



UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata **Advanced Robotics Laboratory**

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

**Wireless webcam con
FoxBoard LX416**

Elaborato di esame di:

**Alberto Schena, Nicola
Fraccaroli, Giovanni Dall'Oglio
Risso**

Consegnato il:

22 luglio 2009

Sommario

Il lavoro svolto ha avuto come obiettivo la ricerca di una metodologia di compilazione, installazione e configurazione di software idoneo alla messa in funzione di una scheda FoxBoard con sistema operativo Linux, una webcam ed un adattatore di rete wireless, al fine di realizzare una “wireless webcam” adatta all’installazione a bordo di robot mobili, come ad esempio Speedy e Morgul presenti in ARLBS, Laboratorio di Robotica Avanzata dell’Università degli Studi di Brescia. La visione soggettiva a distanza così ottenuta può fornire supporto alla video-sorveglianza del laboratorio, oppure essere utilizzata in specifiche funzioni quali l’auto-localizzazione dei robot mediante punti cospicui visuali.

1. Introduzione

Il progetto è stato commissionato dal Laboratorio di Robotica Avanzata dell'Università degli Studi di Brescia (ARLBS), in particolare dal docente R. Cassinis, al gruppo di lavoro composto dagli studenti Alberto Schena, Nicola Fraccaroli e Giovanni Dall'Oglio Riso.

L'intento è di dotare un robot mobile di un dispositivo per la ripresa e la trasmissione di immagini ad una postazione fissa. Il dispositivo deve essere leggero, di ridotte dimensioni e non deve richiedere l'utilizzo di cavi tra il robot e la postazione fissa. Pertanto si avrà bisogno di un sistema embedded cui collegare una webcam ed un'antenna wireless.

Un progetto analogo era stato condotto nel settembre 2007 dall'ormai ex-studente Marco Iora, il cui lavoro prevedeva l'utilizzo di un mini-calcolatore a bordo di un robot mobile per due funzioni:

- la trasmissione video, ed
- il controllo remoto del robot mobile via rete wireless grazie all'interfacciamento di una linea seriale, tra robot e scheda, attraverso un socket TCP.

Il mini-calcolatore in questione avrebbe sostituito il laptop installato precedentemente sul corpo dei robot del laboratorio ed utilizzato per il solo controllo remoto, comportando:

- la riduzione del carico a bordo del robot (da oltre 1kg a circa 100g),
- la riduzione dell'ingombro dell'apparecchiatura di controllo,
- il risparmio energetico durante il movimento, grazie al minor carico trasportato, e
- il risparmio energetico grazie al ridotto consumo di potenza da parte del nuovo dispositivo.

Si sarebbero presentati però svantaggi legati a:

- la minor potenza di calcolo disponibile (da circa 2Ghz di frequenza di clock a 100MHz)
- il minore spazio di memorizzazione (da decine di GB a 4MB)
- la minor velocità di comunicazione.

La ricerca di Iora, tuttavia, non ha avuto successo a causa della mancanza in quel periodo di software opportuno che fosse completamente funzionante. Egli ha raccolto i risultati del proprio studio in un elaborato, intitolato "Configurazione FoxLX416 Board" e scaricabile all'indirizzo Internet [1], di cui si raccomanda la lettura (dato che vi si farà spesso riferimento e che verranno riportati aggiornamenti, correzioni ed integrazioni ad alcune sue parti). Tale elaborato ha costituito una preziosa fonte di informazione per la conduzione del presente lavoro, nonché per il superamento di alcune difficoltà.

Rispetto alla sperimentazione condotta da Iora, il progetto assegnato al nostro gruppo di lavoro è stato sgravato dall'utilizzo dell'interfaccia seriale per il controllo remoto del robot mobile, richiedendo solamente la funzione relativa alla trasmissione video.

Con questo elaborato si intende fornire al lettore una descrizione il più possibile dettagliata del procedimento mediante il quale il gruppo di lavoro è riuscito ad ottenere una "wireless webcam" funzionante avendo a disposizione il seguente hardware:

- 1 scheda FoxBoard LX416 dotata di alimentatore da rete,
- 1 webcam USB Philips ToUCam II PCVC840,
- 1 adattatore di rete WiFi USB D-Link DWL-G122 (rev. C1, 54 Mb/s),
- 1 PC dotato di scheda di rete Ethernet 10/100 Mb/s e di un adattatore di rete WiFi, ed
- 1 cavo di rete Ethernet RJ45 cross-over.

I primi tre strumenti sono stati gentilmente concessi da ARLBS al gruppo di lavoro, mentre i rimanenti due erano già in possesso dei membri del gruppo.

FoxBoard LX416 è una scheda madre di piccole dimensioni (66x72mm), prodotta da AcmeSystems (il cui sito ufficiale è [2]), che monta il seguente hardware:

- 1 processore RISC con architettura Axis ETRAX 100LX (32 bit, 100MHz di frequenza di clock), prodotto da Axis Communications, il cui sito ufficiale è [3],
- 4 MB di memoria flash e 16 MB di memoria RAM (da cui la sigla “416”),
- 1 presa di alimentazione da 5V, 280mA (equivalenti a circa 1W di potenza),
- 2 interfacce USB 1.1,
- 1 interfaccia Ethernet (10/100 Mb/s), ed
- 1 adattatore per porta seriale (TTL 3.3V to RS232).

Questa scheda è adattabile a moltissimi utilizzi, che spaziano dal semplice lettore MP3 a veri e propri web server, ed è espandibile grazie a componenti hardware aggiuntivi (quali ad esempio display, espansioni di memoria, antenne GSM/GPRS) messi in commercio da AcmeSystems ed altre ditte.

La scheda FoxBoard è progettata espressamente per ospitare un sistema operativo Linux. Come è noto, il *kernel* di questo sistema operativo offre un elevato grado di adattabilità. Per poter funzionare sulla scheda, il sistema operativo deve essere cross-compilato¹ in un formato binario compatibile con l'architettura del processore.

Sono disponibili in Internet alcuni tool di sviluppo, o SDK², per la configurazione e la cross-compilazione del sistema operativo Linux per FoxBoard. Gli SDK che utilizzeremo consistono in una *toolchain*³ atta a svolgere le seguenti operazioni:

- la configurazione del kernel del sistema operativo Linux e dei suoi moduli;
- la cross-compilazione del software, sia esso il sistema operativo, un modulo aggiuntivo o un driver di periferica, oppure un programma applicativo utente;
- il *flashing* di *firmware* (ovvero la trasmissione via rete dal PC, nel ruolo di boot-server, alla scheda dell'immagine di memoria del sistema operativo).

¹ Cross-compilazione: tecnica mediante la quale si compila un codice sorgente con un cross-compilatore, ottenendo così un file binario eseguibile su un elaboratore (*target*) con architettura diversa da quella della macchina (*host*) su cui si è lanciato il cross-compilatore stesso.

² SDK: Software Development Kit, insieme di strumenti applicativi di supporto all'attività di sviluppo software.

³ Toolchain: insieme di tool di sviluppo usati nella produzione di software secondo uno schema a catena, ovvero tale che l'output di un tool rappresenti l'input per il successivo (ad es. un editor di testo, un compilatore ed un linker formano una toolchain).

2. Il problema affrontato

Date le premesse, per raggiungere l'obiettivo si sono rese necessarie le seguenti attività:

- l'installazione sulla scheda di opportuni driver per l'utilizzo della webcam,
- l'installazione sulla scheda di opportuni driver per l'utilizzo dell'adattatore di rete wireless,
- l'utilizzo di librerie grafiche che consentano rispettivamente alla scheda ed al PC di acquisire e visualizzare le immagini.

A titolo informativo puntualizziamo che in Internet sono attualmente disponibili molti ambienti di sviluppo con supporto alla FoxBoard, ovvero all'architettura Axis ETRAX. Fra essi elenchiamo:

Ambiente	Versione	Link	Data rilascio	Kernel	Maintainer
Axis SDK	2.01	[4]	sett. 2005	2.4.31, 2.6.15	John Crispin
	2.10	[5]	marzo 2007	2.6.19	
	2.20	[6]	febb. 2009	2.6.26	
OpenWRT	7.06	[7]	giugno 2007	2.6.19.2	Team OpenWRT
	8.09.1		giugno 2009	2.6.26	
CrisOs	Alpha	[8]	febb. 2008	2.6.19.8	Claudio Mignanti
	Alpha R2		aprile 2008	2.6.19.8	
	DarwinFox 8.09		sett. 2008	2.6.25.20	
	Aquila		maggio 2009	2.6.29	

Tabella 1 - Ambienti di sviluppo di supporto ad Axis ETRAX

Il progetto OpenWRT consiste in un sistema operativo reperibile sia in forma di tarball dei sorgenti che di fimage general-purpose pre-compilate per utenti che non necessitano di configurazioni particolari del kernel o dei kernel module. Sono disponibili fimage per varie architetture target, fra le quali anche la nostra Axis ETRAX, mentre per quanto riguarda il main project, denominato "Kamikaze" e di cui sono disponibili solamente i sorgenti, non è stata eseguita la cross-compilazione per la verifica del porting verso tale architettura. Per ciascuna versione del sistema OpenWRT sono inoltre disponibili moltissimi pacchetti software aggiuntivi pre-compilati, scaricabili dal repository in formato ".ipk" ed installabili separatamente, ovviamente seguendo a discesa l'albero delle dipendenze.

