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)

Configurazione FoxLX416 Board

Elaborato di esame di:

Marco Iora

Consegnato il: **11 Settembre 2007**

Sommario

Il lavoro svolto ha avuto come obiettivo la configurazione della scheda FoxBoard LX 416 ai fini di sostituire i pc onboard di Speedy e Morgul. La FoxBoard è una scheda madre di piccole dimensioni (66x72mm) completa di processore, ram e memoria persistente di tipo flash, porte usb, seriale ed ethernet. Il sistema operativo utilizzato sulla piattaforma è Linux e l'obiettivo del lavoro è rendere operativa una connessione wireless e accessibili, tramite questa connessione, i flussi dati di una webcam e della porta seriale.

1. Introduzione

Il Laboratorio di Robotica Avanzata mette a disposizione per l'attività di studio e ricerca nel campo della robotica mobile alcuni robot Pioneer della MobileRobots [1].

Due di questi, Speedy e Morgul, vengono controllati tramite porta seriale direttamente da pc portatili installati sul corpo dei robot stessi. Su questi pc sono installati il sistema operativo Linux e le librerie Aria per il controllo dei robot Pioneer.

Il lavoro descritto in questo documento nasce nell'intento di sostituire questi pc portatili con una piattaforma più semplice, piccola e leggera tramite la quale sia possibile controllare i robot.

Questa scelta offre i seguenti vantaggi:

- riduzione del carico trasportato dai robot da oltre 1 kg a meno di 100 g
- riduzione dell'ingombro dell'apparecchiatura di controllo del robot da ca. 400x300x50mm a 128x72x23mm
- riduzione del consumo delle batterie in diretta dipendenza dalla riduzione del carico trasportato
- riduzione del consumo delle batterie per l'effettivo utilizzo di minore energia da parte del nuovo dispositivo di controllo rispetto ad un pc portatile.

A fronte di questi vantaggi va notato che le risorse di calcolo, memoria e comunicazione disponibili sulla piattaforma sono decisamente limitate. Si passa infatti da processori CISC Intel con frequenza di clock di 2 GHz (per Speedy) e 1,6 GHz (per Morgul) ad un processore Axis ETRAX 100LX RISC con frequenza di clock di 100MHz. Per quanto riguarda lo spazio di memorizzazione si passa invece da decine di Gigabyte a 4Megabyte.

2. Il problema affrontato

Vista la scarsa potenza di calcolo disponibile sulla nuova piattaforma, si è deciso di remotizzare la maggior parte delle funzionalità di controllo mantenendo presso il robot solo lo stretto necessario.

In particolare, la libreria Aria risiederà su un calcolatore remoto e si provvederà ad instaurare un tunnel di comunicazione su rete senza fili per permettere alla libreria di raggiungere la porta seriale posta sul robot.

In questo modo sarà possibile compilare ed eseguire i programmi che utilizzano la libreria Aria per il controllo del robot direttamente dal calcolatore remoto.

Oltre al controllo dei robot, i pc portatili attualmente in uso si occupano di fornire le immagini provenienti dalle webcam poste sui robot. Per garantire questa funzionalità anche con l'utilizzo della scheda FoxBoard, come terminale a bordo dei robot, sarà necessario fare in modo che il software sulla scheda acquisisca immagini da una webcam ad essa collegata e quindi le invii sulla rete senza fili verso il calcolatore remoto.

I due problemi fin qui descritti fanno riferimento all'utilizzo di una connessione di rete senza fili. La scheda FoxBoard non integra un adattatore di rete wireless, sarà per questo necessario utilizzarne uno esterno da collegare alla porta usb.

In particolare i dispositivi che verranno utilizzati per questo progetto oltre la scheda FoxBoard sono:

- Un adattatore di rete wireless usb D-Link DWL-G122 (rev. C1).
- Una webcam Philips TouCam II PCVC840.
- Un adattatore per la porta seriale (TTL 3.3 V to RS232)

Per quanto riguarda il software, scheda e processore sono supportati dal sistema operativo Linux ed esistono kit di sviluppo per compilare i propri programmi per questa piattaforma.

I software che si intende compilare/modificare per l'utilizzo con la FoxBoard sono:

- Il programma remserial [4] per trasportare la comunicazione seriale su un canale TCP/IP.
- La libreria VisLib [6] per acquisire le immagini dalla webcam.
- Il driver PWC [5] per l'utilizzo della webcam.
- Il driver RT73 [7] per l'utilizzo dell'adattatore wireless.

3. La soluzione adottata

Prima di analizzare le soluzioni specifiche per i problemi analizzati nel secondo capitolo è necessario spendere alcune parole sulla modalità di programmazione e di installazione del software su di un dispositivo Linux Embedded, come la scheda FoxBoard.

A tale scopo, possiamo considerare la scheda come un computer in miniatura privo di mouse, tastiera e monitor, con un processore, della memoria e delle porte di comunicazione.

Come ogni computer ha bisogno che vi sia installato del software per poter svolgere compiti di una qualche utilità. L'installazione del sistema operativo e del software aggiuntivo viene di norma svolta con una metodologia nota col nome di flashing. Questa consiste nell'avviare il dispositivo in una particolare modalità nella quale esso si aspetta di ricevere dati sulla porta seriale o su quella di rete. Da un calcolatore, utilizzando un software opportuno, è possibile inviare al dispositivo in questo stato un file binario che esso salverà sovrascrivendo l'intera propria memoria ed utilizzerà successivamente per avviare il sistema operativo ed i programmi in esso salvati.

L'importanza della correttezza del file binario che viene inviato al dispositivo è massima. Per questo i kit di sviluppo per dispositivi embedded si preoccupano anche di creare questi file chiamati "immagine" o firmware, oltre che di compilare il codice del software che si vuole installare in modo che sia interpretabile dal processore del dispositivo.

La tabella che segue mostra i kit di sviluppo disponibili che supportano la FoxBoard.

	AcmeSystems SDK 2.01	Axis SDK 2.10	OpenWRT Head
Versione kernel Linux	2.4.31 o 2.6.15	2.6.19	2.6.19.2
Maintainer	John Crispin	John Crispin	Team OpenWRT

Tabella 3.1

Lo SDK (Software Development Kit) di AcmeSystems é basato sulla versione 2.01 del SDK realizzato da Axis, l'azienda produttrice del processore montato sulla FoxBoard. Rispetto alla versione di Axis, quella di AcmeSystems vanta l'integrazione di driver aggiuntivi per il supporto di alcuni dispositivi prodotti dalla stessa AcmeSystems, utilizzabili per ampliare le possibilità di utilizzo della FoxBoard.

Va notato che questo SDK offre però versioni del kernel Linux piuttosto vecchie (al momento della stesura di questa relazione il kernel Linux è disponibile in versione 2.4.35.1 e 2.6.22.6) e che non è nelle intenzioni di AcmeSystems aggiornare il kit di sviluppo con kernel più recenti.

