Università di Padova  Facoltà di Ingegneria

Corso di Laurea in Ingegnera Informatica
Tesi di Laurea

AUDIOCITOFONIA REMOTA BASATA SUL PROTOCOLLO SIP

Relatore: **Chiar.mo Prof. Lorenzo Vangelista**

Laureando: **Fabio Montemaggiore**

Anno Accademico: 2009-2010

Indice

Capitolo 1

Introduzione

1.1 Introduzione

Le principali conquiste della moderna tecnologia informatica stanno rivoluzionando il nostro modo di vivere. L'incessante evoluzione tecnologica e la disponibilità di elettronica a basso costo hanno infatti portato all'utilizzo di un enorme numero di dispositivi elettronici nei vari settori e nei vari ambienti della nostra vita.

Questi sistemi vengono definiti **sistemi embedded**. Ossia sistemi di calcolo con una forte integrazione tra hardware e software e progettati per garantire ottime prestazioni nello svolgimento di un compito specifico. La parola inglese "embedded" significa "incluso" e riflette il fatto che questi sistemi sono spesso una parte integrante di un sistema molto più grande che, incorporato in qualche dispositivo viene chiamato anch'esso, sistema embedded.

I sistemi embedded si stanno velocemente evolvendo verso architetture hardware sempre più complesse in grado di ospitare una varietà di applicativi software fino a poco tempo fa impensabili con lo scopo di migliorare la qualità della vita, da un lato automatizzando in modo intelligente ed affidabile compiti critici e dall'altro fornendo funzionalità sempre più ricche per l'utente.

Senza rendercene quasi conto ne siamo costantemente circondati, li utilizziamo giornalmente e a molti di loro deleghiamo perfino il controllo delle nostre azioni; basti pensare al sistema della frenata di un'automobile (meglio conosciuto come ABS) che, sostituendosi ai nostri comandi, garantisce una guida più sicura in quei momenti in cui l'istinto umano potrebbe fallire.

Anche le comunicazioni via Internet hanno avuto un grosso sviluppo in questi ultimi anni. La tecnologia **VOIP** ha aperto e sta aprendo grosse opportunità di business per ciò che riguarda sia gli operatori di telecomunicazioni già presenti sul mercato, sia per quelli emergenti, dando loro la possibilità di entrare da subito nel mondo della telefonia senza quei grossi

investimenti iniziali necessari per raggiungere la totalità degli utenti.
VOIP ha aperto la strada ad un nuovo modo di pensare nei sistemi di comunicazione embedded grazie alla possibilità di costruire piattaforme di comunicazione "ad-hoc" pensate per soddisfare le esigenze di enti, aziende o società con sedi sparse nel territorio nazionale o internazionale, consentendo loro di ridurre la spesa relativa alla comunicazione tra i vari uffici, ed avere una soluzione scalabile e facilmente configurabile per future espansioni, non rinunciando all'affidabilità ed alla qualità del servizio offerti dalla rete telefonica tradizionale (PSTN).

Nel mondo VOIP troviamo molti protocolli di comunicazione per la voce, ma uno in particolare è riconosciuto come standard ed è il **SIP** (Session Initiation Protocol). SIP (Session Initiation Protocol) è un protocollo di livello applicativo che può stabilire, modificare e terminare sessioni multimediali, quali, ad esempio, le telefonate su IP. SIP è un protocollo standard della IETF (RFC 3261), indipendente dal protocollo di trasporto (UDP o TCP) e programmabile, quindi facilmente estendibile. A differenza di altri protocolli, SIP dà supporto a servizi avanzati come il controllo di presenza, la ricerca del chiamato e il reinstradamento della comunicazione. SIP consente all'utente di decidere come e dove desidera essere raggiunto, aggiornando sul sistema, in modo dinamico, un registro con lo stato di tutti gli utenti.

Nel mondo VOIP oltre a SIP troviamo anche un concorrente molto diffuso, **Skype**. Skype (nato nel 2002) è un software proprietario libero di messaggistica istantanea e VOIP. Esso unisce caratteristiche presenti nei client più comuni (chat, salvataggio delle conversazioni, trasferimento di file) ad un sistema di telefonate basate su una network peer-to-peer. Gli sviluppatori sono gli stessi che hanno realizzato il popolare client di file sharing Kazaa. Uno dei vantaggi di Skype rispetto al SIP è la facilità di configurazione delle impostazioni di connessione, infatti basta solamente registrarsi e il collegamento viene effettuato, ed è questo uno dei motivi per cui Skype ha preso molto piede sia a livello domestico che aziendale rispetto al SIP. Al contrario però Skype utilizza un protocollo VOIP proprietario per trasmettere le chiamate e questo lo rende poco integrabile in sistemi più ampi. La sola possibilità di interagire con Skype è attraverso le sue API che rendono le funzionalità di personalizzazione disponibili veramente limitate.

La presente tesi propone quindi un'integrazione di questi tre aspetti, cioè la realizzazione di un sistema embedded in cui si realizza una comunicazione VOIP tramite il protocollo SIP ma con la possibilità di estendere la comunicazione anche alla rete Skype. La realizzazione in particolare vede la **creazione di un campanello/citofono** il quale è un client SIP che comunica con altri client SIP oppure con un utente Skype.

La realizzazione di questo citofono SIP risulta vantaggiosa rispetto ad un citofono tradizionale perché fornisce la possibilità di gestire il citofono da remoto e a basso costo. Ipotizziamo la situazione in cui non siamo presenti in casa e qualcuno ci suona il citofono. Con questo sistema possiamo rispondere al citofono da qualsiasi parte del mondo e comunicare con la persona presente. Supponiamo per esempio il caso di una consegna importante di un pacco in cui al momento della consegna non possiamo essere presenti. Con questo sistema possiamo indicare al corriere di lasciare il pacco al nostro vicino, ma il punto principale è che il costo di questo servizio di comunicazione è basso rispetto ad altri mezzi di comunicazione, come per esempio al cellulare.
Altra situazione molto importante è quando dobbiamo gestire la videosorveglianza, cioè quando abbiamo il citofono, delle telecamere di controllo, la gestione dell'apertura dei cancelli e delle luci. Tutto il controllo di questi dispositivi possono avvenire tramite il citofono e con una gestione remota di tutto il sistema. Effettuando una chiamata al citofono possiamo attivare la visualizzazione delle telecamere, in più se le telecamere sono motorizzate possiamo muoverle da remoto, possiamo inoltre aprire il cancello nel caso un famigliare suoni il citofono ed ha bisogno di lasciare qualcosa dentro casa.
Pensiamo poi alla grande personalizzazione dell'interfaccia grafica che può essere fatta nel citofono. Se al citofono viene predisposto un sistema di autentificazione (anche attraverso un lettore di impronte digitale, oppure sistemi RFID), nel momento in cui il citofono riconosce la nostra autentificazione ci viene visualizzata una serie di funzionalità che possiamo eseguire sul posto: come l'accensione delle luci, l'apertura del cancello, la possibilità di effettuare una chiamata telefonica all'interno della casa o all'esterno.
Altre funzioni che il citofono può svolgere sono i collegamenti a servizi web, alla email.

Essendo il citofono sviluppato in questo modo è bene ricordare che possiamo aggiornare il firmware e il software da remoto riducendo i tempi e costi dell'aggiornamento.

Infine è bene sottolineare che utilizzando questo sistema possiamo proteggere (criptare) tutte le comunicazioni e il traffico che avviene nella rete. Infatti usando dei sistemi di crittografia possiamo garantire l'autentificazione dell'utente SIP e la riservatezza nelle comunicazioni.

Tutta la gestione remota e le chiamate del citofono avvengono a basso costo. Pensiamo quanto verrebbe a costare se usassimo un cellulare per gestire il citofono quando siamo all'estero? Ma la stessa cosa vale se fossimo nello stesso Paese. La differenza di costo tra una chiamata telefonica attraverso cellulare e attraverso il VOIP non è paragonabile, soprattutto in una realtà in cui le connessioni a Internet hanno preso il soppravvento e possiamo trovare degli Access Point quasi ovunque.

Tutto ciò è stato implementando utilizzando:

- Freescale I.MX27-PDK: scheda hardware per realizzare il citofono
- Asterisk: server VOIP per gestire le varie comunicazioni
- Linphone: client SIP
- SipToSis: gateway software per le chiamate tra SIP e Skype

Nel **capitolo 2** si andrà a presentare la scheda I.MX27; nel **capitolo 3** verrà presentato il tool usato per creare l'immagine di Linux che verrà caricata sulla scheda I.MX; nel **capitolo 4** verrà presentato il server VOIP Asterisk; nel **capitolo 5** verrà presentato il client SIP Linphone; nel **capitolo 6** verrà presentato il gateway SipToSis per far comunicare SIP con Skype; nel **capitolo 7** verrà presentato il programma che si interfaccia con Linphone ed infine nel **capitolo 8** verranno presentate le conclusioni di tale tesi.

Uno dei presupposti della tesi è di utilizzare il più possibile software opensource, in quanto si ha la possibilità di estendere le varie funzionalità dei software usati ed inoltre non si hanno costi di licenza per l'utilizzo del software.

1.2 Strumenti utilizzati

Per quanto riguarda l'hardware del citofono è stata usata la scheda Freescale **i.MX27 PDK** in quanto è completamente integrata e facilita lo sviluppo di software grazie ad un tool di sviluppo chiamato LTIB. La scheda si basa sull'ARM9, con 128 MB di SDRAM e 256 MB di NAND FLASH. È dotata di un display touch-screen da 2,7 pollici con un modulo camera a 2 MPixel ed essendo adatta per applicazioni multimediali ha al suo interno un microfono e degli altoparlanti. Ovviamente tra le varie porte che integra c'è la porta ethernet che ci permette di collegarci alla rete. Questa scheda integra al suo interno un modulo per la gestione del consumo di energia, così riesce a lavorare molto bene a basso costo.

All'interno di questo hardware viene installato un client SIP ovvero un softphone. Tra i vari softphone a disposizione è stato preso in considerazione **Linphone** in quanto oltre ad essere libero è anche opensource, quindi è una soluzione ottimale. Linphone permette di effettuare, oltre alle chiamate voce, anche chiamate con video e di comunicare tramite instant messagging; supporta il protocollo SIP e utilizza vari codec per l'audio e il video. Linphone può essere installato su varie piattaforme tra cui Linux, che è l'ambiente su cui la scheda lavora.

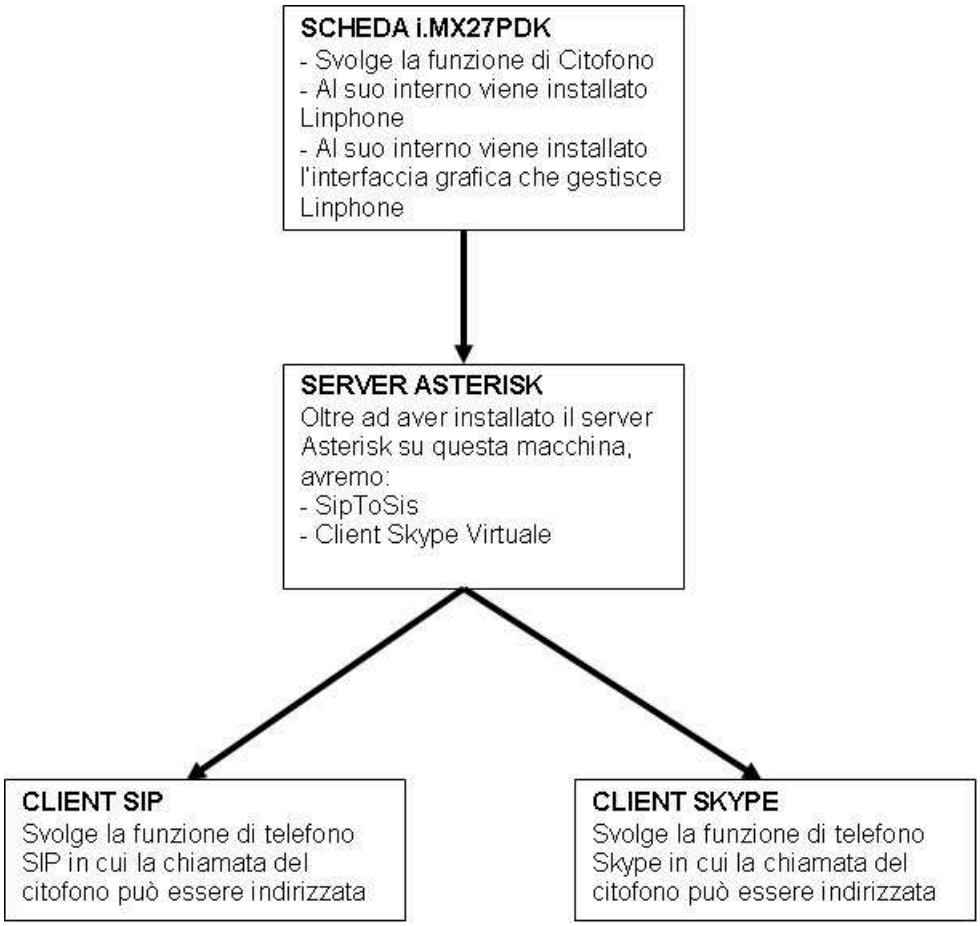
La gestione tra citofono SIP e telefono SIP deve avvenire tramite un server VOIP. È stato scelto di utilizzare un server **Asterisk** locale in quanto permette di sfruttare al massimo le varie potenzialità, senza dover utilizzare un servizio di terze parti che pone dei limiti di utilizzo, oltretutto è libero e open source, quindi è possibile personalizzarlo come che si vuole.