Il progetto CrisOs, derivante da OpenWRT, consiste in una serie di fimage general-purpose per varie architetture. Come per il progetto OpenWRT, anche per CrisOs sono disponibili molti pacchetti software in formato ".ipk" pre-compilati e pronti all'uso, fra cui è facile identificare i driver adatti al nostro scopo (più precisamente *kmod-usb-pwc_...* e *kmod_rt73...*).

Nel caso in cui il lettore intendesse far uso di OpenWRT o CrisOs, è necessario sia a conoscenza del fatto che tali sistemi utilizzano un client DHCP per l'acquisizione di un indirizzo IP dinamico e pertanto è indispensabile la configurazione di un server DHCP presso il PC di lavoro.

3. La soluzione adottata

Dopo una breve analisi si è deciso di affrontare il problema posto ripetendo essenzialmente i medesimi test effettuati da Iora relativamente alla sola parte di trasmissione video, ottimizzando però due variabili relative al software, ossia:

- variando ed integrando lo schema di configurazione da lui proposto circa il sistema operativo;
- utilizzando, talvolta, versioni aggiornate degli stessi software, nella speranza che queste consentano di superare gli ostacoli da lui incontrati utilizzando quelle oggi obsolete.

Prendendo spunto dalla relazione di Iora si è appurato che il software da scaricare ed utilizzare comprendeva:

- uno degli ambienti di sviluppo a supporto della FoxBoard di cui alla Tabella 1;
- la libreria grafica *VisLib*;
- il driver *pwc* per l'utilizzo della webcam USB, scaricabile da [10];
- il driver *rt73* per l'utilizzo dell'adattatore di rete WiFi USB, scaricabile da [11].

Data la molteplicità di ambienti di sviluppo disponibili, si è scelto di seguire un approccio secondo il quale questi sarebbero stati esaminati nell'ordine illustrato in Tabella 1, fino a trovare quello che avesse consentito il funzionamento di *VisLib*, *pwc* e *rt73*, quindi il raggiungimento dell'obiettivo. Di fatto, il primo SDK analizzato, ovvero Axis SDK 2.01, ha dato esito positivo, quindi l'esperimento è terminato senza test sugli ambienti successivi.

3.1. La libreria grafica *VisLib*

Questo software consiste in una libreria grafica open-source (scritta in C per sistemi Linux, ma esistono versioni anche per Windows) per l'acquisizione, l'analisi e la visualizzazione di immagini. La versione di interesse per noi è sviluppata da MobileRobots e disponibile al sito [9]. Le versioni ufficiali non sono tuttavia adatte al nostro scopo in quanto il frame-grabber⁴ ed il front-end di elaborazione e visualizzazione sono implementati per risiedere sulla stessa macchina ed essere eseguiti in un unico processo, utilizzando strutture dati e buffer condivisi in memoria.

Il gruppo ARLBS ha creato alcune branch della libreria ufficiale, aggiungendo il supporto al funzionamento su sistemi embedded con scarse risorse computazionali (quale è la nostra FoxBoard), la trasmissione via rete del flusso video, la conversione dello spazio YUV in RGB ed altre funzionalità.

La peculiarità di questa versione consiste nello streaming video via socket TCP tra un *grab-server*, che acquisisce e trasmette le immagini, ed un *grab-client*, che le riceve e le visualizza in una finestra. Le due sono entità separate, che possono risiedere su macchine distinte.

➤ **La versione di *VisLib* in questione non è presente sul repository dei software di ARLBS, ma ci è stata fornita da Marco Iora. Poiché è stato necessario apportarne ulteriori modifiche, sarà nostro impegno consegnare il prodotto al docente R. Cassinis, in qualità di responsabile di ARLBS, affinché l'utente possa reperirne i sorgenti presso il sito di ARLBS [15] o altra fonte indicata.**

⁴ Frame-grabber: porzione di software responsabile dell'acquisizione dei frame, a partire dai dati forniti dal driver della videocamera, e la memorizzazione in un buffer di memoria.

Marco Iora ci ha fornito tre versioni distinte della libreria modificata da ARLBS, denominate “1.9.1”, “1.9.1-YUV2RGB”, e “1.9.2-RC1”. Per quanto concerne la soluzione adottata, nel presente studio è stata utilizzata la versione 1.9.2-RC1 che, come le altre due, è compilabile in duplice modalità da un unico makefile generando:

- il grab-server: cross-compilato per architettura target Axis ETRAX come eseguibile stand-alone, modificato per occuparsi esclusivamente di acquisizione video e streaming via socket TCP (su porta 3490);
- il grab-client: compilato per architettura x86 come libreria statica, modificato per ricevere lo streaming video dal socket TCP anziché da un driver *v4l* (ovvero video4Linux⁵).

Il front-end di visualizzazione è un eseguibile per architettura x86 che si aggancia alla libreria grab-client e fa uso delle API X11 del X Window System di Linux per il disegno dei frame.

3.2. Il driver *pwc*

Questo driver è responsabile del funzionamento su sistemi Linux di alcuni modelli di webcam USB, fra cui la nostra Philips TouCam II PCVC840. I sorgenti possono essere scaricati dal sito ufficiale [10]. In questo studio è stata utilizzata la versione 10.0.12-RC1.

Come spiegato nel paragrafo 4.3 dell’elaborato di Iora, nonché nel rapporto tecnico ARL-TR-08-03 edito dal Prof. R. Cassinis e scaricabile da [13], in fase di inizializzazione della webcam vengono eseguite le due istruzioni (che si trovano nel file sorgente `pwc-if.c`):

```
...  
pwc_set_brightness(pdev, 0x7fff);  
pwc_set_agc(pdev, 1, 0);  
...
```

Queste sono responsabili dell’abilitazione del controllo automatico della sensibilità della telecamera (AGC: Automatic Gain Control). Questo implica che:

- le prime acquisizioni generino fotogrammi inizialmente neri, via via più chiari finché il guadagno raggiunge il livello di regime;
- i programmi di controllo manuale del guadagno (ad es. `setpwc`) non producano alcun risultato apprezzabile quando si tenta di impostarne il valore, in quanto esso viene istantaneamente ricalcolato in base alla luminosità dell’immagine acquisita.

La soluzione proposta consiste nel commentare tali righe prima della cross-compilazione del software `pwc`:

```
...  
// pwc_set_brightness(pdev, 0x7fff);  
// pwc_set_agc(pdev, 1, 0);  
...
```

Il gruppo di lavoro ha scelto, per motivi di semplicità, di mantenere attivo l’AGC della webcam, tralasciando la procedura sopra descritta. Il lettore può tuttavia decidere quale approccio sia più indicato per le proprie esigenze.

⁵ video4Linux, ovvero “video for Linux”: strato software di interfacciamento ai driver di periferiche video sotto sistemi Linux.

Si raccomanda inoltre la lettura del rapporto tecnico ARL-TR-08-01, edito dal Prof. R. Cassinis e scaricabile da [12], in merito all'utilizzo del programma `setpwc` per l'impostazione manuale di framerate e guadagno della webcam.

Riteniamo doveroso informare il lettore che Axis SDK 2.01 contiene già i sorgenti di una versione del driver `pwc` (attivabile mediante opportune impostazioni di configurazione del kernel Linux) ma dai test eseguiti si è appurato che questo non funziona correttamente in quanto le immagini acquisite erano completamente grigie e statiche. Per questo motivo è necessario scaricare e compilare una differente versione del driver, più aggiornata e sperabilmente funzionante.

3.3. Il driver `rt73`

Il nome “`rt73`” è storicamente attribuito ad un tipo di chipset, prodotto dalla RalinkTech (chiamata più brevemente Ralink, il cui sito ufficiale è riportato in [11]), montato all'interno di una serie di adattatori wireless, di cui fa parte il nostro D-Link DWL-G122 (rev. C1). La stessa Ralink era sviluppatrice anche dei firmware e degli omonimi driver, `rt73` appunto, necessari al funzionamento del chipset. Tuttavia questi software soffrivano di instabilità e malfunzionamenti, così da spingere in tempi più recenti un team di sviluppo di SerialMonkey a rielaborarli e migliorarli dando vita ad un progetto denominato “Rt2x00”.

I risultati di tale lavoro sono oggi definiti “legacy drivers” ed hanno conosciuto negli anni un'enorme diffusione, giustificata dalla loro stabilità e praticità di utilizzo, nonché dalla reperibilità dei sorgenti da un CVS giornaliero di SourceForge (di cui riportiamo il link alla voce [11]).

Data la diffusione dei legacy driver, il team SerialMonkey ha recentemente intrapreso lo sviluppo di una nuova generazione di questi stessi driver, definiti “in-kernel” poiché progettati per essere inseriti direttamente nel kernel Linux, pronti all'uso senza alcuna necessità di compilazione di sorgenti da parte dell'utente.

Nel frattempo, i legacy driver sono stati dichiarati permanentemente deprecati il 24 aprile 2009 scorso e sul CVS è disponibile il tarball della loro ultima release (di cui riportiamo il link alla voce [11]).

Nella proprio elaborato, Iora spiega come vi fosse incompatibilità tra il legacy driver Rt2x00 ed il driver del controller USB della scheda FoxBoard: l'adattatore wireless cessava di funzionare sotto alto carico di trasmissione, situazione assai critica e frequente in ambito di streaming video a basso o nullo rapporto di compressione.

Relativamente al problema postoci ed alla soluzione individuata, Axis SDK 2.01 contiene già i sorgenti di una versione funzionante del driver `rt73`, pertanto non sono necessari il download e la compilazione dei sorgenti di versioni più aggiornate di questo software per tale SDK.