Verranno comunque in seguito illustrate le modalità di installazione ed utilizzo del kit di sviluppo di AcmeSystems. Va detto che con tale distribuzione non è stato possibile portare a termine il lavoro, così come descritto nei primi due capitoli, a causa di incompatibilità tra il driver dell'adattatore wireless e il gestore del controller usb nei kernel Linux disponibili.

Il secondo kit di sviluppo è lo SDK 2.10 prodotto da Axis. Questa distribuzione permette di creare le immagini personalizzate per il flashing di vari dispositivi prodotti da Axis (principalmente telecamere di rete, videoserver per videosorveglianza e server di stampa).

Il kernel più recente offre prestazioni e funzionalità migliori, ma il lavoro per utilizzare questo SDK per i nostri scopi è appesantito dall'installazione dei driver per webcam e adattatore wireless, non presenti di default nella distribuzione.

Anche per questo kit verranno illustrate le modalità di installazione ed utilizzo. Purtroppo, come nel precedente kit, dei bug nella gestione del controllore usb non hanno permesso il compimento del progetto.

L'ultimo kit di sviluppo è ben noto nel mondo Linux, e dell'Open-source in generale, in quanto utilizzato ormai da tempo su moltissimi dispositivi embedded, in particolare su apparati di rete, print server ed embedded web server.

Il progetto OpenWRT [8] é nato nel Gennaio 2004 dopo il rilascio, sotto licenza GPL da parte di Linksys (divisione di Cisco Systems), dei sorgenti del firmware utilizzato nei router wireless WRT54G. Da allora molto lavoro è stato svolto dalla comunità di sviluppo di OpenWRT, tanto che oggi viene considerata una distribuzione Linux a tutti gli effetti, che supporta un gran numero di router normalmente in commercio e molti dispositivi embedded.

Dopo il rilascio dell'ultima versione stabile di OpenWRT (la 7.06 soprannominata Kamikaze) John Crispin, maintainer dei kit di sviluppo per la scheda FoxBoard, é entrato a far parte del team di sviluppo del progetto ed ha iniziato il porting del supporto per le schede basate su processori Axis all'interno del kit di sviluppo OpenWRT.

L'installazione e l'utilizzo di questa distribuzione verranno discussi in seguito, ma vale la pena di notare fin da ora che, al momento della stesura di questo documento, il supporto per la scheda FoxBoard, offerto da OpenWRT, è ancora in fase di test e non è stato ufficialmente annunciato dai suoi produttori (AcmeSystems).

La documentazione di AcmeSystems specifica espressamente di utilizzare lo SDK 2.01 se si ha intenzione di adottare una D-Link DWL-G122.

Oltre all'installazione ed all'utilizzo di un opportuno SDK il lavoro per la realizzazione del progetto è suddividibile in quattro obiettivi principali a cui vengono dedicate le sezioni seguenti:

- trasmissione bidirezionale su socket TCP del flusso dati della porta seriale;
- messa in funzione della webcam tramite installazione di opportuni driver;
- troncamento delle funzionalità offerte dalla libreria vislib tra il livello di accesso al device e quello di elaborazione dati con installazione della parte di accesso al device sulla scheda FoxBoard;
- messa in funzione della connettività wireless tramite adattatore usb D-Link DWL-G122 (rev. C1) grazie all'installazione di opportuni driver.

3.1. Remotizzazione della porta seriale

Per rendere accessibile tramite socket TCP la porta seriale è stato scelto il software Remserial [4].

Il programma Remserial funziona sostanzialmente da ponte tra una porta di rete TCP/IP e un device Linux a caratteri, come ad esempio una porta seriale.

Il programma può funzionare in due distinte modalità: come server, accettando connessioni di rete da altre macchine, oppure come client, connettendosi alla macchina remota dove sta girando il programma Remserial o qualche altro programma che accetta una connessione di rete "raw" (grezza). Sul canale TCP di rete, i dati fluiscono così come sono, dato che Remserial non svolge alcun particolare controllo.

Copie multiple del programma Remserial possono essere avviate su una singola macchina, a patto che utilizzino porte TCP e device a caratteri differenti.

L'utilizzo che si farà di Remserial consiste, quindi, nella compilazione, installazione ed avvio sulla FoxBoard in modalità server, in modo da rendere accessibile la porta seriale tramite una porta TCP.

Remserial sarà invece installato ed avviato in modalità client sul pc remoto, dove risiedono le librerie Aria ed i programmi per l'azionamento dei robot.

Remserial supporta le seguenti opzioni:

```
remserial [-r machinename] [-p netport] [-s "stty params"] device

-r machinename          The remote machine name to connect to.  If not
                    specified, then this is the server side.
-p netport              Specifiy IP port# (default 23000)
-s "stty params" If serial port, specify stty parameters, see man stty
-d                      Run as daemon programs
-x debuglevel           Set debug level, 0 is default, 1,2 give more info
-l linkname             If the device is /dev/ptmx, creates a symbolic link
                        to the corresponding slave pseudo-tty so that another
                    application has a static device name to use.
device                  Character oriented device node such as /dev/ttyS0.
```

Il comando per avviarlo sulla FoxBoard sarà quindi:

```
remserial -d -p <IP_Port> -s "9600 raw" /dev/ttyS0 &
```

Sul pc remoto invece verrà avviato come segue:

```
remserial -d -r <FoxBoard_Name_or_IP> -p <IP_Port> -s "9600 raw" -l
/dev/remserial /dev/ptmx &
```

Una volta collegata la FoxBoard alla porta seriale del robot ed avviato così come specificato, sarà possibile utilizzare il device virtuale /dev/remserial come se vi fosse collegata direttamente la porta seriale del robot.

3.2. Messa in funzione della webcam

La webcam utilizzata per il progetto é una Philips PCVC840 (ToUCam II). È possibile utilizzare tale webcam in ambiente Linux installando il driver PWC [5]. Il driver in questione é disponibile nella distribuzione ufficiale del kernel Linux (per versione 2.6.x), ma la versione inclusa nel kernel 2.6.15 (disponibile nel SDK 2.01) non funziona correttamente ed è quindi necessario installare tale driver come modulo a se stante, compilandolo da una versione aggiornata dei sorgenti.

3.3. Troncamento della VisLib

La libreria opensource VisLib [6] è sviluppata da MobileRobots. Tale libreria permette di acquisire immagini da un dispositivo video (utilizzando l'interfaccia video4Linux) e visualizzarle in finestra (utilizzando le librerie X11) oppure salvandole come file, anche dopo aver eseguito eventuali analisi e rielaborazioni.

La versione originale disponibile da MobileRobots della VisLib è numerata 1.8. La versione utilizzata per questo lavoro è invece la 1.9, che deriva da ulteriori modifiche sul codice svolte dal gruppo ARLBS.