Infine il sistema per far comunicare SIP con Skype è stato fatto utilizzando il programma **SipToSis**. Anche qui c'erano varie soluzioni a disposizione ma molte erano proprietarie, mentre SipToSis è open source. SipToSis è un'applicazione sviluppata in Java in grado di connettere un SIP device al network di Skype ed effettuare e ricevere chiamate. Il software dispone di un convertitore in grado di convertire l'audio SIP RTP nell'audio Skype PCM e viceversa. SipToSis essendo sviluppato in Java è multipiattaforma, quindi lo possiamo installare sia su ambiente Windows che Linux.

La gestione di Linphone avverrà tramite un'interfaccia grafica sviluppata in C che andrà ad interfacciarsi con la console di Linphone e permetterà quindi di gestire la chiamata dal citofono al telefono SIP o Skype.

1.3 Schema di realizzazione

Andiamo quindi a schematizzare la soluzione finale dell'intero sistema:



Capitolo 2

Piattaforma I.MX27-PDK

2.1 Caratterestiche generali

La piattaforma i.MX27 PDK è un hardware completamente integrato ed ha una soluzione software progettata per semplificare lo sviluppo del prodotto in modo da poter concentrarsi sullo sviluppo software necessario per il successo di mercato. La piattaforma i.MX27 PDK è progettata da Freescale. Freescale ha introdotto il sistema di piattaforma a 3-stack, il quale viene usato per sviluppare applicazioni multimediali e di connettività usando l'Application Processor i.MX27, l'audio MC13783 e un dispositivo di gestione dell'energia.

Il sistema di piattaforma a 3-stack ci permette di ridurre il tempo tra il primo sviluppo e il prodotto finale grazie alla piattaforma di sviluppo sia per il software che per l'hardware.

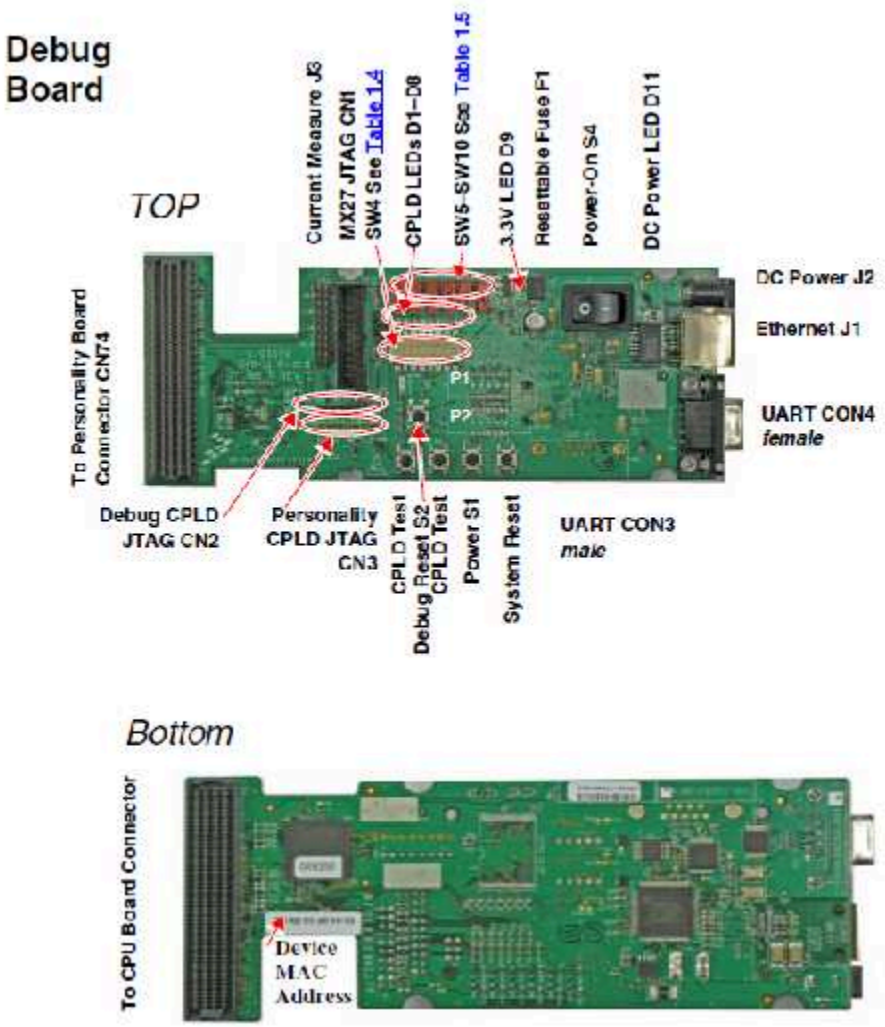
Il sistema di Freescale a 3-stack è formato da tre piccole board: una CPU, una board di Debug e una board Personality.

- La board CPU contiene la CPU i.MX27, le memorie e il PMIC MC13783 (Power Management IC)

Top

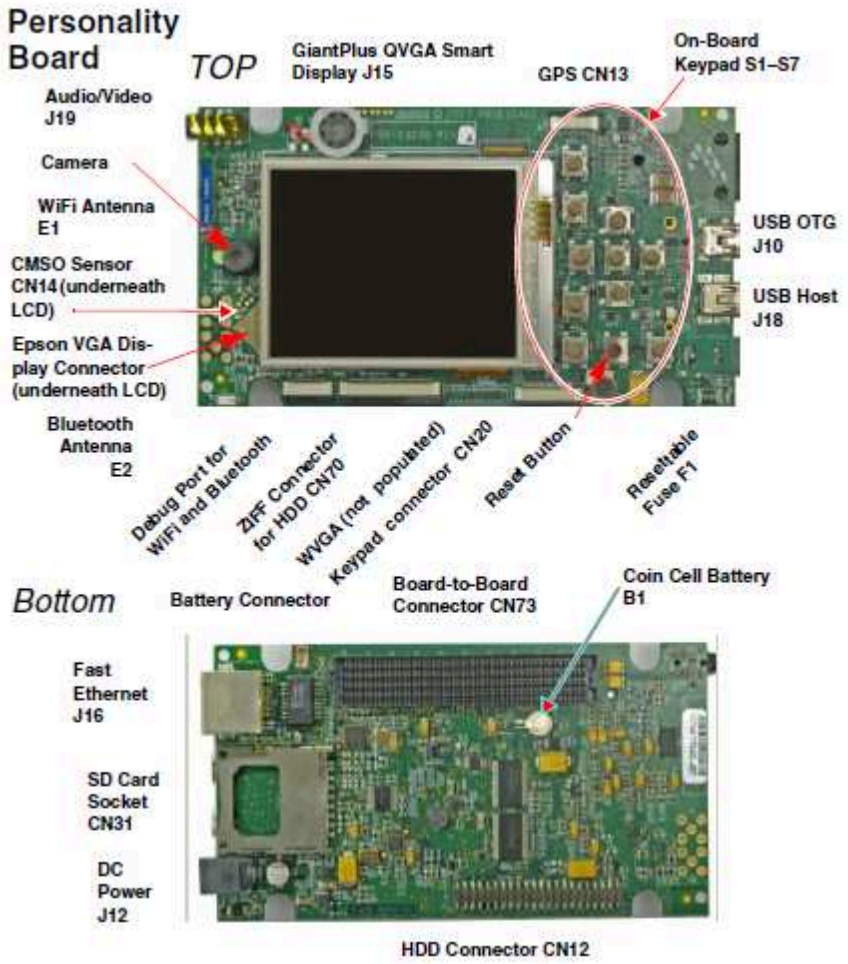
Bottom

- La board Debug fornisce le interfacce di debug (come JTAG), e inoltre ha un CPLD (Complex Programmable Logic Device) che implementa una Ethernet esterna e un controllore seriale per scopi di debug



- La board Personality implementa le funzionalità del sistema a 3-stack, e contiene l'hardware per la connettività Wi-Fi, il ricevitore FM, e così via. La board Personanality può essere modificata per soddisfare dei requisiti specifici senza la necessità di modificare le altre due board (CPU e Debug). La board Personality è stata progettata per supportare applicazioni multimediali comuni, ed ha un display VGA a 2.8 pollici, una videocamera, una scheda Wi-Fi IEEE 802.11 b/g standard, un ricevitore FM, un connettore SD card, una USB OTG , un USB host, un connettore display QVGA 2.4, un connettore ATA e

un connettore TV-out. Con l'evoluzione della piattaforma 3-stack abbiamo anche più board personalità che vengono create per soddisfare nuove richieste multimediali.



2.2 Pacchetti a supporto della scheda (BSP)

Lo scopo di questo software, come il nome BSP (Board Support Package) fa riferimento, è di fornire Linux alla famiglia i.MX dei circuiti integrati e alle piattaforme associate (3-stack board). Il BSP fornisce il software necessario per interfacciarsi al kernel Linux standard open-source per l'hardware i.MX. L'obiettivo di questo porting è di poter personalizzare il sistema rapidamente basandosi sui dispositivi i.MX che usano il sistema operativo Linux.

Il BSP non è una piattaforma o un prodotto di riferimento per l'implementazione. Non contiene tutti i driver specifici, gli stack-software indipendenti dall'hardware, le componenti GUI, la JVM, e le applicazioni richieste per un prodotto. Alcune di queste sono rese disponibili nella forma open-sorce come parte del kernel di base.

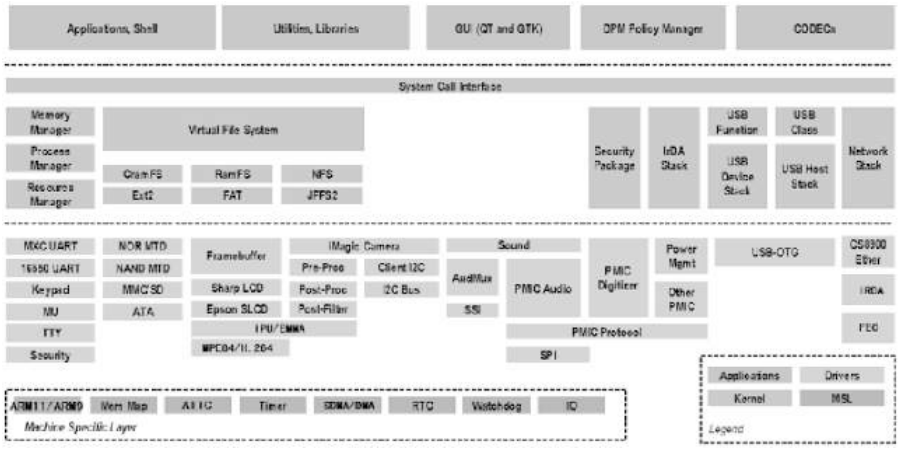
Il BSP i.MX è basato sulla versione del kernel Linux 2.6.22 fornita dal sito ufficiale dei kernel Linux (http://www.kernel.org). Il kernel è stato poi ottimizzato con le caratteristiche fornite da Freescale.

2.3 Architettura

Il BSP supporta tutte le piattaforme in un singolo ambiente di sviluppo, ma non tutti i driver sono supportati da tutti i processori. I driver comuni per tutte le piattaforme sono riferiti come driver i.MX e i driver specifici per una piattaforma vengono riferiti al nome della piattaforma.

2.3.1 Diagramma a blocchi di Linux BSP

La figura sotto mostra l'architettura del BSP per la famiglia dei processori i.MX. Essa consiste di uno spazio-utente eseguibile, dai componenti standard del kernel che provengono dalla comunità Linux, dai driver hardware specifici, e dalle funzionalità fornite da Freescale per la famiglia i.MX dei processori.



2.3.2 Kernel

Il kernel Linux i.MX è basato sullo standard kernel Linux. Il kernel supporta molte caratteristiche previste in molti sistemi embedded moderni:

- Gestione dei processi e dei thread
- Gestione della memoria (memory mapping, allocation/deallocation, MMU, e L1 cache control)
- Gestione delle risorse (interrupt)
- Gestione dell'alimentazione
- File system (VFS, cramfs, ext, ramfs, NFS, devfs, jffs2, fat)
- Modelli di driver
- API standardizzate
- Stack di rete

La personalizzazione del kernel Linux ARM supporta ogni piattaforma includendo una configurazione del kernel personalizzato e l'implementazione del livello macchina (MSL).

