Del Dreambox 7000S al Nokia 770 Internet Tablet

Port de VideoLAN Client a Maemo per a la reproducció d'un flux mpeg2 per mitjà d'una xarxa sense fils

per Josep Torra Vallès

jtorra@uoc.edu

UOC, Primavera 2006
Tutor: Victor Carceler Hontoria
vcarceler@uoc.edu

Introducció

L'objectiu d'aquest projecte és connectar el sintonitzador de televisió per satèl·lit Dreambox 7000S amb el Nokia 770 Internet Tablet.

El propòsit d'aquesta aplicació és utilitzar el Dreambox 7000S com una font de *streaming* mpeg2 i el Nokia 770 com a reproductor portàtil.

Per assolir aquest objectiu s'ha portat el programari VideoLAN Client, d'ara en davant VLC, i diverses llibreries de que en depèn a la plataforma maemo, a més s'han codificat de nou rutines específiques optimitzant-les per el maquinari concret del Nokia 770.

Principalment aquest projecte m'ha permès assolir amb un cert grau de profunditat: coneixements de la plataforma maemo, aspectes d'arquitectura del Nokia 770 i nocions sobre la tecnologia mpeg2.

Aquesta memòria està estructurada en diversos capítols que introdueixen un determinat tema, i posteriorment el relaciona amb aspectes concrets realitzats en el projecte.

Capítol 1
: Aquest primer capítol proporciona una visió general del maquinari utilitzat en el projecte, introdueix diversos aspectes interns de l'arquitectura del Nokia 770.

Capítol 2
: En aquest capítol es fa una introducció a maemo i proporciona les bases necessàries per començar a desenvolupar en aquest entorn.

Capítol 3
: En el text d'aquest capítol es descriu el programari VideoLAN Client i les diverses llibreries que s'han portat i modificat.

Capítol 4
: Aquí es descriu breument les tècniques de codificació mpeg2 utilitzat per l'emissió digital de la televisió per satèl·lit.

Capítol 5

Per concloure aquest treball, en aquest capítol es sintetitzen les idees que no s'han abordat i les limitacions en que s'ha enfrontat el projecte.

Apèndix A

En aquest annex es descriu el guió de l'interpret de comandes que s'ha construït per automatitzar la descàrrega del codi font i la compilació per a l'entorn maemo del projecte.

Apèndix B

En aquest annex es detallen els canvis introduits en el codi font del Video LAN Client per tal d'implementar les adaptacions al Nokia 770.

Apèndix C

En aquest annex es detallen els canvis introduits en el codi font de la llibreria libmpeg2 per tal de millorar el rendiment de la descodificació en al Nokia 770.

Josep Torra Vallès
UOC - primavera 2006

Índex

Introducció **ii**

1 Nokia 770 Internet Tablet i Dreambox 7000S **1**
1.1 Nokia 770 Internet Tablet 1
1.1.1 TI OMAP 1710 2
1.2 Dreambox 7000S 4

2 Descobrint Maemo **6**
2.1 El *framework* d'aplicació Hildon 7
2.2 Llibreries base de la plataforma 8
2.3 Entorn de desenvolupament 9
2.4 Arquitectura multimèdia 10
2.5 Maemo 2.0 i Internet Tablet 2006 12
2.6 Instal·lació, comença el projecte 13

3 El projecte VideoLAN Client **16**
3.1 Respecte al projecte 18
3.1.1 Modificació de l'interfície d'usuari 19
3.1.2 Modificació en el mòdul V.Out-SDL 19

4 Una introducció a MPEG2 **21**
4.1 Els *frames* 21
4.2 La predicció 22
4.3 Els macroblocs 24
4.4 DCT: Discrete Cosine Transformation 25
4.5 Els contenidors 27
4.6 Respecte al projecte 28

5 Conclusions **30**

A Descripció del guió build.sh **33**
A.1 build.sh . . . 33
A.1.1 Preludi . . . 33
A.1.2 Funció ask_to_do() . . . 33
A.1.3 Funció makedir() . . . 34
A.1.4 Funció downpackage() . . . 34
A.1.5 Funció uncompress_gzip() . . . 34
A.1.6 Funció uncompress_bzip2() . . . 35
A.1.7 Funció patch_X11_include_dir() . . . 35
A.1.8 Funció patch_source() . . . 35
A.1.9 Funció build_libmad() . . . 35
A.1.10 Funció build_libmpeg2 . . . 36
A.1.11 Funció build_libdvbpsi() . . . 36
A.1.12 Funció build_vlc() . . . 37
A.1.13 Funció test_vlc() . . . 39
A.1.14 Secció principal del guió . . . 39

B Canvis en el Video LAN Client **41**
B.1 Modificacions en el fitxer configure.ac . . . 41
B.2 Modificacions en la interfície d'usuari . . . 41
B.2.1 Fitxers de capçalera . . . 42
B.2.2 Declaració de variables auxiliars . . . 42
B.2.3 Instanciació de l'aplicació . . . 42
B.2.4 Instanciació del viewbox principal . . . 43
B.2.5 Empaquetat de la barra d'eines . . . 43
B.2.6 Redimensionat de diversos widgets . . . 43
B.2.7 Vinclatge de la vista amb l'aplicació . . . 43
B.3 Modificacions en el mòdul de renderitzat SDL . . . 44
B.3.1 Declaració de funcions . . . 44
B.3.2 Canvis en la funció Display() . . . 44
B.3.3 Inclusió de l'estructura private_yuvhwdata . . . 45
B.3.4 Canvis a la funció SDL_DisplayYUVOverlayAlter() . . . 45
B.3.5 Canvis en la funció Display1X() . . . 46

C Canvis en la llibreria libmpeg2 **47**
C.1 Canvis en la carpeta ./ . . . 47
C.1.1 Modificacions en el fitxer configure.in . . . 47
C.2 Canvis en la carpeta ./include . . . 47
C.2.1 Mòdul n770.h . . . 47
C.3 Canvis en la carpeta ./libmpeg2 . . . 49
C.3.1 Modificacions en el fitxer Makefile.am . . . 49

C.3.2 Modificacions en el fitxer mpeg2_internal.h 49
C.3.3 Modificacions en el fitxer idct.c 49
C.3.4 Mòdul idct_n770.c 50

Bibliografia **56**

Capítol 1

Nokia 770 Internet Tablet i Dreambox 7000S

En aquest capítol es descriu breument el maquinari implicat en el desenvolupament d'aquest projecte.

El Nokia 770 és un dispositiu portàtil semblant a una PDA i el Dreambox 7000S és un receptor digital de televisió per satèl·lit; aquests dos dispositius tenen en comú que funcionen amb el sistema operatiu Linux i programari lliure.

Amb l'objectiu de proporcionar un context adient es proporciona una visió del Nokia 770 i del seu processador, l'OMAP 1710 de Texas Instruments, així com un resum de les principals característiques del receptor Dreambox 7000S.

1.1 Nokia 770 Internet Tablet

A principis de l'any 2006, la companyia finlandesa Nokia va llançar al mercat un nou dispositiu portàtil, el Nokia 770 Internet Tablet. Aquest dispositiu amb característiques similars a una PDA està orientat a proporcionar un accés a Internet des de qualsevol lloc amb un elevat grau de qualitat i comoditat.

Una de les característiques més rellevants d'aquest dispositiu és el fet que funciona amb una distribució de Linux i un conjunt d'eines de codi font obert que han estat portades a aquesta plataforma.

Per altre banda, cal destacar-ne també les reduïdes dimensions i les possibilitats de connectivitat a xarxes sense fils i a dispositius bluetooth.

Un altre detall que fa molt atractiu aquest aparell és l'excel·lent pantalla tàctil d'alta resolució (800 x 480 punts) i fins a 65.536 colors. Aquestes característiques fan que sigui possible representar la majoria de pàgines web

sense problemes.

Actualment, la versió de sistema operatiu que utilitza el Nokia 770 és l'Internet Tablet 2005, que en el moment de la redacció d'aquest treball és a punt de ser-ne alliberada la nova versió denominada Internet Tablet 2006.

La nova versió del sistema operatiu incorporarà noves possibilitats molt interessants, com ara el suport a veu sobre IP, versions més modernes del programari existent en l'actualitat, un nou gestor de paquets i un nou format binari.

1.1.1 TI OMAP 1710

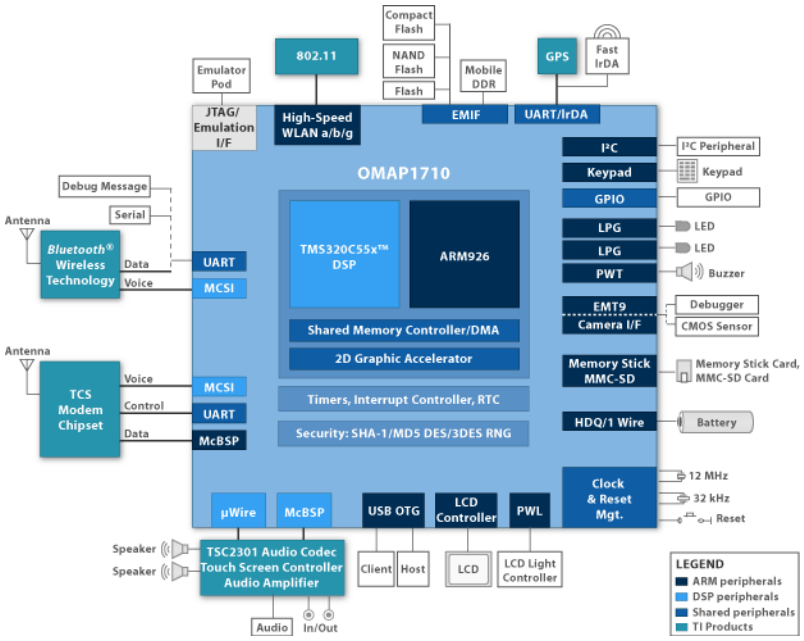

Figura 1.1: Esquema funcional del OMAP 1710

El motor del Nokia 770 és el processador de Texas Instruments OMAP 1710[1]; aquesta solució en un sol xip, incorpora un sistema de doble nucli heterogeni constituït per un processador de propòsit general ARM926TEJ i processador de senyals digitals (DSP) TMS320C55x. A més dels dos nuclis l'OMAP 1710, incorpora en el mateix xip la infraestructura per crear un sistema sencer. La figura 1.1 mostra un esquema funcional d'aquesta solució. En el Nokia 770 el processador de propòsit general ARM926TEJ executa el sistema operatiu Linux i la resta d'aplicacions mentre que el DSP està destinat principalment a les tasques multimèdia, diversos codecs de vídeo i àudio estan implementats per aquest nucli.

El nucli ARM926TEJ

Aquest és un processador RISC de 32 bits que pot funcionar a 250 MHz, amb una arquitectura ARM versió 5 que inclou una memòria cau de 32 KB per instruccions i 16 KB per a dades.

Inclou la extensió Jazelle per accelerar via maquinari l'execució de bytecode Java.

Aquest processador és capaç d'executar dos conjunts d'instruccions de 32 bits i 16 bits (mode thumb).

Disposa d'unitats de gestió de memòria (MMU) per dades i codi, i de dos taules de traducció d'adreces (TLB) de 64; entrades, amb aquest maquinari es proporciona el suport a la memòria virtual de sistemes operatius com Linux.

També disposa d'extensió DSP[5] amb unitats aritmètiques que permeten realitzar multiplicacions i multiplicació-suma (MAC) en un sol cicle. Aquesta característica s'ha utilitzat en el projecte per tal d'optimitzar la implementació de l'algoritme IDCT[1] que és una de les seccions crítiques de la descodificació del sistema mpeg2.

Per obtenir informació detallada sobre l'ARM926 visitar www.arm.com, es pot obtenir informació tècnica als manuals[2][3] i informació sobre el llenguatge assemblador[4].

Aquesta és la unitat funcional on s'hi executarà el port de VLC i les diverses llibreries necessàries per el projecte.

El nucli DSP TMS320C55x

Aquest és un processador de senyal digital amb aritmètica de punt fix, amb un alt rendiment i un consum d'energia reduït. És un processador superescalar amb 7 etapes que pot funcionar a una freqüència variable fins a 250 MHz.

Les característiques principals d'aquest DSP són:

- Una arquitectura de bus múltiple amb un bus de memòria intern per el codi i cinc busos de dades (tres dedicats a la lectura i dos a l'escriptura).
- Dos multiplicadors de 17x17 bits acoblats a sumadors de 40 bits per realitzar operacions de multiplicació-suma (MAC) en un sol cicle de rellotge.
- Dos generadors d'adreces amb 8 registres auxiliars i dos registres auxiliars amb unitats aritmètiques.

[1] Inverse Discrete Cosine Transformation

- Capacitat per executar fins a dos instruccions en un sol cicle.

Aquest nucli està destinat principalment a l'execució de codecs d'àudio i vídeo, malauradament la informació tècnica respecte als codecs incorporats en el Nokia 770 no està disponible de moment.

Per obtenir informació detallada sobre el DSP TMS320C55x llegir els manuals de Texas Instruments[6][7]

La comunicació entre els dos nuclis es pot realitzar per mitja de l'API proporcionada per DSP Gateway[8].

1.2 Dreambox 7000S

El dispositiu Drembox 7000S és un receptor de televisió per satèl·lit de tercera generació, en realitat es tracta d'un petit ordinador de saló que funciona amb el sistema operatiu Linux i diverses aplicacions específiques.

Aquest aparell disposa d'un processador PowerPC a 250 MHz, 2 lectors de targetes smartcard, un slot pcmcia, un lector de targetes de memòria compact flash, una controladora IDE/ATAPI on s'hi pot connectar un disc dur estàndard, porta integrat una controladora ethernet 10/100, disposa també d'una connexió USB, una connexió RS-232 i una connexió ps/2.

Disposa també d'una sortida de so digital òptica i una sortida de so stereo analògica. A més de 2 euroconectors controlables per programari.

Aquest maquinari i el programari que l'acompanya permet la gravació digital dels programes en el disc dur i utilitzar-lo com una font de *streaming* cap a la connexió de xarxa.

El programari del Dreambox 7000S incorpora una interfície web que permet gestionar remotament l'aparell per mitjà de la connexió de xarxa.

Figura 1.2: A l'esquerra el Dreambox 7000S, a la dreta el portàtil Dell D600 executant Ubuntu, l'entorn desenvolupament Maemo i el port de VLC, sobre el portàtil la consola GP2X i el Nokia 770 executant el port del joc Nethack.