Dall'analisi del codice della libreria si evince che non è possibile utilizzarla remotamente tramite uno strumento quale remserial. Nonostante il device utilizzato sia a caratteri, la libreria fa uso di alcune ioctl, non trasportabili con remserial, per impostare risoluzione, frame rate ed altre caratteristiche di acquisizione sulla webcam.

Per questo motivo, si è scelto di dividere la libreria in due stub: uno lato server, che si occupi della sola acquisizione delle immagini, ed uno lato client, che si occupi, dapprima della modifica dello spazio di colore utilizzato (da YUV a RGB), e poi della sua elaborazione o visualizzazione. I due spezzoni comunicheranno tramite socket TCP.

Lo stub lato server diverrà, di fatto, un programma avviabile che sarà compilato grazie al SDK, affinché sia eseguibile dalla FoxBoard. Lo stub lato client rimarrà, invece, in forma di libreria, ma non effettuerà più accesso ad un device locale, bensì si connetterà tramite socket al server e utilizzerà un semplice ed opportuno protocollo per remotizzare le ioctl.

3.4. Messa in funzione della connettività wireless

Vista l'esigenza di usufruire di una connessione wireless, é stato scelto appositamente l'adattatore D-Link DWL-G122 (rev. C1), poiché supportato da AcmeSystems. Come per la webcam, anche per questo dispositivo è necessario compilare gli opportuni driver, inclusi nello SDK distribuito da AcmeSystems. Inoltre, é necessario compilare ed installare sulla scheda il pacchetto wireless-tools, necessario per la configurazione dell'interfaccia di rete wireless.

Purtroppo, è proprio il supporto di questo dispositivo che non ha permesso il completamento del lavoro, così come progettato.

Infatti, nonostante fosse reclamato come supportato, si è scoperto, durante il lavoro, che il driver del controllore usb e quello dell'adattatore wireless non riescono a gestire il flusso di dati derivante dallo streaming video non compresso.

Questo problema, taciuto sul sito ufficiale di AcmeSystems, era ben noto ai produttori e vi era menzione solo nei forum e nella mailing list di supporto al prodotto.

Il problema inizialmente ci era stato detto essere legato esclusivamente ai kernel 2.4 e 2.6.15 (quindi alla versione 2.01 del SDK).

I test sulla versione 2.10 ha, però, dimostrato che anche questa non è immune al medesimo problema.

4. Modalità operative

Nei prossimi paragrafi viene documentato tecnicamente il lavoro svolto per la realizzazione del progetto.

Dapprima verrà spiegato come installare ed utilizzare i vari SDK disponibili. Quindi, si adranno ad analizzare i dettagli relativi all'integrazione dei singoli software necessari per il progetto: Remserial, VisLib, driver PWC e driver rt73.

4.1. SDK

4.1.1. SDK 2.01

4.1.1.1 Modalità di installazione

In linea teorica, é possibile installare lo SDK di AcmeSystems su qualsiasi pc ove possa girare una versione di Linux oppure Windows XP.

La procedura, qui riportata, fa riferimento esclusivo all'ambiente Linux con architettura i386.

Per poter utilizzare il kit di sviluppo è necessario installare dapprima i seguenti software:

GNU gcc, GNU make, GNU wget, Subversion, Awk o GNU awk, Bc, Byacc o Yacc, Lex o Flex, Bison, Perl, Sed. Tar, Zlib, Md5sum, Ncurses, Which, Pmake.

Tutti questi sono comodamente installabili con gli strumenti di gestione pacchetti della distribuzione in uso. Nel caso dei computer del laboratorio, basati su debian, si è utilizzato il tool apt con il seguente comando (da utente con privilegi amministrativi):

```
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
```

Confermando quando viene richiesto se si vogliono installare tutte le dipendenze, vengono scaricati tutti i pacchetti ed installati nel sistema.

È necessario quindi installare il compilatore che verrà utilizzato da questo, così come dagli altri SDK, per creare codice macchina eseguibile dal processore della FoxBoard.

Il compilatore in questione è stato realizzato da John Crispin ed è scaricabile sia in forma di sorgente che in forma pacchettizzata, per le principali distribuzioni Linux, presso il sito di sviluppo [9] di Axis.

Per installarlo sul pc del laboratorio è sufficiente eseguire i seguenti comandi (dpkg va eseguito con i privilegi amministrativi):

```
wget http://developer.axis.com/download/compiler/cris-dist_1.63-1_i386.deb
dpkg –i cris-dist_1.63-1_i386.deb
```

A questo punto è possibile procedere con l'installazione del SDK con i seguenti comandi:

```
wget http://www.AcmeSystems.it/download/install_svn_SDK.sh
chmod +x install_svn_SDK.sh
./install_svn_SDK.sh
```

Una volta completata l'esecuzione il kit di sviluppo sarà installato nella subdirectory devboard-R2_01.

4.1.1.2 Configurazione e compilazione del firmware

La fase di configurazione permette di determinare quali programmi si vuole vengano inseriti nel firmware.

Per accedere al menu di configurazione basta digitare il comando:

```
make menuconfig
```

Il menu propone di scegliere la tipologia di prodotto di destinazione, nel caso della scheda utilizzata in questo progetto la scelta corretta è FOXBOARD LX416.

Le scelte nei menu successivi dipendono direttamente da quali siano le risorse, i programmi e le caratteristiche del kernel che si vuole utilizzare nel firmware che verrà generato.

L'ultima configurazione utilizzata per i test del progetto vede selezionate le seguenti caratteristiche:

- Linux Kernel: Linux 2.6.x
- Standard C Library: uClibc
- Driver Settings:
    - Enable WLAN Support
    - Wireless Networking:
        - Wireless tools
        - DLink DWL-G122 (C1)
    - Enable webcam support
    - Webcam tools – Michel Xharad (servfox & co)
- Network Settings:
    - WLAN Settings:
        - (Robotica) ESSID
        - (5) Channel
        - WLAN mode (Managed)

        - WEP key (None)
        - (192.0.2.12) IP
        - (255.255.255.0) Subnet
        - Use WLAN as default route
        - (192.0.2.1) Gateway
    - Nameserver Settings:
        - Use custom nameserver
        - (194.25.2.129) Nameserver
    - MAC-Address
        - Set a MAC
        - (00:40:8C:00:00:00) MAC
- Applications:
    - Networking:
        - Enable web server
        - Enable OpenSSL 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
- Use pregenerated ssh keys

Scegliendo quindi EXIT e specificando YES quando viene chiesto se salvare la procedura di configurazione è terminata e si è pronti a compilare il sistema operativo ed il software che verrà installato sulla FoxBoard.