Configurazione

In questo caso la configurazione del kernel, per il BSP, è stata fatta attraverso LTIB Le seguenti caratteristiche sono alcune delle impostazioni di configurazioni diverse da quelle standard (la porta seriale è chiamata UART-DCE su i.MX27 PDK):

- Modalità embedded
- Modulo di loading/unloading
- ARM9
- Formato dei file supportati: ELF binary, a.out e ECOFF
- Block device: Loopback, Ramdisk
- i.MX Internal UART
- File system: ext2, dev, proc, sysfs, cramfs, ramfs, jffs2, fat, pramfs
- Framebuffer
- Debug del kernel

- Caricamento automatico del modulo kernel
- Gestione dell'alimentazione
- Support della MTD (Memory Technology Device)

Livello macchina specifico (MSL)

Il Livello Macchina fornisce un'implementazione dipendente dalla macchina come richiesto dal kernel Linux, come la memory map, gli interrupt, e il timer. Ogni piattaforma ARM ha la propria directory MSL sotto la directory:

```
<ltib_dir>/rpm/BUILD/linuxarch/arm/mach-mx27.
```

Prima che il kernel inizia a lavorare sullo spazio virtuale, deve avvenire il mapping tra gli indirizzi fisici e quelli virtuali per le periferiche di I/O che devono essere fornite alla MMU per effettuare la traduzione per gli accessi di memoria/registri. Il mapping viene fatto attraverso una struttura a tabella nella MSL per la particolare piattaforma, per ogni entry viene specificato l'indirizzo iniziale della periferica degli indirizzi virtuali, vengono inizializzati gli indirizzi fisici, la dimensione della regione della memoria e il tipo di regione.

Il kernel standard Linux contiene codici ARM comuni per gestire gli interrupt. Il MSL contiene implementazioni di funzioni specifiche per la piattaforma per interfacciare il kernel Linux al controllore degli interrupt vettoriale ARM11 (AVIC) o al controllore degli interrupt dell'ARM9 (AITC), in base alla piattaforma. I codici ARM insieme alle funzioni specifiche forniscono le seguenti capacità:

- Inizializzazione AVIC
- Inizializzazione AITC
- Controllo degli enable/disable interrupt
- ISR binding
- ISR dispatch
- Catena degli interrupt
- Standard API Linux per accedere alle funzioni interrupt
- Mapping statica di una sorgente di interrupt come FIQ (Fast Interrupt reQuest)

Il GPT (General Purpose Timer) è impostato per generare un interrupt ogni 10 msec per fornire un tick al sistema operativo. Questo timer viene anche usato dal kernel per aggiungere eventi temporali. Linux definisce le API del timer MSL richieste per il tick del sistema operativo, ma non mostra l'implementazione del tick del kernel. Linux inoltre facilita l'uso del timer attraverso varie funzioni, come il ritardo di tempo (delay), la misurazione, gli eventi, e gli allarmi. Il GPT è anche usato come sorgente per supportare le caratteristiche di alta risoluzione temporale. Gli interrupt del timer-tick sono disabilitati quando è in modalità a basso consumo rispetto a quando è in modalità di attesa (idle).

Il componente di I/O nella MSL fornisce un livello di astrazione tra i vari driver e le configurazione e l'utilizzazione del sistema, includendo GPIO, IOMUX, e le board esterne di I/O. Il modulo software di I/O è una board specifica, e risiede nel livello MSL come un insieme indipendente di file. Tale modulo fornisce le seguenti caratteristiche come parte di uno spazio kernel di API personalizzate:

- Inizializzazione per la configurazione di default I/O dopo il boot
- Funzioni per configurare i vari I/O per usi attivi
- Funzioni per configurare i vari I/O per la modalità a basso consumo
- Funzioni per controllare e campionare GPIO e la board di I/O
- Funzioni per autorizzare, non autorizzare, e obbligare le funzioni di callback per GPIO e interrupt EDIO (modulo che riconosce i segnali asincroni esterni come sorgenti di interrupt)
- Funzioni per supportare diversi livelli di priorità durante la registrazione ISR (Interrupt Service Routine) per diversi moduli; se avvengono più interrupt nello stesso momento, l'ISR di priorità maggiore avviene per primo
- Funzioni di aiuto per la configurazione GPIO, EDIO e IOMUX

Queste funzioni sono organizzate in base alle loro funzionalità, e non attraverso il loro pin o porta. Questo permette alla configurazione di I/O di cambiare, per essere centralizzata nel modulo GPIO senza richiedere cambiamenti nei vari driver. Queste funzionalità sono usate da altri device driver nello spazio del kernel. Il programma a livello utente non ha accesso alle funzionalità del modulo GPIO. Le API e le implementazioni sono diverse su ogni piattaforma per soddisfare diversi hardware, driver e board. Questo modulo è continua evoluzione. Molte funzionalità sono richieste da questo modulo con l'aumentare di driver che vengono aggiunti.

La SPBA (Shared Peripherical Bus Arbiter) fornisce un meccanismo arbitrario per avere accesso alla periferica condivisa fra più master. L'implementazione di SPBA sotto MSL definisce le API per permettere diversi master di prendere o lasciare il possesso della periferica condivisa. Queste funzionalità sono anche esportate in modo tale che possono essere usate da altri moduli.

2.3.3 Driver

Ci sono molti driver forniti da Freescale che sono specifici per le periferiche della famiglia dei processori i.MX o per le piattaforme di sviluppo. Molti di questi driver sono comuni per tutte le piattaforme. Molti possono essere compilati nel kernel o compilati come moduli i quali vengono caricati dinamicamente da un file system attraverso insmod o modprobe. I moduli possono essere caricati automaticamente quando richiesti usando le caratteristiche dell'auto-load del kernel. Il BSP contiene un file module.dep e un file modprobe.conf che contiene le informazioni di dipendenza per i moduli. I processori di applicazioni multimediali i.MX hanno parecchie classi di driver, che verranno spiegati di seguito.

Caracter device driver

La famiglia dei processori i.MX supporta un driver UART (Universal Asynchronous Receive/Transmitter). I driver dei dispositivi dei caratteri sono il driver UART 16C652 e il driver UART.

Il driver UART 16C652 fornisce l'interfaccia all'UART SC16C652 esterno su tutte le board i.MX PDK. Inoltre fornisce il driver standard API seriale di Linux per entrambe le porte UART esterne. Fornisce anche le seguente caratteristiche:

- Trasmissione/ricezione di caratteri guidati da interrupt
- Standard baud rate di Linux da 460.8K baud a 50 baud
- Due porte UART su 16C652
- Ogni porta UART 16C652 può essere programmata per 1,4,8 o 14 byte di livelli per interrupt
- Supporta la trasmissione e la ricezione di caratteri con lunghezza di 7-bit e 8-bit
- Supporta la trasmissione di 1, 1.5 o 2 bit stop
- Supporta la parità pari e dispari
- Supporta il controllo del flusso software XON/XOFF

- Supporta il controllo del flusso hardware CTS/RTS
- Spedisce e riceve i caratteri di break attraverso le API seriali standard di Linux
- Riconosce gli errori di frame e di parità
- Abilita di ignorare i caratteri con break, parità e errori di frame
- Ottiene e imposta le informazioni della porta UART attraverso le ioctls TIOCGSSERIAL e TIOCSSERIAL TTY
- Supporta le chiamate standard TTY
- Include la console di supporto che è necessaria per ottenere il prompt dei comandi attraverso le porte UART

Il driver UART, invece, interfaccia il driver seriale API Linux per tutte le porte UART. Supporta le seguenti caratteristiche:

- Trasmissione/ricezione di caratteri guidati da interrupt
- Il baud rate standard di Linux fino a 1.5 Mbps
- Trasmissione e ricezione di caratteri con lunghezza di caratteri di 7-bit o 8-bit
- Trasmissione di 1 o 2 bit stop
- Ha la parità dispari o pari
- Ha il controllo del flusso software XON/XOFF
- Ha il controllo del flusso hardware CTS/RTS
- Ha ioctl TIOCMGET per leggere le linee di controllo del modem
- Ha ioctl TIOCMSET per impostare le linee di controllo del modem
- Spedisce e riceve i caratteri di break attraverso le API standard seriali Linux
- Riconosce gli errori di frame e di parità
- Abilita di ignorare i caratteri con break, e gli errori di frame e di parità
- Ottiene e imposta le informazioni della porta UART attraverso le ioctls TTY TIOCGSSERIAL e TIOCSSERIAL
- Ha la gestione del consumo di energia, sospende e riprende le porte UART

- Supporta le chiamate standard TTY

- Include la console di supporto che è necessaria per ottenere il prompt dei comandi attraverso le porte UART

Grazie ad un parametro della configurazione del kernel si ha la possibilità di scegliere il driver UART, e anche di scegliere se l'UART deve essere usato come sistema di console. Tutte le porte UART possono essere usate attraverso i file dei dispositivi /dev/ttymxc0 (/dev/ttymxcX dove X è il numero dell'UART). /dev/ttymxc0 si riferisce all'UART 1.

Il clock real-time (RTC=Real Time Clock) è il clock che mantiene aggiornata la data e l'ora mentre il sistema è in esecuzione e spesso quando il sistema non è attivo. L'implementazione RTC supporta le chiamate ioctl per leggere l'ora, per impostare l'ora, per impostare gli interrupt periodici, e per impostare gli allarmi. Linux definisce le API RTC.

Il timer Watchdog (WDOG) protegge il sistema contro gli errori dovuti a eventi non previsti o errori di programmazione. L'implementazione software WDOG fornisce routine per servizi del timer WDOG, in modo tale che il time-out non possa avvenire. WDOG è attivo se viene attivato prima del boot del kernel Linux (attivato dal boot-loader o dalla ROM) con intervalli di servizio che devono essere configurati. In più, le opzioni del compile-time specifica se il kernel Linux deve attivare il watchdog, e se così fosse, quali parametri devono essere usati. Se è presente un secondo WDOG l'interrupt di priorità maggiore viene assegnata all'interrupt WDOG.

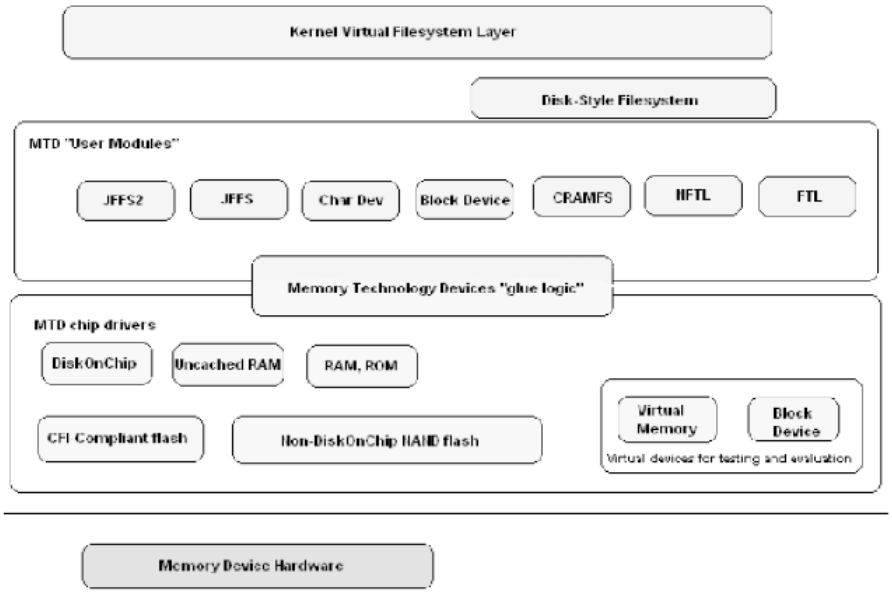
Il sistema operativo Linux ha una interfaccia standard WDOG che permette ad un driver WDOG, per una specifica piattaforma, di essere supportata. Questo è supportato sotto tutte le piattaforme i.MX.

Sound driver

I componenti del sottosistema audio sono le applicazioni, le Advanced Linux Sound Architecture (ALSA), i driver audio e l'hardware. Le applicazioni si interfacciano con l'ALSA, e l'ALSA si interfaccia con i driver audio, il quale controlla l'hardware del sottosistema audio.
I driver audio girano su processori ARM11 e ARM9. I dati audio digitali sono portati sul collegamento di interfaccia audio al codec hardware, questo gestito dal driver audio. Ci possono essere uno o più stream audio, che dipendono dal codec, come la voce o lo stereo DAC. Il driver audio anche configura il sample rate, l'audio MUXing, il formato, e il clock audio. Il driver audio gestisce anche l'impostazione del codec e del suo controllo, l'impostazione DMA e del suo controllo, e il controllo degli accessori audio, come la rilevazione delle cuffie e del microfono. L'unione di più stream può essere supportato, questo dipende dal codec.

Memory Tecnology Device Driver

Il Memory Technology Device (MTD) in Linux ricopre tutti gli aspetti dei dispositivi di memoria, come la RAM, la ROM, e i diversi generi di Flash. Come ogni dispositivo di memoria ha le proprie caratteristiche di lettura e scrittura, il sottosistema MTD fornisce un unico e uniforme accesso ai vari dispositivi di memoria.