Capítol 2

Descobrint Maemo

Maemo[9] és una plataforma de desenvolupament oberta per crear aplicacions destinades al Nokia 770 Internet Tablet i altres dispositius de mà compatibles que puguin existir en un futur.

La plataforma proporciona un entorn de desenvolupament senzill i una interfície d'usuari orientada a dispositius de mà, especialment optimitzada.

Maemo està composat per un conjunt components programari lliure que són molt utilitzats en les distribucions de Linux actuals orientades a escriptori. Maemo ha adaptat aquests components a l'entorn dels dispositius de mà i ha contribuït amb eines senzilles per desenvolupar envers plataformes diferents a l'arquitectura x86.

Un dels pilars de maemo és el *framework* d'aplicacions Hildon, que està basat en la tecnologia GNOME. El programari lliure GNOME proporciona un entorn d'escriptori molt intuïtiu i atractiu de la plataforma d'escriptori tradicional basada en Linux, i un *framework* molt potent per crear aplicacions integrades amb l'escriptori. Maemo ha adaptat aquesta tecnologia als dispositius de mà amb extensions i modificacions que han permès evolucionar el *framework* de l'interfície d'usuari per que s'adapti millor a la categoria dels dispositius de mà.

Amb maemo es disposa d'un entorn de desenvolupament hoste que executa el mateix programari que està disponible en el dispositiu físic, eliminant la necessitat d'emular el maquinari, també permet la utilització d'eines de desenvolupament estàndard de la comunitat de codi font obert reduint així considerablement la corba d'aprenentatge.

Alternativament maemo proporciona dos entorns de desenvolupament addicionals, el primer mitjançant el programari QEMU és possible executar el codi binari ARM via emulació en l'entorn de desenvolupament. El segon entorn de desenvolupament permet connectar el Nokia 770 a l'ordinador personal amb la finalitat d'executar el codi directament al dispositiu des de

l'entorn de desenvolupament, aquesta tècnica s'anomena transparència.

En la figura 2.1 presenta un esquema de la plataforma, es pot observar la divisió lògica entre el nucli intermediari i la interfície d'usuari referida habitualment com a hildon.

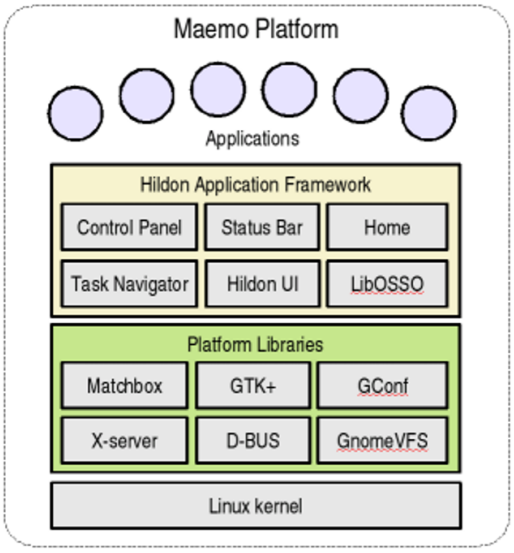

Figura 2.1: Plataforma Maemo

El nucli de components intermediari no orientat a l'interfície d'usuari està constituït principalment per projectes de programari lliure com glibc, la pila bluez bluetooth, la pila estàndard de xarxa de Linux (ppp, autoip, openobex, iptables, wlan, etc), Xserver, el parsejador XML expat, D-BUS, per anomenar-ne uns quants. Alguns del components s'han modificat per tal d'adequar-los les restriccions imposades per el maquinari.

2.1 El *framework* d'aplicació Hildon

El *framework* d'aplicació hildon està basat en les tecnologies bàsiques de GNOME com els widgets GTK+ que s'han ampliat i modificat, el motor de temes d'escriptori, Pango per proporciona el suport de text en diverses llengües i la localització, etc.

La interfície d'usuari Hildon està constituïda per el conjunt estàndard de widgets GTK+, modificats per suportar millor la pantalla tàctil del Nokia 770 i al conjunt reduït de botons que incorpora el dispositiu.

En resum el *framework* d'aplicació Hildon està compost per:

- Task Navigator - El navegador de tasques per engegar i commutar les aplicacions. També incorpora altres habilitats com la possibilitat de mostrar les capçaleres dels correus electrònics o la llista d'enllaços del navegador d'internet en el propi menú.

- Home - Proporciona el fons d'escriptori on s'hi poden incrustar diferents plugins, per exemple el rellotge.

- Home Status Bar - La barra d'estat principal proporciona una interfície on s'hi poden incrustar plugins que proporcionin informació d'estat, per exemple l'estat de la connexió amb xarxes sense fils.

- Control Panel - El panell de control proporciona un *framework* per executar applets que s'utilitzen per canviar les preferències de l'usuari. Aquests applets són llibreries que proporcionen una interfície d'usuari per canviar o establir la configuració del dispositiu.

- Hildon UI - Proporcionen un conjunt addicional de widgets sobre els de GTK+ i les modificacions de tema necessàries per obtenir una consistència estètica unificada.

- LibOSSO - Llibreria addicional que conté diversos serveis auxiliars d'ajuda per que les aplicacions s'integrin millor amb la plataforma.

2.2 Llibreries base de la plataforma

Maemo empra el gestor de finestres Matchbox que és més adequat per les restriccions de maquinari dels dispositius de mà que altres solucions disponibles en Linux.

Una diferencia notable entre maemo i els entorns d'escriptori estàndard és la substitució de bonobo i les tocologies associades construïdes al voltant de CORBA per D-BUS, que proporciona un protocol per a la missatgeria entre aplicacions molt més simple i lleuger.

En resum el conjunt de llibreries base està compost per:

- GTK+ - Proporciona les eines necessàries per crear interfícies d'usuari gràfiques.

- Matchbox - Proporciona un gestor de finestres X11 lleuger adequat per a dispositius de mà.

- Xserver - És la part de la plataforma que s'encarrega de la representació per pantalla tenint en compte les característiques específiques del maquinari.
- D-BUS - Constituït per diversos dimonis que proporcionen una infraestructura lleugera per a la comunicació entre aplicacions i aplicacions amb el sistema, per exemple notificacions del sistema com la indicació de bateria baixa.
- GnomeVFS - Proporciona una abstracció per accedir de forma uniforme a diferents sistemes de fitxer i diversos protocols de xarxa com ara http o ftp.
- GConf - És el sistema per emmagatzemar en una base de dades les preferències de les aplicacions.

2.3 Entorn de desenvolupament

L'entorn de desenvolupament està basat en Scratchbox[14], al voltant del conjunt d'eines i el compilador creuat que proporciona. Aquesta eina habilita un entorn aillat[1] que s'executa sobre un entorn Linux estàndard, maemo.org proporciona un bastidor arrel[2] que conté totes les llibreries compilades per a les plataformes x86 i ARM. Això permet realitzar el desenvolupament i la depuració en la plataforma x86, i finalment la validació i l'empaquetat per la plataforma ARM de cara el desplegament de les aplicacions en el dispositiu.

La habilitat de poder desenvolupar en un entorn x86 retalla en gran mesura el temps necessari, ja que es poden utilitzar eines molt potents i conegudes per la comunitat de programari lliure.

Desenvolupar aplicacions gràfiques per a la plataforma maemo no és gaire diferent que desenvolupar-les per a la plataforma d'escriptori basada en GNOME/GTK+ utilitzant el llenguatge C. Maemo proporciona tan sols un nou conjunt obert de widgets que està orientat als dispositius de mà.

Maemo està construït al voltant dels paquets de debian i de les eines d'empaquetat relacionades, això permet actualitzar l'entorn de desenvolupament amb noves versions d'una forma senzilla.

Una de les característiques úniques de scratchbox és que simplifica la compilació d'aplicacions basades en les eines autoconf/automake. Aquestes eines utilitzen mètodes sofisticats per determinar les capacitats de la plataforma objectiu i adequar el codi font a les característiques determinades per

[1] sandbox
[2] rootstrap

els casos de prova. Normalment això suposa un problema quan s'utilitzen compiladors creuats amb un entorn per a la compilació i validació que no és el mateix maquinari objectiu. Scratchbox proporciona una solució a aquesta circumstància utilitzant QEMU o transparència[3]. Aquest fet redueix significativament el temps necessari per portar aplicacions de l'entorn d'escriptori tradicional als dispositius de mà.

2.4 Arquitectura multimèdia

En aquesta secció es descriu breument l'arquitectura multimèdia en el Nokia 770 i en l'entorn de desenvolupament, ja que tenen lleugeres diferències.

En el diagrama de la figura 2.2 es mostra de forma simplificada l'arquitectura multimèdia implementada en el dispositiu.

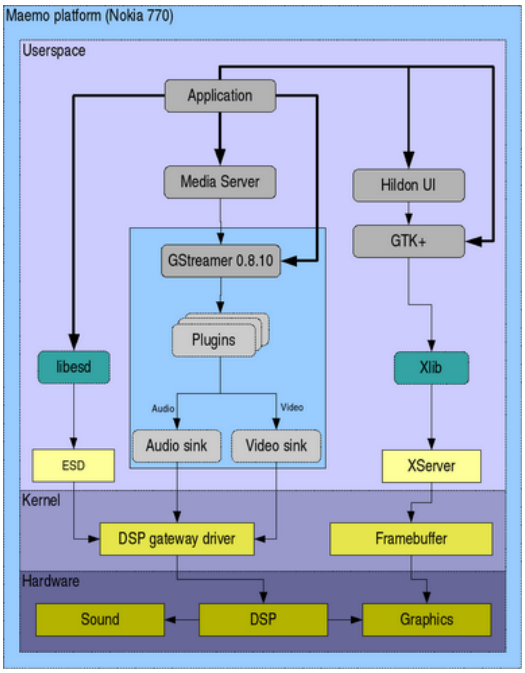

Figura 2.2: Arquitectura multimèdia del Nokia 770.

- *Application* és el programari que utilitza les capacitats multimèdia del dispositiu.

[3]CPU Transparency

- *Media Server* és un dimoni que s'executa en el dispositiu i és utilitzat per la plataforma per tal de produir so. El *Media Server* no està inclòs en l'entorn de desenvolupament.
- *GStreamer* és el *framework* de processament de continguts multimèdia.
- *Plugins* són els mòduls de GStreamer que fan possible tractar diferents tipus de continguts i el processament dels *streams* d'àudio i vídeo.
- *Audio* i *Video sinks* son els plugins de baix nivell de GStreamer que es comuniquen amb el maquinari per mitjà dels controladors del kernel.
- *libesd* és una llibreria que permet a les aplicacions utilitzar el dimoni ESD directament.
- *ESD* és el dimoni ESound que habilita la possibilitat que diferents *streams* simultanis d'àudio PCM puguin utilitzar el maquinari, és un mesclador per programari. Aquest dimoni es substituirà en un futur per el suport d'ALSA.
- *DSP gateway driver* és un modul del kernel que permet la comunicació amb el nucli DSP.
- *Hildon UI* és el *framework* per l'interfície gràfica d'usuari especifica per dispositius mòbils construïda sobre GTK+.
- *GTK+* és un framwork per l'interfície gràfica d'usuari multi plataforma.
- *Xlib* és la llibreria de baix nivell per representar elements de l'interfície gràfica.
- *X server* és la part de la plataforma que dóna suport al dibuixat de gràfics en la pantalla. El servidor X de maemo s'ha optimitzat específicament per el maquinari del Nokia 770.
- *Framebuffer* és l'àrea de memòria on s'escriu la informació que es vol mostrar.

En maemo es disposa també d'un port de la llibreria SDL[4], aquesta és una llibreria de codi font obert multi plataforma de baix nivell que proporciona una abstracció sobre el maquinari d'àudio, vídeo, teclat, etc...

Aquesta llibreria és utilitzada per programari de reproducció multimèdia com el VLC, emuladors de consoles i molts jocs de codi font obert.

[4]Simple Directmedia Layer

El port de la llibreria en el Nokia 770 s'ha construït sobre la Xlib i per tant molta de la funcionalitat que proporciona està implementada per programari.

Un altre aspecte a destacar és que el codi font dels plugins de GStreamer que utilitzen el DSP i el codi dels codecs implementats en el DSP no està disponible actualment.

En el diagrama de la figura 2.3 es mostra de forma simplificada l'arquitectura multimèdia implementada per a l'entorn de desenvolupament.

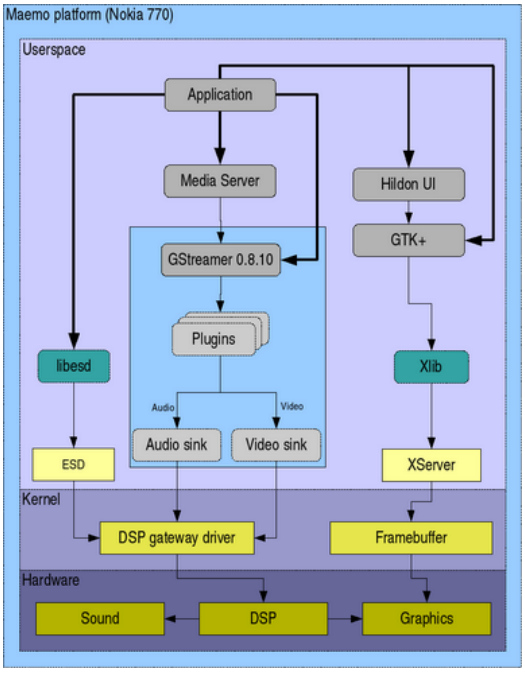

Figura 2.3: Arquitectura multimèdia en l'entorn de desenvolupament x86.

- *osssink*, quan GStreamer està configurat per utilitzar OSS o l'emulació de OSS proporcionada per ALSA, és l'ultim element d'àudio abans que s'envii al controlador OSS del kernel.
- *X server* és el servidor X que executa el sistema operatiu hoste, habitualment XFree86 o XOrg.

2.5 Maemo 2.0 i Internet Tablet 2006

Aquest projecte s'ha realitzat amb la versió 1.1 de maemo i s'executa sobre el sistema Internet Tablet 2005, però en les dates de la redacció d'aquesta memòria és a punt de ser alliberada la nova versió del programari, l'última

notícia fa referència que es presentarà al món durant la GUADEC 2006, que enguany es celebra a Vilanova i la Geltrú.

En aquesta nova versió[12] s'esperen millores importants, tant en l'entorn de desenvolupament com per l'usuari final, entre elles cal destacar el suport a veu sobre IP amb la inclusió del programari Telepathy[16] i Farsight[17].