Ora lo SDK è pronto per creare il firmware. È sufficiente richiamare i seguenti comandi ed attendere il completamento.

```
./configure
make
```

Al termine dell'operazione viene creata l'immagine firmware utilizzabile per flashare la FoxBoard.

4.1.1.3 Flashing

Per installare l'immagine sulla FoxBoard il metodo migliore è quello di utilizzare lo script boot_linux presente nel SDK per rendere il pc sul quale lo si esegue un network boot server.

Poiché dopo la compilazione di un firmware questo sarà salvato nella directory principale del SDK è sufficiente dare il comando seguente (da utente con privilegi amministrativi) per attivare la modalità di network boot:

```
./boot_linux –d eth0 –F –i fimage
```

In tale comando, *–d eth0* specifica il device di rete da utilizzare per il flashing (eth0 sotto Linux per la prima interfaccia di rete), *-F* specifica di sovrascrivere l'intero contenuto della memoria a stato solido della scheda, *-f* invece permette di sovrascrivere solo il contenuto di sistema salvaguardando i file di configurazione in /etc; infine *–i fimage* specifica quale sia il file binario da utilizzare per l'operazione.

A questo punto, è necessario avviare o riavviare la FoxBoard avendo cura di porre un jumper a cavallo dei pin indicati sulla scheda con la sigla J8. In questo modo sarà una porzione di codice, residente nella memoria di sola lettura all'interno del processore Axis, ad occuparsi di salvare in cache i dati che vengono inviati in broadcast dal pc e quindi darli in pasto al processore permettendo l'avvio ed il salvataggio del nuovo firmware.

Una volta completata l'operazione la FoxBoard dovrebbe essere raggiungibile via ssh o telnet, a seconda di come la si è configurata al passo precedente. L'username abilitato di default è *root* con password *pass*.

4.1.1.4 Cross-Compilazione e installazione software di terze parti

La libreria VisLib, il programma Remserial e, nel caso di questo SDK, il driver PWC non sono inclusi nella distribuzione software di default del kit di sviluppo. Per questo motivo, è necessario scrivere opportuni Makefile in modo che la compilazione sia efficace per l'utilizzo sul processore Axis.

La documentazione AcmeSystems consiglia di copiare l'albero sorgente del software che si vuole compilare per la FoxBoard, nella sottodirectory "apps" del kit di sviluppo e quindi di modificare o creare un nuovo Makefile che richiami le definizioni presenti nel SDK.

Il Makefile che segue è l'esempio utilizzabile per cross-compilare la tipica applicazione helloworld. Si notino in particolare le prime due righe con le quali viene specificato quali siano le librerie C utilizzabili e richiamato un ulteriore Makefile (Rules.axis), ove sono impostate le variabili e le regole necessarie alla corretta compilazione.

```
AXIS_USABLE_LIBS = UCLIBC GLIBC
include $(AXIS_TOP_DIR)/tools/build/Rules.axis

PROGS = helloworld

all: $(PROGS)
$(PROGS): $(PROGS).o
  $(CC) $(LDFLAGS) $^ $(LDLIBS) -o $@

clean:
  rm -f $(PROGS) *.o core
```

Una volta realizzato un file di make adeguato al proprio programma, è sufficiente dare i seguenti comandi per cross-compilare il codice:

```
. init_env
cd apps/directory_software
make cris-axis-Linux-gnuuclibc
make
```

Per installare il file binario sulla FoxBoard è sufficiente copiarlo con scp, ponendolo nella porzione scrivibile della memoria con il seguente comando:

```
scp nomeprogramma root@FoxBoard_IP:/mnt/flash
```

Infine collegandosi tramite ssh alla FoxBoard, sarà possibile eseguire il programma come un qualsiasi altro specificandone il percorso completo oppure inserendo /mnt/flash nella variabile d'ambiente PATH.

Nelle sezioni 4.2, 4.3 e 4.4 e 4.5, dove vengono trattate nello specifico le installazioni e le modifiche di remserial, driver per la webcam, vislib e driver per l'adattatore wireless, verranno riportati i Makefile necessari per cross-compilarli.

4.1.2. SDK 2.10

4.1.2.1 Modalità di installazione

I prerequisiti per installare la versione 2.10 del kit di sviluppo sono gli stessi già elencati per la versione 2.01.

Una volta installati tutti gli strumenti di sistema, si può procedere a scaricare dal sito Axis i due file necessari:

- http://www.axis.com/techsup/dl.php?prodid=1587&url=../ftp/pub/axis/dev/soft_dist/R2_10/devboard-R2_10.tar.gz
- http://www.axis.com/techsup/dl.php?prodid=1587&url=../ftp/pub/axis/dev/soft_dist/R2_10/devboard-R2_10-distfiles.tar.gz

Una volta scaricati e decompressi questi file, verranno create la directory *devboard-R2_10* e *distfiles*. Viene richiesto di creare un link simbolico che punta alla directory *distfiles* all'interno della directory *devboard-R2_10*. Per fare tutto ciò è sufficiente eseguire i seguenti comandi:

```
tar xvfz devboard-R2_10.tar.gz
tar xvfz devboard-R2_10-distfiles.tar.gz
cd devboard-R2_10
ln -s ../distfiles
```

A questo punto, poiché questo SDK supporta l'intera gamma di prodotti di Axis, è necessario eseguire un primo comando di configurazione che si occuperà di impostare il kit di sviluppo, in maniera tale da essere adatto alla compilazione per un prodotto specifico. Nel caso della FoxBoard il comando da dare è il seguente:

```
DEV_BOARD_PRODUCT=fox416 ./configure
```

Svolte queste operazioni lo SDK è pronto per essere utilizzato.

4.1.2.2 Configurazione

Come per la versione 2.01 lo SDK 2.10 è configurabile semplicemente invocando il comando *make menuconfig*. Il menu contiene molte voci aggiuntive rispetto a quelle della versione 2.01, questo è dovuto al fatto che tale versione è stata modificata dal team di AcmeSystems e personalizzata per il proprio prodotto.

Anche per la versione 2.10 sono state testate varie configurazioni e viene riportata di seguito quella con la quale si sono ottenuti i risultati migliori (vengono riportate solo le modifiche rispetto alla configurazione di default proposta o le voci che hanno importanza particolare per il risultato)

- Product Information
  - Product Type (Developer Board)
- Hardware Configuration
  - Processor (ETRAX 100LX)
- General Configuration
  - Linux kernel (Linux 2.6.x)
  - Shell (ash (BusyBox))
  - Start a console shell
  - Enable serial test utilità
  - Enable EasyEdit support
  - Create SCSI device nodes
  - Create TTYUSB device nodes

- Network Configuration
  - Network Hardware Configuration
    - Enable Ethernet support
    - (1) Network interfaces
    - (eth0) Device name for interface 1
  - Network Protocol Configuration
    - Enable web server
    - Enable SSH support
    - Enable TELNETD support
- Web Configuration
  - Sample web pages (Sample web pages for Acme Systems FOX board)
- Libraries Configuration
  - Standard C library (uClibc)

Una volta specificata e salvata la configurazione, è possibile compilare l'immagine firmware per la FoxBoard con i seguenti comandi:

```
./configure
make
```

4.1.2.3 Flashing

Il metodo per installare il firmware sulla scheda è analogo a quello spiegato per il SDK 2.01 e si rimanda a tale descrizione per la procedura.