Come già anticipato, l'utente può comunque scaricare e cross-compilare i sorgenti della “final release” del legacy driver, in caso avesse bisogno di particolari esigenze. Illustreremo la procedura di cross-compilazione ed installazione di `rt73` nel paragrafo 4.9.3.

4. Modalità operative

4.1. Scelta del sistema operativo

La cross-compilazione di software per la scheda FoxBoard richiede l'installazione di un cross-compilatore da architettura x86 verso architettura ETRAX 100LX. Il cross-compilatore in questione è denominato CRIS⁶-compiler, è sviluppato da John Crispin per conto della Axis ed è scaricabile dal sito [14]. Come si può osservare, il compilatore è disponibile nel repository sia come package (in due versioni, una “.deb” per Debian ed Ubuntu, ed una “.rpm” per Red Hat), che come tarball di sorgenti, pertanto il suo utilizzo è in teoricamente possibile su qualsiasi distribuzione Linux.

Per semplicità, il gruppo ha ritenuto conveniente scaricare ed installare direttamente la versione pacchettizzata, nonché eseguire gli esperimenti servendosi esclusivamente di sistemi Linux Ubuntu e Linux Debian, quindi della versione “.deb” del CRIS-compiler. In particolare si è osservato che gli Axis SDK:

- non funzionano correttamente sotto Linux Ubuntu e distribuzioni derivate (Kubuntu, Xubuntu, ecc.): gli errori riscontrati presentavano messaggi piuttosto criptici, inutili a determinare le cause del problema; si è così giunti alla decisione di passare a Linux Debian;
- hanno dato risultati migliori sotto Linux Debian: in questo caso la compilazione di fimage e software accessorio ha avuto esito positivo.

Il gruppo ha così condotto i successivi esperimenti dapprima servendosi del sistema Debian GNU/Linux 4.0 (denominato “etch” ed avente versione del kernel 2.6.24) per poi passare a Debian GNU/Linux 5.0 (denominato “lenny”, rilasciato l'11 aprile 2009 ed avente versione del kernel 2.6.26). I tool di sviluppo hanno comunque dato prova di funzionare correttamente con entrambe queste versioni di Linux Debian.

4.2. Installazione dei prerequisiti software

Prima di procedere all'installazione di qualsiasi Axis SDK è necessario installare alcuni pacchetti software di base. La dipendenza degli Axis SDK da tali pacchetti non sempre è resa esplicita nella documentazione in linea o nell'output di compilazione: a fronte di errori, spesso, si è reso necessario cercare soluzione ispezionando script di configurazione e makefiles, oppure consultando fonti on-line alternative. In questo modo sono emerse nuove dipendenze rispetto alla lista dei package stilata da Iora.

Cominciamo con l'installazione dei pacchetti da lui individuati con il comando:

```
# apt-get install gcc libc6-dev make wget subversion gawk bc byacc flex bison perl sed tar zlib1g zlib1g-dev md5sum libncurses5 libncurses5-dev which pmake
```

Per quanto riguarda `md5sum`, si è ottenuto il seguente messaggio di errore: “Impossibile trovare `md5sum`”. Inoltre, il comando `apt-cache search md5sum` ha trovato questo software in vari pacchetti, tra cui il più adatto sembra essere `coreutils`.

Installiamo poi i pacchetti aggiuntivi con il comando:

```
# apt-get install coreutils build-essential dpkg-dev autoconf automake xutils xutils-dev xorg-dev wireless-tools
```

dove:

⁶ CRIS: Code Reduced Instruction Set, acronimo che identifica architettura ed instruction set dei processori RISC della famiglia Axis ETRAX.

- `autoconf` è un generatore automatico di script di shell `configure`;
- `automake` è un generatore di `makefile`;
- `makedepend` (in `xutils-dev`) è un generatore di dipendenze nei `makefile`;
- i tre software precedenti sono necessari al funzionamento degli Axis SDK;
- `xorg-dev` è un insieme di librerie di sviluppo per l'X Window System e sarà necessario alla compilazione del `grab-client` di *VisLib*;
- `wireless-tools` è un insieme di tool di gestione della rete wireless (ad es.: `iwlist`, `iwconfig`) che saranno utili più avanti per stabilire la connessione non cablata con la FoxBoard.

Successivamente scarichiamo da [14] ed installiamo il CRIS-compiler. La versione attualmente disponibile, nonché quella usata nel presente lavoro è la 1.64 (mentre quella usata da lora era la 1.63). Per far ciò digitiamo i seguenti comandi (con i privilegi di root):

```
# wget http://www.axis.com/ftp/pub/axis/tools/etrax100lx/compiler-kit/cris-dist_1.64-1_i386.deb
# dpkg -i cris-dist_1.64-1_i386.deb
```

Ora è possibile procedere con il download e l'utilizzo degli SDK. Come già anticipato, il gruppo di lavoro ha portato a completamento il lavoro assegnato servendosi di Axis SDK 2.01. In seguito descriveremo la prassi seguita ed i risultati ottenuti.

4.3. Download del SDK

Per ottenere Axis SDK 2.01 è necessario seguire una procedura automatizzata che fa uso di un apposito script di shell, `install_svn_sdk.sh`, il quale scarica da [4] (se non presenti) e decomprime nella medesima directory i due file `devboard-R2_01.tar.gz` e `devboard-R2_01-distfiles.tar.gz`, scarica alcuni file aggiuntivi (tra cui i sorgenti del kernel Linux 2.6) e prepara il kit di sviluppo alla configurazione del kernel stesso. Digitiamo i seguenti comandi (preferibilmente con privilegi di root):

```
# wget http://www.acmesystems.it/download/install_svn_sdk.sh
# chmod +x install_svn_sdk.sh
# ./install_svn_sdk.sh
# cd devboard-R2_01
# make menuconfig
```

Durante l'esecuzione dell'ultimo comando viene chiesto all'utente se `etrax100boot` dovrà essere eseguito come "setuid root" (ovvero come root): si suggerisce di rispondere "y" + INVIO (per "yes") a tale quesito.

Si è inoltre individuata una procedura manuale (simile a quella relativa ad Axis SDK 2.10 descritta nella relazione di Iora), che non si serve dello script `install_svn_sdk.sh`, ma che consta dei seguenti comandi:

```
# wget http://www.acmesystems.it/download/devboard-R2_01.tar.gz
# wget http://www.acmesystems.it/download/devboard-R2_01-distfiles.tar.gz
# tar xvfz devboard-R2_01.tar.gz
# tar xvfz devboard-R2_01-distfiles.tar.gz
# cd devboard-R2_01
# ln -s ../distfiles
# DEV_BOARD_PRODUCT=fox ./configure
# make menuconfig
```

Durante il penultimo comando viene chiesto se cambiare prodotto a “fox” e durante l’ultimo se eseguire o meno `etrax100boot` e `fsboot` come “setuid root” (indipendentemente dai privilegi assunti durante questa procedura): rispondere “y” + INVIO ad entrambe le domande. Questa procedura tuttavia non fa uso del kernel 2.6, non genera le stesse voci del menu di configurazione rispetto alla procedura automatizzata, pertanto è da considerarsi errata.

➤ **Errata corrige: in 4.1.2.1 Iora imposta `DEV_BOARD_PRODUCT` al valore stringa “fox416” prima di invocare lo script `./configure`. Questo valore non è riconosciuto dallo script, che pertanto ritorna un messaggio di errore. Il valore esatto da impostare è infatti “fox_416” (con underscore).**

4.4. Configurazione del sistema operativo

Una volta terminata la procedura automatizzata compare nella finestra del terminale il menu grafico di configurazione del kernel. Inseriamo impostazioni simili a quelle specificate nel paragrafo 4.1.1.2 della relazione di Iora. Riportiamo in Tabella 2 le impostazioni che l’utente deve controllare con particolare attenzione:

• ACME FOXBOARD selection.....	FOXBOARD LX416
• Linux kernel.....	Linux 2.6.x
• Standard C Library	uClibc
• Driver settings (♥)	
○ Enable WLAN support.....	[*]
○ Wireless networking (WLAN)	
▪ Wireless tools.....	[*]
▪ DLink DWL-G122.....	C1
○ Enable webcam support.....	[*]
○ Webcam tools – Michel Xharad (servfox & co)	[*]
• Network settings	
○ Ethernet Settings	
▪ Use custom IP settings for eth0	[]
○ WLAN Settings	
▪ ESSID	Robotica
▪ Channel	11
▪ WLAN mode	Ad-Hoc
▪ WEP key	NONE
▪ IP	192.168.5.90
▪ Subnet.....	255.255.255.0
▪ Use WLAN as default route?.....	[*]
▪ Gateway.....	192.168.5.1
○ Nameserver Settings	
▪ Use custom nameserver	[*]
▪ Nameserver	194.25.2.129
○ MAC-Address	
▪ Set a MAC.....	[*]
▪ MAC	00:40:8C:00:00:00
○ Firewall Script	
▪ Enable firewall script	[]
• Applications	
○ Libraries (♦)	
○ Applications (♦)	
○ Networking (♥)	
▪ Enable web server	[*]
▪ Enable Point to Point Protocol (PPP) support	[]
▪ Enable IPtables support	[]
▪ Enable SMTP support	[]
▪ Enable FTP support	[]
▪ Enable OpenSSL support	[*]
▪ Enable ipsetd support	[]
▪ Enable DHCP support.....	[]
▪ Enable HTTPS support.....	[]
▪ Enable SSH support.....	[*]
▪ Enable TELNETD support	[*]
○ System tools (♥)	
▪ Enable EasyEdit support.....	[*]
▪ Enable BusyBox support.....	[*]
○ Device nodes	
▪ Create SCSI device nodes	[*]
▪ Create TTYUSB device nodes.....	[*]
• Advanced settings (♣)	
• Use pregenerated SSH keys.....	[*]

♥ - Deselezionare le voci non riportate.
♦ - Deselezionare tutto.
♣ - Non modificare.