Respecte al desenvolupament, aquesta nova versió incorporarà la tecnologia EABI[13] que pot tenir un impacte positiu millorant el rendiment de les aplicacions amb alt contingut d'operacions de coma flotant, ja que s'utilitzarà *softfloat* en comptes de l'emulació realitzada per el nucli de Linux. L'inconvenient associat a aquesta tecnologia és un nou format binari que obligarà a tornar a compilar totes les aplicacions existents fins ara.

Aquesta nova versió també incorpora un nou sistema de paquets basat amb les eines de debian apt, dpkg i gnupg. En aquest cas també es trenca amb la versió anterior i caldrà reempaquetar totes les aplicacions.

A part dels canvis comentats s'actualitzarà la versió de la majoria dels components del sistema.

2.6 Instal·lació, comença el projecte

Per la realització del projecte de portar el Video LAN Client a la plataforma maemo s'ha utilitzat un portàtil Dell Latitude D600 amb una instal·lació de la distribució Linux Ubuntu Dapper Drake 6.06 beta.

Sobre aquest entorn s'ha instal·lat l'entorn de desenvolupament maemo seguint els passos descrits en el tutorial[10] i s'ha procedit realitzar els passos d'introducció a l'entorn, exemple "hello world".

Un cop familiaritzat amb les eines de compilació he procedit a estudiar els aspectes relatius al port d'aplicacions i al desenvolupament d'una interfície gràfica d'usuari basada en GTK+/Hildon. Com a petit projecte per assolir aquest objectiu he resolt diverses errades en el port del joc Nethack (gtk2hack) al Nokia 770 i he contribuït millorant la interfície d'usuari per adaptar-la millor a les característiques del Nokia 770, el resultat es pot obtenir en el meu blog[15].

Un cop arribat a aquest estadi, seguint les indicacions del tutorial[19], s'ha procedit a realitzar les primeres temptatives del port amb el codi font del VLC 0.7.2. i els components associats a aquesta versió; en la figura 2.4 és pot veure un exemple d'execució en l'entorn de desenvolupament.

Per realitzar aquest projecte ha estat necessari l'estudi, en un cert grau de profunditat, del codi font de diversos components per tal de determinar quines millores i adaptacions podia realitzar per tal d'incrementar el rendiment de la aplicació en el maquinari i com es podien aprofitar els recursos

del Nokia 770.

En els següents capítols d'aquest document es detallaran els components i les modificacions realitzades, com a resum per proporcionar una visió global a continuació hi ha una llista dels components estudiats.

- VLC - Primer sobre la versió 0.7.2 i finalment sobre la versió 0.8.6 que es troba en desenvolupament actualment. Proporciona el GUI i integra tots els components.
- libdvbpsi - Proporciona demultiplexat del *stream* mpeg2-ts
- libmpeg2 - Proporciona el codec de vídeo mpeg2
- ffmpeg - Proporciona diversos codecs d'àudio i vídeo, s'ha descartat un cop realitzat un estudi preliminar, és un component complex i no aporta millores significatives.
- libmad - Proporciona el codec d'àudio mpeg2, en l'estudi preliminar s'han detectat aspectes millorables, però la millor opció per l'àudio hauria de ser poder utilitzar el codec implementat al DSP que ja disposa el Nokia 770, malauradament el codi font dels plugins de GStreamer[11] que utilitzen el DSP no han estat alliberats encara.
- libSDL - Proporciona una API per accedir al dispositiu gràfic, proporciona un overlay YUV i conversió de l'espai de colors abans d'enviar el quadre a la pantalla.

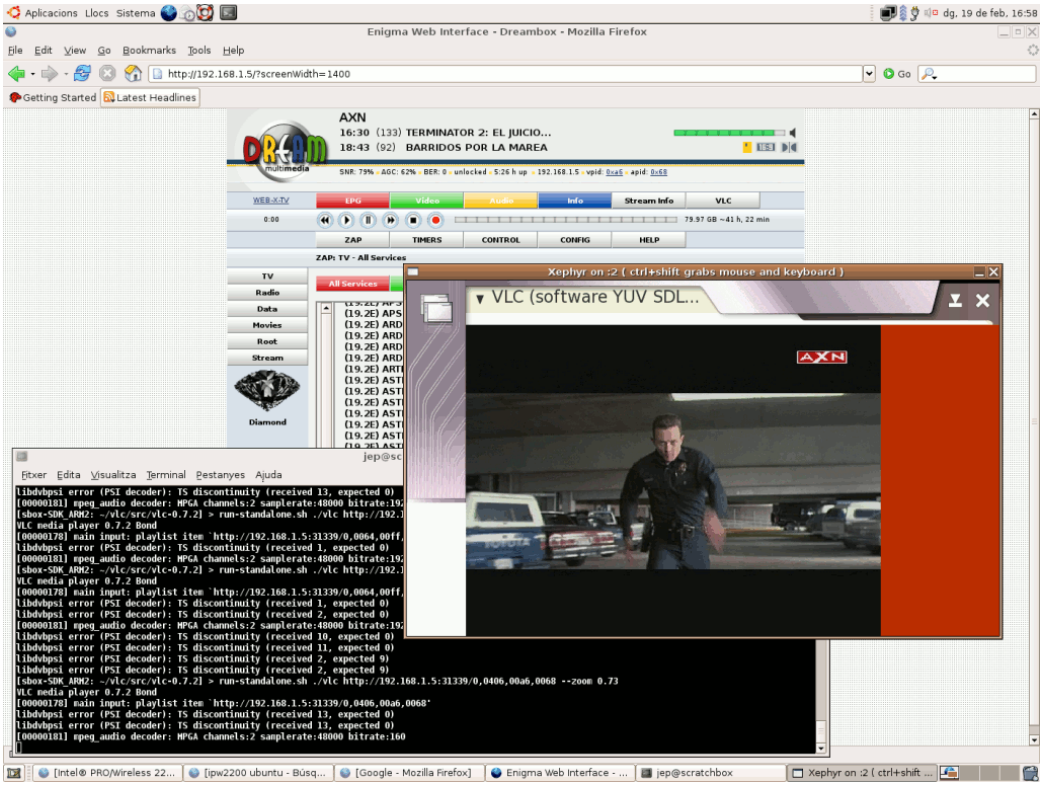

Figura 2.4: En aquesta captura de pantalla es pot observar l'entorn de desenvolupament, en execució el port de VLC executant-se amb QEMU reproduint un canal rebut per satèl·lit amb el Dreambox 7000S, el *stream* es rep per la xarxa sense fils del portàtil. En un segon pla el navegador obert amb la pàgina de gestió remota del Dreambox 7000S.

Capítol 3

El projecte VideoLAN Client

El projecte VideoLAN[18] va començar com a projecte escolar el 1996 per estudiants de "Ecole Centrale, Paris". Aquests estudiants volien poder veure la televisió en els seus ordinadors personals, per això van començar per escriure el VideoLAN Server (VLS) i el VideoLAN Client (VLC) per generar *streams* i reproduir *streams* MPEG2. Van aconseguir servir i llegir el primer *stream* el 1998.

Aquestes dues aplicacions es van dissenyar per ser modular, en essència el nucli de les aplicacions consisteix en la comunicació entre els diferents mòduls; aquest fet ha permès que fos relativament senzill portar les aplicacions a diferents plataformes i sistemes operatius.

El 2001 es va alliberar el projecte sota la llicencia GPL, fet que ha facilitat la creació d'una comunitat de desenvolupadors a internet. Un d'ells va realitzar el port a Win32 amb només 6 mesos.

El VideoLAN Client és un reproductor multimèdia de codi font obert altament portable. Suporta diversos formats d'àudio i de vídeo com MPEG-1, MPEG-2, MPEG-4 DivX, mp3, ogg, etc, així com DVDs, VCDs i diversos protocols de *streaming* en xarxa.

El VLC no es tracta d'un projecte autònom sinó que es serveix de diferents projectes de codi font obert per tal de construir-se, en essència aquest projecte integra diverses interfícies d'usuari i funcionalitats encarnades en mòduls sobre una arquitectura basada en plugins.

Es tracta d'una aplicació multi fil amb un gran nombre de mòduls, alguns d'ells disponibles únicament per a determinades plataformes.

El diagrama d'arquitectura de la figura 3.1 descriu gràficament la divisió en mòduls.

Els mòduls és poden classificar en les categories següents:

- Interfície d'usuari: Consola, GTK+, Gnome, QT, KDE, WxWidgets,

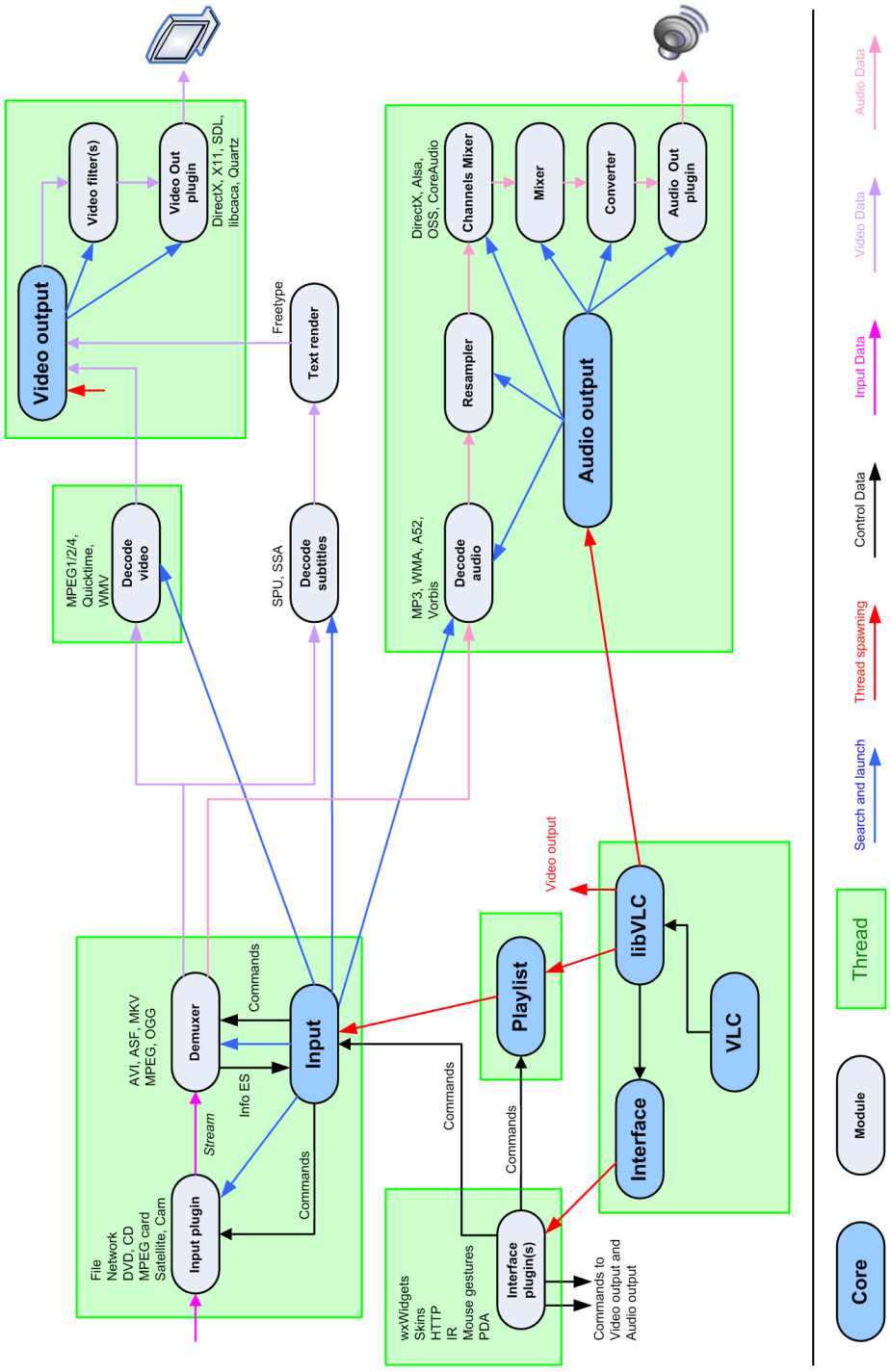

Figura 3.1: Arquitectura del VideoLAN Client.

HTTP/Webpage, GTK+/PDA, control remot, port IR.

- Origen de dades: Fitxer, protocols de xarxa (UDP, RTP, HTTP, FTP, MMS), DVD, VCD, SVCD, AudioCD, DVB-S/C/T, Capturadores de Vídeo.
- Formats de dades: MPEG ES/PS/TS/PVA/mp3, AVI, ASF/wmv/wma, Ogg/OGM/Annodex, MP4/MOV, Matroska, Real, Wav, Raw DV, Raw AAC, FLAC audio,...
- Codec de vídeo: MPEG 1 i 2, DivX 1,2 i 3, MPEG-4/DivX 5/XviD/3ivX D4, H.264, DV, Cinepak, Theora, H.263/H.263i, MJPEG, WMV 1, 2 i 3...
- Subtítols: DVD, SVCD, CVD, DVB, OGM, Matroska, MicroDVD, Vobsub, SubRIP, SSA1-4,...
- Filtres de vídeo: desentrellaçat, crop, rotació,...
- Sortides de vídeo: DirectX, GDI,, X11, XVideo, SDL, Framebuffer, ASCII art, ASCII art acolorit, MGA, GGI
- Codecs d'àudio: MPEG Layer 1 i 2, MP3, AC3, DTS, LPCM, AAC, Vorbis, WMA 1/2, ADPCM, DV Audio, FLAC, QDM2/QDMC, MACE, AMR, Speex,...
- Sortides d'àudio: OSS, ALSA, DirectX, S/PDIF, SDL, ESD, aRts, JACK.
- Sortides de steaming: UDP, RTP, HTTP, MMSH.
- Serveis d'informació: SAP/SDP, SLP

3.1 Respecte al projecte

Respecte al projecte s'han fet modificacions en el codi del VideoLAN client enfocades en dues direccions:

1. Modificar la interfície d'usuari per tal de que sigui adequada per al Nokia 770 utilitzant el *framework* d'aplicació Hildon i aplicant la guia d'estil corresponent.
2. Modificar el mòdul de sortida de vídeo per tal d'optimitzar-lo i mirar d'incrementar el nombre de quadres per segon obtinguts en el dispositiu.

