public License v2. Compatible con GNU GPL v3.
- **Crtmpserver:** GNU General Public License v3.
- **VLC:** GNU Lesser General Public License (LGPL). Compatible con GNU GPL v3.
- **Video-js:** Apache License v2.0. Es compatible con GPL v3 siempre que el trabajo resultante sea publicado bajo la lincecia GPL v3.
- **Flowplayer:** GNU General Public License v3.
- **Mutt:** GNU General Public License (GPL).