L'unica differenza da notare è che mentre il firmware viene sempre salvato con nome *fimage* nella directory *devboard-R2_10*, lo script *boot_linux* risiede ora nella directory *devboard-R2_10/tools/build/bin* che viene inserita nella variabile d'ambiente *PATH* quanto si richiama il seguente comando:

```
. init_env
```

Per questo motivo lo script *boot_linux* va invocato senza il *./* prefisso.

4.1.2.4 Cross-Compilazione e installazione software di terze parti

Il metodo di cross-compilazione con l'utilizzo del SDK 2.10 é sostanzialmente identico a quello spiegato per lo SDK 2.01. Particolare nota va fatta alle novità del Makefile (viene riportato il file per la compilazione di un generico programma helloworld scritto in C).

```
AXIS_USABLE_LIBS = GLIBC UCLIBC
include $(AXIS_TOP_DIR)/tools/build/rules/common.mak

PROGS     = hello
INSTDIR   = $(prefix)/bin/
INSTMODE  = 0755
INSTOWNER = root
INSTGROUP = root

OBJS      = hello.o

all: $(PROGS)

$(PROGS): $(OBJS)
        $(CC) $(LDFLAGS) $^ $(LDLIBS) -o $@

install: $(PROGS)
```

```
        $(INSTALL) -d $(INSTDIR)
        $(INSTALL) -m $(INSTMODE) -o $(INSTOWNER) -g $(INSTGROUP) $(PROGS)
$(INSTDIR)

clean:
        rm -f $(PROGS) *.o core
```

Le differenze rispetto alla versione 2.01 stanno nel nuovo file da includere (*common.mak*, non più *Rules.axis*) e nella possibilità di utilizzare il target install che permette di specificare quali file del software vanno installati e dove è necessario che vengano salvati nel firmware che sarà utilizzato sul dispositivo.

Per fare in modo che il software venga inserito nell'immagine firmware è necessario, oltre che specificare correttamente il target *install* nel Makefile, aggiungere una riga nel file *configure-files/post* dove viene riportato il nome della directory dove risiedono il Makefile ed il codice sorgente del software in questione. Per esempio per inserire nella distribuzione il programma helloworld, è necessario dalla directory principale del SDK digitare i seguenti comandi:

```
echo sub apps/helloworld >> configure-files/post
make
```

Seguendo la procedura appena descritta, una volta eseguito il flashing del dispositivo, sarà possibile richiamare il software installato senza doverlo copiare (via scp o ftp) successivamente. Nel caso del programma helloworld sarà sufficiente collegarsi al dispositivo ed eseguire il comando hello:

```
ssh root@192.168.0.90
hello
```

4.1.2.5 Configurazione del kernel

Poiché il menu di configurazione illustrato nel paragrafo 4.1.2.2 non menziona i moduli kernel necessari per il funzionamento dei dispositivi esterni richiesti da questo progetto, è stato necessario individuare un modo per aggirare tale configurazione e fare in modo che venissero compilati.

Questa tecnica non é documentata, né ufficialmente, né su alcun forum o newsletter.

Il menu di configurazione del kernel Linux è accessibile eseguendo i seguenti comandi:

```
. init_env
cd os/Linux-2.6
make menuconfig
```

Per fare poi in modo che le scelte effettuate siano rispecchiate dall'effettiva compilazione del firmware, è necessario riportare il file di configurazione nascosto che *make menuconfig* crea prima di terminare nella directory, dove gli script di compilazione del firmware si aspettano che sia. È possibile svolgere questa procedura e ricreare il firmware con i seguenti comandi:

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

4.1.2.6 Creazione di un device node

Durante la realizzazione dell'elaborato è emerso che, nonostante fossero stati compilati i driver per la webcam Philips, così come spiegato nel paragrafo precedente, era impossibile accedervi, poiché il firmware era privo del device node necessario (*/dev/video0*).

Per fare in modo che questo venisse creato, è stata trovata soluzione nella modifica del file *packages/devices/axis-2.4-R1_0_10/Makefile*. Per effettuare questa operazione è sufficiente eseguire il seguente comando:

```
echo '$(MKNOD) -m 0666 $(DEV)/video0 c 81 0' >> packages/devices/axis-2.4-
R1_0_10/Makefile
```

4.1.3. OpenWRT HEAD

4.1.3.1 Modalità di installazione

I prerequisiti per l'installazione del sistema OpenWRT sono i medesimi degli SDK sviluppati da Axis e AcmeSystems.

Per procedere all'installazione è sufficiente creare una directory ed eseguire il checkout del codice dal repository subversion di sviluppo del progetto:

```
mkdir openwrt
cd openwrt
svn co https://svn.openwrt.org/openwrt/trunk
```

Per ottenere anche un gran numero di pacchetti già preparati dal team OpenWRT per essere compilati con lo SDK, è possibile eseguire anche il checkout del modulo package.

```
svn co https://svn.openwrt.org/openwrt/package
```

4.1.3.2 Configurazione

Al pari dei SDK precedenti, anche OpenWRT necessita di configurazione, al fine di determinare non solo quali software compilare ed inserire nella distribuzione firmware, ma anche quale sia l'architettura hardware target dell'installazione.

Questo perché con OpenWRT è possibile creare firmware con distribuzioni personalizzate di Linux per oltre dieci diverse architetture, principalmente utilizzate in apparati di rete destinati all'home networking o alle piccole e medie imprese.

Per accedere al menu di configurazione è sufficiente, come per gli altri SDK, eseguire *make menuconfig* all'interno della directory trunk. Per ottenere un firmware utilizzabile sulla piattaforma FoxBoard, sarà necessario selezionare dal menu l'architettura *Axis ETRAX*. Di particolare importanza è la scelta del filesystem da utilizzare per il firmware: in particolare è stato verificato il corretto funzionamento di immagini con filesystem *jffs2* e *squashfs*.

Una volta salvata una configurazione, è possibile creare l'immagine con il solito comando *make*.

Le informazioni riportate per questo SDK sono destinate ad essere obsolete nel breve termine, poiché l'unica versione di OpenWRT che include il supporto per la piattaforma ETRAX è la versione di sviluppo.

Al momento della stesura di questa relazione (svn rev. 8686) vi sono alcuni problemi che non permettono la corretta creazione del firmware o il suo corretto funzionamento sulla FoxBoard.