La figura sopra descrive il sottosistema MTD. Il modulo utenteïon deve essere confuso con il modulo kernel o qualsiasi altro tipo di astrazione software. Il termine modulo utente MTD si riferisce ai moduli software con all'interno il kernel che attiva l'accesso al driver chip MTD di basso livello attraverso il riconoscimento dell'interfaccia e l'astrazione di un livello maggiore del kernel, oppure, in alcuni casi, allo spazio utente.

Il driver MTD chip si registra con il sottosistema MTD fornendo un insieme di chiamate predefinite di ritorno (callback) e proprietà attraverso argomenti mtd_info passate alle funzioni add_mtd_device(). Le callback sono fornite dal sottosistema MTD che portano fuori le operazioni come cancellazione, lettura, scrittura e sincronizzazione.

Il driver NAND MTD si interfaccia con il controllore NAND sul processore i.MX. Può supportare vari file system, come CRAMFS e JFFS. L'implementazione del driver supporta le operazioni di più basso livello sul chip NAND flash esterno. A causa che i blocchi in una NAND flash non sono garantiti ad essere buoni, il driver NAND MTD è anche capace di individuare i blocchi guasti e riportare quella informazione al livello più alto per gestire i blocchi corrotti. Questo driver è parte dell'immagine kernel.

Driver di rete

Il driver ethernet SMSC LAN9217 interfaccia le funzioni di SMSC LAN9217 con il modulo standard di network del kernel Linux. Il LAN9217 è formato da un singolo controller chip Ethernet 10/100, progettato per applicazioni embedded, in cui le prestazioni, la flessibilità, la facilità di integrazione e il controllo del costo del sistema, sono richiesti. Il LAN9217 è stato specificatamente progettato per fornire le più alte prestazioni possibili per qualsiasi applicazione a 16-bit. Il LAN9217 ricopre pienamente le caratteristiche di IEEE 802.3 10BASE-T e 802.3u 100BASE-TX, e supporta l'HP Auto-MDIX. Il driver Ethernet SMSC LAN9217 ha le seguenti caratteristiche:

- L'efficiente architettura PacketPage può operare in I/O e nello spazio di memoria, e come un slave DMA
- Supporta le operazioni in full duplex
- Supporta buffer per la trasmissione e la ricezioni di frame su chip RAM
- Supporta le caratteristiche di trasmissione programmabile come la ritrasmissione automatica in caso di collisione e generazione automatica di CRC
- Supporta EEPROM per la configurazione
- Supporta l'impostazione del MAC address
- Supporta le statistiche del dispositivo, come le collisioni di trasmissione

Questo adattatore di rete può avere accesso attraverso il comando ifconfig con il nome di interfaccia (eth0).

Disk driver

I driver disk comprendono i driver ATA. Il suo principale uso è di interfacciarsi con i dispositivi hard disk. Il driver ATA ricopre le caratteristiche dello standard ATA-6, e supporta i seguenti protocolli:

- PIO mode 0, 1, 2, 3, e 4
- Multiword DMA mode 0, 1 e 2
- Ultra DMA mode 0, 1, 2, 3 e 4 con il bus clock di 50 MHz o superiore
- Ultra DMA mode 5 con bus clock di 80 MHz o superiore

Driver sicurezza

Il livello di sicurezza è costituito da due moduli, il modulo Secure RAM e il modulo Secure Monitor. Il modulo Secure RAM fornisce un modo sicuro di memorizzare i dati sensibili in una memoria RAM on-chip e off-chip. I dati su on-chip possono essere cancellati se necessario per prevenire accessi non autorizzati. I dati su off-chip sono memorizzati in modo crittografato usando una chiave di crittografia che è unica per ogni dispositivo ed è accessibile solo attraverso il modulo Secure RAM. Il Controller Security (SCC) è una parte dell'architettura di sicurezza indipendente dalla piattaforma (PISA) di Freescale. Il modulo SCC può solo venire usato dall'ARM11.

L'HACC (Hash Accelerator Controller) è un acceleratore hardware progettato per assistere nell'hashing delle Flash esterne o RAM. Viene eseguito l'algoritmo SHA-1 su un dato input per produrre un hash a 160 bit. Il HACC include le seguenti caratteristiche:

- Accelera la generazione di un hash SHA-1 su contenuti di memoria selezionati
- Lavora su 512 bit alla volta, e alla fine può aumentare il blocco finale con una sequenza finale in modo tale che il blocco risultante sia di 512 bit
- Ha la flessibilità per lo hash usando 16 parole burst, oppure con burst incrementali per memorie non capaci di manipolare 16 parole burst
- I 160 bit risultanti di hash possono essere letti da 5 registri HSH di 32-bit
- Calcola l'hash SHA-1 su un numero grande, potenzialmente con segmenti non continui di memoria
- Gestisce il consumo di energia

Il test di integrità in run-time (RTIC) è parte della famiglia PISA dei componenti della sicurezza della piattaforma. Il suo scopo è di assicurare l'integrità delle periferiche di memoria, contenute e assistite con l'autenticazione di boot. La RTIC ha la capacità di verificare il contenuto della memoria durante il boot del sistema e durante l'esecuzione in runtime. Se il contenuto delle memoria in runtime fallisce nel match della firma di hash, viene generato un errore nel monitor di sicurezza.

- La RTIC include le seguenti caratteristiche:
- Messaggio di autenticazione SHA-1
- Insieme di dati segmentati per supportare i dati non contigui nella memoria
- Lavoro con il processo di boot ad alta sicurezza
- Supporto per 4 blocchi di memoria indipendenti
- Bus DMA programmabile e timer watchdog

Driver generali

Il driver Card Multimediale (MMC)/SD implementa una parte del driver standard Linux come un driver di blocco che si interfaccia al controller MMC/SHDC. L'interfaccia per i livelli superiori seguono il driver standard API di Linux:

- Il modulo SDHC supporta MMC e SD card
- La specifica MMC versione 3.0 è supportata. La specifica SD Memory Card e la specifica SD I/O card 1.0 sono supportate
- L'hardware contiene 32x16 bit di data buffer
- È supportata il plug and play
- 100 Mbps massimo di data rate in modalità 4-bit
- 1/4 bit di operazioni
- Per accedere alla SD card, solo la modalità del bus SD è supportata. La modalità SPI non è supportata
- Supporta gli eventi dell'inserimento della card e della rimozione
- Supporta i comandi standard MMC/SD/SDIO
- Supporta la gestione di energia

- Supporta l'impostazione o il reset della password o i comandi di lock/unlock della card

L'Inter-IC(I2C) driver bus è un interfaccia di livello inferiore che viene usata per interfacciarsi con il bus I2C. Questo driver è invocato dal driver chip I2C; non viene esposto nello spazio utente. Il kernel standard Linux contiene un modulo I2C che viene usato dal driver chip per accedere al driver bus per trasferire i dati sul bus I2C. Il driver chip usa lo spazio API dello standard kernel che viene fornito dal kernel Linux per accedere al modulo I2C. Il driver bus supporta le seguenti caratteristiche:

- Compatibilità con lo standard bus I2C
- Supporta il bit rate fino a 400 kbps
- Genera o individua i segnali di start e stop
- Genera o riconosce il bit acknowledge
- Supporta la modalità standard master I2C
- Supporta la gestione delle caratteristiche di energia attraverso la sospensione e la ripresa di I2C

La modalità slave I2C non è supportata da questo driver.

Il driver Digital Audio Multiplexer (AUDMUX) di basso livello fornisce un personalizzato spazio kernel di API per il modulo AUDMUX. Supporta tutte le caratteristiche del modulo hardware.

Il driver di sincronizzazione dell'interfaccia seriale (SSI) fornisce un personalizzato spazio kernel per i moduli SSI. Supporta tutte le caratteristiche dei moduli hardware includendo l'attivazione/disattivazione della richiesta degli eventi DMA. Il driver configura i canali DMA attraverso le API DMA.

Il driver di Configurazione dell'Interfaccia della Periferica Seriale (CSPI) interfaccia uno spazio personalizzato del kernel API per i moduli CSPI. Supporta le seguenti caratteristiche:

- Trasmissione/ricezione di frame SPI guidati da interrupt
- Gestione di multi-client
- Gestione delle priorità tra client
- Configurazione dei dispositivi SPI per client

IL DMA non è supportato.

La riduzione della frequenza è una parte importante per aumentare la vita della batteria di un dispositivo portatile, ma anche per ridurre il consumo di energia. Questo driver (CPUFreq) permette alla frequenza della CPU

di cambiare dinamicamente secondo il carico della CPU o di lasciare all'utente di impostarla. Opzionalmente, è anche possibile cambiare il voltaggio della CPU che dipende dai selettori della frequenza per ridurre il consumo di energia.

2.3.4 Boot Loader

Un boot loader è un piccolo programma che viene eseguito subito dopo che la CPU è stata avviata. Un boot loader è richiesto per avviare un sistema ARM Linux. Il boot loader per il Linux ARM ha due scopi:

- Impostare il sistema, come le Interfacce AHB Lite IP (AIPS) e il Multi Layer Cross Bar Switch (MAX), la memoria, e i diversi clock
- Ottenere le corrette informazioni per il kernel Linux prima di passare al kernel

Un boot loader fornisce le seguenti funzioni:

- Impostare AIPS e MAX
- Impostare Phase-Locked Loop (PLL) per i vari sistemi di clock
- Impostare ed inizializzare la RAM
- Inizializzare le porte seriali
- Individuare il tipo di macchina
- Impostare la lista delle etichette del kernel
- Saltare all'immagine del kernel

Il primo passo, l'impostazione di AIPS e MAX, è un passo richiesto per un boot loader per ottenere l'accesso alle corrette periferiche, come Timer e UART. Il MAX dovrebbe anche impostare correttamente le priorità dei diversi bus master. Il secondo passo, l'impostazione di PLL, è necessario perché l'impostazione di PLL di default potrebbe non essere ottimale. Il boot loader dovrebbe regolare le impostazioni prima di tentare di eseguire l'immagine per impostare il clock desiderato.

Nota che nell'ultimo passo, il salto all'immagine kernel, il boot loader chiama l'immagine kernel direttamente, indipendentemente se il kernel è compresso. Per un kernel compresso (zImage), l'espansione viene fatta dal codice circostante l'immagine del kernel durante la costruzione del kernel.

Il boot loader fornito da BSP è RedBoot. RedBoot scarica l'immagine usando o la connessione seriale o l'Ethernet, gestendo la decompressione dell'immagine e gli scripting, e memorizzando l'immagine nella Flash. RedBoot è principalmente usato per lo sviluppo di software.

RedBoot

Redboot è un firmware boot open source basato su eCos Hardware Abstraction Layer. É progettato per essere molto portabile, estendibile, e configurabile. Alcune delle sue caratteristiche sono:

- Connessione dell'host attraverso RS-232 o Ethernet
- Interfaccia a linea di comando attraverso RS-232 o Telnet
- Download dell'immagine attraverso http, TFTP, X-Modem, o Y-Modem
- Supporto per le compressioni di immagine
- Sistema di immagine Flash per gestire più immagini Flash
- La configurazione è memorizzata nella Flash
- Esecuzione dei script al boot
- GDB (per il debugging)
- BOOTP (per il boot da rete)
- Manutenzione watchdog

2.3.5 Interfaccia grafica utente

La GUI risiede nello spazio applicazioni a livello utente e interagisce con i driver video in modo trasparente. Tuttavia, ci sono certe parti della GUI che necessitano di essere portate, come i driver del touch screen e i driver della keypad.

Qtopia 4.3 sono le componenti della QT/Embedded per gestire le finestre.

2.3.6 Tools

La GCC ARM e la tool-chain cross-compiler sono usate per compilare il kernel, associare i driver, le librerie e le applicazioni. La tool-chain lavora su un PC host con Linux. L'ARM ADS è usata per eseguire il debugging a livello di kernel e GDB per il debugging a livello di applicazione.

2.3.7 Root File System

Il root file system è costruito come un'immagine cramfs o jffs2. Cramfs è un filesystem Linux progettato per essere semplice, piccolo e per una buona compressione. Viene usato su un buon numero di sistemi embedded e piccoli dispositivi.

RAM è montato come ramfs. Questo è usato per /tmp, /var e /home. C'è anche un supporto per ramdisk e file system jffs2.