3.1.1 Modificació de l'interfície d'usuari

Respecte a l'adaptació de la interfície d'usuari s'ha aplicat l'experiència adquirida amb el port del gtk2hack, fent les modificacions necessàries en el mòdul d'interfície GTK+/PDA original del VideoLAN Client.

Els canvis en el mòdul estan controlats per la definició del preprocessador **HILDON** i s'ha afegit al guió de les eines autoconf/automake un nou mòdul per tal de poder compartir el codi en les dues interfícies.

En la figura 3.2 es pot veure la interfície d'usuari modificada executant-se en l'entorn de desenvolupament.

Figura 3.2: Interfície d'usuari PDA-Hildon.

3.1.2 Modificació en el mòdul V.Out-SDL

Degut a les limitacions del maquinari del Nokia 770 gran part del projecte s'ha enfocat a simplificar la complexitat i optimitzar els recursos disponibles.

Amb la intenció d'augmentar els quadres per segon obtinguts en el dispositiu, s'ha realitzat un anàlisi en profunditat del modul de renderització implementat sobre la llibreria SDL; així com la implementació d'aquesta sobre Xlib respecte el codi font del port de maemo.

La funcionalitat principal proporcionada per aquest mòdul de cara al projecte és facilitar un overlay YUV on el codec escriu els quadres descodificats, realitzar la conversió de color YUV a RGB, escalar la imatge i mostrar el quadre en la pantalla.

Per tant es tracta d'una secció crítica amb un alt us del bus de dades i accés a la memòria. Per altre banda també és significatiu el cost del càlcul necessari per realitzar la conversió de l'espai de color.

De l'estudi del codi font de SDL es va descobrir que la conversió de color i l'escalat es realitzava per dos passos diferents consecutius i memòria intermèdia auxiliar addicional, això suposa una gran quantitat d'accessos a memòria.

En conseqüència d'aquests fets el mòdul del VLC de renderitzat s'ha modificat, per tal d'implementar part de la funcionalitat proporcionada per la llibreria SDL amb l'objectiu de realitzar l'escalat i la conversió de color en un sol pas, reduint així el nombre accessos a memòria.

En la implementació presentada s'ha optat per reduir la resolució original a la meitat, descart alternatiu de píxels, reduint a una quarta part el nombre d'operacions necessàries per realitzar la conversió de color i és realitza en un sol pas llegint l'overlay YUV un sol cop.

Respecte a l'escriptura del quadre, s'ha modificat per escriure directament en el *framebuffer* de la Xlib. Per tal de proporcionar la il·lusió d'un millor aprofitament de la pantalla, els píxels es doblen horitzontalment, això fa que quedi modificada la relació d'aspecte final a l'estil del que succeïx en els televisors panoràmics quan mostren una emissió 4:3.

Capítol 4

Una introducció a MPEG2

La codificació de vídeo MPEG es descriu aquí per tal de proporcionar les bases per entendre la complexitat del sistema. També es fa una breu descripció dels formats contenidors.

4.1 Els *frames*

Respecte a l'algoritme de compressió mpeg[1] es diferencien tres classes de *frame*:

- Els **I-frames** poden ser reconstruïts sense necessitar cap referència a altres *frames*.
- Els **P-frames** són prediccions cap a endavant[2], no es poden reconstruir sense la informació de l'últim I-frame o P-frame.
- Els **B-frames** són alhora prediccions endavant i endarrere[3] relatives a l'últim/proper I-frame o P-frame, per tan fan falta dos *frames* per reconstruir-los.

Els **I-frames** són *intra coded*, els **P-frames** i els **B-frames** són *inter coded*.

Això significa que per descodificar un flux de dades MPEG fa falta com a mínim memòria intermedia per contenir tres quadres, un per a la predicció cap a endavant i un per a la predicció enderrere. El tercer conté el *frame* que s'ha de reconstruir a continuació. En la figura 4.1 es pot observar

[1] codec
[2] forward predicted
[3] forward/backward predicted

esquemàticament diversos *frames* en l'ordre que són mostrats. Tal com es mostra en la figura semblaria que hem de descartar els **B-frame** fins al següent **P-frame** o **B-frame**; afortunadament l'ordre de reproducció no és l'ordre de codificació. Els *frames* apareixen en un flux de dades MPEG en l'ordre tal que els *frames* referits precedeixen als *frames* que s'hi refereixen. Com a exemple, la seqüència de la figura es transforma en l'ordre: I P B B B P B B B.

Una altre de les tasques que ha de realitzar el descodificador MPEG és reordenar els *frames*, per aconseguir-ho cada *frame* va enumerat de forma ascendent (mòdul 1024).

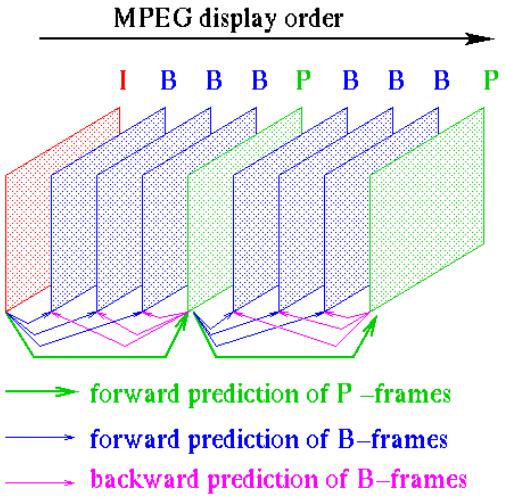

Figura 4.1: Exemple d'un conjunt de *frames* MPEG en l'ordre de reproducció

4.2 La predicció

La predicció pren significat quan en el cas de la imatge en moviment i semblança d'informació entre *frames* consecutius.

Imaginem un I-frame que mostra un triangle sobre un fons blanc, un P-frame consecutiu podria mostrar el mateix triangle però en una altre posició, la figura 4.2 ens ho mostra. La predicció significa que proporcionant un vector de moviment[4] que expliqui com es mou el triangle del I-frame per obtenir el P-frame és tota la informació necessària.

Aquest vector de moviment és part del sistema MPEG i està dividit en dues parts: moviment horitzontal i moviment vertical. Un valor positiu

[4]motion vector

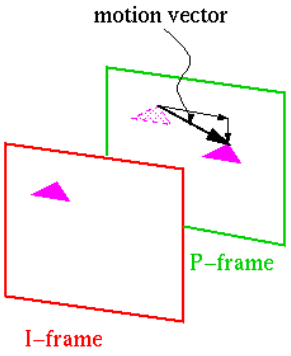

Figura 4.2: Exemple de vector de moviment

significa moviment a la dreta o cap avall i un valor negatiu moviment cap a l'esquerra o cap amunt, respectivament. Aquest valors de moviment estan compresos entre -64 i +63, per tant, com a màxim, l'àrea desplaçada es pot trobar a 64 píxels de distància.

Però aquest model es basa en l'assumpció que cada canvi entre *frames* es pot expressar simplement com un desplaçament de píxels. La figura 4.3 ens mostra un exemple on no només hi ha un desplaçament de píxels sinó que a més hi ha un rotació a la dreta, en aquest cas es produirà un error de predicció; per solucionar aquest problema el sistema MPEG incorpora una matriu per compensar aquest error de predicció. La matriu de compensació de l'error de predicció descriu els canvis de color que s'han d'aplicar als píxels afectats.

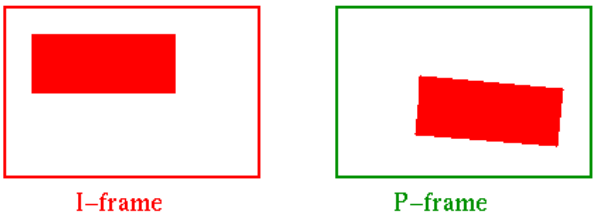

Figura 4.3: Exemple de predicció

En resum, la reconstrucció dels cuadres *inter coded* es realitza en dos passos, com es pot veure en la figura 4.4:

1. Aplicar el vector de moviment al *frame* de referència.

2. Afegir la matriu amb la compensació de l'error de predicció al resultat.

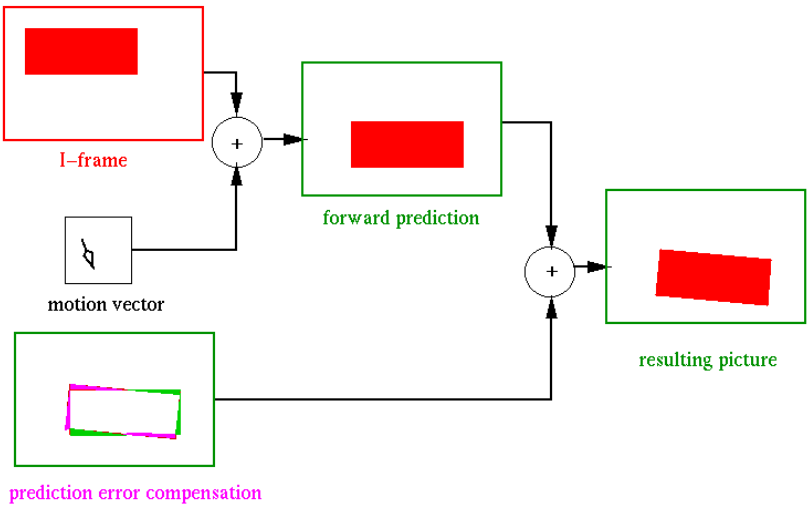

Figura 4.4: Exemple de reconstrucció del P-frame

Cal destacar que la matriu de compensació de l'error de predicció ocupa menys bytes que tot el *frame* sencer per que les parts amb valor zero són descartades del flux MPEG gràcies a la compressió DCT[5] descrita més endavant.

4.3 Els macroblocs

Que passa quan una part del **I-frame** es mou en una direcció i una altre part en una de diferent?

Davant d'aquesta situació es veu clarament que el vector de moviment per tot el *frame* no és suficient. El sistema MPEG resol aquesta situació dividint el *frame* en macroblocs de 16x16 píxels. Cada macrobloc té el seu vector de moviment; està clar que això no elimina la situació de moviment contradictori però en minimitza la probabilitat.

I si un moviment contradictori succeeix?

En aquesta situació caldrà aportar tota la informació del macrobloc, el sistema MPEG permet decidir individualment en els P-frames si els macroblocs han de ser *inter coded* o *intra coded* en funció de la grandària de l'error de predicció.

Cada macrobloc es codifica amb 4 blocs de luminescència i 2 blocs de cromatografia, com es veu en la figura 4.5. Cada bloc té una dimensió de 8x8 píxels. Els blocs de luminescència contenen la informació de la intensitat per

[5]Discrete Cosine Transformation

cada punt del macrobloc. Els blocs amb informació de cromatografia contenen la informació del color per a cada 2x2 píxels ja que l'ull humà és incapaç de distingir-ho. El valor del color el dividim en dos parts representades per C_b i C_r.

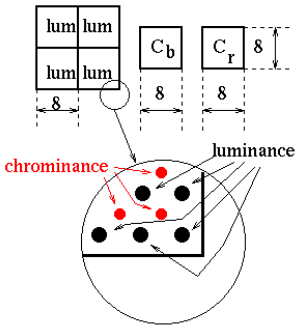

Figura 4.5: Exemple de macrobloc

Depenent del tipus de macrobloc la informació continguda és relativa als píxels reals o a la correcció del color per a la compensació dels errors de predicció. En tots dos casos la informació es comprimeix per mitjà de l'algoritme DCT[6].

4.4 DCT: Discrete Cosine Transformation

L'algoritme per comprimir la informació del sistema MPEG és l'anomenat DCT; aquest algoritme realitza una transformació entre dos dominis obtenint cert grau de redundància que es pot eliminar en la transmissió de la informació.

Figura 4.6: Exemple de bloc amb informació de luminescència.

Suposarem un bloc amb informació de luminescència com el de la figura 4.6 i que codifiquem la intensitat de cada punt en un interval de 0 a 255[7], on 0 representa el negre i 255 el blanc.

[6]Discrete Cosine Transformation
[7]MPEG2 estableix un interval -256 a 256

120	108	90	75	69	73	82	89
127	115	97	81	75	79	88	95
134	122	105	89	83	87	96	103
137	125	107	92	86	90	99	106
131	119	101	86	80	83	93	100
117	105	87	72	65	69	78	85
100	88	70	55	49	53	62	69
89	77	59	44	38	42	51	58

Figura 4.7: Matriu de luminescència.

En aquesta situació podem representar la imatge mitjançant la matriu de valors de la figura 4.7.

En aquest exemple podrem definir els 64 valors de la matriu amb només 5 enters si apliquem la següent formula matemàtica, anomenada *discrete cosine transformation*.

$$F(u,v) = \frac{C_u}{2}\frac{C_v}{2}\sum_{y=0}^{7}\sum_{x=0}^{7} f(x,y)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

amb:

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & \text{si} \quad u = 0 \\ 1 & \text{si} \quad u > 0 \end{cases} ; C_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{si} \quad v = 0 \\ 1 & \text{si} \quad v > 0 \end{cases}$$

Quan $f(x,y)$ és el valor de la intensitat de llum de cada punt a la posició $[x,y]$ de la matriu, el resultat és la matriu F representat en la figura 4.8.

Com es pot veure, molts dels valors són zero i els valors significatius es troben concentrats en la part superior esquerra de la matriu.

700	90	100	0	0	0	0	0
90	0	0	0	0	0	0	0
-89	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 4.8: Matriu comprimida per DCT.

Llavors si realitzem un recorregut en zigzag tal com es mostra en la figura

4.9 tindrem els valors 700 90 90 -89 0 100 0 0 0 ... 0, i podrem descartar els zeros de la dreta substituint-los per una marca de final de bloc.

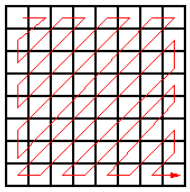

Figura 4.9: Recorregut en zigzag.

El decodificador MPEG pot reconstruir el valor dels píxels utilitzant la següent formula matemàtica, anomenada *inverse discrete cosine transformation(IDCT o DCT $^{-1}$)*.

$$f(x,y) = \sum_{u=0}^{7} \sum_{v=0}^{7} F(u,v) \frac{C_u}{2} \frac{C_v}{2} \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

On $F(u,v)$ és el valor de la matriu de transformació a la posició $[u,v]$. Els resultats són exactament els valors originals. Per tant, el sistema compressió MPEG es podria dir que no té pèrdua de qualitat però això no és cert ja que el valors són escalats ja que el DCT dóna com a resultat valors fins a 2047. Per reduir-los a la longitud d'un byte es divideix el resultat del DCT per un valor i el descodificador torna a multiplicar per aquest valor per tal de recuperar el valor inicial, en aquest pas s'introdueixen errors d'arrodoniment. Tot i així l'error introduït és imperceptible per l'ull humà.