Il primo problema sta nella erronea configurazione di default del sistema OpenWRT quando viene selezionata l'architettura Axis ETRAX. La configurazione include la compilazione del tool di firewalling iptables, ma non vengono invece compilati i moduli del kernel necessari al suo corretto funzionamento. Questo comporta la non raggiungibilità della FoxBoard qualora venga flashata con un firmware OpenWRT compilato con una configurazione standard.

Un semplice workaround per questo problema è deselezionare il pacchetto iptables dal menu di configurazione, oppure selezionare tutti i moduli del kernel necessari.

Un secondo problema comporta la creazione di immagini firmware erronee e quindi non avviabili sulla FoxBoard. In fase di creazione dell'immagine, gli script di OpenWRT non trovano un eseguibile fondamentale (mkfimage), poiché non risiede in una delle directory specificate nella variabile d'ambiente PATH. Per ovviare a questo problema, si è trovata soluzione nella modifica del file *target/Linux/etrax/image/Makefile*. Per replicare la modifica è necessario cambiare la riga

```
mkfimage $(KDIR)/vmlinuz_$$$$f $(KDIR)/vmlinuz_$$$$f.tmp ; \
```

con:

```
$(STAGING_DIR)/bin/mkfimage $(KDIR)/vmlinuz_$$$$f $(KDIR)/vmlinuz_$$$$f.tmp ; \
```

Un altro bug presente tra i file di configurazione realizzati da John Crispin per il supporto ETRAX di OpenWRT è il non funzionamento dei parametri di configurazione relativi agli indirizzi di rete da assegnare alle interfacce del dispositivo target.

Se l'IP di default 192.168.0.90 va bene per l'applicazione che si intende realizzare, non c'è bisogno di attuare alcuna modifica. Altrimenti il modo migliore è modificare manualmente il file *target/Linux/etrax/base-files/default/etc/config/network*. Il contenuto del file di default viene di seguito riportato:

```
# Copyright (C) 2006 OpenWrt.org

config interface loopback
        option ifname   lo
        option proto    static
        option ipaddr   127.0.0.1
        option netmask  255.0.0.0

config interface lan
        option ifname   eth0
        option proto    static
        option ipaddr   192.168.0.90
        option netmask  255.255.255.0
        #option dns      192.168.0.1
        #option gateway  192.168.0.1
```

4.1.3.3 Flashing

Nella documentazione di OpenWRT non c'è menzione di dove vengano create le immagini firmware e come sia possibile installarle sul dispositivo target per quanto riguarda l'architettura Axis ETRAX.

Analizzando a fondo i sorgenti del kit, è stato trovato che le immagini vengono create nella directory *bin,* dove viene inoltre copiato lo script *boot_linux* per eseguire la procedura di flashing così come spiegata per gli SDK precedentemente trattati.

Se con gli altri SDK poteva essere di poca importanza l'opzione *–i* dello script *boot_linux,* con OpenWRT è fondamentale poiché non viene salvato alcun file dal nome *fimage*. L'elenco dei file che invece vengono creati nella directory *bin* è invece il seguente:

```
boot_linux
etrax100boot
openwrt-etrax-2.6-jffs2-64k-fimage_416
openwrt-etrax-2.6-jffs2-64k-fimage_816
openwrt-etrax-2.6-jffs2-64k-fimage_832
openwrt-etrax-2.6-jffs2-64k-fimage_custom
openwrt-etrax-2.6-jffs2-64k-fimage_MCM
openwrt-etrax-2.6-squashfs-fimage_416
openwrt-etrax-2.6-squashfs-fimage_816
openwrt-etrax-2.6-squashfs-fimage_832
openwrt-etrax-2.6-squashfs-fimage_custom
openwrt-etrax-2.6-squashfs-fimage_MCM
```

Poiché la scheda FoxBoard in dotazione per questo progetto ha 4Mb di memoria utilizzabile per il flashing le immagini che è possibile utilizzare sono le seguenti:

```
openwrt-etrax-2.6-jffs2-64k-fimage_416
openwrt-etrax-2.6-squashfs-fimage_416
```

4.1.3.4 Cross-compilazione e installazione software di terze parti

Il kit OpenWRT consta di un gran numero di pacchetti già pronti per essere compilati ed installati per il dispositivo target. Tali pacchetti si trovano categorizzati all'interno del modulo package, di cui è stato spiegato come eseguire il checkout da repository subversion nel paragrafo 4.1.3.1.

Per attivare uno di questi pacchetti è sufficiente creare un link simbolico all'interno della directory package del SDK (trunk/package) che punti alla directory del pacchetto in questione.

Per esempio se si vuole attivare lo spool di stampa cups si può creare il link come segue:

```
ln -s ../package/net-print/cups package/cups
```

Alla successiva chiamata di *make menuconfig,* cups risulterà tra i pacchetti installabili e, se selezionato, verrà inserito nel firmware al successivo *make*.

Se il software non esiste tra i pacchetti già preparati dal team OpenWRT, è possibile crearne uno nuovo. Per farlo è sufficiente aggiungere una directory all'interno di *package* e inserirvi un Makefile che specifichi al sistema OpenWRT dove scaricare i sorgenti, dove e come compilarli ed installarli.

È possibile anche creare pacchetti per moduli aggiuntivi del kernel e vi sono molti pacchetti d'esempio per scrivere il Makefile in modo corretto, oltre ad una completa documentazione presso il wiki di OpenWRT [10].

4.2. Remserial

L'installazione di Remserial è possibile mediante cross-compilazione ed installazione da effettuarsi con le metodologie riportate nei paragrafi relativi agli SDK. Vengono di seguito riportati i Makefile relativi ad ognuno degli SDK utilizzati durante il progetto.

SDK 2.01:

```
AXIS_USABLE_LIBS = UCLIBC GLIBC
include $(AXIS_TOP_DIR)/tools/build/Rules.axis

all: remserial

REMOBJ=remserial.o stty.o
remserial: $(REMOBJ)
        $(CC) $(LDFLAGS) -o remserial $(REMOBJ)

clean:
        rm -f remserial *.o
```

SDK 2.10

```
AXIS_USABLE_LIBS = UCLIBC GLIBC
include $(AXIS_TOP_DIR)/tools/build/rules/common.mak

PROGS     = remserial
INSTDIR   = $(prefix)/bin/
INSTMODE  = 0755
INSTOWNER = root
INSTGROUP = root

OBJS=remserial.o stty.o

all: $(PROGS)

$(PROGS): $(OBJS)
        $(CC) $(LDFLAGS) $^ $(LDLIBS) -o $@

install: $(PROGS)
        $(INSTALL) -d $(INSTDIR)
        $(INSTALL) -m $(INSTMODE) -o $(INSTOWNER) -g $(INSTGROUP) $(PROGS)
$(INSTDIR)

clean:
        rm -f $(PROGS) *.o core
```