Tabella 2 - Impostazioni di configurazione del sistema operativo

Si fa notare che, poiché l'opzione "Use custom IP settings for eth0" è disabilitata, la FoxBoard avrà per default indirizzo IP statico 192.168.0.90, subnet mask 255.255.255.0 ed IP del gateway 192.168.0.1, mentre non farà uso di DHCP o ipsetd per l'impostazione di IP dinamici.

Selezioniamo infine EXIT per uscire dal menu, poi YES per salvare le impostazioni configurate.

4.5. Compilazione del sistema operativo

Procediamo ora alla compilazione del sistema operativo (ed alla creazione della fimage) digitando i comandi:

```
# ./configure
# make
```

di cui il primo essenzialmente prepara alcuni file di codice sorgente e crea i makefile di compilazione del sistema applicando le impostazioni appena inserite, mentre il secondo procede alla compilazione vera e propria. Al termine della compilazione (dopo alcuni minuti) si dovrebbe ottenere il seguente messaggio:

```
-----SDK VERSION INFO-----
The image was build against SDK version 30
-----
```

e nella directory `path-sdk/devboard-R2_01` dovremmo trovare un file privo di estensione, denominato `fimage`, delle dimensioni di 4 MB (che coincide con la dimensione dello spazio di memoria flash sulla nostra scheda). Questa è la tipica immagine del sistema operativo Linux adatta alla nostra FoxBoard.

4.6. Configurazione del kernel

Arrivati a questo punto è necessario modificare alcune impostazioni di configurazione del kernel. Per far ciò l'SDK mette a disposizione un menu di configurazione a sé stante da quello utilizzato precedentemente per il sistema operativo in generale.

Dalle prove effettuate, la procedura di configurazione del kernel che andremo a seguire sembra funzionare solo dopo aver compilato almeno una volta il sistema operativo ed è per questo motivo che ne stiamo trattando solo ora.

La procedura che andremo ora ad effettuare non è stata documentata da Iora o sui forum Axis, ma ricavata per tentativi successivi basati sulla procedura descritta da Iora per la configurazione del kernel Linux sotto una differente versione di Axis SDK, ovvero la 2.10.

La necessità di modificare il kernel è dovuta ai seguenti fatti:

- il driver `pwc` fa uso della libreria `v4l` e, poiché questa viene di default installata nella fimage come kernel module, non riesce ad importare correttamente alcuni simboli in essa definiti; dai tentativi svolti si è osservato che l'installazione del modulo `pwc` va invece a buon fine qualora `v4l` viene installato in user mode e non come kernel module;
- il driver `pwc` incluso nella distribuzione Linux di Axis SDK 2.01 non funziona correttamente e quindi può essere escluso dalla compilazione;
- il driver `rt73` incluso nella distribuzione Linux di Axis SDK 2.01, benché attivato, necessita di configurazione;

Dalla directory `path-sdk/devboard-R2_01` digitiamo i seguenti comandi:

```
# . init_env
# cd os/linux-2.6
# make menuconfig
```

Dopo alcuni messaggi simili a `"make[1]: Nothing to be done for '...'"` (dovuti al fatto che alcuni target risultano già completati dal precedente `make` totale eseguito nel paragrafo 4.5) otteniamo il menu di configurazione del kernel 2.6.15. Riportiamo in Tabella 3 le impostazioni che l'utente deve controllare con particolare attenzione, avendo cura di mantenere invariate le voci non menzionate:

• Networking	
○ Networking support.....	[*]
○ Generic IEEE 802.11 Networking Stack.....	<M>
▪ IEEE 802.11 WEP Encryption (802.1x).....	< >
▪ IEEE 802.11i CCMP support.....	< >
▪ IEEE 802.11i TKIP support.....	< >
• Generic driver options	
○ Hotplug firmware loading support.....	< * >
• Network device support	
○ Network device support.....	[*]
▪ Wireless LAN (non hamradio) (♥)	
• Wireless LAN drivers (non hamradio) & Wireless Extensions.....	[*]
• Multimedia devices	
○ Video For Linux.....	< * > ⁷
▪ Video For Linux	
• CPiA Video For Linux.....	< >
▪ Radio adapters	
• Maestro on board radio.....	< >
○ Digital Video Broadcasting Devices	
▪ DVB For Linux.....	[]
• USB support	
○ USB Philips cameras.....	< > ⁸
○ USB SPCA5XX Sunplus/Vimicro/Sonix jpeg Cameras.....	< >
• Acmesystems	
○ Acmesystems third party drivers (♥)	
▪ Support for RALINK RT73-USB WIFI D-Link C1.....	[*]
♥ - Deselezionare le voci non riportate.	

Tabella 3 - Impostazioni di configurazione del kernel

Successivamente usciamo dal menu di configurazione selezionando EXIT e YES per salvare le nuove impostazioni. Una volta tornati alla riga di comando digitiamo:

```
# cp .config ../../kernelconfig-2.6
# cd ../../
# make
```

Ciò “esporta” le nuove impostazioni del kernel ed avvia nuovamente la compilazione del sistema operativo, che questa volta durerà molto meno rispetto alla precedente (poiché ora solo alcune parti del kernel devono essere aggiornate) al termine della quale la nuova fimage sarà pronta per il trasferimento sulla FoxBoard.

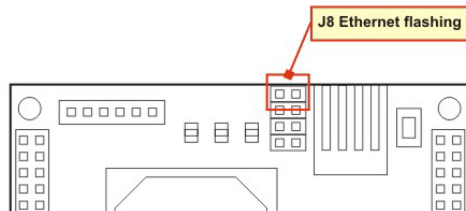
4.7. Flashing

Provvediamo ora a trasferire la fimage sulla FoxBoard. Per far ciò occorre seguire la seguente prassi:

⁷ Il software *video4Linux* dovrebbe risultare inizialmente installato come kernel module (con checkbox <M>). Per installare *v4l* senza modularizzazione è sufficiente selezionare questa voce e premere il tasto “Y” (la checkbox dovrebbe diventare < * >).

⁸ Questo trattasi del modulo *pwc* non funzionante descritto a fine paragrafo 3.2. Ne disabilitiamo la compilazione perché andremo a sostituirlo con una versione funzionante appositamente compilata.

1. impostiamo IP statico 192.168.0.1 sul PC desktop;
2. colleghiamo la FoxBoard al PC con il cavo Ethernet RJ45 cross-over;
3. posizioniamo un jumper sulla coppia di pin J8 della scheda, come illustrato nell'immagine seguente:



(questo collegamento attiva la modalità di “network boot” con registrazione in memoria flash della fimage che sarà ricevuta via Ethernet);

4. colleghiamo la FoxBoard all'alimentatore di rete (la spia LED verde dovrebbe emettere luce continua);
5. avviamo il trasferimento digitando a PC il comando (come root dalla directory `path-sdk/devboard-R2_01`):

```
# ./boot_linux -d eth0 -F -i fimage
...
0x80360000: Writing 0x00010000 bytes
0x80370000: Writing 0x00010000 bytes
0x80380000: No need to write
0x80390000: No need to write
0x803a0000: Writing 0x00010000 bytes
0x80000000: Verifying...OK
JUMP
0x00000000
END
Exiting with code 0
```

(durante il flashing le tre spie LED, gialla, verde e rossa, emettono luce continua);

6. al termine del flashing (sperabilmente con il messaggio “Exiting with code 0”, che indica esito positivo) scollegiamo l'alimentatore dalla scheda;
7. rimuoviamo il jumper dai pin J8;
8. ricollegiamo la FoxBoard all'alimentatore (ora le tre spie dovrebbero rimanere accese per qualche istante, successivamente la gialla e la rossa si dovrebbero spegnere: questo è il segnale che il bootstrap è terminato, il kernel funziona ed il sistema è in stato di “prompt”).

La scheda dovrebbe ora essere raggiungibile da PC via SSH o TELNET. Andrà d'ora in poi tenuto a mente che per default l'utente di accesso sulla scheda ha username “root” e password “pass”.

4.8. Accesso alla console

Per accedere alla console sulla scheda digitiamo i comandi:

```
# ssh root@192.168.0.90
root@192.168.0.90's password:          (qui abbiamo digitato "pass" + INVIO)
[root@axis ~]#
```

Ci troviamo quindi nella home directory sulla FoxBoard.

La prima volta che effettuiamo login sulla scheda, prima della richiesta della password compare un messaggio del protocollo SSH simile a:

```
The authenticity of host '192.168.0.90 (192.168.0.90)' can't be established.
RSA key fingerprint is xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx.
Are you sure you want to continue connecting (yes/no)?      (qui digitiamo "yes" + INVIO)
Warning: Permanently added '192.168.0.90' (RSA) to the list of known hosts.
```

Ciò è dovuto al fatto che il nostro PC non possiede ancora l'opportuna chiave RSA pre-generata per le future sessioni di lavoro SSH con la scheda. Una volta autenticato l'host 192.168.0.90, il login dovrebbe andare a buon fine.

Nota: Per l'introduzione della password è previsto il timeout di 1 minuto circa, oltre al quale la FoxBoard interrompe la connessione con il client e l'accesso viene negato. Al successivo tentativo di login compare il messaggio:

```
Connection closed by 192.168.0.90
```

Per ritentare successivamente il login è necessario ripetere il comando `ssh`.