MPEG defineix una matriu de quantització que defineix diferents valors d'escalat per cada valor de transformació en funció de la seva posició.

En quant a l'algoritme que resolgui el DCT i l'IDCT, la definició de MPEG no especifica quin s'ha d'utilitzar; existeixen diferents algoritmes amb major o menor complexitat dissenyats per assolir la qualitat requerida dins del marges de tolerància respecte l'error estadístic permès, establert per l'especificació de MPEG.

4.5 Els contenidors

En els apartats anteriors d'aquest capítol s'ha descrit les tècniques de compressió del sistema MPEG, aquest és un sistema complert i també proporciona una definició respecte diferents contenidors de la informació multimèdia orientats a diferents aplicacions.

Es defineixen així tres tipus de contenidors: ES, PS i TS.

Quan es reprodueix un vídeo MPEG d'un DVD, el *stream* MPEG està actualment compost per diversos *streams* anomenats *Elementary Streams* (ES): hi ha un *stream* per el vídeo, un *stream* per l'àudio i un altre per els subtítols. Aquests *streams* diferents estan barrejats, tots junts en un únic *Program Stream* (PS). Per tant els fitxers .VOB que es poden trobar en un DVD són actualment fitxers MPEG-PS. Però el format PS no és adequat per a l'emissió de vídeo en un xarxa o des d'un satèl·lit, per aquesta situació hi ha un altre format anomenat *Transport Stream* (TS) dissenyat per poder ser transportat per aquest tipus de canals.

4.6 Respecte al projecte

En aquest projecte he utilitzat la implementació del codec de vídeo MPEG, proporcionat per la llibreria libmpeg2[20].

En aquesta llibreria s'hi ha introduït diverses modificacions per tal d'optimitzar la descodificació en el Nokia 770; els canvis s'han orientat principalment en accelerar la implementació del IDCT tenint en compte les característiques del ARM926TEJ i l'ample de banda del bus de dades.

La llibreria libmpeg2 proporciona una implementació genèrica en C de l'algoritme IDCT, a més de versions optimitzades per aprofitar les extensions MMX i altivec. Aquestes implementacions estan basades en el càlcul amb enters de punt fix amb una precisió de 16 bits.

En resum les modificacions realitzades a la llibreria són:

- Afegir una nova arquitectura (ARM) a la llibreria, modificació dels fitxers de configuració de les eines autoconf/automake.
- Afegir una nova implementació de l'IDCT associant-la a la nova arquitectura a partir de la implementació genèrica en C.

Respecte a la implementació de l'IDCT optimitzada s'han utilitzat les següent tècniques:

- Reutilització de variables per disminuir-ne el nombre i eliminar els accessos a la pila per manca de registres.
- Desplegat de bucles.
- Reduir el nombre d'accessos a memòria canviant l'estratègia d'escriptura del resultat del IDCT.

- Implementar les multiplicacions i multiplicacions amb acumulació per mitjà de les instruccions assemblador del conjunt de les extensions EDSP de l'ARM926TEJ, utilització d'assemblador en línia i macros del preprocessador.

- Valors de les constants de 16bits aparellades en memòria i càrrega en un únic registre.

El procés seguit, s'ha iniciat amb l'estudi del codi font per tal de determinar les seccions millorables, introduir petits canvis, estudiar el codi assemblador generat per el compilador i realització de testo d'unitat per determinar la millora introduïda.

Les proves d'unitat s'han realitzat utilitzant el programa d'exemple d'us de la llibreria mpeg2dec per tal de descodificar sense renderització un fitxer mpeg2 en una resolució 800x480 píxels i 25 quadres per segon. Les proves s'han realitzat principalment utilitzant QEMU des de l'entorn de desenvolupament i en el Nokia 770 per les versions candidates.

Capítol 5

Conclusions

L'objectiu perseguit en aquest projecte, utilitzar el Nokia 770 com a reproductor portàtil, s'ha assolit només parcialment degut a les limitacions del maquinari.

Els resultats finals obtinguts en l'entorn de desenvolupament han estat: aproximadament 17 quadres per segon amb so i 21 quadres per segon sense so.

Respecte l'execució en el dispositiu real només hem pogut aconseguir 3 quadres per segon sense so i l'àudio sense el vídeo.

Durant el desenvolupament del projecte s'han realitzat diversos experiments i temptatives per tal d'augmentar el nombre de quadres per segon en el dispositiu, l'estratègia que s'ha explorat ha consistit en eliminar el color.

En aquest sentit s'ha intentat els canvis següents:

- Modificar el codec de vídeo (libmpeg2) per tal que calculi l'IDCT només per els blocs de luminescència.
- Modificar el mòdul de renderització per fer la conversió de color ràpida $C(r, g, b) = (L >> 3, L >> 3, L >> 3)$

Tot i així no s'ha aconseguit una millora significativa i s'ha obtingut també de l'ordre de 3 quadres per segon quan s'executa en el dispositiu.

Aquest resultat fa pensar que el límit està determinat principalment per l'ample de banda del bus de dades (16 bits), més que en la capacitat de càlcul del processador.

Per altre banda la nova versió de maemo pot ser que impacti lleugerament de forma positiva amb la millora esperada per els càlculs de coma flotant.

Diverses idees no s'han pogut abordar degut a la no disponibilitat de temps, a la magnitud d'algunes d'elles o a la manca d'informació suficient, aquestes són:

- Implementar el codec en el DSP, es tractaria d'un projecte d'una magnitud bastant elevada, amb uns requeriments de coneixements i experiència fora del meu abast.

- Utilitzar el codec d'àudio mpeg2 ja implementat en el DSP i balancejar la càrrega entre els dos nuclis. El codi font del codec i el codi font del plugin de GStreamer que l'utilitza no està publicat, tampoc hi ha informació tècnica disponible de moment de la forma en que es pot utilitzar el codec del DSP directament sense GStreamer. Per altre banda l'arquitectura del VLC pot no ser la més adequada per interactuar amb un codec d'àudio que accedeix directament al dispositiu de sortida al mateix temps que descodifica.

- Un replantejament global de la solució, descartar el VLC i implementar els components com a plugins del GStreamer. Implementar una aplicació basada en GStreamer per dur a terme la reproducció.

En les figures 5.1 i 5.2 es pot observar el dispositiu en funcionament.

Figura 5.1: Nokia 770 en funcionament.

Figura 5.2: Nokia 770 sobre el televisió, tots dos reproduint el mateix programa; hi ha un cert decalatge degut a la comunicació per la xarxa.

Apèndix A

Descripció del guió build.sh

A.1 build.sh

Aquest guió interactiu és l'encarregat de la descarrega des de Internet del codi font i de la compilació del port de VLC a Maemo.

S'ha estructurat el guió en diverses funcions genèriques de suport i altres especifiques per a la generació dels diversos components.

A continuació es facilita una descripció de les funcions i el codi font corresponent a cada una d'elles.

A.1.1 Preludi

Aquest llistat corresponent al preludi del guió de comandes, aquí es defineixen les variables yes_all i no_all que controlen part del procés d'execució i s'inclou el modul question_func.sh per a la funció genèrica d'interrogació.

```
#!/scratchbox/tools/bin/bash
yes_all=""
no_all=""
. /scratchbox/etc/question_func.sh
```

A.1.2 Funció ask_to_do()

Aquest funció genèrica s'utilitza en el guió per tal d'interrogar a l'usuari sobre els passos a realitzar.

```
function ask_to_do()
{
    echo
    if [ "$yes_all" = "true" ]; then
        echo $1
        return 1
    fi
    if [ "$no_all" = "true" ]; then
        echo $1
        return 0
    fi
```

```
    ask="`question_\"$1 (y/n/q/ya/na) ?\"_2_\"y\"`"
    resp="$ask"
    if [ "$resp" = "y" ]; then
        return 1
    fi
    if [ "$resp" = "ya" ]; then
        yes_all="true"
        return 1
    fi
    if [ "$resp" = "n" ]; then
        return 0
    fi
    if [ "$resp" = "na" ]; then
        no_all="true"
        return 0
    fi

    if [ "$resp" = "q" ]; then
        exit 1
    fi
}
```

A.1.3 Funció makedir()

Aquesta funció serveix per crear directoris en el cas de que no existeixin.

```
# Create $1 dir if it not exists
function makedir
{
    if test ! -d $1
    then
        echo "Creating_$1_directory"
        mkdir $1
    fi
}
```

A.1.4 Funció downpackage()

Aquesta funció serveix per descarregar un paquet des de l'url especificada si no s'ha descarregat anteriorment.

```
# Download package $1 from $2 url if it not exists
function downpackage
{
    if test ! -f $1
    then
        echo "Downloading_$1_from_$2"
        wget -c $2
    fi
}
```

A.1.5 Funció uncompress_gzip()

Aquesta funció serveix per descomprimir paquets gzip.

```
# Uncompress package $1 if $2 dir not exists (gzip file)
function uncompress_gzip
{
    if test ! -d $2
    then
        echo "Uncompress_$1"
        tar -zxvf ../packages/$1
    fi
}
```

A.1.6 Funció uncompress_bzip2()

Aquesta funció serveix per descomprimir paquets bzip2.

```
# Uncompress package $1 if $2 dir not exists (bzip2 file)
function uncompress_bzip2
{
    if test ! -d $2
    then
        echo "Uncompress $1"
        tar -jxvf ../packages/$1
    fi
}
```

A.1.7 Funció patch_X11_include_dir()

Aquest és un pegat per evitar una error de compilació en el sistema maemo per algunes del les llibreries de que depèn VLC.

```
# Create a symlink for X11 include directory
function patch_X11_include_dir
{
    if test ! -d /usr/include/X11
    then
        echo "Adding symlink /usr/include/X11 -> /usr/X11R6/include/X11"
        lastpwd="$PWD"
        cd /usr/include
        ln -s /usr/X11R6/include/X11
        cd "$lastpwd"
    fi
}
```

A.1.8 Funció patch_source()

Aquesta és una funció genèrica per aplicar els pegats de la carpeta de pegats amb les modificacions realitzades.

```
# Patch source code on $1 directory
function patch_source
{
    if test -f .patch_$1
    then
        return 0
    fi
    echo "Patching $1"
    cp -rf ../patches/$1/* $1
    echo "done" > .patch_$1
}
```

A.1.9 Funció build_libmad()

Aquesta funció s'encarrega de construir el component libmad, codec d'àudio.

```
# Build libmad
function build_libmad
{
    target="libmad-0.15.1b"
    if test -f .build_$target
    then
        return 0
    fi
    cd $target

    ask_to_do "$target : configure";
```

```
    if [ "$?" = "1" ]; then
        make distclean
        ./configure --enable-release --prefix=/usr \
            --enable-static \
            --enable-shared \
            --enable-speed \
            --enable-fpm=arm \
            --enable-sso \
            --disable-debugging

    fi

    ask_to_do "$target : make";
    if [ "$?" = "1" ]; then
        make
        if [ "$?" = "0" ]; then
            echo "done" > ../.build-$target
        fi
    fi

    ask_to_do "$target : make install";
    if [ "$?" = "1" ]; then
        make install
    fi
    cp /usr/lib/libmad* .
    cd ..
}
```

A.1.10 Funció build_libmpeg2

Aquesta funció s'encarrega de construir el component libmpeg2, codec de vídeo.

```
# Build libmpeg2
function build_libmpeg2
{
    target="mpeg2dec-0.4.0"
    if test -f .build-$target
    then
        return 0
    fi
    patch_source $target
    cd $target

    ask_to_do "$target : configure";
    if [ "$?" = "1" ]; then
        ./bootstrap
        make distclean
        ./configure --enable-release --prefix=/usr \
            --without-x
    fi

    ask_to_do "$target : make";
    if [ "$?" = "1" ]; then
        make
        if [ "$?" = "0" ]; then
            echo "done" > ../.build-$target
        fi
    fi

    ask_to_do "$target : make install";
    if [ "$?" = "1" ]; then
        make install
    fi

    cd ..
}
```

A.1.11 Funció build_libdvbpsi()

Aquesta funció s'encarrega de construir el component libdvbpsi, demuxer per els contenidors en format MPEG.

```
# Build libdvbpsi
function build_libdvbpsi
{
    target="libdvbpsi4-0.1.5"
    if test -f .build_$target
    then
        return 0
    fi

    cd $target

    ask_to_do "$target : configure";
    if [ "$?" = "1" ]; then
        ./bootstrap
        make distclean
        ./configure --enable-release --prefix=/usr \
            --with-x \
            --x-includes=/usr/X11R6/include \
            --x-libraries=/usr/X11R6/lib
    fi

    ask_to_do "$target : make";
    if [ "$?" = "1" ]; then
        make
        if [ "$?" = "0" ]; then
            echo "done" > ../.build_$target
        fi
    fi

    ask_to_do "$target : make install";
    if [ "$?" = "1" ]; then
        make install
    fi

    cd ..
}
```

A.1.12 Funció build_vlc()