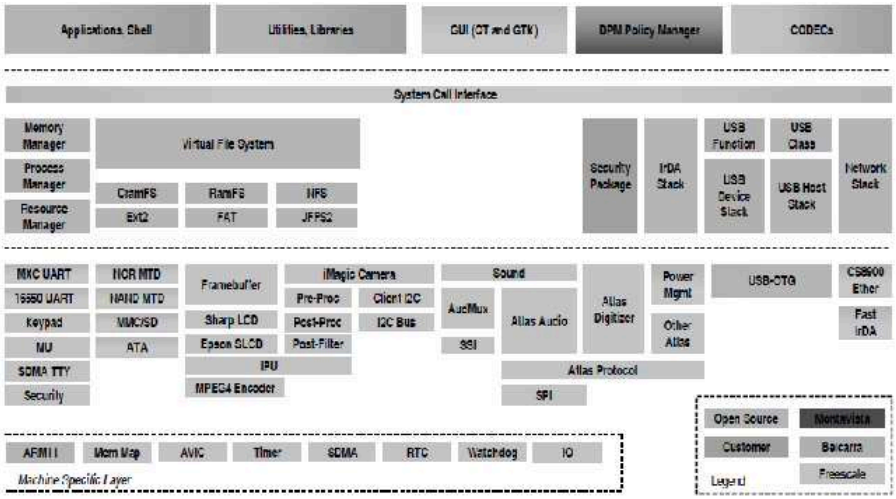
La cramfs è un filesystem di sola lettura (read-only). Diversamente da altri filesystem può essere creato con il suo contenuto. L'utilità mkcramfs è usata per costruire l'immagine cramfs la quale poi può essere scritta nella Flash/ROM e caricata con il comando:

```
mkcramfs dir img.cramfs
```

dove **dir** è il nome di una directory che contiene i file e le sottodirectory che devono essere aggiunte all'immagine cramfs, e img.cramfs è il nome del file dell'immagine.

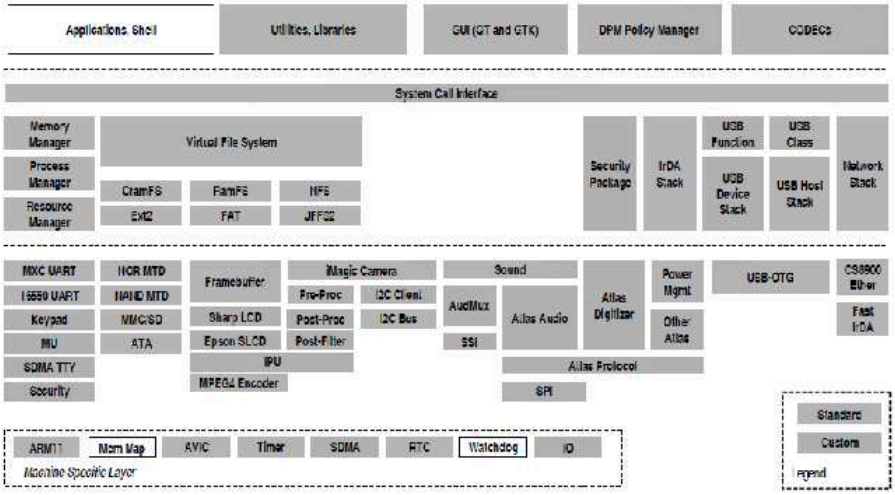
2.3.8 Sorgenti dei componenti di Linux BSP

La figura sotto mostra i sorgenti del codice per ogni componente di Linux BSP



2.3.9 Le API di Linux BSP

La figura sotto mostra una vista di alto livello dei componenti di Linux BSP



Capitolo 3

LTIB

3.1 Caratteristiche generali

Il progetto LTIB (Linux Target Image Builder) è un semplice tool che può essere usato per sviluppare e utilizzare per i BSP (Board Support Packages) per varie piattaforme di sviluppo. Usando questo tool un utente può sviluppare un'immagine GNU/Linux per la sua piattaforma. LTIB ha le seguenti caratteristiche:

- Open source (GPL)
- Viene eseguito sulle distribuzioni Linux più popolari (x86 32/64 e alcuni PPC)
- Usa un'interfaccia a linea di comando con una grafica a caratteri
- Supporta più architetture hardware (PPC, ARM, Coldfire)
- La piattaforma target (la board) viene scelta da un menu
- Ci sono più di 200 pacchetti selezionabili dall'utente
- Fornisce la costruzione del boot loader e dell'immagine del kernel
- Tutti i pacchetti vengono costruiti come utente non-root
- La configurazione e selezione dei singoli pacchetti avviene attraverso un menu
- Avviene la risoluzione delle dipendenze dei pacchetti
- Avviene anche la risoluzione dei conflitti dei pacchetti
- Avviene la re-installazione o la disinstallazione dei pacchetti quando viene cambiato l'albero delle dipendenze

- La tool-chain è selezionabile al momento della configurazione
- La configurazione del kernel Linux usa il linguaggio di configurazione nativo
- La configurazione del sistema target avviene dall'host (indirizzo IP, servizi, etc.)
- Supporta un file pre-configurato il quale permette ai sviluppatori di memorizzare diversi sistemi di configurazione (selezionare la toochain, selezionare il kernel, selezionare i pacchetti, etc.)
- Supporta dei profili, questo permette ai pacchetti dell'utente di essere impostati e riconfigurati. Questo è vantaggioso per l'autobuilding o lo scambio di configurazione specifiche
- Tutti i pacchetti sono costruiti come rpms e gestisti usando rpm
- La gestione dei file di immagine usa un database rpm privato per l'istanza di LTIB sull'host
- Il modo di fornire i singoli pacchetti viene fatto attraverso i comandi prep/scbuild/scdeploy
- Viene fornito un file spec in cui vengono specificati gli update e/o patch
- Viene fornita una shell che permette di eseguire tutti i comandi nell'ambiente di LTIB
- Fornisce un impiego incrementale (su NFS)
- Fornisce la creazione di immagini JFFS2 e RAMDISK
- Supporta il file system ad essere read-only
- LTIB è solo un meta-data, tutti i sorgenti sono presi usando http oppure una cache locale in un area comune dell'host
- Fornisce un supporto alle risorse remote attraverso anche un proxy
- Supporta glibc e ulibc
- Tutti i formati meta-data sono open-source
- L'architettura BSP è modulare (quindi è facile aggiungere nuovi BSPs)
- Supporta la modalità batch e –continue per l'auto-builder
- Supporta –dry per la preview delle azioni di ltib

- Fornisce –dltest per testare la disponibilità di source/patch
- Fornisce listpkgs per mostrare la lista di tutti i pacchetti, inoltre indica se sono selezionati e le relative licenze
- Fornisce una modalità release, questo incapsula il progetto LTIB in una immagine che non richiede l'accesso alla rete

La tool di LTIB è rilasciata sotto GNU, General Public License (GPL).

LTIB è disponibile come immagine da Freescale o in formato CVS da Savannah. Le immagini iso hanno il vantaggio che sono auto-costruite, i pacchetti per quel target sono pre-costruiti e hanno anche molta documentazione specifica per quel BSP. Hanno però lo svantaggio che sono specifiche per quel target e quindi non contengono gli ultimi aggiornamenti della tool-chain di LTIB. LTIB di Savannah ha il vantaggio di essere aggiornata e fornire il supporto per più piattaforme. Lo svantaggio è che gli head di CVS possono non essere sempre stabili, e non contengono source/patch per la costruzione. Questo significa che se viene usato CVS, qualsiasi tool-chain/source/patch necessita di essere selezionata e scaricata da Internet per la configurazione.

Prima di installare LTIB è consigliato avere circa 1 GB di spazio libero per costruire un BSP completo. I pacchetti richiesti per l'host sono:

Pacchetto	**Versione**	**Commento**
Perl	>=5.6.1	Per eseguire gli script ltib
Glibc	>=2.2.x.	Per costruire/eseguire i pacchetti host
Glibc-headers	>=2.2.x	Per costruire/eseguire i pacchetti host
Glibc-devel	>=2.2.x	Per costruire/eseguire i pacchetti host
Binutils	>=2.11.93	Per costruire i pacchetti host
Libstdc++	Qualsiasi	Per costruire i rpm-fs pacchetti host
Libstdc++-dev	Qualsiasi	Per costruire i rpm-fs pacchetti host
Gcc	>=2.96	Per costruire i pacchetti host
Gcc-c++	>=2.26	Per costruire i pacchetti rpm-fs host
Sudo	Qualsiasi	Per eseguire la fase "rpm install"
Zlib	Qualsiasi	Per costruire i pacchetti rpm-fs, mtd-utils
Zlib-devel	Qualsiasi	Per costruire i pacchetti rpm-fs, mtd-utils
Rpm	Qualsiasi	Per costruire i pacchetti iniziali rpm-fs
Rpm-build	Qualsiasi	Per costruire i pacchetti iniziali rpm-fs
Patch	Qualsiasi	Usato da rpm
Wget	Qualsiasi	Per scaricare pacchetti/patch su richiesta
Ncurses	>=5.1	Per costruire i pacchetti lkc
Ncurses-devel	>=5.1	Per costruire i pacchetti lkc
M4	Qualsiasi	Può essere richeisto da bison

Pacchetto	Versione	Commento
Bison	Qualsiasi	Per costruire i pacchetti lkc (config language)
Flex	Qualsiasi	Non richiesto ma consigliato per lkc
Textinfo	Qualsiasi	Per costruire pacchetto host genext2fs
Gettext	Qualsiasi	Genext2fs pacchetto target
Autoconf	>=2.54	Non richiesto, installiamo automake
Libtool	>=1.4.2	Non richiesto, installiamo gibus
Glib2-devel	Qualsiasi	Necessario se vogliamo costruire glib2

3.2 Installare LTIB

LTIB dovrebbe girare su molte distribuzioni Linux che hanno installato glibc-2.2.x o successivi. Sulle seguenti piattaforme è già stato testato:

- Redhat: 7.3, 8.0, 9.0, Enterprise release 4
- Fedora Core: 1, 2, 3, 4, 5, 8
- Debian: 3.1, unstable, 4.0
- Suse: 8.2, 9.1, 9.2, 10.0, 10.2, 10.3
- Ubuntu: 6.10, 7.04, 8.04

Per i PPC è stato testato su Debian 3.1r0 (stable) e unstable.

Per installare LTIB da CVS bisogna scaricare il pacchetto e poi eseguire i seguenti comandi:

```
$ cd ltib
$ ./ltib
```

Per installare LTIB da un'immagine iso, quindi avendo già un binario, è possibile seguire questi passaggi:

- Scaricare l'immagine iso
- Caricare l'immagine iso come root

  ```
  $ mount <bspname.iso> /mnt/cdrom -o loop
  ```

- Installare entrando nella cartella del target (come utente non-root)

  ```
  $ ./mnt/cdrom/install
  ```

  Seguire le istruzioni del prompt e impostare la directory dove deve essere installato LTIB e tutte le sottodirectory

- Una volta che LTIB ha terminato di copiare i suoi file nella directory, è possibile eseguire:

  ```
  $ cd <directory_target>/ltib
  $ ./ltib
  ```

3.3 Configurazione

3.3.1 Aspetti generali

Per configurare LTIB è possibile usare i seguenti comandi:

./ltib	sulla prima esecuzione per avere una istanza di CVS
./ltib -m config	per configurare solo
./ltib –configure	per configurare e costruire

Molte delle informazioni di navigazione sono mostrate in alto della configurazione. In aggiunta è possibile usare:

- **/:** questo ricerca una parola chiave nella configurazione e nel caso la trova mostra la locazione e le informazioni di dipendenza
- **s:** ricerca il testo del menu e si riposiziona nella parola ricercata in ogni match. È possibile ripetere questa operazione per continuare la ricerca.
- **page-up/page-down:** possono essere usati per saltare in alto o in basso nella lista visualizzata a video. La corrispondenza di queste chiavi varia da sistema a sistema.