OpenWRT

```
include $(TOPDIR)/rules.mk

PKG_NAME:=remserial
PKG_VERSION:=1
PKG_RELEASE:=1

PKG_SOURCE=$(PKG_NAME).tar.gz
```

```
PKG_SOURCE_URL=http://lpccomp.bc.ca/remserial
PKG_MD5SUM:=45da28bd2f551b9102a9ec0d62d92424
TAR_OPTIONS += -C $(BUILD_DIR)/remserial

PKG_BUILD_DIR:=$(BUILD_DIR)/remserial

include $(INCLUDE_DIR)/package.mk

define Package/remserial
  SECTION:=multimedia
  CATEGORY:=Multimedia
  TITLE:=Remserial
  DESCRIPTION:=Remserial communication bridge between TCP/IP and char devices.
  URL:=http://lpccomp.bc.ca/remserial/
endef

define Package/remserial/install
        $(INSTALL_DIR) $(1)/usr/bin
        $(INSTALL_BIN) $(PKG_BUILD_DIR)/remserial $(1)/usr/bin/
endef

$(eval $(call BuildPackage,remserial))
```

Va notato che nel caso degli SDK 2.01 e 2.10, il Makefile qui riportato deve essere utilizzato in alternativa a quello fornito nel pacchetto Remserial. Il Makefile originale deve essere utilizzato per creare la versione di Remserial da utilizzare sui computer ove si installano anche le librerie Aria per il controllo del robot.

Nel caso di OpenWRT invece, il Makefile riportato va salvato nella directory *package/remserial*. Il Makefile in questione non è sostitutivo di quello del pacchetto remserial.

Una volta salvato il Makefile per OpenWRT e lanciato il comando *make menuconfig,* sarà possibile selezionare la nuova opzione *remserial* dal menu Multimedia. Al successivo *make,* gli script di OpenWRT, interpretando il Makefile qui riportato, si occuperanno di scaricare dalla rete Internet il pacchetto sorgente di Remserial, scomprimerlo in una posizione conveniente, cross-compilarlo ed installarlo nell'immagine firmware.

4.3. Supporto webcam Philips

Il modulo del kernel *pwc,* che supporta la webcam Philips, è presente e funziona correttamente sia nel SDK 2.10 che in OpenWRT. Per inserirlo nel firmware è necessario seguire la procedura di configurazione kernel descritta nel paragrafo 4.1.2.5 se si lavora con lo SDK 2.10, è invece attivabile dal menu di configurazione di OpenWRT se si lavora con quest'ultima distribuzione.

Per quanto riguarda invece lo SDK 2.01, è necessario scaricare il codice sorgente del modulo ed estrarre il pacchetto in una sottodirectory di *apps*. A questo punto sarà possibile compilare il modulo ed installarlo seguendo la procedura descritta nel paragrafo 4.1.1.4 con il seguente codice di Makefile.

```
MODULE = pwc.o

pwc-y = pwc-if.o pwc-misc.o pwc-ctrl.o pwc-v4l.o pwc-uncompress.o pwc-dec1.o pwc-
dec23.o pwc-kiara.o pwc-timon.o

EXTRA_CFLAGS=-Wall -I$(PWD)

EXTRA_CFLAGS += -DNOKERNEL $(USER_OPT)

PREVENT_RECURSIVE_INCLUDE = 1
include $(AXIS_TOP_DIR)/modules/rules.build_modules
```

A prescindere dal SDK utilizzato, e quindi dalla versione del modulo *pwc*, il codice del driver, in fase di inizializzazione della webcam, comprende il comando *pwc_set_agc* che attiva il controllo automatico del guadagno. Per questo, l'immagine acquisita dopo l'attivazione del modulo è completamente nera e servono acquisizioni successive per portare il guadagno a regime.

La soluzione a questo problema è descritta nel Rapporto tecnico ARL-TR-07-03 [13].

4.4. Vislib

La libreria di acquisizione, analisi e visualizzazione immagini è stata modificata per essere efficace nonostante le scarse risorse del sistema target.

Come già descritto nel primo capitolo, è stato creato un programma stand-alone lato server, che si occupa solo dell'acquisizione video. Lato client invece la libreria è rimasta in tale forma con la sola differenza che non accede ad un device video4Linux, bensì ad un socket TCP per il recupero delle immagini.

I nuovi file grab-server.c e grab-client.c sono così andati a sostituire il grab.c della distribuzione di base.

Il Makefile è stato modificato per compilare esclusivamente il server (con target etrax) oppure esclusivamente la libreria client (con target host).

Viene riportata la parte di codice di interesse per questo punto:

```
DEPLOYMENT = client

# Deployment dependent make
ifeq (server,$(DEPLOYMENT))
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
SOURCES  = common.c             \
     convolution.c      \
     display.c          \
     filter.c           \
     format.c           \
     grab-client.c              \
     morph.c            \
     motion.c           \
     object.c           \
     utility.c          \
     blob.c
```

Impostando *server* o *client* nella variabile *DEPLOYMENT* definita all'inizio del codice è possibile selezionare quale sia la modalità di compilazione da adottare. È necessario selezionare *server* quando si tenta la compilazione con gli SDK al fine di ottenere lo stand-alone grab-server.

Per quanto riguarda la compilazione con OpenWRT della vislib è stato realizzato il seguente script Makefile:

```
include $(TOPDIR)/rules.mk

PKG_NAME:=vislib
PKG_VERSION:=1
PKG_RELEASE:=1

PKG_BUILD_DIR:=$(BUILD_DIR)/vislib

include $(INCLUDE_DIR)/package.mk

define Package/vislib
  SECTION:=multimedia
  CATEGORY:=Multimedia
  TITLE:=Server stub of Vislib
  DESCRIPTION:=\
        Manage ethernet bridging: a way to connect networks together to \\\
        form a larger network.
  URL:=http://bridge.sourceforge.net/
```

```
endef

define Build/Prepare
        mkdir -p $(PKG_BUILD_DIR)
        $(CP) ./files/* $(PKG_BUILD_DIR)/
endef


define Build/Compile
        $(TARGET_CC) -o $(PKG_BUILD_DIR)/src/grab-server $(TARGET_CCFLAGS) -
I$(PKG_BUILD_DIR)/include $(PKG_BUILD_DIR)/src/grab-server.c
endef

define Package/vislib/install
        $(INSTALL_DIR) $(1)/usr/bin
        $(INSTALL_BIN) $(PKG_BUILD_DIR)/src/grab-server $(1)/usr/bin/
endef

$(eval $(call BuildPackage,vislib))
```

A differenza dello script creato per Remserial, questo richiede che il codice sorgente della libreria vislib, modificato secondo quanto specificato fin qui, risieda nella directory *package/vislib/files*.