Aquesta funció s'encarrega de construir el VideoLAN Client.

```
# Build vlc
function build_vlc
{
    target="vlc-0.8.6-svn"
    if test -f .build_$target
    then
        return 0
    fi
    patch_source $target
    cd $target

    ask_to_do "$target : configure";
    if [ "$?" = "1" ]; then
        ./bootstrap
        make distclean
        ./configure  --prefix=/usr \
            --disable-release \
            --enable-debug \
            --disable-static \
            --enable-fast-install \
            --disable-st \
            --disable-hal \
            --enable-dbus \
            --enable-mostly-builtin \
            --enable-optimize-memory \
            --enable-optimizations \
            --disable-debug \
            --disable-sout \
            --disable-shout \
            --disable-httpd \
            --disable-vlm \
            --disable-growl \
            --disable-livedotcom \
            --disable-dv \
            --disable-dvd \
            --disable-dvdread \
```

```
--disable-dvdplay \
--disable-dvdnav \
--disable-smb \
--enable-dvbpsi \
--disable-v4l \
--disable-pvr \
--enable-gnomevfs \
--disable-libcdio \
--disable-cddax \
--disable-libcddb \
--disable-vcdx \
--disable-cdda \
--disable-vcd \
--disable-dvb \
--disable-screen \
--disable-ogg \
--disable-mkv \
--disable-mod \
--disable-mpc \
--disable-gme \
--enable-mad \
--disable-ffmpeg \
--disable-ffmpegaltivec \
--disable-faad \
--disable-twolame \
--disable-quicktime \
--disable-real \
--disable-realrtsp \
--disable-a52 \
--disable-dts \
--disable-flac \
--enable-libmpeg2 \
--disable-vorbis \
--disable-tremor \
--disable-speex \
--disable-tarkin \
--disable-theora \
--disable-dirac \
--disable-png \
--disable-x264 \
--disable-cmml \
--enable-x11 \
--disable-xvideo \
--disable-glx \
--disable-xinerama \
--disable-opengl \
--enable-sdl \
--disable-freetype \
--disable-fribidi \
--disable-libxml2 \
--disable-svg \
--disable-snapshot \
--disable-qte \
--disable-hd1000v \
--disable-directx \
--disable-fb \
--disable-mga \
--disable-svgalib \
--disable-directfb \
--disable-ggi \
--disable-glide \
--disable-aa \
--disable-caca \
--disable-wingdi \
--enable-oss \
--disable-alsa \
--enable-esd \
--disable-portaudio \
--disable-arts \
--disable-waveout \
--disable-macosx-audio \
--disable-hd1000a \
--disable-jack \
--disable-cyberlink \
--disable-upnp \
--disable-skins2 \
--enable-pda \
--enable-hildon \
--disable-wxwidgets \
--disable-opie \
--disable-qnx \
--disable-ncurses \
--disable-xosd \
```

```
            --disable-visual \
            --disable-galaktos \
            --disable-goom \
            --disable-daap \
            --disable-bonjour \
            --disable-lirc \
            --disable-corba \
            --disable-gnutls \
            --disable-slp \
            --disable-loader \
            --disable-activex \
            --disable-mozilla \
            --disable-mediacontrol-python-bindings \
            --disable-java-bindings \
            --enable-plugins \
            --disable-testsuite \
            --with-dvbpsi-tree=../libdvbpsi4-0.1.5 \
            --with-mad-tree=../libmad-0.15.1b \
            --with-libmpeg2-tree=../mpeg2dec-0.4.0 \

    fi

    ask_to_do "$target : make";
    if [ "$?" = "1" ]; then
        make
        if [ "$?" = "0" ]; then
            echo "done" > ../.build_$target
        fi
    fi

    cd ..
}
```

A.1.13 Funció test_vlc()

Aquesta funció executa el vlc amb diversos paràmetres de funcionament.

```
function test_vlc
{
    target="run-vlc"
    yes_all=""
    no_all=""

    ask_to_do "$target: run-standalone_vlc";
    if [ "$?" = "1" ]; then
        run-standalone.sh ./vlc -vvv
    fi

    ask_to_do "$target: run-standalone_stream(noaudio)";
    if [ "$?" = "1" ]; then
        run-standalone.sh ./vlc --intf rc --vout sdl --noaudio
            http://192.168.1.5:31339/0,405,a5,64
    fi

    ask_to_do "$target: run-standalone_stream";
    if [ "$?" = "1" ]; then
        run-standalone.sh ./vlc --intf rc --vout sdl
            http://192.168.1.5:31339/0,405,a5,64
    fi

    ask_to_do "$target: af-sb-init_start";
    if [ "$?" = "1" ]; then
        af-sb-init.sh start
    fi
}
```

A.1.14 Secció principal del guió

Aquesta és la secció principal del guió on s'orquestra tot el procés.

```
# Preparing
makedir "packages"
makedir "src"
```

```
makedir "dist"
makedir "dist/modules"

# Download sources
cd packages
downpackage vlc-snapshot-20060504.tar.gz
    http://nightlies.videolan.org/build/source/vlc-snapshot-20060504.tar.gz
downpackage libdvbpsi4-0.1.5.tar.bz2
    http://download.videolan.org/pub/libdvbpsi/0.1.5/libdvbpsi4-0.1.5.tar.bz2
downpackage mpeg2dec-0.4.0b.tar.gz
    http://libmpeg2.sourceforge.net/files/mpeg2dec-0.4.0b.tar.gz
downpackage libmad-0.15.1b.tar.gz
    ftp://ftp.mars.org/pub/mpeg/libmad-0.15.1b.tar.gz
cd ..

# Uncompressing sources
cd src
uncompress_gzip vlc-snapshot-20060504.tar.gz vlc-0.8.6-svn
uncompress_bzip2 libdvbpsi4-0.1.5.tar.bz2 libdvbpsi4-0.1.5
uncompress_gzip mpeg2dec-0.4.0b.tar.gz mpeg2dec-0.4.0
uncompress_gzip libmad-0.15.1b.tar.gz libmad-0.15.1b
cd ..

# Patch /usr/include directory for X11
patch_X11_include_dir

# Set CFLAGS
export CFLAGS="-march=armv5te"

# Move to source dir
cd src

# Build
build_libmad
build_libmpeg2
build_libdvbpsi
build_vlc

cp vlc-0.8.6-svn/vlc ../dist
find vlc-0.8.6-svn/modules/ -name *.so -exec cp -f '{}' ../dist/modules \;
# Move to root
cd ..

# Test
cd dist
test_vlc
cd ..
```

Apèndix B

Canvis en el Video LAN Client

B.1 Modificacions en el fitxer configure.ac

En aquest guió per a les eines autoconf/automake s'ha afegit el codi necessari per afegir la definició del preprocessador **HILDON**, aquesta funcionalitat s'encarna en el paràmetre –enable-hildon del configure.sh.

També s'altera la compilació per tal que les llibreries que son necessàries en l'entorn maemo siguin afegides al procés de compilació.

```
dnl
dnl  HILDON-PDA module
dnl
AC_ARG_ENABLE(hildon,
  [  --enable-hildon          HILDONize PDA interface, needs PDA and Gtk2 support
  (default disabled)])
if test "${enable_hildon}" = "yes"
then
  VLC_ADD_CFLAGS([pda],[`pkg-config hildon-libs --cflags` `pkg-config libosso --cflags`
    `pkg-config dbus-1 --cflags`])
  VLC_ADD_CFLAGS([pda],[-DHILDON])
  VLC_ADD_LDFLAGS([pda],[`pkg-config hildon-libs --libs` `pkg-config libosso --libs`
    `pkg-config dbus-1 --libs`])
fi
```

B.2 Modificacions en la interfície d'usuari

En aquest apartat es descriuen el canvis que s'han introduït al codi font per tal d'adequar la interfície d'usuari a les característiques del Nokia 770.

Els canvis afecten al fitxer pda-interface.c que es troba en la carpeta modules/gui/pda del codi font del Video LAN Client. Aquest mòdul defineix una única funció create_pda que encarna el constructor de l'objecte.

B.2.1 Fitxers de capçalera

Per tal de portar una interfície GTK+ a Hildon, cal primer de tot incloure diversos fitxers de capçalera. També afegirem les definicions de MAEMO_PACKAGE i MAEMO_VERSION que utilitzarem per registrar l'aplicació.

```
#ifdef HILDON
#    include <hildon-widgets/hildon-app.h>
#    include <hildon-widgets/hildon-appview.h>
#    include <libosso.h>
#    define MAEMO_PACKAGE "maemoVLC"
#    define MAEMO_VERSION "0.1"
#endif

#include <gdk/gdkkeysyms.h>
#include <gtk/gtk.h>
```

B.2.2 Declaració de variables auxiliars

Declaració de diverses variables punter auxiliars per referir objectes del framework d'aplicacions Hildon.

```
#ifdef HILDON
    HildonApp           *p_hildon_app = NULL;
    HildonAppView       *p_hildon_app_view = NULL;
    osso_context_t      *p_osso_context = NULL;
#endif
```

B.2.3 Instanciació de l'aplicació

Creació de l'objecte aplicació hildon i inicialització dels serveis OSSO.

```
#ifdef HILDON
  p_hildon_app = HILDON_APP( hildon_app_new() );
  hildon_app_set_title( p_hildon_app, _("maemoVLC") );
  hildon_app_set_two_part_title ( p_hildon_app, TRUE );

  /* Initialize maemo application */
  p_osso_context = osso_initialize( MAEMO_PACKAGE, MAEMO_VERSION, TRUE, NULL);

  /* Check that initialization was ok */
  if (p_osso_context == NULL)
    {
      g_print( "osso_initialize_failed" );
    }
  g_assert(p_osso_context);

  pda = GTK_WIDGET( p_hildon_app );
#else
  pda = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_widget_set_size_request (pda, 240, 320);
  gtk_window_set_title (GTK_WINDOW (pda), _("VLC media player"));

  pda_icon_pixbuf = create_pixbuf ("vlc16x16.png");
  if (pda_icon_pixbuf)
    {
      gtk_window_set_icon (GTK_WINDOW (pda), pda_icon_pixbuf);
      gdk_pixbuf_unref (pda_icon_pixbuf);
    }
#endif
```

B.2.4 Instanciació del viewbox principal

```
#ifdef HILDON
   gtk_container_set_border_width(GTK_CONTAINER(vbox), 0);
   p_hildon_app_view = HILDON_APPVIEW(hildon_appview_new( _("maemoVLC") ));
   hildon_appview_set_fullscreen_key_allowed( p_hildon_app_view, TRUE );
#else
   gtk_container_add (GTK_CONTAINER (pda), vbox);
   gtk_widget_show (vbox);
#endif
```

B.2.5 Empaquetat de la barra d'eines

```
#ifdef HILDON
   gtk_box_pack_end (GTK_BOX (p_hildon_app_view->vbox), toolbar, TRUE, TRUE, 0);
#else
   gtk_box_pack_start (GTK_BOX (vbox), toolbar, FALSE, FALSE, 5);
   gtk_widget_set_size_request (toolbar, 240, 22);
#endif
```

B.2.6 Redimensionat de diversos widgets

```
#ifdef HILDON
   gtk_widget_set_size_request (hseparator15, 0, 10);
#else
   gtk_widget_set_size_request (hseparator15, -2, 10);
#endif
```

```
#ifdef HILDON
   gtk_widget_set_size_request (labelDescription, 650, 150);
#else
   gtk_widget_set_size_request (labelDescription, 300, 112);
#endif
```

```
#ifdef HILDON
   gtk_widget_set_size_request (labelAuthors, 650, 80);
#else
   gtk_widget_set_size_request (labelAuthors, 208, 32);
#endif
```

```
#ifdef HILDON
   gtk_widget_set_size_request (labelCopyright, 650, 32);
#else
   gtk_widget_set_size_request (labelCopyright, 208, 16);
#endif
```

```
#ifdef HILDON
   gtk_widget_set_size_request (labelProgramName, 650, 32);
#else
   gtk_widget_set_size_request (labelProgramName, 152, 16);
#endif
```

B.2.7 Vinclatge de la vista amb l'aplicació

```
#ifdef HILDON
   gtk_container_add( GTK_CONTAINER(p_hildon_app_view), vbox);
   hildon_app_set_appview( p_hildon_app, p_hildon_app_view );
   gtk_widget_show_all( GTK_WIDGET(p_hildon_app) );
#endif
```

B.3 Modificacions en el mòdul de renderitzat SDL

En aquest apartat es descriuen el canvis que s'han introduït al codi font per tal de reduir la complexitat i l'us de memòria d'aquesta secció crítica.

Els canvis afecten al fitxer sdl.c que es troba en la carpeta modules/video_output del codi font del Video LAN Client.

B.3.1 Declaració de funcions

```
static int SDL_DisplayYUVOverlayAlter( SDL_Overlay *overlay, SDL_Rect *dstrect);
static void Display1X( int *colortab, Uint32 *rgb_2_pix,
                                    unsigned char *lum, unsigned char *cr,
                                    unsigned char *cb, unsigned char *out,
                                    int rows, int cols, int mod );
```

B.3.2 Canvis en la funció Display()

Els canvis en aquesta funció estan dirigits a utilitzar la funcionalitat sobre l'overlay YUV alternativa a la proporcionada per SDL.

```
/*****************************************************************************
 * Display: displays previously rendered output
 *****************************************************************************
 * This function sends the currently rendered image to the display.
 *****************************************************************************/
static void Display( vout_thread_t *p_vout, picture_t *p_pic )
{
    SDL_Rect disp;
    double elapsed;
    frame_count++;

    if ((frame_count%200) == 0) {
        time(&current);
        elapsed = (double) difftime(current,start);
        printf("VLC: Display Frames:%i Elapsed:%f FPS:%f \n", \
            frame_count, \
            elapsed, \
            (double)frame_count / elapsed);
    }

    disp.x = 0;
    disp.y = 0;
    disp.w = p_vout->p_sys->i_width;
    disp.h = p_vout->p_sys->i_height;

    if( p_vout->p_sys->p_overlay == NULL )
    {
        /* RGB picture */
        SDL_Flip( p_vout->p_sys->p_display );
    }
    else
    {
        /* Overlay picture */
        SDL_UnlockYUVOverlay( p_pic->p_sys->p_overlay);
//        SDL_DisplayYUVOverlay( p_pic->p_sys->p_overlay , &disp );
        SDL_DisplayYUVOverlayAlter( p_pic->p_sys->p_overlay , &disp );
        SDL_LockYUVOverlay( p_pic->p_sys->p_overlay);
    }
}
```

B.3.3 Inclusió de l'estructura private_yuvhwdata

Aquesta estructura s'ha extret de la llibreria SDL i s'ha traslladat al mòdul per tal de poder accedir als diversos punters i poder reimplementar la funcionalitat sobre l'overlay YUV.

```
struct private_yuvhwdata {
        SDL_Surface *stretch;
        SDL_Surface *display;
        Uint8 *pixels;
        int *colortab;
        Uint32 *rgb_2_pix;
        void (*Display1X)(int *colortab, Uint32 *rgb_2_pix,
                          unsigned char *lum, unsigned char *cr,
                          unsigned char *cb, unsigned char *out,
                          int rows, int cols, int mod );
        void (*Display2X)(int *colortab, Uint32 *rgb_2_pix,
                          unsigned char *lum, unsigned char *cr,
                          unsigned char *cb, unsigned char *out,
                          int rows, int cols, int mod );

        /* These are just so we don't have to allocate them separately */
        Uint16 pitches[3];
        Uint8 *planes[3];
};
```

B.3.4 Canvis a la funció SDL_DisplayYUVOverlayAlter()