3.3.2 Videata di più alto livello

Il livello più alto della videata di configurazione mostra i punti di configurazione specifici della piattaforma. L'esatta composizione di questa videata varia da piattaforma a piattaforma in base alle opzioni disponibili che vengono offerte. L'esempio sotto è stato preso dalla piattaforma mpc8548cds, quindi verrà presa ogni sezione della videata e verrà fornita una spiegazione:

```
--- Choose the target C library type
      C library type (glibc) --->
```

Molte piattaforme offrono sia glibc o uClibc per le librerie C. Questa opzione condiziona la scelta delle tool-chain disponibili.

```
--- Choose your toolchain
        Toolchain (gcc-3-4/glibc-2.3.4 e500 (DPFP)) --->
        (-mcpu=8548 -hard-float -mfloat-gprs=double)
        enter any CFLAGS for gcc
```

Questa entry permette di scegliere la tool-chain da quelle disponibili. La scelta è data dalla entry CFLAGS che segue. È possibile comunque sovrascrivere il valore di CFLAGS. Quando LTIB viene eseguito, può scaricare (se richiesto) e installare la tool-chain che viene selezionata. Nota che ogni piattaforma include un'opzione per selezionare una tool-chain personale. Questo

permette di usare molti cross-compilatori che sono nel sistema. Gli usi più comuni di questa opzione è di testare nuove tool-chain prima di inserirle in una nuova tool-chain della piattaforma. Quando viene usata una tool-chain personalizzata, si deve inserire:

- **tool-chain path:** questo è il percorso base assoluto della tool-chain. Per esempio, se vogliamo usare: /opt/mtwk/usr/local/gcc-3.4.3-glibc-2.3.4/arm-linux/bin/arm-linux-gcc, la corretta entry dovrebbe essere: /opt/mtwk/usr/local/gcc-3.4.3-glibc-2.3.4/arm-linux
- **cross tools prefix:** questo è il valore del prefisso che tutte le tool-chain standard eseguono per una data tool-chain. Così per l'esempio appena dato, questo dovrebbe essere powerpc-linux-
- **CFLAGS:** normalmente questo può essere lasciato in bianco

```
--- Bootloader
[*] Build a boot loader
    U-Boot options --->
```

Se disponibile, questo permette di selezionare e costruire un boot loader per la piattaforma.

```
    Kernel (Linux 2.6.11+pq3 patches) --->
[] Include kernel headers
(linux_2.6.11_mpc8548_cds_def.config) kernel preconfig
[] Configure the kernel
[] Leave the source after building
```

- **Kernel**: permette di selezionare il kernel disponibile per la piattaforma.
- **Include kernel headers**: imposta LTIB di installare i file header del kernel che sono stati costruiti in rootfs/usr/src/linux/include.
- **Kernel preconfig**: mostra il file di configurazione del kernel che può essere usato quando viene costruito il kernel. Questo è automaticamente selezionato secondo la scelta del kernel. Questo file inoltre può essere sovrascritto.
- **Configure the kernel**: attiva LTIB di entrare nella configurazione del kernel quando LTIB costruisce i pacchetti.
- **Leave the sources after building**: questa opzione lascia il sorgente per il kernel spacchettato dopo che il pacchetto del kernel è stato costruito e installato.

```
--- Package selection
    Package list --->
--- Target System Configuration
    Options --->
--- Target Image Generation
    Options --->
```

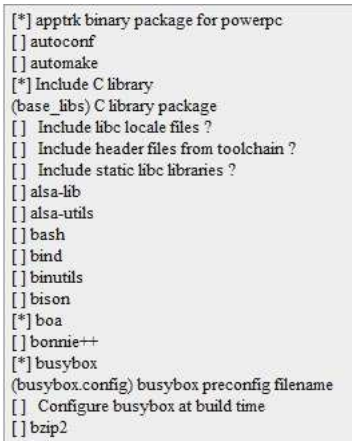
Queste voci permettono di entrare nelle videate delle specifiche configurazioni che saranno discusse nelle prossime sezioni.

Nota: in alto della finestra vedremo:
Load an Alternate Configuration File

Questo può essere usato per importare una configurazione completa per la piattaforma. Per esempio, è possibile salvare una particolare configurazione in un file, per essere riusato successivamente. Questo file può essere messo nella directory config/platform.

```
--- Package selection
    Package list --->
```

Selezionando la voce Package List si entra nella videata in cui si possono selezionare i pacchetti. Usando questa videata è possibile aggiungere o rimuovere i pacchetti che si vuole. Quasi tutti i pacchetti sono comuni a tutte le piattaforme, eccetto per alcuni pacchetti che sono condizionati dalla scelta della piattaforma. Se un pacchetto richiede un altro pacchetto, viene selezionato anche l'altro pacchetto in modo automatico (cioè viene risolto l'auto-dipendenza). Abbiamo qui un esempio della parte iniziale della videata:



3.3.3 Videata di configurazione per i target comuni

```
--- Target System Configuration
    Options --->
```

Selezionando *Options* si entra nelle opzioni di configurazione del sistema. Lo scopo di questa videata è di permettere di configurare il sistema target. Abbiamo preso questa videata come esempio per ogni item:

```
(mpc8548cds) target hostname
```

Questa opzione permette di configurare l'host-name del target

```
[*] boot up with a tty and login
(::respawn:/sbin/getty -L ttyS1 115200 vt100)
Enter your inittab startup line
```

Se non viene impostata, il target si avvia direttamente al prompt dei comandi. Il vantaggio di questo è che non necessita di conoscere il baud rate o il device tty del target. Molto spesso questa opzione viene lasciata non selezionata durante lo sviluppo iniziale di un BSP. Se selezionata, viene impostata una entry di default per la linea di *inittab*. Questa può variare da piattaforma a piattaforma. È comunque possibile sovrascrivere tale linea.

```
() load these modules at boot
```

È possibile inserire una lista di moduli che si desidera che vengano caricati al boot della piattaforma.

```
[*] start networking
    Network setup --->
```

Selezionando start networking, si ha la possibilità di entrare e configurare i parametri della rete. Questa videata permette di configurare al massimo cinque dispositivi. La schermata seguente mostra un esempio per eth0.

```
[*] Enable interface 0
(eth0) interface
[ ] get network parameters using dhcp
(192.168.1.100) IP address
(255.255.255.0) netmask
(192.168.1.255) broadcast address
(192.168.0.1) gateway address
(192.168.0.1) nameserver ip address
```

Per default molte piattaforme sono configurate non con un indirizzo IP fisso. Nel caso si volesse tale configurazione è possibile cambiare tali parametri in modo appropriato. Ritornando alla videata di configurazione del sistema, dopo le impostazioni di rete, si vede un insieme di opzioni che permettono di attivare/disattivare dei servizi in esecuzione sulla board. La lista esatta dipende da quali pacchetti si hanno selezionato. Di seguito vienne riporato un esempio:

```
[*] set the system time at startup
(ntp.cs.strath.ac.uk) NTP server name/ipaddress
[*] start syslogd/klogd
[*] start inetd
() Enter command line arguments for inetd startup
[*] start portmap
[*] start boa (webserver)
(-c /etc) command line arguments for boa
```

3.3.4 Generazione dell'immagine per il target

```
--- Target Image Generation screen
    Options --->
```

Selezionando Options si entra nelle opzioni per la generazione delle immagini. Lo scopo di questa videata è di permettere di configurare certi aspetti dell'immagine target. Per default, un'immagine NFS viene sempre generata, ma in aggiunta si può scegliere se costruire un'immagine come JFFS2 oppure RAMDISK ext2. Di seguito viene riportato un esempio per un sistema che usa RAMDISK

```
--- Choose your root file system image type
    Target image: (ext2.gz ramdisk) --->
```

Qui è possibile scegliere tra NFS in sola lettura, JFFS2 o ext2.gz RAMDISK

```
[ ] read-only root file system
```

Selezionando questa opzione si ottiene che il root file system viene montato in sola lettura. Certe directory che devono avere il permesso di scrittura vengono copiate in una directory particolare, come ramfs/tmpfs per essere lette e scritte al boot e successivamente caricate.

```
[*] create a ramdisk that can be used by u-boot
```

Questa opzione è disponibile se viene usato u-boot per il target. Viene messo in RAMDISK in un formato che u-boot possa essere letto

```
() rootfs target directory
```

Questa opzione copia l'immagine di uscita (RAMDISK o JFFS2) in una directory scelta. Ovviamente si deve avere i permessi per accedere a quella cartella.

```
[ ] Keep temporary staging directory
```

Questo viene usato principalmente durante il debugging.
Usa una directory temporanea che viene usata come directory per l'immagine RAMDISK/JFFS2.

```
[*] remove man pages etc from target image
[*] remove the /boot directory
[*] remove the /usr/src/ directory
[*] remove the /usr/include directory
() remove these directory
() remove these file [*] remove the static libraries
[*] strip any remaining binaries or libraries in the
    target image
```

Queste opzioni permettono di rimuovere certi file/directory dall'immagine RAMDISK/JFFS2. L'idea è di liberare spazio cambiando piattaforma.

```
(0) Allocate extra space (Kbytes)
```

Questa opzione permette di riservare spazio nell'immagine JFFS2.

3.3.5 Riconoscere il kernel in LTIB

L'esempio che segue è stato preso per la piattaforma mpc8548cds

- Eseguire LTIB

  ```
  ./ltib --configure
  ```

- Localizzare la sezione come questa e selezionare *Configure the kernel*

  ```
  --- Choose your Kernel
      Kernel (Linux 2.6.11+pq3 patches) --->
  [ ] Include kernel headers
  [ ] Configure the kernel
  [ ] Leave the source after building
  ```

- Uscire e salvare.

Il kernel viene costruito e si ferma nel menuconfig. In questo modo è possibile impostare i valori che sono memorizzati in:

```
config/platform/mpc8548cds/linux_2.6.11_mpc8548_cds_def.config
```

Dopo avere selezionato e cambiato qualche valore, LTIB li salva e ritorna a:

```
config/platform/mpc8548cds/
                      linux_2.6.11_mpc8548_cds_def.config.dev
```

Se si vuole tentare di cambiare un numero diverso di opzioni del kernel, ma non si vuole ricostruire completamente il kernel ogni volta, allora si può seguire questi passi:

- Selezionare l'opzione

  ```
  [ ] Configure the kernel
  ```

- Eseguire

  ```
  ./ltib -p kernel-2.6.11-pq3 -m prep
  ```

- Infine:

  ```
  ./ltib -p kernel-2.6.11-pq3 -m scbuild
  ```

Questo permettere di impostare i valori del kernel. Se c'è un file *config/platform/mpc8548cds/linux_2.6.11_mpc8548_cds_def.config.dev*, questo file può essere usato come punto d'inizio. Il file salvato ritorna con lo stesso nome del file, in modo tale che sia possibile cambiare i valori in modo incrementale.

- Per copiare il kernel costruito nella directory rootfs/boot, possiamo eseguire il comando

  ```
  ./ltib -p kernel-2.6.11-pq3 -m scdeploy
  ```

Se viene cambiato qualche codice sorgente in *rpm/BUILD/linux-xxx/* si deve attivare i cambiamenti eseguendo:

```
./ltib --p kernel-2.6.11-pq3 -m patchmerge
```

Questo può generare una patch dei cambiamenti e aggiornare il file spec. Per rendere i cambiamenti permanenti, si deve eseguire:

```
mv config/platform/mpc8548cds/
                       linux_2.6.11_mpc8548_cds_def.config.dev
config/platform/mpc8548cds/linux_2.6.11_mpc8548_cds_def.config
```

3.4 Istruzioni specifiche per l'utilizzo

Alcuni target hanno delle istruzioni specifiche nella loro directory della piattaforma, questo file è normalmente chiamato deployment_instructions.txt, per esempio:

`config/platform/tmp8231/deployment_instructions.txt`

Le seguenti istruzioni sono un caso generale (basate su tqm823l).

Il bootloader, il kernel e il root file system non si trovano in una directory standard perché varia da target a target (dipende da defconfig), ma andiamo a vedere il caso generale:

- **Root filesystem**: il root filesystem è una directory chiamata solitamente rootfs. I files sotto questa directory sono generalmente di proprietà di root, con permessi di group etc., impostati come richiesto ad essere eseguiti sulla board target. Lo scopo di questa directory è di fornire un punto che può essere l'NFS esportato dall'host e l'NFS caricato come il root filesystem del target.

- **Bootloader**: il bootloader (se costruito) si trova sotto la cartella rootfs/boot/, per esempio molte piattaforme PowerPc dovrebbero avere le seguenti componenti di bootloader:

  1. Rootfs/boot/u-boot: versione di u-boot per il debug
  2. Rootfs/boot/u-boot.bin: l'immagine binario di u-boot che dovrebbe essere installato nella Flash (normalmente non è necessario fare questo)

- **Kernel**: il kernel (se costruito) si trova nella cartella *rootfs/boot*, per esempio molte piattaforme powerpc hanno i seguenti componenti di bootloader:

  1. Vmlinux: l'immagine del kernel per il debug
  2. uImage: l'immagine del kernel
  3. system.map: il file di map per il kernel
  4. linux.config: la configurazione usata per costruire l'immagine del kernel

I più comuni modi per usare il proprio kernel e root filesystem nel target è di caricare il kernel sulla board usando tftp e poi fare il boot con il kernel con appropriati parametri che caricano il root filesystem dall'host, usando NFS.

I passi da seguire per esportare il root file system dall'host usando NFS dipende da distribuzione a distribuzione, ma in generale, i passi standard sono i seguenti:

- Scelta di un indirizzo IP della rete per la board. Dobbiamo essere sicuri di aver configurato LTIB in modo tale che il root filesystem usi questo indirizzo IP. La board può spesso usare il DHCP per ottenere il suo indirizzo IP, tuttavia se viene selezionato DHCP nella configurazione di LTIB e si vuole usare l'NFS, quello che viene fatto è di bypassare le impostazioni della scheda di rete e lasciare che le impostazioni vengano ereditate dai parametri di boot del kernel
- Annotare l'indirizzo IP dell'host. Per questo esempio, si assumono i seguenti valori:
  - Indirizzo IP della board: 192.168.0.254
  - Indirizzo IP dell'host: 192.168.0.204
  - Posizione del root filesystem: */ltib/rootfs*
- Creare un collegamento da */tftpboot* al root filesystem

  ```
  ln -s /ltib/rootfs /tftpboot/192.168.0.254
  ```

- Essere sicuri che il server NFS e il portmap siano installati sul sistema
- Esportare questa directory editando */etc/exports* e aggiungendo questa entry:

  ```
  /home/<utente>/ltib/rootfs *(rw,no\_root\_squash)
  ```

  Nota: questo esempio ha tutti i livelli di sicurezza disattivati. Qualsiasi host potrebbe accedere alla directory esportata.
- Riavviare NFS:

  ```
  sh /etc/rc.d/init.d/nfs restart
  ```

Molti bootloader caricano il kernel attraverso una connessione di rete usando il protocollo tftp. Così il bootloader della board può accedere al kernel che è stato costruito, ma quindi si deve essere sicuri di avere il demone tftp in esecuzione sull'host. Si deve quindi eseguire dei test per verificare questo:

1. Testare che il server tftp è presente sul sistema, eseguendo questi comandi:

   ```
   $ls /usr/sbin/in.tftp
   /usr/sbin/in.tftp
   ```

   Nel caso non fosse presente è necessario installare il pacchetto tftp-server.