Dopo alcuni esperimenti con file e directory si nota che il filesystem è read-only a meno delle subdirectory in `/mnt/flash`. Sarà dunque in `/mnt/flash` ove in seguito andremo a copiare (tramite il comando `scp`) i file necessari all'uso di webcam e adattatore wireless. In ogni momento sarà possibile tornare alla shell del nostro sistema desktop digitando il semplice comando `exit`.

Informiamo che i file di configurazione del sistema (che di norma si trovano in `/etc` o nelle sue subdirectory) sono modificabili in quanto sulla FoxBoard la directory `/etc` altro non è che un link alla directory `/mnt/flash/etc`.

4.9. Cross-compilazione ed installazione del software

Come specificato nella relazione di Iora al paragrafo 4.1.1.4, relativamente alla cross-compilazione di software con Axis SDK 2.01, è necessario copiare l'intero albero di directory dei sorgenti nella subdirectory `path-sdk/devboard-R2_01/apps` del SDK, nonché modificare i makefile in modo da utilizzare le regole di compilazione previste dallo stesso Axis SDK 2.01 (definite in `path-sdk/devboard-R2_01/tools/build/bin/Rules.axis`).

Per avviare la compilazione è necessario eseguire i seguenti comandi dalla directory `path-sdk/devboard-R2_01`:

```
# . init_env
# cd apps/dir-software
# make cris-axis-linux-gnuclibc
# make
```

Il primo comando esegue lo script di shell `init_env` che si occupa di impostare le variabili d'ambiente `AXIS_TOP_DIR` e `AXIS_KERNEL_DIR` rispettivamente ai valori `path-sdk/devboard-R2_01` e `path-sdk/devboard-R2_01/os/linux-2.6`; inoltre "prepende" (ovvero aggiunge in testa) i percorsi del CRIS-compiler `/usr/local/cris/bin` e `path-sdk/devboard-R2_01/tools/build/bin` alla variabile d'ambiente `PATH`.

➤ **Non è necessario eseguire `init_env` ad ogni compilazione, ma è importante ricordarsi di farlo ogniqualvolta si intenda cross-compilare utilizzando un nuovo terminale, poiché questo vede re-inizializzata la propria variabile `PATH`.**

4.9.1. Cross-compilazione ed installazione di *VisLib*

Una volta ottenuti i sorgenti della versione di *VisLib* modificata da ARLBS è possibile procedere alla compilazione di *grab-server* e *grab-client*, per i quali suggeriamo di lavorare su due copie distinte delle directory dei sorgenti.

La versione base fornitaci era stata modificata dal gruppo ARLBS suddividendo il *frame-grabber* in *grab-server* e *grab-client*: mentre il *grab-server* implementa varie funzioni di controllo utilizzando chiamate dirette al driver *v4l* attraverso le cosiddette *ioctl*, il *grab-client* può invocare tali funzioni da remoto inviando sul socket (a caratteri) opportuni codici di controllo, seguiti da eventuali parametri specifici per ciascun comando. Esempi di questi comandi sono: inizio e fine trasmissione, acquisizione di un nuovo frame, impostazione del guadagno (*gain*) e della frequenza di refresh dell'otturatore elettronico (*framerate*).

Purtroppo la versione base appariva incompleta. Infatti, le funzioni di regolazione *gain* e *framerate* (aventi rispettivamente firma `vlGrabSetGain(int)` e `vlGrabSetShutter(int)`) risultavano implementate solo in `grab-server.c`, mentre `grab-client.c` ne era sprovvisto e, poiché i programmi front-end grafici richiamaivano tali funzioni dalla libreria *grab-client*, la loro compilazione terminava con errori.

Si è pertanto dovuto intervenire su alcune parti della libreria base per:

- introdurre in `vislib.h` i due codici di controllo `NET_VLSETGAIN` e `NET_VLSETSHUTTER`;
- implementare in `grab-client.c` le funzioni omologhe `vlGrabSetGain(int)` e `vlGrabSetShutter(int)` (che inviano sul socket i rispettivi codici di controllo `NET_VLSETGAIN` e `NET_VLSETSHUTTER` seguiti dal valore intero fornito in input);
- aggiungere in `grab-server.c`, nel ciclo di lettura dal socket, le chiamate alle funzioni `vlGrabSetGain(int)` e `vlGrabSetShutter(int)` proprie del *grab-server* in corrispondenza alla lettura dei rispettivi codici di controllo `NET_VLSETGAIN` e `NET_VLSETSHUTTER`.

Dalle prove effettuate a seguito della compilazione, ora terminata con successo, si è constatato il corretto funzionamento di *grab-client* e *grab-server*. Riportiamo in Appendice le porzioni di codice modificate relativamente ai vari file di interesse.

4.9.1.1 Il *grab-server*

Relativamente alla cross-compilazione del *grab-server*, come spiegato poc'anzi è necessario copiare l'intero albero di directory di *VisLib* nella subdirectory `path-sdk/devboard-R2_01/apps` e modifichiamo il file `path-vislib/src/Makefile` come illustrato nel listato sottostante:

```

CC      = gcc
CFLAGS  = -funroll-loops -O3 -pedantic -Wall -W -I../include
LDFLAGS =
LDLIBS  = -I/usr/X11R6/include -L/usr/X11R6/lib # -lX11 -lXext

# = client for compiling grab-client.
# = server for compiling grab-server.
DEPLOYMENT = server

ifeq (server,$(DEPLOYMENT))

#
# GRAB-SERVER
#
AXIS_USABLE_LIBS = UCLIBC GLIBC
include $(AXIS_TOP_DIR)/tools/build/Rules.axis
CFLAGS += -I../include
SOURCES = grab-server.c
all: grab-server
grab-server: grab-server.c

else

#
# GRAB-CLIENT
#
SOURCES = common.c convolution.c display.c filter.c format.c grab-client.c \
          morph.c motion.c object.c utility.c blob.c
INCLUDES = $(wildcard ../include/*.h)
OBJECTS = $(SOURCES:.c=.o)
LIB_DIR = ../lib
LIB_TARGET = $(LIB_DIR)/libvislib.a
all: $(LIB_TARGET)
bt848check:
    @echo " Verifying that BT848 directory exists. If this test fails,"
    @echo " check that BT848_HOME is correctly defined in src/Makefile."
    test -e $(BT848_HOME)
    @echo " Success! It appears that BT848_HOME is properly set..."
$(LIB_TARGET): $(LIB_DIR) $(LIB_TARGET) $(OBJECTS)
$(LIB_DIR):
    mkdir $@

endif

tags:
    find . -name \*.[ch] -print | xargs etags -o TAGS
clean:
    rm -f $(PROGS) $(OBJECTS) $(LIB_TARGET) *~ TAGS core *.bak
    rm -rf $(LIB_DIR)

```

Facciamo notare che impostando nel listato precedente `DEPLOYMENT = server` oppure `DEPLOYMENT = client` (o qualsiasi altra stringa diversa da `server`) otterremo la generazione del `grab-server` e del `grab-client`, rispettivamente.

Il `grab-server` generato sarà il file eseguibile privo di estensione `path-vislib-grab-server/src/grab-server`. Dalla directory `path-sdk/devboard-R2_01` diamo i comandi:

```

# . init_env                                     (se necessario...)
# cd apps/dir-vislib-grab-server
# make cris-axis-linux-gnuuclibc
# make

```

Siamo ora pronti a trasferire l'eseguibile `path-vislib-grab-server/src/grab-server` sulla FoxBoard. A scheda avviata digitiamo, sempre dalla directory `path-vislib-grab-server`, il comando `scp` (*secure cp*) di copiatura di file via SSH:

```
# scp src/grab-server root@192.168.0.90:/mnt/flash
root@192.168.0.90's password: (qui abbiamo digitato "pass" + INVIO)
```

Verifichiamo poi l'effettiva presenza del file sulla scheda:

```
# ssh root@192.168.0.90
root@192.168.0.90's password: (qui abbiamo digitato "pass" + INVIO)
[root@axis ~]# cd /mnt/flash
[root@axis /mnt/flash]# ls
etc      grab-server  root
[root@axis /mnt/flash]# exit
```

4.9.1.2 Il grab-client

Apriamo il file `grab-client.c` e controlliamo che la riga `#define HOST` contenga il corretto indirizzo IP dell'interfaccia WLAN della scheda (impostato in Tabella 2), ovvero:

```
...
#define HOST "192.168.5.90" // IP address of FoxBoard's WLAN interface
...
```

Possiamo altresì impostare il `grab-client` affinché si connetta all'indirizzo IP relativo all'interfaccia Ethernet alla FoxBoard:

```
...
#define HOST "192.168.0.90" // IP address of FoxBoard's Ethernet interface
...
```

Generiamo ora il `grab-client` impostando `DEPLOYMENT = client` nel `makefile` corrispondente e digitando i comandi:

```
# cd path-vislib-grab-client
# make
```

Il `grab-client` generato consiste nella libreria statica `path-vislib-grab-client/lib/libvislib.a`, referenziata dai `makefile` di alcuni programmi dimostrativi (situati in `path-vislib-grab-client/examples` e `path-vislib-grab-client/user_programs`) che ne illustrano le funzionalità, e che saranno utilizzati in ultima analisi per verificare il buon esito dell'intero progetto.

Poiché abbiamo compilato dalla directory principale di `VisLib`, si sono compilati anche questi programmi dimostrativi.