Aquesta funció s'ha extret de la llibreria SDL i s'ha modificat per tal de escalar els quadres i fer la conversió de color en un sol pas.

```
static int SDL_DisplayYUVOverlayAlter( SDL_Overlay *overlay, SDL_Rect *dstrect)
{
        SDL_Surface *display;
        Uint8 *lum, *Cr, *Cb;
        Uint8 *dst;
        int mod;
        SDL_Rect disp;

        disp.x = 0;
        disp.y = 0;
        disp.w = dstrect->w;
        disp.h = dstrect->h / 2;

        display = overlay->hwdata->display;

        lum = overlay->pixels[0];
        Cr =  overlay->pixels[1];
        Cb =  overlay->pixels[2];

        if ( SDL_MUSTLOCK(display) ) {
                if ( SDL_LockSurface(display) < 0 ) {
                        return(-1);
                }
        }

        dst = (Uint8 *)display->pixels
                + dstrect->x * display->format->BytesPerPixel
                + dstrect->y * display->pitch;

        mod = (display->pitch / display->format->BytesPerPixel);

        mod -= overlay->w;
        Display1X(overlay->hwdata->colortab, overlay->hwdata->rgb_2_pix,
                          lum, Cr, Cb, dst, overlay->h, overlay->w,mod);

        if ( SDL_MUSTLOCK(display) ) {
                SDL_UnlockSurface(display);
        }

        SDL_UpdateRects(display, 1, &disp);

        return(0);
}
```

B.3.5 Canvis en la funció Display1X()

Aquesta funció s'ha extret de la llibreria SDL i s'ha modificat per tal de escalar els quadres i fer la conversió de color en un sol pas.

```
static void Display1X( int *colortab, Uint32 *rgb_2_pix,
                                    unsigned char *lum, unsigned char *cr,
                                    unsigned char *cb, unsigned char *out,
                                    int rows, int cols, int mod )
{
    unsigned short* row1;

    int x, y;
    int cr_r;
    int crb_g;
    int cb_b;
    int cols_2 = cols / 2;

    row1 = (unsigned short*) out;

    y = rows / 2;
    while( y-- )
    {
        x = cols_2;
        while( x-- )
        {
            register int L;
            register unsigned int c;

            L = *lum;
            cr_r   = 0*768+256 + colortab[ *cr + 0*256 ];
            crb_g  = 1*768+256 + colortab[ *cr + 1*256 ]
                               + colortab[ *cb + 2*256 ];
            cb_b   = 2*768+256 + colortab[ *cb + 3*256 ];
            ++cr; ++cb;

            c = (rgb_2_pix[ L + cr_r ] |
                        rgb_2_pix[ L + crb_g ] |
                        rgb_2_pix[ L + cb_b ]);
            *row1++ = c;
            *row1++ = c;
            lum+=2;
        }

        /*
         * These values are at the start of the next line, (due
         * to the ++'s above), but they need to be at the start
         * of the line after that.
         */
        lum  += cols;
        row1 += mod;
    }
}
```

Apèndix C

Canvis en la llibreria libmpeg2

C.1 Canvis en la carpeta ./

C.1.1 Modificacions en el fitxer configure.in

En aquest guió per a les eines autoconf/automake s'ha afegit el codi necessari per afegir l'arquitectura ARM.

```
    dnl arch-specific flags
    case "$host" in
    i?86-* | k?-* | x86_64-*)
        AC_DEFINE([ARCH_X86],,[x86 architecture])
        case "$host" in
        i386-*) TRY_CFLAGS="$OPT_CFLAGS -mcpu=i386";;
        i486-*) TRY_CFLAGS="$OPT_CFLAGS -mcpu=i486";;
        i586-*) TRY_CFLAGS="$OPT_CFLAGS -mcpu=pentium";;
        i686-*) TRY_CFLAGS="$OPT_CFLAGS -mcpu=pentiumpro";;
        k6-*)   TRY_CFLAGS="$OPT_CFLAGS -mcpu=k6";;
        esac
        AC_TRY_CFLAGS([$TRY_CFLAGS $CFLAGS],[OPT_CFLAGS="$TRY_CFLAGS"]);;
[...]
    arm*)
        AC_DEFINE([ARCH_ARM],,[arm architecture]);;
    esac
```

C.2 Canvis en la carpeta ./include

C.2.1 Mòdul n770.h

Aquesta capçalera proporciona part de la funcionalitat EDSP del nucli ARM per mitjà de definicicions del preprocessador i assemblador en linea.

```
/*
 * n770.h
 * Copyright (C) 2006-2007 Josep Torra <jtorra@uoc.edu>
 *
 * This file is part of mpeg2dec, a free MPEG-2 video stream decoder.
 * See http://libmpeg2.sourceforge.net/ for updates.
 *
 * mpeg2dec is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
```

```
 *
 * mpeg2dec is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

/* signed 16x16 -> 32 multiply and accumulate */
#define MAC16bb(rt, ra, rb, rc) __asm__ __volatile__ ("smlabb\t%0,%1,%2,%3;\t@MAC\n" \
                                                      : "=&r"(rt) \
                                                      : "%r"(ra),"r"(rb),"r"(rc))

/* signed 16x16 -> 32 multiply */
#define MUL16bb(rt, ra, rb) __asm__ __volatile__ ("smulbb\t%0,%1,%2;\t@MUL\n" \
                                                      : "=&r"(rt) \
                                                      : "%r"(ra),"r"(rb))

/* signed 16x16 -> 32 multiply and accumulate */
#define MAC16tt(rt, ra, rb, rc) __asm__ __volatile__ ("smlatt\t%0,%1,%2,%3;\t@MAC\n" \
                                                      : "=&r"(rt) \
                                                      : "%r"(ra),"r"(rb),"r"(rc))

/* signed 16x16 -> 32 multiply */
#define MUL16tt(rt, ra, rb) __asm__ __volatile__ ("smultt\t%0,%1,%2;\t@MUL\n" \
                                                      : "=&r"(rt) \
                                                      : "%r"(ra),"r"(rb))

/* signed 16x16 -> 32 multiply and accumulate */
#define MAC16tb(rt, ra, rb, rc) __asm__ __volatile__ ("smlatb\t%0,%1,%2,%3;\t@MACt\n" \
                                                      : "=&r"(rt) \
                                                      : "%r"(ra),"r"(rb),"r"(rc))

/* signed 16x16 -> 32 multiply */
#define MUL16tb(rt, ra, rb) __asm__ __volatile__ ("smultb\t%0,%1,%2;\t@MULt\n" \
                                                      : "=&r"(rt) \
                                                      : "%r"(ra),"r"(rb))


/* signed 16x16 -> 32 multiply and accumulate */
#define MAC16bt(rt, ra, rb, rc) __asm__ __volatile__ ("smlabt\t%0,%1,%2,%3;\t@MACt\n" \
                                                      : "=&r"(rt) \
                                                      : "%r"(ra),"r"(rb),"r"(rc))

/* signed 16x16 -> 32 multiply */
#define MUL16bt(rt, ra, rb) __asm__ __volatile__ ("smulbt\t%0,%1,%2;\t@MULt\n" \
                                                      : "=&r"(rt) \
                                                      : "%r"(ra),"r"(rb))

#define BLOCK_CLEAR(block) __asm__ __volatile__ ( \
        "mov r1, #0\n" \
        "mov r2, #0\n" \
        "mov r3, #0\n" \
        "mov r4, #0\n" \
        "mov r5, #0\n" \
        "mov r6, #0\n" \
        "mov r7, #0\n" \
        "mov r8, #0\n" \
        "mov r9, #0\n" \
        "mov r10, #0\n" \
        "mov r11, #0\n" \
        "mov r12, #0\n" \
        "stmia %[dst]!, {r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12}\n" \
        "stmia %[dst]!, {r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12}\n" \
        "stmia %[dst]!, {r1, r2, r3, r4, r5, r6, r7, r8}\n" \
        : [dst] "+&r" ((int32_t *)block) \
        : \
        : "cc", "memory")
```

C.3 Canvis en la carpeta ./libmpeg2

C.3.1 Modificacions en el fitxer Makefile.am

En aquest guió per a les eines autoconf/automake s'hi ha afegit idct_n770.c per incloure'l en la compilació de la llibreria.

```
libmpeg2arch_la_SOURCES = motion_comp_mmx.c idct_mmx.c \
                          motion_comp_altivec.c idct_altivec.c \
                          motion_comp_alpha.c idct_alpha.c \
                          motion_comp_vis.c \
                          idct_n770.c \
                          cpu_accel.c cpu_state.c
```

C.3.2 Modificacions en el fitxer mpeg2_internal.h

Afegeix la declaració de les funcions que implementen la versió del IDCT per l'arquitectura ARM del Nokia 770.

```
/* idct_n770.c */
void mpeg2_idct_copy_n770 (int16_t * block, uint8_t * dest, int stride);
void mpeg2_idct_add_n770 (int last, int16_t * block,
                          uint8_t * dest, int stride);
void mpeg2_idct_n770_init (void);
```

C.3.3 Modificacions en el fitxer idct.c

Canvis per incorporar la nova arquitectura ARM i la implementació de l'IDCT per el Nokia 770.

```
void mpeg2_idct_init (uint32_t accel)
{
#ifdef ARCH_X86
    if (accel & MPEG2_ACCEL_X86_MMXEXT) {
        mpeg2_idct_copy = mpeg2_idct_copy_mmxext;
        mpeg2_idct_add = mpeg2_idct_add_mmxext;
        mpeg2_idct_mmx_init ();
    } else if (accel & MPEG2_ACCEL_X86_MMX) {
        mpeg2_idct_copy = mpeg2_idct_copy_mmx;
        mpeg2_idct_add = mpeg2_idct_add_mmx;
        mpeg2_idct_mmx_init ();
    } else
#endif
#ifdef ARCH_PPC
    if (accel & MPEG2_ACCEL_PPC_ALTIVEC) {
        mpeg2_idct_copy = mpeg2_idct_copy_altivec;
        mpeg2_idct_add = mpeg2_idct_add_altivec;
        mpeg2_idct_altivec_init ();
    } else
#endif
#ifdef ARCH_ALPHA
    if (accel & MPEG2_ACCEL_ALPHA_MVI) {
        mpeg2_idct_copy = mpeg2_idct_copy_mvi;
        mpeg2_idct_add = mpeg2_idct_add_mvi;
        mpeg2_idct_alpha_init ();
    } else if (accel & MPEG2_ACCEL_ALPHA) {
        int i;

        mpeg2_idct_copy = mpeg2_idct_copy_alpha;
        mpeg2_idct_add = mpeg2_idct_add_alpha;
        mpeg2_idct_alpha_init ();
        for (i = -3840; i < 3840 + 256; i++)
            CLIP(i) = (i < 0) ? 0 : ((i > 255) ? 255 : i);
    } else
#endif
#ifdef ARCH_ARM
```

```
    if (1) {
        mpeg2_idct_copy = mpeg2_idct_copy_n770;
        mpeg2_idct_add = mpeg2_idct_add_n770;
        mpeg2_idct_n770_init ();
    } else
#endif
    {
        extern uint8_t mpeg2_scan_norm[64];
        extern uint8_t mpeg2_scan_alt[64];
        int i, j;

        mpeg2_idct_copy = mpeg2_idct_copy_c;
        mpeg2_idct_add = mpeg2_idct_add_c;
        for (i = -3840; i < 3840 + 256; i++)
            CLIP(i) = (i < 0) ? 0 : ((i > 255) ? 255 : i);
        for (i = 0; i < 64; i++) {
            j = mpeg2_scan_norm[i];
            mpeg2_scan_norm[i] = ((j & 0x36) >> 1) | ((j & 0x09) << 2);
            j = mpeg2_scan_alt[i];
            mpeg2_scan_alt[i] = ((j & 0x36) >> 1) | ((j & 0x09) << 2);
        }
    }
}
```