2. Testare se inetd è impostato a eseguire il server tftp eseguendo il seguente commando

```
$ netstat -a | greptftp
udp 0 0 *:tftp (questa dovrebbe essere l'uscita)
```

   Se non viene visualizzato niente sullo schermo, si deve essere sicuri che non sia disabilitato in /etc/xinetd.d/tftp, cioè che non ci sia la riga:

```
disable = no
```

3. Essere sicuri che se c'è un firewall sull'host che non blocchi i pacchetti in ingresso dal target

Generalmente il server tftp viene configurato a chroot nella directory /tftpserver. Questo significa che abbiamo bisogno di copiare il kernel in quella directory, o in una sottodirectory di quella directory. Nel nostro esempio, si deve eseguire:

```
$ cp rootfs/boot/uImage /tftpboot
```

Per usare NFS si deve inoltre impostare gli argomenti del bootloader per il kernel. L'esempio mostrato è per un sistema che usa uboot/ppcboot con le impostazioni precedenti. Si impostano quindi i parametri della rete eseguendo:

```
=> setenv ipaddr 192.168.0.254
=> setenv severi 192.168.0.204
=> setenv bootargs root=/dev/nfs
   nfsaddrs=192.168.0.254:192.168.0.204
```

Infine si deve caricare il kernel sulla board target ed eseguire il boot, eseguendo i seguenti comandi:

```
=> tftp100000 vmlinux.gz.uboot
...
=> bootm 100000
```

3.5 Opzioni della linea di comando

3.5.1 Operazioni generiche

Eseguendo il comando ./ltib –help, verranno visualizzate tutte le opzioni di LTIB che ora si andrà a riassumere:

```
$ ./ltib --help

This script is used to manage the building of BSPs with common target
root filesystems.  The rpms are installed as they are built
in the directory /home/seh/ltib_bsps/ltib-dev/rootfs (unless overriden in the resource file)

Edit the file .ltibrc in this directory to change the default system
configuration, or .ltibrc in your home directory.

ltib [-m <mode>] [options....]
  Where:
    --mode|m
     Where mode is either:
      prep       just prep the package
      scbuild    rpmbuild -bc --short-circuit
      scinstall  rpmbuild -bi --short-circuit
      scdeploy   does an scinstall followed by an install to the rootfs
      patchmerge  generate and merge a patch (requires -p <pkg>)
      clean      clean/uninstall target packages
      distclean  full cleanup, removes nearly everything
      listpkgs   list packages (alphanumeric)
      release    make a binary release iso image
      config     use with --configure to do configuration only
      shell      enter ltib shell mode (sets up spoofing etc)
    --pkg|p      : operate on this package only
    --configure|c : run the interactive configuration
    --preconfig  : configuration file to build from (defaults to .config)
    --profile    : profile file.  This is used to select an alternate
                   set of userspace packages, this is saved and used
                   on later runs of ltib (e.g config/profiles/max.config)
    --rcfile|r   : use this resource file
    --batch|b    : batch mode, assume yes to all questions
    --force|f    : force rebuilds even if they are up to date
    --reinstall|e : re-install rpms (but don't force rebuild)
    --nodeps|n   : turn off install/uninstall dependency checks
    --conflicts|k : don't force install rpms that have file conflicts
    --keepsrpms|s : keep the srpms after the build (deleted by default)
    --verbose|v  : more output
    --dry-run|d  : mostly a dry run (calls to system are just echos)
    --continue|C  : try to continue on package build errors (autobuilds)
    --version|V  : print the application version and quit
    --noredir|N  : do not redirect any output
    --deploy|D   : run the deploy scripts even if build is up to date
    --dlonly     : just download the packages only
    --dltest     : test that the BSP's packages are available
    --leavesrc|l : leave the sources unpacked (only valid for pkg mode)
    --hostcf     : (re)configure/build/install the host support package set
    --help|h     : help on usage
```

Eseguendo ./ltib senza nessun argomento, si avrà quanto segue:

1. Si installano i pacchetti comuni (solo la prima volta)
2. Ci viene chiesto di scegliere la piattaforma target (solo la prima volta e solo attraverso CVS)
3. Si entra nel menu di configurazione principale e i valori impostati sono quelli di default
4. Si costruisce/installa i pacchetti scelti come richiesto

-m config
Si entra nel menu di configurazione della piattaforma. LTIB non prosegue a costruire i pacchetti dopo che siamo usciti dal menu di configurazione.

—**configure**
Si entra nel menu di configurazione della piattaforma. Dopo l'uscita dal menu, LTIB costruisce/installa i pacchetti scelti come richiesto.

-m clean
Questo comando disinstalla tutti i pacchetti rpm per questo target. Questo effettivamente cancella tutto il contenuto della directory rootfs. Nota che:

- Ci possono rimanere dei file in rootfs se abbiamo l'NFS caricato sul target (per esempio /var/log/xx).
- Non vengono cancellati i file binari rpm, così si può rieseguire ltib, senza nessun cambiamento, reinstallando i binari rpm selezionati.

-m distclean
Questo modo è usato per rimuovere completamente tutti i file del progetto LTIB corrente, come dovrebbe essere da una nuova installazione. Non rimuove nessun file dall'area comune. Questo modo viene generalmente usato con la versione CVS in modo tale che si possa ritornare alla videata di selezione di una diversa piattaforma.

-m release
Questo incapsula il progetto LTIB corrente in un'immagine iso in modo tale che non venga utilizzato l'accesso di rete.

-m shell
Questo è una funzione per lo sviluppatore. Permette di entrare nella shell del prompt con l'ambiente impostato esattamente come se stesse costruendo un file spec di una rom. In questa situazione, tutte le variabili di ambiente, e il compilatore sono impostati come si desidera. Questo può essere importante quando viene prima sviluppato un pacchetto e si sta tentando di apportare delle modifiche così che possa essere poi cross-compilato.

-m listpkgs
Questo modo è usato per mostrare tutti i pacchetti disponibili, se sono selezionati e le loro licenze. Qui abbiamo un esempio della piattaforma tqm823l:

```
$ ./ltib -m listpkgs
------------------- ------------ ------- ------ -----------------------------------------
Package             Spec file    Enabled License Summary
------------------- ------------ ------- ------ -----------------------------------------
DirectFB-0.9.24-1   DirectFB         n   LGPL   DirectFB is a graphics library for embedded syst
NAS-config-1.0-1    NAS-config       n   GPL    NAS setup scripts and instructions
alsa-lib-1.0.10-0   alsa-lib         n   distributab A libraries for ALSA (Advanced Linux Sound Archi
```

—**preconfig**
Per configurare e costruire la piattaforma viene usato il file di configurazione specifico. Questa configurazione deve essere consistente con il target corrente. Lo scopo di questa opzione è di permettere un metodo veloce di cambiamento tra un insieme di punti di configurazione e un altro. Gli sviluppatori possono usare questo come mezzo di modifiche per le loro configurazioni.

—**profile**
Questo è simile al preconfig, eccetto che cambia solo la selezione dei pacchetti nello spazio utente. Lo scopo principale di questo sistema è la possibilità di creare dei profili, i quali vengono successivamente usati per certi specifici tipi di sistema. Per esempio, un root filesystem minimale, un sistema completo, etc.

—**rcfile**
Questa opzione viene raramente usata perché permette di sovrascrivere il file di default di LTIB.

—**force|-f**
Questa opzione forza tutti i pacchetti o alcuni ad essere ricostruiti.

—**reinstall|-e**
Questa opzione forza tutti i pacchetti o alcuni ad essere reinstallati. Non forza la ricostruzione dei pacchetti.

–**nodeps|-n**
Questo comando raramente viene usato, perché disattiva la funzione di testare la dipendenza di installare/disinstallare gli rpm.

—**conflicts|-k**
Normalmente LTIB potrebbe permettere di sovrascrivere file che sono stati installati. Questa è una scelta di progettazione, la quale permette di scalare il filesystem da piccole a grandi dimensioni. Questo raramente viene usato, perché disattivare questa opzione può permettere ad un pacchetto di installare i suoi file ed andare in conflitto con un altro pacchetto.

—**keepsrpms|-s**
Normalmente LTIB costruisce i binari rpm. Questa opzione raramente viene usata perché causa la costruzione dei srpms. Questo può essere utile se un sviluppatore vuole condividere un pacchetto sorgente.

—**verbose|-v**
Usando questa opzione si possono ottenere un maggiore numero di informazioni di debug da LTIB.

—**dry-run|-d**
Questa opzione viene usata per testare l'esecuzione di LTIB. Può mostrare quali comandi possono essere eseguiti, senza eseguirli direttamente.

—**continue|-C**
Normalmente LTIB si ferma quando incontra un errore quando viene costruito un pacchetto. Con questa opzione, LTIB riporta l'errore, ma continua l'esecuzione. Lo scopo principale di questa opzione è di fornire un'autobuilding.

—**version|-V**
Questa opzione mostra il numero di versione di LTIB.

—**noredir|N**
Normalmente, durante la prima esecuzione di LTIB, quando i pacchetti comuni vengono costruiti e installati, l'output viene indirizzato al file host_config.log.

—**deploy|-D**
Questa opzione forza l'esecuzione di entrare nella sezione deploy di LTIB, spesso se la build è stata aggiornata.

—**dlonly**
Questa opzione viene raramente usata perché fornisce un modo per scaricare tutti i sorgenti e le patch che possono essere richiesti per costruire la configurazione di LTIB corrente, senza eseguire il build. La principale ragione per questa opzione è di supporto alle persone che hanno una connessione temporanea (come VPN/proxy). Questo permette a loro di ottenere tutta l'attività di rete prima che vengano costruiti.

—**dltest**
Questa opzione è usata per testare se i pacchetti, che sono configurati per la costruzione, possono essere scaricati dalla rete. Lo scopo principale di questo test è di vedere se i source/patch sono disponibili all'indirizzo http://bitshrine.org/gpp

—**hostcf**
Questa opzione viene raramente usata perché permette ad un utente di riconfigurare/ricostruire/reinstallare i pacchetti comuni. Viene principalmente usata dai sviluppatori di LTIB come tool per LTIB quando viene testato su diverse distribuzioni di host.

—**help**
Questa mostra le opzioni della linea di comando.

3.5.2 Operazioni su un singolo pacchetto

–p pkg
Build/installa il pacchetto. Lo scopo è di permettere alle persone di sviluppare nuovi pacchetti e collegarli ai loro file spec, prima che il pacchetto venga completamente aggiunto al sistema. Nota:

- In questo caso, pkg è attualmente il nome del file spec, senza l'estensione .spec
- Questa opzione non lascia il codice sorgente spacchettato

-p pkg –leavesrc|-l
Questa opzione informa LTIB di lasciare i sorgenti di un pacchetto spacchettati dopo che è stato costruito e installato. -p pkg -p prep In questo modo,

il pacchetto viene spacchettato e vengono applicate le patch. I sorgenti si trovano in rpm/BUILD/xxx

-p pkg -m scbuild

In questo modo, l'esecuzione passa alla sezione %Build del file spec. Questo comando esegue l'opzione -scbuild dei rpm. Questa opzione assume che prima si sia eseguito il prep del pacchetto.

-p pkg -m scinstall

In questo modo, si esegue la sezione %Install del file spec. Questo comando esegue l'opzione -scinstall dei rpm. Questa opzione assume che prima siano stati eseguiti prep e scbuild del pacchetto.

-p pkg -m scdeploy

In questo modo, LTIB internamente esegue la sezione %Install del file spec del pacchetto, poi crea un rpm temporaneo e lo installa nell'area rootfs. Questo modo assume che sia stato eseguito prima il prep e poi scbuild del pacchetto.

-p pkg -m patchmerge

In questo modo, LTIB può generare una patch delle differenze tra i sorgenti spacchettati (dopo aver eseguito make distclean) e i sorgenti di questo pacchetto prima dei cambiamenti. Questa patch viene salvata in LPP e il file spec viene aggiornato con i riferimenti di questa nuova patch. Lo scopo è di permettere ai sviluppatori di prelevare i cambiamenti per un pacchetto dopo che hanno completato il loro lavoro/test.