4.9.2. Cross-compilazione ed installazione di *pwc*

Il sistema Linux per FoxBoard supporta alcuni semplici programmi di gestione dei moduli che si sono rivelati fondamentali per il nostro studio e la risoluzione di molti problemi in questo ambito. Saranno altresì indispensabili allo svolgimento del lavoro da compiere in seguito. Questi programmi sono:

- `lsmod`: per il listing dei moduli correntemente installati ed attivati;
- `insmod`: per l'installazione di un nuovo modulo;
- `rmmod`: per la disattivazione di un modulo in funzione;
- `dmesg`: per il listing dei debug-log relativi alla gestione ed al funzionamento di firmware e moduli.

➤ **L'installazione di un modulo con `insmod` non è permanente: dovremo ripetere il comando se riavviamo la scheda FoxBoard.**

Nel sistema non sono invece presenti programmi di gestione più elaborati, come `depmod`.

Dopo essersi procurati i sorgenti di questo driver dal sito ufficiale [10], si procede alla cross-compilazione. Analogamente a quanto svolto per *VisLib*, decomprimiamo il tarball, copiamo la directory dei sorgenti nella subdirectory `path-sdk/devboard-R2_01/apps` e modifichiamo il file `path-pwc/Makefile` come illustrato nel listato sottostante:

```
MODULE = pwc.o
pwc-y = pwc-if.o pwc-misc.o pwc-ctrl.o pwc-v4l.o pwc-uncompress.o pwc-decl.o \
      pwc-dec23.o pwc-kiara.o pwc-timon.o
EXTRA_CFLAGS = -Wall -I$(PWD) -DNOKERNEL $(USER_OPT)
PREVENT_RECURSIVE_INCLUDE = 1
include $(AXIS_TOP_DIR)/modules/rules.build_modules
```

Segue la cross-compilazione dei sorgenti.

Controlliamo che i percorsi del SDK siano presenti nella variabile d'ambiente `PATH` (che contiene una lista di percorsi separati da colon, “:”); dovremmo ottenere un risultato simile a quello sotto riportato se stiamo usando il terminale con cui abbiamo cross-compilato il `grab-server`:

```
# echo $PATH
/usr/local/cris/bin:path-sdk/devboard-R2_01/tools/build/bin:
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Se i due percorsi `/usr/local/cris/bin` e `path-sdk/devboard-R2_01/tools/build/bin` mancano dall'output, è necessario ripetere il seguente comando dalla directory `path-sdk/devboard-R2_01`:

```
# . init_env
```

Diamo ora i seguenti comandi dalla directory `path-sdk/devboard-R2_01`:

```
# cd apps/dir-pwc
# make cris-axis-linux-gnuclibc
# make
```

Al termine della compilazione dovremmo constatare la presenza nella directory di `pwc` alcuni file oggetto (con estensione “.o”), nonché del file `pwc.ko`. Questo è il file del kernel module che dovrà essere trasferito ed installato sulla FoxBoard. A scheda avviata, dalla directory `path-sdk/devboard-R2_01/apps/dir-pwc` digitiamo i comandi:

```
# scp pwc.ko root@192.168.0.90:/mnt/flash
root@192.168.0.90's password: (qui abbiamo digitato "pass" + INVIO)
```

Verifichiamo poi l'effettiva presenza del file sulla scheda e procediamo con l'installazione del modulo utilizzando il comando `insmod`:

```
# ssh root@192.168.0.90
root@192.168.0.90's password: (qui abbiamo digitato "pass" + INVIO)
[root@axis ~]# cd /mnt/flash
[root@axis /mnt/flash]# ls
etc      grab-server  pwc.ko      root
[root@axis /mnt/flash]# insmod pwc.ko
Using pwc.ko
[root@axis /mnt/flash]#
```

Se non compaiono ulteriori messaggi dopo “Using `pwc.ko`” significa che il modulo in questione è stato correttamente installato e risulta ora attivo.

Per maggior sicurezza è possibile dare il comando `lsmod` e verificare l'effettiva presenza del modulo “`pwc`” nella lista visualizzata.

Si informa che, a prescindere dalla versione di `pwc` utilizzata, questo driver crea automaticamente il device node `/dev/video0` relativo alla webcam USB.

Se non avessimo tolto la modularizzazione di `v4l` nel menu di configurazione del kernel, attraverso la procedura descritta a paragrafo 4.6, avremmo ora ottenuto il seguente messaggio di errore relativo ad `insmod`:

```
Using pwc.ko
insmod: cannot insert 'pwc.ko': Unknown symbol in module (-1): No such file or
directory
```

Analizzando il listato di `dmesg` avremmo inoltre trovato queste ultime righe:

```
...
pwc: Unknown symbol video_devdata
pwc: Unknown symbol video_unregister_device
pwc: Unknown symbol video_device_alloc
pwc: Unknown symbol video_register_device
pwc: Unknown symbol v4l2_ioctl_names
pwc: Unknown symbol video_usercopy
pwc: Unknown symbol video_device_release
```

Da questi log si evince che il module loader non è in grado di risolvere tali simboli, che sono usati nei sorgenti di `pwc` e si scopre facilmente esser definiti in `v4l`, il quale è in tal caso installato come kernel module.

4.9.3. Cross-compilazione ed installazione di `rt73`

Come spiegato da Iora nel paragrafo 4.5 del proprio elaborato, Axis SDK 2.01 include già il driver `rt73`, pertanto questo step è da considerarsi superato.

Si puntualizza che se non avessimo fornito le impostazioni relative al “hotplug firmware loading” nel menu di configurazione del kernel, attraverso la procedura descritta a paragrafo 4.6, l'adattatore wireless D-Link non funzionerebbe dopo il plug alla FoxBoard: le spie di Link e Activity resterebbero spente. Analizzando il listato di `dmesg` avremmo inoltre trovato una coppia di righe per ogni plug/unplug:

```
rt73: Unknown symbol release_firmware
rt73: Unknown symbol request_firmware
```

Da ciò si avrebbe dedotto il mancato caricamento del firmware Ralink, richiesto dal driver `rt73` (eppure in funzione senza di esso), al momento del plug.

Per completezza riportiamo la procedura di compilazione ed installazione anche per questo software, peraltro molto simile a quella seguita per il driver `pwc`.

Scaricati i sorgenti dal sito [11], decomprimiamo il tarball nella subdirectory `path-sdk/devboard-R2_01/apps` e modifichiamo il file `path-rt73/Module/Makefile` come di seguito illustrato. Tale makefile corrisponde esattamente a quello riportato a paragrafo 4.5 della relazione di Iora; nonostante riguardi la compilazione di `rt73` sotto Axis SDK 2.10, esso risulta valido anche per il nostro Axis SDK 2.01 (in particolare il percorso simbolico `$(AXIS_TOP_DIR)/modules/rules.build_modules` è corretto in entrambe le versioni del SDK e identifica il file di regole per la compilazione di moduli).

```
MODULE = rt73.o
rt73-y = rtmp_main.o mlme.o connect.o rtusb_bulk.o rtusb_io.o sync.o assoc.o \
        auth.o auth_rsp.o rtusb_data.o rtmp_init.o sanity.o rtmp_wep.o rtmp_info.o \
        rtmp_tkip.o wpa.o md5.o
PREVENT_RECURSIVE_INCLUDE = 1
include $(AXIS_TOP_DIR)/modules/rules.build_modules
```

Procediamo ora alla cross-compilazione dei sorgenti.

Controlliamo che i percorsi del SDK siano presenti nella variabile d'ambiente PATH (che contiene una lista di percorsi separati da colon, “:”); dovremmo ottenere un risultato simile a quello sotto riportato se stiamo usando il terminale con cui abbiamo cross-compilato il grab-server o *pwc*:

```
# echo $PATH
/usr/local/cris/bin:path-sdk/devboard-R2_01/tools/build/bin:
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Se i due percorsi `/usr/local/cris/bin` e `path-sdk/devboard-R2_01/tools/build/bin` mancano dall'output, è necessario ripetere il seguente comando dalla directory `path-sdk/devboard-R2_01`:

```
# . init_env
```

Diamo ora i seguenti comandi dalla directory `path-sdk/devboard-R2_01`:

```
# cd apps/dir-rt73/Module
# make cris-axis-linux-gnuuclibc
# make
```

Al termine della compilazione dovremmo constatare la presenza nella directory `path-rt73/Module` alcuni file oggetto (con estensione “.o”), nonché dei file `rt73.ko` e `rt73.bin`. Il primo file è il kernel module, mentre il secondo è il firmware del chipset Ralink. Entrambi dovranno essere trasferiti sulla FoxBoard. Sempre dalla subdirectory `path-sdk/devboard-R2_01/apps/dir-rt73` diamo i comandi:

```
# scp rt73.ko rt73.bin root@192.168.0.90:/mnt/flash
root@192.168.0.90's password: (qui abbiamo digitato "pass" + INVIO)
```

Verifichiamo poi l'effettiva presenza del file sulla scheda e procediamo con l'installazione del nuovo modulo `rt73` utilizzando il comando `insmod`:

```
# ssh root@192.168.0.90
root@192.168.0.90's password: (qui abbiamo digitato "pass" + INVIO)
[root@axis ~]# cd /mnt/flash
[root@axis /mnt/flash]# ls
etc      grab-server  pwc.ko      root      rt73.bin    rt73.ko
[root@axis /mnt/flash]# insmod rt73.ko
Using rt73.ko
[root@axis /mnt/flash]#
```

Si informa che, a prescindere dalla versione di `rt73` utilizzata, questo driver crea automaticamente il device `node /dev/rausb0` relativo all'adattatore USB di rete wireless.