4.5. Supporto adattatore wireless D-Link

L'adattatore wireless usb prodotto da D-Link è supportato da Linux grazie a diverse versioni del driver rt73. Si riconoscono principalmente tre versioni, o branch, del codice di supporto a questa periferica: i driver ufficiali scritti dal produttore del chipset (RalinkTech) disponibili presso il loro sito [11] (ora non più in sviluppo attivo), i driver sviluppati dal team del progetto opensource Rt2x00 basati sul codice originale ralink ed infine i driver di nuova generazione sviluppati sempre dal team del progetto Rt2x00, completamente riscritti e basati sulle nuove API del kernel Linux mac80211 e ieee80211. Il lavoro del gruppo Rt2x00 è disponibile presso il loro sito ufficiale [13].

Sulla periferica FoxBoard sono state testate le prime due versioni del driver con ognuno dei tre SDK. La terza versione del driver non è stata testata poiché le API mac80211 sono state introdotte nel Kernel Linux a partire dalla versione 2.6.21, mentre il kernel più aggiornato disponibile negli SDK utilizzati è il 2.6.19.2 con OpenWRT.

Tra le due versioni testate quella migliorata dal team Rt2x00 sembra essere la più stabile, ma, in ogni caso, le incompatibilità tra il driver rt73 ed il driver del controller usb della FoxBoard fanno si che la wireless smetta di funzionare quando viene sottoposta ad un alto carico di trasmissione.

Mentre il driver risulta essere presente nel SDK 2.01, è necessario utilizzare il seguente Makefile per la compilazione con lo SDK 2.10.

```
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
```

Nel caso di OpenWRT é necessario creare un pacchetto con il seguente Makefile (*package/rt73/Makefile*):

```
include $(TOPDIR)/rules.mk
include $(INCLUDE_DIR)/kernel.mk

PKG_NAME:=rt73
PKG_VERSION:=cvs-daily
PKG_RELEASE:=1

PKG_SOURCE:=$(PKG_NAME)-$(PKG_VERSION).tar.gz
PKG_SOURCE_URL:=http://rt2x00.serialmonkey.com/
#PKG_MD5SUM:=a5f08e33cd4115129cbe811f697e0af1

PKG_BUILD_DIR:=$(KERNEL_BUILD_DIR)/$(PKG_NAME)-cvs
```

```
include $(INCLUDE_DIR)/package.mk

define KernelPackage/rt73
  SUBMENU:=Wireless Drivers
  TITLE:=Driver for ralink usb wireless chipsets
  DEPENDS:=@LINUX_2_6 @USB_SUPPORT +wireless-tools
  DESCRIPTION:= \
    This package contains a driver for ralink rt73 usb chipsets.
  URL:=http://rt2x00.serialmonkey.com/
  VERSION:=$(LINUX_VERSION)+$(PKG_VERSION)-$(BOARD)-$(PKG_RELEASE)
  FILES:= \
    $(PKG_BUILD_DIR)/Module/rt73.$(LINUX_KMOD_SUFFIX)
  AUTOLOAD:=$(call AutoLoad,50,rt73)
endef

define Package/rt73-firmware
        SECTION:=net
        CATEGORY:=Network
        TITLE:=rt73 wifi firmware
endef

define Build/Prepare
        $(call Build/Prepare/Default)
        $(CP) $(PKG_BUILD_DIR)-*/* $(PKG_BUILD_DIR)/
endef


define Build/Compile
    $(MAKE) -C $(LINUX_DIR) \
        ARCH="$(LINUX_KARCH)" \
        CROSS_COMPILE="$(TARGET_CROSS)" \
        PATCHLEVEL="$(LINUX_PATCHLEVEL)" \
        KERNDIR="$(LINUX_DIR)" \
        SUBDIRS="$(PKG_BUILD_DIR)/Module" \
        modules
endef

define Package/rt73-firmware/install
        $(INSTALL_DIR) $(1)/lib/firmware
        $(INSTALL_DATA) $(PKG_BUILD_DIR)/Module/rt73.bin $(1)/lib/firmware
endef

$(eval $(call BuildPackage,rt73-firmware))
$(eval $(call KernelPackage,rt73))
```

Il Makefile così realizzato crea due pacchetti attivabili via *make menuconfig*: quello relativo al modulo del kernel rt73 e quello relativo al firmware rt73.bin che viene utilizzato per il flashing dell'adattatore wireless quando viene messo in funzione.

La versione del driver utilizzata è lo snapshot del cvs notturno del progetto Rt2x00.

5. Conclusioni e sviluppi futuri

Il progetto ha dato buoni risultati. Dapprima sono state applicate le modifiche alla libreria VisLib e ne è stata verificata la funzionalità con immagini firmware prodotto con ognuno dei tre SDK. Lo stesso vale per le funzionalità offerte da Remserial.

L'unico problema ancora irrisolto é relativo al funzionamento della connettività Wireless sotto alto carico di dati. Forse la via per la soluzione sta nell'attesa di una maggiore stabilità del supporto da parte di OpenWRT della piattaforma AXIS Etrax.

Purtroppo a questo riguardo la collaborazione con John Crispin, maintainer dei due SDK "ufficiali" e maintainer dello sviluppo per piattaforma Etrax in OpenWRT, non è stata decisamente delle migliori. Alcuni dei problemi affrontati in queste pagine e dei fix trovati, sono stati riportati a Crispin affinché li introduca nelle prossime versioni.

Problemi individuati e risolti, relativi al SDK OpenWRT, non sono stati riportati a Crispin in quanto vi sono attualmente problemi di comunicazione con questo ultimo, a cui purtroppo non riesco a porre rimedio.

Bibliografia

[1] Sito web di MobileRobots: http://www.mobilerobots.com

[2] Sito web di AcmeSystems: http://www.AcmeSystems.it

[3] Sito web di Axis: http://www.axis.com

[4] Sito web di remserial: http://lpccomp.bc.ca/remserial/

[5] Sito web del driver pwc: http://www.saillard.org/Linux/pwc/

[6] Sito web della vislib: http://robots.mobilerobots.com/vislib/

[7] Sito web del driver rt73: http://rt2x00.serialmonkey.com/

[8] Sito web del progetto OpenWRT: http://OpenWRT.org/

[9] Repository del compilatore CRIS: http://developer.axis.com/download/compiler/

[10] Wiki del progetto OpenWRT: http://wiki.openwrt.org/

[11] Driver ufficiali Ralink: http://www.ralinktech.com/ralink/Home/Support/Linux.html

[12] Sito ufficiale progetto Rt2x00: http://rt2x00.serialmonkey.com/wiki/index.php?title=Main_Page

[13] R. Cassinis, Ricompilazione driver per webcam Philips, Rapporto tecnico ARL-TR-07-03, http://www.ing.unibs.it/~cassinis/docs/techreps/ARL-TR-07-03

Indice