C.3.4 Mòdul idct_n770.c

Aquesta és la reimplementació de la versió genèrica de l'IDCT amb l'optimització per el Nokia 770.

```
/*
 * idct_n770.c
 * Copyright (C) 2006-2007 Josep Torra <jtorra@uoc.edu>
 *
 * This file is part of mpeg2dec, a free MPEG-2 video stream decoder.
 * See http://libmpeg2.sourceforge.net/ for updates.
 *
 * mpeg2dec is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * mpeg2dec is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

#include <stdio.h>
#include "config.h"

#ifdef ARCH_ARM

#include <stdlib.h>
#include <inttypes.h>

#include "n770.h"

#include "mpeg2.h"
#include "attributes.h"
#include "mpeg2_internal.h"


#define W1 2841 /* 2048 * sqrt (2) * cos (1 * pi / 16) */
#define W2 2676 /* 2048 * sqrt (2) * cos (2 * pi / 16) */
#define W3 2408 /* 2048 * sqrt (2) * cos (3 * pi / 16) */
#define W5 1609 /* 2048 * sqrt (2) * cos (5 * pi / 16) */
#define W6 1108 /* 2048 * sqrt (2) * cos (6 * pi / 16) */
#define W7 565  /* 2048 * sqrt (2) * cos (7 * pi / 16) */

#define subW2W6 1568
#define addW2W6 3784

#define ROW_SHIFT 12
```

```
#define COL_SHIFT 17

int32_t W[] = { (W7<<16) | W3, (-W1<<16) | W1, (-W2<<16) | W2, (-W3<<16) | W3, 0,
    (-W5<<16) | W5, (-W6<<16) | W6, (-W7<<16) | W7 };

/*
 * In legal streams, the IDCT output should be between -384 and +384.
 * In corrupted streams, it is possible to force the IDCT output to go
 * to +-3826 - this is the worst case for a column IDCT where the
 * column inputs are 16-bit values.
 */
extern uint8_t mpeg2_clip[3840 * 2 + 256];
#define CLIP(i) ((mpeg2_clip + 3840)[i])

static inline void idct_row (int16_t * const block)
{
    uint32_t const *dblock = (uint32_t*)block;
    uint32_t dd0, dd1, dd2, dd3;
    int a0, a1, a2, a3, b0, b1, b2, b3;
    int t0, t1, t2, t3;
    int x, w0, w1;

    asm volatile ("@ idct_row read_row\n");
    dd0 = dblock[0];
    dd1 = dblock[1];
    dd2 = dblock[2];
    dd3 = dblock[3];

    asm volatile ("@ idct_row begin\n");
    /* shortcut */
    asm volatile ("@ idct_row(shortcut) begin\n");
    x = ( dd1 | dd2 | dd3 ) ;
    if ( !(x | dd0 ) ){
        asm volatile ("@ zero_row\n");
        return;
    }

    if ( !(x | block[1]) ) {
        asm volatile ("@ only_first_word_on_row\n");
        uint32_t tmp = (uint16_t) (block[0] >> 1);
        tmp |= tmp << 16;
        ((uint32_t*)block)[0]=((uint32_t*)block)[1] =
            ((uint32_t*)block)[2]=((uint32_t*)block)[3] = tmp;
        return;
    }
    asm volatile ("@ idct_row(shortcut) end\n");

    asm volatile ("@ idct_row a0 = a1 = a2 = a3 = 2048 \n");
    a0 = a1 = a2 = a3 = 2048;

    asm volatile ("@ idct_row test dd0 | dd1 != 0 \n");
    x = ( dd0 | dd1 );
    if ( x )
    {
        x = block[0] << 11;
        a0 = a1 += x;
        x = block[2] << 11;
        a3 = a0 += x;
        a2 = a1 -= x;

        asm volatile ("@ idct_row w0 = W6 & w1 = W2 \n");
        w0 = W[6];
        w1 = W[2];
        // BUTTERFLY (t2, t3, W6, W2, d3, d1);
        MUL16bt(x , w0 , dd1);          // W2 * block[3]
        MAC16bt(x , w1 , dd0 , x);      // W6 * block[1]
        a0 = a0 + x;
        a3 = a3 - x;

        MUL16bt(x , w0 , dd0);          // W6 * block[1]
        MAC16tt(x , w1 , dd1 , x);      // -W2 * block[3]
        a1 = a1 + x;
        a2 = a2 - x;
    }

    asm volatile ("@ idct_row w0 = W7W3 & w1 = W1 \n");
    w0 = W[0]; // W7 and W3
    w1 = W[1];
    // BUTTERFLY (t0, t1, W7, W1, d3, d0);
    MUL16tt(t0 , w0 , dd3);             // W7 * block[7]
    MAC16bb(t0 , w1 , dd2 , t0);        // W1 * block[4]

    MUL16tb(t1 , w0 , dd2);             // W7 * block[4]
```

```
    MAC16tt(t1 , w1 , dd3 , t1);          // -W1 * block[7]

    asm volatile ("@ idct_row w0 = W7W3 & w1 = W5 \n");
    w1 = W[5];
    // BUTTERFLY (t2, t3, W3, W5, d1, d2);
    MUL16bt(t2 , w0 , dd2);               // W3 * block[5]
    MAC16bb(t2 , w1 , dd3 , t2);          // W5 * block[6]

    MUL16bb(t3 , w0 , dd3);               // W3 * block[6]
    MAC16tt(t3 , w1 , dd2 , t3);          // -W5 * block[5]

    b0 = t0 + t2;
    b3 = t1 + t3;
    t0 -= t2;
    t1 -= t3;
    asm volatile ("@ idct_row b1 = ... & b2 = ... \n");
    b1 = ((t0 + t1) >> 8) * 181;
    b2 = ((t0 - t1) >> 8) * 181;

    asm volatile ("@ idct_row update block \n");
    block[0] = (a0 + b0) >> ROW_SHIFT;
    block[7] = (a0 - b0) >> ROW_SHIFT;
    block[1] = (a1 + b1) >> ROW_SHIFT;
    block[6] = (a1 - b1) >> ROW_SHIFT;
    block[2] = (a2 + b2) >> ROW_SHIFT;
    block[5] = (a2 - b2) >> ROW_SHIFT;
    block[3] = (a3 + b3) >> ROW_SHIFT;
    block[4] = (a3 - b3) >> ROW_SHIFT;

    asm volatile ("@ idct_row end\n");
}


static inline void idct_col_put (int16_t * const block, uint8_t * dest, const int stride)
{
    int d0, d1, d2, d3;
    int a0, a1, a2, a3, b0, b1, b2, b3;
    int t0, t1, t2, t3;
    int w0, w1;

    d0 = (block[8*0] << 11) + 65536;
    d1 = block[8*1];
    d2 = block[8*2] << 11;
    d3 = block[8*3];

    t0 = d0 + d2;
    t1 = d0 - d2;

    asm volatile ("@ idct_col_put w0 = W6 & w1 = W2 \n");
    w0 = W[6];
    w1 = W[2];
    // BUTTERFLY (t2, t3, W6, W2, d3, d1);
    MUL16bb(t2 , w0 , d3);                // W6
    MAC16bb(t2 , w1 , d1 , t2);           // W2

    MUL16bb(t3 , w0 , d1);                // W6
    MAC16tb(t3 , w1 , d3 , t3);           // -W2

    a0 = t0 + t2;
    a1 = t1 + t3;
    a2 = t1 - t3;
    a3 = t0 - t2;

    d0 = block[8*4];
    d1 = block[8*5];
    d2 = block[8*6];
    d3 = block[8*7];

    asm volatile ("@ idct_col_put w0 = W7W3 & w1 = W1 \n");
    w0 = W[0]; // W7 and W3
    w1 = W[1];
    // BUTTERFLY (t0, t1, W7, W1, d3, d0);
    MUL16tb(t0 , w0 , d3);                // W7
    MAC16bb(t0 , w1 , d0 , t0);           // W1

    MUL16tb(t1 , w0 , d0);                // W7
    MAC16tb(t1 , w1 , d3 , t1);           // -W1

    asm volatile ("@ idct_col_put w0 = W7W3 & w1 = W5 \n");
    w1 = W[5];
    // BUTTERFLY (t2, t3, W3, W5, d1, d2);
    MUL16bb(t2 , w0 , d1);                // W3
    MAC16bb(t2 , w1 , d2 , t2);           // W5
```

```
    MUL16bb(t3 , w0 , d2);              // W3
    MAC16tb(t3 , w1 , d1 , t3);         // -W5

    b0 = t0 + t2;
    b3 = t1 + t3;
    t0 -= t2;
    t1 -= t3;

    b1 = ((t0 + t1) >> 8) * 181;
    b2 = ((t0 - t1) >> 8) * 181;

    dest[0] = CLIP( (a0 + b0) >> COL_SHIFT );
    dest+=stride;
    dest[0] = CLIP( (a1 + b1) >> COL_SHIFT );
    dest+=stride;
    dest[0] = CLIP( (a2 + b2) >> COL_SHIFT );
    dest+=stride;
    dest[0] = CLIP( (a3 + b3) >> COL_SHIFT );
    dest+=stride;
    dest[0] = CLIP( (a3 - b3) >> COL_SHIFT );
    dest+=stride;
    dest[0] = CLIP( (a2 - b2) >> COL_SHIFT );
    dest+=stride;
    dest[0] = CLIP( (a1 - b1) >> COL_SHIFT );
    dest+=stride;
    dest[0] = CLIP( (a0 - b0) >> COL_SHIFT );

    block[8*0] = block[8*1] = block[8*2] = block[8*3] = 0;
    block[8*4] = block[8*5] = block[8*6] = block[8*7] = 0;
}


static inline void idct_col_add (int16_t * const block, uint8_t * dest, const int stride)
{
    int d0, d1, d2, d3;
    int a0, a1, a2, a3, b0, b1, b2, b3;
    int t0, t1, t2, t3;
    int w0, w1;

    d0 = (block[8*0] << 11) + 65536;
    d1 = block[8*1];
    d2 = block[8*2] << 11;
    d3 = block[8*3];

    t0 = d0 + d2;
    t1 = d0 - d2;

    asm volatile ("@ idct_col_add w0 = W6 & w1 = W2 \n");
    w0 = W[6];
    w1 = W[2];
    // BUTTERFLY (t2, t3, W6, W2, d3, d1);
    MUL16bb(t2 , w0 , d3);              // W6
    MAC16bb(t2 , w1 , d1 , t2);         // W2

    MUL16bb(t3 , w0 , d1);              // W6
    MAC16tb(t3 , w1 , d3 , t3);         // -W2

    a0 = t0 + t2;
    a1 = t1 + t3;
    a2 = t1 - t3;
    a3 = t0 - t2;

    d0 = block[8*4];
    d1 = block[8*5];
    d2 = block[8*6];
    d3 = block[8*7];

    asm volatile ("@ idct_col_add w0 = W7W3 & w1 = W1 \n");
    w0 = W[0]; // W7 and W3
    w1 = W[1];
    // BUTTERFLY (t0, t1, W7, W1, d3, d0);
    MUL16tb(t0 , w0 , d3);              // W7
    MAC16bb(t0 , w1 , d0 , t0);         // W1

    MUL16tb(t1 , w0 , d0);              // W7
    MAC16tb(t1 , w1 , d3 , t1);         // -W1

    asm volatile ("@ idct_col_add w0 = W7W3 & w1 = W5 \n");
    w1 = W[5];
    // BUTTERFLY (t2, t3, W3, W5, d1, d2);
    MUL16bb(t2 , w0 , d1);              // W3
    MAC16bb(t2 , w1 , d2 , t2);         // W5
```

```
    MUL16bb(t3 , w0 , d2);                  // W3
    MAC16tb(t3 , w1 , d1 , t3);             // -W5

    b0 = t0 + t2;
    b3 = t1 + t3;
    t0 -= t2;
    t1 -= t3;

    b1 = ((t0 + t1) >> 8) * 181;
    b2 = ((t0 - t1) >> 8) * 181;

    dest[0] = CLIP( ((a0 + b0) >> COL_SHIFT) + dest[0] );
    dest+=stride;
    dest[0] = CLIP( ((a1 + b1) >> COL_SHIFT) + dest[0] );
    dest+=stride;
    dest[0] = CLIP( ((a2 + b2) >> COL_SHIFT) + dest[0] );
    dest+=stride;
    dest[0] = CLIP( ((a3 + b3) >> COL_SHIFT) + dest[0] );
    dest+=stride;
    dest[0] = CLIP( ((a3 - b3) >> COL_SHIFT) + dest[0] );
    dest+=stride;
    dest[0] = CLIP( ((a2 - b2) >> COL_SHIFT) + dest[0] );
    dest+=stride;
    dest[0] = CLIP( ((a1 - b1) >> COL_SHIFT) + dest[0] );
    dest+=stride;
    dest[0] = CLIP( ((a0 - b0) >> COL_SHIFT) + dest[0] );

    block[8*0] = block[8*1] = block[8*2] = block[8*3] = 0;
    block[8*4] = block[8*5] = block[8*6] = block[8*7] = 0;
}

void mpeg2_idct_copy_n770 (int16_t * block, uint8_t * dest,
                           const int stride)
{
    int i;

    for (i = 0; i < 8; i++)
        idct_row (block + 8 * i);
    for (i = 0; i < 8; i++)
        idct_col_put (block + i, dest + i, stride);

}

void mpeg2_idct_add_n770 (const int last, int16_t * block,
                          uint8_t * dest, const int stride)
{
    int i;

    if (last != 129 || (block[0] & (7 << 4)) == (4 << 4)) {
        for (i = 0; i < 8; i++)
            idct_row (block + 8 * i);

        for (i = 0; i < 8; i++)
            idct_col_add (block + i, dest + i, stride);

    } else {
        int DC;

        DC = (block[0] + 64) >> 7;
        block[0] = block[63] = 0;
        i = 8;
        do {
            dest[0] = CLIP (DC + dest[0]);
            dest[1] = CLIP (DC + dest[1]);
            dest[2] = CLIP (DC + dest[2]);
            dest[3] = CLIP (DC + dest[3]);
            dest[4] = CLIP (DC + dest[4]);
            dest[5] = CLIP (DC + dest[5]);
            dest[6] = CLIP (DC + dest[6]);
            dest[7] = CLIP (DC + dest[7]);

            dest += stride;
        } while (--i);
    }
}

void mpeg2_idct_n770_init (void)
{
    extern uint8_t mpeg2_scan_norm[64];
    extern uint8_t mpeg2_scan_alt[64];
    int i, j;
```

```
    for (i = -3840; i < 3840 + 256; i++)
        CLIP(i) = (i < 0) ? 0 : ((i > 255) ? 255 : i);
    for (i = 0; i < 64; i++) {
        j = mpeg2_scan_norm[i];
        mpeg2_scan_norm[i] = ((j & 0x36) >> 1) | ((j & 0x09) << 2);
        j = mpeg2_scan_alt[i];
        mpeg2_scan_alt[i] = ((j & 0x36) >> 1) | ((j & 0x09) << 2);
    }
}

#endif /* ARCH_ARM */
```

Bibliografia

[1] Product page of OMAP1710,
http://focus.ti.com/general/docs/wtbu/wtbuproductcontent.tsp?templateId=6123&navigationId=11991&contentId=4670

[2] ARM926EJ-S Product Overview,
http://www.arm.com/pdfs/DVI0035B_926_PO.pdf

[3] ARM926EJ-S Technical Reference Manual,
http://www.arm.com/pdfs/DDI0198D_926_TRM.pdf

[4] ADS 1.2: Assembler Guide,
http://www.arm.com/pdfs/DUI0068B_ADS1_2_Assembler.pdf

[5] ARM DSP-Enhanced Extensions,
http://www.arm.com/pdfs/ARM-DSP.pdf

[6] TMS320C55x DSP Functional Overview,
http://www-s.ti.com/sc/psheets/spru312/spru312.pdf

[7] TMS320C55x Technical Overview,
http://www-s.ti.com/sc/psheets/spru393/spru393.pdf

[8] DSP Gateway homepage,
http://dspgateway.sourceforge.net/pub/index.php

[9] Maemo homepage,
http://www.maemo.org

[10] Maemo SDK tutorial,
http://maemo.org/platform/docs/tutorials/Maemo_tutorial.html

[11] Maemo Multimedia Architecture explained,
http://www.maemo.org/platform/docs/multimedia/multimedia_architecture.html

[12] Maemo Development Platform Roadmap,
http://maemo.org/platform/docs/roadmap.html

[13] Debian Wiki: ArmEabiPort,
http://wiki.debian.org/ArmEabiPort

[14] Scratchbox homepage,
http://www.scratchbox.org

[15] The Hitchhiker's Guide to the N770 Galaxy,
http://n770galaxy.blogspot.com/

[16] Telepathy: IM/VOIP Integration Framework,
http://telepathy.freedesktop.org/wiki/

[17] FarSight,
http://farsight.sourceforge.net/

[18] VideoLAN Project,
http://www.videolan.org/

[19] ARM Cross-Compile HOWTO,
http://developers.videolan.org/vlc/arm-crosscompile.html

[20] libmpeg2 - a free MPEG-2 video stream decoder,
http://libmpeg2.sourceforge.net/

[21] MAD: MPEG Audio Decoder,
http://www.underbit.com/products/mad/

[22] Simple Directmedia Layer,
http://www.libsdl.org/index.php