4.10. Risultati

Abbiamo terminato l'installazione del software sulla FoxBoard. Procediamo ora con il test finale sul funzionamento di quanto abbiamo finora prodotto.

Supponendo di partire a scheda spenta, colleghiamo webcam e adattatore wireless alle due uscite USB montate sulla scheda. Successivamente la alimentiamo, attendiamo il termine del boot ed accediamo tramite SSH nella directory `/mnt/flash`:

```
# ssh root@192.168.0.90
root@192.168.0.90's password:          (qui abbiamo digitato "pass" + INVIO)
[root@axis ~]# cd /mnt/flash
[root@axis /mnt/flash]#
```

Installiamo ora il modulo `pwc.ko` con il comando:

```
[root@axis /mnt/flash]# insmod pwc.ko
Using pwc.ko
[root@axis /mnt/flash]#
```

Possiamo constatare che il modulo `rt73` è già attivo: ciò non risulta dal comando `lsmod` (che visualizza soltanto `"pwc"`), ma dal comando `ps -a`, che mostra la lista di tutti i processi attivi in entrambe le modalità utente e supervisore. Troviamo infatti due processi denominati `"[rt73]"`, per i quali le parentesi graffe indicano appunto la modalità supervisore.

Avviamo successivamente il `grab-server`:

```
[root@axis /mnt/flash]# ./grab-server
```

Da questo momento la console è bloccata perché il processo `grab-server` gira finché non viene interrotto da CTRL+C. In alternativa avremmo potuto avviare il `grab-server` in background, ri-ottenendo immediatamente il prompt della console.

Apriamo ora una nuova finestra di terminale e digitiamo il comando:

```
# path-vislib-grab-client/examples/videoin
```

Questo avvia il programma client (denominato appunto `"videoin"`) di visualizzazione delle immagini inviate dalla FoxBoard. A causa di AGC, discusso a paragrafo 3.2, l'immagine appare inizialmente nera e si schiarisce via via dopo alcuni secondi mostrando le effettive immagini riprese dalla webcam.


4.10.1. Modifiche apportate a *VisLib*

Si è riscontrato un problema durante il primo tentativo di messa in funzione del `grab-client`. Al suo avvio fa uso delle API del sistema X11 per rilevare il depth dello schermo grafico in modo da richiamare poi le funzioni di disegno dei frame adatte al numero di colori rilevato. Nel file `display.c` di *VisLib* è definita una funzione chiamata `"_vlDisplayImage"` responsabile del disegno di un'immagine sullo schermo: questa è utilizzata dai programmi front-end per disegnare i frame del flusso video. Attualmente la funzione supporta solo 8, 16 e 24 bpp (bit-per-pixel), corrispondenti a 256, 65.536 e 16.777.216 colori, rispettivamente. Il problema sorge poiché il server X rileva la presenza di 32bpp di profondità dello schermo indipendentemente dal depth realmente impostato. Una volta connesso al `grab-server`, quindi, il `grab-client` non visualizza le immagini ricevute ma ripete continuamente nel proprio terminale il seguente messaggio di errore:

```
_vlDisplayImage: depth=32 is not supported
vlShow: error: could not display image
```

Una semplice soluzione a questo problema, tuttavia assai poco elegante, consiste nel riavviare il server X con 24bpp di depth e nel modificare la funzione `_vlDisplayImage` affinché supporti i 32bpp, pur tuttavia utilizzando lo stesso codice di disegno

valido per i 24bpp. In questa funzione, infatti, il disegno viene differenziato all'interno di un semplice ed intuitivo blocco condizionale:

<pre>... switch(depth){ case 24: ... break; case 16: ... break; case 8: ... break; } ...</pre>		<pre>... switch(depth){ case 24, 32: ... break; case 16: ... break; case 8: ... break; } ...</pre>
--	---	--

Dopo aver ricompilato il grab-client, questo visualizzava correttamente le immagini.

4.10.2. Lo script di avvio automatico

Per garantire l'avvio automatico del grab-server all'accensione della FoxBoard, è sufficiente creare lo script `/etc/init.d/boottime/start-grab-server` contenente i comandi:

```
/sbin/insmod /mnt/flash/pwc.ko
/mnt/flash/grab-server
```

Com'è facile capire, il primo carica il kernel module `pwc`, mentre il secondo avvia il grab-server in foreground (ciò non preclude la possibilità di accedere alla scheda via terminale remoto, dato che ora il processo grab-server non sarebbe figlio del demone SSH, e quest'ultimo non deve attenderne il termine per proseguire con la sessione di lavoro).

Questo script, assieme a tutti i file eventualmente presenti in `/etc/init.d/boottime`, viene eseguito dallo script di shell `/bin/start_ok.sh` all'avvio del sistema. Esso è infatti a sua volta invocato da `/etc/inittab` nel runlevel 3, ovvero "full multiuser mode". In tal modo, la FoxBoard entra in funzione come "wireless webcam" dall'istante in cui termina il bootstrap. Risulta ovviamente necessario che webcam e adattatore wireless siano collegati alla scheda sin dall'accensione della scheda.

4.10.3. Alcune considerazioni sulla configurazione della WLAN su PC

Per poter impostare manualmente l'indirizzo IP sull'interfaccia WLAN sul PC è necessario eseguire i seguenti passi:

- Controllare che l'utente di lavoro appartenga al gruppo "netdev" di gestione delle reti, eventualmente dando il comando:

```
# adduser nome-utente netdev
```

Nel nostro caso, trattandosi dell'utente `root`, tale comando è inutile poiché questo utente speciale fa già parte del suddetto gruppo.

- Rimuovere (sotto Gnome) il programma Network Manager per la gestione automatica delle connessioni di rete, poiché questo impedisce la connessione alla FoxBoard tramite IP statico e tenta comunque un DHCP discover, che fallisce impostando un indirizzo link-local 169.254.xxx.xxx (cosa ovvia non essendo attivo sulla rete alcun un DHCP server).

```
# apt-get remove network-manager network-manager-gnome
```

- Modificare il file `/etc/network/interfaces` in modo da configurare `wlan0` (l'interfaccia WLAN) affinché instauri una connessione tramite IP fisso 192.168.5.1⁹ alla rete wireless con ESSID "Robotica" creata dalla FoxBoard:

```
auto wlan0
iface wlan0 inet static
    network 192.168.5.0
    address 192.168.5.1
    netmask 255.255.255.0
    wireless-essid Robotica
    wireless-mode ad-hoc
    wireless-channel 11
pre-up ifconfig wlan0 up
```

Attributi aggiunti dal package `wireless-tools`.

e successivamente far ripartire il servizio di networking con il comando:

```
# /etc/init.d/networking restart
```

4.10.4.Osservazioni finali

Il programma front-end di visualizzazione delle immagini utilizzato (`path-vislib-grab-client/examples/videoin`) mostra a cadenza regolare nel terminale il numero medio di frame ricevute per secondo. Si è deciso di configurare una variante del `grab-client` per la trasmissione via interfaccia Ethernet, utilizzando la procedura descritta nel paragrafo 4.9.1.2, allo scopo di confrontarne le performance rispetto alla trasmissione wireless, utilizzando come indicatore di performance proprio il numero di frame/sec.

I test comparativi condotti sulle versioni, "wired" e "wireless", di questo front-end hanno mostrato che la trasmissione nel secondo caso ha velocità notevolmente più bassa (di quanto ci si aspettava) rispetto al primo:

- Frequenza media su rete wired¹⁰: ~5.4 frame/sec
- Frequenza media su rete wireless¹¹: ~0.9 frame/sec

Le scarse performance del canale wireless potrebbero essere attribuite a vari fattori, quali:

- l'inadeguatezza delle dimensioni del frame-buffer in transito, considerando le latenze introdotte da tempi di propagazione, rumore, ritrasmissioni, ecc...
- la mancata ottimizzazione dei driver `rt73`, dato che si è utilizzata la versione di base prevista da Axis SDK 2.01.

⁹ Tale indirizzo appartiene alla stessa subnet dell'indirizzo IP configurato sulla FoxBoard, che è 192.168.5.90 ed ha subnet mask 255.255.255.0 (come specificato a paragrafo 4.4).

¹⁰ Test condotti con cavo diretto RJ45 cross-over (5m, SFTP, categoria 5, 100MHz), nessun dispositivo di rete intermedio.

¹¹ Test condotti su distanza in linea d'aria di 1m fra le due antenne, nessun ostacolo frapposto.

5. Conclusioni e sviluppi futuri

Il progetto ha dato risultati molto positivi e pertanto può essere considerato concluso. Tuttavia non può dirsi esentato da sviluppi futuri.

Alla luce di quanto detto in precedenza, infatti, uno di questi potrebbe riguardare il miglioramento delle performance del sistema di trasmissione. Un tentativo in questo senso potrebbe essere condotto cross-compilando l'ultima release del driver *rt73*, procedura descritta a paragrafo 4.9.3.

Un ulteriore ambito di ricerca potrebbe vertere sulla re-introduzione dell'interfacciamento della linea seriale su socket TCP e la coesistenza nella banda disponibile del traffico video con quello di controllo remoto del robot.

In entrambi i casi, per quanto concerne lo streaming video si consiglia la messa a punto di tecniche di ottimizzazione dei parametri di trasmissione¹², oppure di riduzione del carico dati. Queste ultime potrebbero riguardare:

- l'utilizzo di immagini con risoluzione inferiore, in base alle esigenze imposte dal proprio ambito di utilizzo;
- la riduzione del depth da 24bpp a 8bpp o 16bpp, in base alle esigenze imposte dal proprio ambito di utilizzo;
- l'introduzione di un algoritmo di compressione video.

A causa delle limitate risorse di calcolo del processore Axis ETRAX, l'ultimo metodo proposto è tuttavia sconsigliato.

¹² Parametri di trasmissione: quali ad esempio dimensione o frequenza di flush (invio) del frame-buffer, ove questo è inteso come unità di trasferimento dati del flusso video. La dimensione del frame-buffer dovrebbe essere calibrata tenendo conto sia delle probabilità di ritrasmissione di pacchetti danneggiati o persi, che dell'*overhead* introdotto da header e trailer del frame 802.11.

Bibliografia

- [1] Relazione tecnica “Configurazione FoxLX416 Board”, Marco Iora, 11 settembre 2007.
http://www.ing.unibs.it/~arl/docs/projects/Sau_03.pdf
- [2] Sito ufficiale di Acme Systems, ditta produttrice della scheda FoxBoard.
<http://www.acmesystems.it>
- [3] Sito ufficiale di Axis Communications, ditta produttrice dei processori ETRAX.
<http://www.axis.com>
- [4] Tarball dei sorgenti di Axis SDK 2.01.
Sito web:
http://www.acmesystems.it/download/install_svn_sdk.sh
http://www.acmesystems.it/download/devboard-R2_01.tar.gz
http://www.acmesystems.it/download/devboard-R2_01-distfiles.tar.gz
Repository:
http://www.axis.com/ftp/pub/axis/dev/soft_dist/R2_01/devboard-R2_01.tar.gz
http://www.axis.com/ftp/pub/axis/dev/soft_dist/R2_01/devboard-R2_01-distfiles.tar.gz
- [5] Tarball dei sorgenti di Axis SDK 2.10.
Sito web:
http://www.acmesystems.it/download/devboard-R2_10.tar.gz
http://www.acmesystems.it/download/devboard-R2_10-distfiles.tar.gz
Repository:
http://www.axis.com/ftp/pub/axis/dev/soft_dist/R2_10/devboard-R2_10.tar.gz
http://www.axis.com/ftp/pub/axis/dev/soft_dist/R2_10/devboard-R2_10-distfiles.tar.gz
- [6] Tarball dei sorgenti del Axis SDK 2.20.
Sito web:
http://www.acmesystems.it/download/devboard-R2_20.tar.gz
http://www.acmesystems.it/download/devboard-R2_20-distfiles.tar.gz
Repository:
http://www.axis.com/ftp/pub/axis/dev/soft_dist/R2_20/devboard-R2_20.tar.gz
http://www.axis.com/ftp/pub/axis/dev/soft_dist/R2_20/devboard-R2_20-distfiles.tar.gz
- [7] Repository del sistema OpenWRT.
<http://downloads.openwrt.org>
Fimage per Axis ETRAX e relativi packages:
<http://downloads.openwrt.org/snapshots/trunk/etrax>
<http://downloads.openwrt.org/snapshots/trunk/etrax/packages>
Latest release del progetto Kamikaze:
<http://downloads.openwrt.org/kamikaze/8.09.1>
- [8] Siti dedicati al sistema CrisOs.
Sito web:
<http://crisos.org/dokuwiki/doku.php>
Repository:
<http://download.tuxfamily.org/crisos>
- [9] Sito web della libreria *VisLib* di Mobile Robots.
<http://robots.mobilerobots.com/wiki/VisLib>
- [10] Sito ufficiale del driver *pwc* (per webcam USB Philips).
<http://www.saillard.org/linux/pwc>
Tarball dei sorgenti di *pwc* 10.0.12-RC1 (versione utilizzata nel progetto).
<http://www.saillard.org/linux/pwc/files/pwc-10.0.12-rc1.tar.bz2>

- [11] Sito ufficiale del progetto Rt2x00.
<http://rt2x00.serialmonkey.com>
CVS giornaliero SourceForge dei legacy driver.
http://sourceforge.net/project/showfiles.php?group_id=107832
Tarball dei sorgenti della final release del legacy driver *rt73* (CVS SourceForge).
http://sourceforge.net/project/downloading.php?group_id=107832&filename=rt73-cvs-daily.tar.gz&a=55363250
Sito ufficiale di RalinkTech, ditta produttrice di chipset per adattatori wireless.
<http://www.ralinktech.com>
- [12] Rapporto tecnico “Alcune considerazioni sul driver pwc” (ARL-TR-08-01),
Prof. R. Cassinis, 14 febbraio 2008.
<http://www.ing.unibs.it/~arl/docs/techreps/ARL-TR-08-01.pdf>
- [13] Rapporto tecnico “Ricompilazione driver per webcam Philips” (ARL-TR-08-03),
Prof. R. Cassinis, 3 giugno 2008.
<http://www.ing.unibs.it/~arl/docs/techreps/ARL-TR-08-03.pdf>
- [14] Repository del CRIS-compiler per architettura Axis ETRAX.
<http://www.axis.com/ftp/pub/axis/tools/etrax100x/compiler-kit>
- [15] Sito web di ARLBS (**A**dvanced **R**obotics **L**aboratory of **B**rescia University).
<http://www.ing.unibs.it/~arl>

Appendice

src/grab-server.c

```

...
int vlGrabSetGain(int value){
    // Calls ioctl(device_ID, VIDIOCPWCSAGC, &value)
    ...
}

int vlGrabSetShutter(int value){
    // Calls ioctl(device_ID, VIDIOCPWCSSHUTTER, &value)
    ...
}

...

int main(void){
    ...
        // 'buf' is the buffer (char array) read from TCP socket.
    switch (buf[0]) {
        case NET_VLGRABINIT:
            ...
            break;
        case NET_VLGRABSHUTDOWN:
            ...
            break;
        case NET_VLGRABIMAGE:
            ...
            break;
        case NET_VLGRABLASTIMAGE:
            ...
            break;
        case NET_VLSETGAIN:
            char gain_str[4];
            int gain;
            strncpy(buf+1, gain_str, 3);
            gain_str[3] = 0;
            gain = atoi(gain_str);
            result = vlGrabSetGain(gain);
            if(send(new_fd, &result, 1, 0) == -1)
                perror("send_set_gain_result");
            break;
        case NET_VLSETSHUTTER:
            char shutter_str[4];
            int shutter;
            strncpy(buf+1, shutter_str, 3);
            shutter_str[3] = 0;
            shutter = atoi(shutter_str);
            result = vlGrabSetShutter(shutter);
            if(send(new_fd, &result, 1, 0) == -1)
                perror("send_set_shutter_result");
            break;
        default:
            printf("Messaggio non valido dal client.");
            close(new_fd);
            exit(1);
    }
    ...
}

```

src/grab-client.c

```

...

int vlGrabSetGain(int value){
    char setMsg[8];
    int result;
    snprintf(setMsg, sizeof(setMsg), "%c%.3d", NET_VLSETGAIN, value);
    if(send(sockfd, setMsg, 1, 0) == -1){
        perror("send_VLSETGAIN");
        return -1;
    }
    if(recv(sockfd, &result, MAXDATASIZE, 0) == -1){
        perror("recv_VLSETGAIN");
    }
    return (result);
}

int vlGrabSetShutter(int value){
    char setMsg[8];
    int result;
    snprintf(setMsg, sizeof(setMsg), "%c%.3d", NET_VLSETSHUTTER, value);
    if(send(sockfd, setMsg, 1, 0) == -1){
        perror("send_VLSETSHUTTER");
        return -1;
    }
    if(recv(sockfd, &result, MAXDATASIZE, 0) == -1){
        perror("recv_VLSETSHUTTER");
    }
    return (result);
}

```

include/vislib.h

```

...

#define NET_VLGRABINIT      'i'
#define NET_VLGRABSHUTDOWN 's'
#define NET_VLGRABIMAGE    'g'
#define NET_VLGRABLASTIMAGE 'l'
#define NET_VLSETGAIN      'a'
#define NET_VLSETSHUTTER  'h'

#endif /* __VISLIB_H__ */

```

Indice

SOMMARIO	1
1. INTRODUZIONE	2
2. IL PROBLEMA AFFRONTATO	4
3. LA SOLUZIONE ADOTTATA	5
3.1. La libreria grafica <i>VisLib</i>	5
3.2. Il driver <i>pwc</i>	6
3.3. Il driver <i>rt73</i>	7
4. MODALITÀ OPERATIVE	8
4.1. Scelta del sistema operativo	8
4.2. Installazione dei prerequisiti software	8
4.3. Download del SDK	9
4.4. Configurazione del sistema operativo	10
4.5. Compilazione del sistema operativo	12
4.6. Configurazione del kernel	12
4.7. Flashing	13
4.8. Accesso alla console	14
4.9. Cross-compilazione ed installazione del software	15
4.9.1. Cross-compilazione ed installazione di <i>VisLib</i>	16
4.9.1.1 <i>Il grab-server</i>	16
4.9.1.2 <i>Il grab-client</i>	18
4.9.2. Cross-compilazione ed installazione di <i>pwc</i>	18
4.9.3. Cross-compilazione ed installazione di <i>rt73</i>	20
4.10. Risultati	21
4.10.1. Modifiche apportate a <i>VisLib</i>	22
4.10.2. Lo script di avvio automatico	23
4.10.3. Alcune considerazioni sulla configurazione della WLAN su PC	23
4.10.4. Osservazioni finali	24
5. CONCLUSIONI E SVILUPPI FUTURI	25
BIBLIOGRAFIA	26
APPENDICE	28
INDICE	30