3.6 Alcuni esempi generali

3.6.1 Esempio di work-flow di installazione di un pacchetto

1. Spacchettare i sorgenti e applicare tutte le patch

   ```
   ./ltib -m prep -p <package>
   ```

2. Editare o aggiungere file sotto la directory rpm/BUILD/package/

3. Costruire il pacchetto con eventuali cambiamenti

   ```
   ./ltib -m scbuild -p <package>
   ```

4. Una volta che il pacchetto è stato costruito con successo, eseguire la fase di installazione

   ```
   ./ltib -m scinstall -p <package>
   ```

5. Testare il pacchetto prima di apportare i cambiamenti

   ```
   ./ltib -m scdeploy -p <package>
   ```

6. Ripetere i passi 2-5 fino a che non si ottiene i risultati desiderati
7. Generare una patch e aggiornare il file spec

```
./ltib -m patchmerge -p <package>
```

8. Manualmente eliminare i file di patch
9. Costruire e installare il pacchetto

```
./ltib --p <package>
```

10. Una volta ottenuto tutti i cambiamenti, caricare la patch in GPP http://www.bitshrine.org/cgi-bin/gpp_upload.cgi
11. Caricare anche il file spec se si ha i permessi di scrivere in CVS

```
cvs commit dist/lfs-5.1/<pkg>/<pkg>.spec
```

Se non si ha i permessi di scrivere in CVS, si può spedire la patch alla mail list di LTIB.

3.6.2 Come aggiungere un nuovo pacchetto

Supponiamo di avere la nostra directory con i file sorgenti:

1. Pulire la directory dei sorgenti e poi costruire una tarball, per esempio:

```
cd <pacchetto>-x.y
make clean
cd ..
tar zcvf <pacchetto>-x.y.atr.gz <pacchetto>-x.y
```

2. Spostare il pacchetto tarball alla LPP così LTIB lo può trovare

```
mv <pacchetto>-x.y.tar.gz /opt/freescale/pkgs/
```

3. Creare un file spec usando il template esistente

```
mkdir dist/lfs-5.1/<pacchetto>
cp dist/lfs-5.1/template/template.spec dist/lfs-5.1/
   <pacchetto>/<pacchetto>.spec
```

4. Editare il file e apportare le modifiche in base alle necessità. I campi da modificare sono:

Campo	**Descrizione**
Summary	Descrizione del pacchetto
Name	Nome del pacchetto (è il nome del tarball)
Version	Versione del pacchetto (specificato x.y nel tarball)
Release	Inizia con 1 e viene modificato ogni volta che il file spec viene modificato
License	Esempio GPL/LGPL/BSD
Group	Se esiste su un rpm, copia da rpm -qi. Nel caso contrario, scegli qualcosa da /usr/share/doc/rpm-/GROUPS
%Build	Spesso è richiesto di aggiungere -host=$CFHOST -build=%_build

5. Spacchettare il nuovo pacchetto

```
./ltib -m prep -p <pacchetto>
```

6. Eseguire qualche cambiamento se è necessario per ottenere la compilazione corretta

7. Costruire il pacchetto con le modifiche

```
./ltib -m scbuild -p <pacchetto>
```

8. Quando il pacchetto compila in modo corretto, passare alla fase di installazione

```
./ltib -m scinstall -p <pacchetto>
```

9. Installare il pacchetto nella directory del root filesystem NFS

```
./ltib -m scdeploy -p <pacchetto>
```

10. Una volta che è stato eseguito tutto correttamente, prelevare le modifiche

```
./ltib -m patchmerge <pacchetto>
```

Qualsiasi modifica che è stata fatta può essere messa in un file patch e copiata in /opt/freescale/pkgs.

11. Caricare i sorgenti in GPP per renderli disponibili al pubblico

```
cvs add dist/lfs-5.1/<pacchetto>
cvs add dist/lfs-5.1/<pacchetto>/<pacchetto>.spec
cvs commit -m "added new_package" dist/lfs-5.1/
    <pacchetto>/<pacchetto>.spec
```

Per introdurre un nuovo pacchetto nel file di configurazione del sistema, si deve seguire i seguenti passi (esempio per il pacchetto *strace*):

1. Editare il file config/userspace/package.lkc, questo è ordinato in modo alfabetico. Appena dopo PKG_SKELL aggiungiamo l'entry:

```
config PKG_STRACE
bool "strace"
```

2. Editare il file config/userspace/pkg_map. Questo è ordinato in base a come vengono costruiti i pacchetti (in base quindi anche alle dipendenze dei pacchetti). Aggiungere quindi il pacchetto in base all'ordine che deve essere costruito e aggiungere una entry al file, per esempio:

```
PKG_GDB = gdb
PKG_STRACE = strace
```

3. Caricare i cambiamenti

```
cvs commit -m "add new_package"
```

3.6.3 Come aggiungere un demone all'avvio

1. Aggiungere il pacchetto che fornisce il servizio (come sopra)

2. Scrivere un script di avvio. La via più semplice è di prendere uno script già esistente e di modificarlo. I script non devono avere dei prefissi numerici. Per esempio:

```
etc/rc.d/init.d/nome
```

3. Aggiungere una entry nel file config/userspace/sysconfig.lkc, per esempio:

```
config SYSCFG_START_NAMED
depend PKG_NAMED
bool "start named (nameserver)"
default y
```

4. Aggiungere un'entry in dist/lfs-5.,1/sysconfig/sysconfig.spec per eseguire questo nuovo servizio:

```
if ["$SYSCFG_START_NAMED" = "y" ]
then
   named=named
fi
```

5. Aggiungere *named* nella linea all_service nella posizione in cui si vuole che venga eseguito

6. Aggiungere *named* nella linea all_services_r (questo è il contrario della linea precedente, per l'arresto della board)

7. Aggiungere *$named* nel file cfg_service e cfg_services_r

3.7 Codici sorgenti

I codici sorgenti se non sono disponibili in un'area comune di cache, vengono scaricati e messi nell'area comune. Se non si usa un'immagine iso, devono essere pre-installati nell'area comune.

Si possono scaricare i sorgenti da GPP/LPP/PPP.
Il PPP (Private Package Pool) è un accesso privato al server http che può essere opzionalmente usato come un'area di memoria per sorgenti/patch private.
Il GPP (Global Package Pool) è un server http pubblico che è usato per memorizzare source/patch per LTIB.
Il LPP (Local Package Pool) è un'area locale comune che viene usata per memorizzare tutte le sorgenti/patch che sono state eseguite da LTIB. Quando LTIB necessita di costruire rpms, il file di spec viene analizzato. Dopo essere stato analizzato, il costruttore (builder) ha una lista di sorgenti e patch che sono necessarie per costruire il pacchetto. Il costruttore poi esegue quanto segue:

- Testa se il file/collegamento è presente in rpm/SOURCES, altrimenti

- Testa se il file è presente nella directory LPP (vedi la entry &lpp nel file .ltibrc). Se esiste, viene fatto un link verso rpm/SOURCE e viene usato, altrimenti

- Il costruttore (builder) scarica il file da PPP (se specificato da &ppp_url in .ltibrc) in LPP usando wget. Se ha successo, viene creato il link verso rpm/SOURCE e viene usato. Se il download non ha successo allora:

- Il costruttore (builder) scarica il file da GPP (se specificato da &gpp_url in .ltibrc) in LPP usando wget. Se ha successo, viene creato il link verso rpm/SOURCE e viene usato. Se il download non ha successo allora:
- Tutti i passi sono stati terminati e quindi il costruttore terminare con un errore

Nota: è possibile inserire un server proxy nel file .ltibrc. Questo può essere necessario se si sta lavorando all'interno di una VPN e si vuole ottenere un GPP esterno su Internet.

3.8 Supporto

La mail-list è il mezzo più usato per ottenere supporto, se si sta usando LTIB da Savannah la mail list la si può trovare a questo indirizzo:

`http://lists.nongnu.org/mailman/listinfo/ltib`

Se si sta usando una immagine iso da http://www.freescale.com è possibile scrivere le richieste di informazione seguendo questi passi:

- Andare alla pagina http://www.freescale.com
- Cliccare su *Support / Technical support*
- Cliccare su *Submit a Service Request*
- Registrarsi con un username ed una password
- Eseguire il login con i dati inseriti precedentemente
- Sulla pagina *New Service Request*:
  - Categoria: Technical Request
  - Argomento: Linux BSP
  - Clicca su *Continue*
- Riempire tutte le informazioni per il servizio richiesto
- Cliccare sul bottone *Submit* in alto della pagina

Capitolo 4

Asterisk

4.1 Introduzione

Asterisk è un progetto PBX open source, progettato dalla Digium. Il suo manutentore principale è Mark Spencer, ma la comunità sta contribuendo al continuo sviluppo di tale progetto. La Digium oltre a dedicarsi allo sviluppo di Asterisk commercializza anche vari componenti hardware in grado di operare con Asterisk. Questi componenti sono delle schede PCI che permettono di connettere il computer alle normali linee telefoniche analogiche, come le linee ISDN. Asterisk lavora principalmente su piattaforma Linux e altre piattaforme Unix, con o senza l'hardware per connettersi alla tradizionale linea telefonica, PSTN.

Il suo nome proviene dal simbolo dell'asterisco *, che negli ambienti Unix ed in DOS rappresenta una specie di carattere "jolly", che può referenziare qualunque carattere o sequenza di caratteri; in analogia, Asterisk è progettato per interfacciarsi con qualunque hardware o software per telefonia.

Con Asterisk è possibile realizzare un PBX di alto livello, che con i normali centralini telefonici non si può ottenere. Con Asterisk è possibile, per esempio:

- Connettere i dipendenti che lavorano da casa con l'ufficio tramite una connessione a banda larga;
- Connettere uffici presenti in vari stati su un unico server, in modo tale di poter comunicare tra i vari uffici a costo zero;
- Fornire a tutti i dipendenti una voicemail, cioè una casella di posta vocale, integrata con una gestione Web;
- Costruire applicazioni di IVR (Interactive Voice Response);
- Spedire SMS;

- Tradurre un testo in un messaggio vocale;
- Avere una musica di attesa personalizzata, tramite file MP3;
- Mettere le chiamate in arrivo in coda;
- Monitorare una chiamata;
- Ottenere il dettaglio delle chiamate, CDR (Call data record).

Asterisk utilizza vari tipi di canali per la connessione, sia con il VOIP che con la linea tradizionale PSTN. Per quanto riguarda la connessione VOIP, Asterisk utilizza principalmente quattro protocolli, che sono: SIP, IAX, MGCP e H.323; mentre per quanto riguarda la connessione alla tradizionale linea telefonica PSTN, utilizza degli hardware come Zaptel, ISDN BRI e PRI.

La connessione delle chiamate in arrivo dall'esterno può avvenire utilizzando molte applicazioni o comandi, che vengono utilizzati per creare un vero e proprio PBX. Si può partire da una semplice istruzione Goto, fino ad arrivare ad applicazioni più complesse come Voicemail e chiamate in conferenza.

Il piano delle chiamate, comunemente chiamato DialPlan, viene memorizzato in un file di testo chiamato "extensions.conf". In questo file le azioni sono connesse alle estensioni, inoltre ogni estensione appartiene a un contesto, che può essere un contesto di default o un conteso specifico. Gli utenti connessi a Asterisk definiscono tutti un contesto a cui fare riferimento, specificato nel file di configurazione di quel canale, in cui Asterisk cerca come deve manipolare la chiamata fatta da quel utente, per esempio può testare il giusto accesso alla linea meno costosa. Nel DialPlan si possono impostare tutte le istruzioni e le situazioni che il centralino deve fare. Si possono configurare contesti che lavorano solo durante una parte della giornata, per esempio un contesto che lavora solo durante l'orario di ufficio e un contesto che lavora solo al di fuori dell'orario d'ufficio, oppure si possono anche includere contesti in altri contesti.

Con il DialPlan si possono per esempio:

- Connettere una chiamata alla Voicemail, se un utente non risponde entro i primi 20 secondi;
- Connettere una chiamata a una conferenza, in multi-party;
- Trasferire una chiamata a un altro Asterisk PBX;
- Bloccare le chiamate che non presentano l'ID del chiamante;
- Cercare dati nel database di Asterisk utilizzando una query sul numero di telefono del chiamante, in modo tale magari di decidere a quale agente inoltrare la chiamata;

- Mettere le chiamate in coda e lasciare che vengano gestite dagli agenti.

Asterisk mette a disposizione un'interfaccia API per gestire le comunicazioni client con il server Asterisk, questo tipo di comunicazione avviene tramite una comunicazione TELNET, inviando quindi dei comandi appropriati. Con questi comandi è possibile per esempio:

- Vedere cosa succede nel PBX
- Fare un debug di vari protocolli, per esempio vedere che utenti sono connessi, oppure vedere le chiamate fatte
- Vedere gli utenti attivi e le chiamate attive
- Cambiare i dati nel database di Asterisk
- Ricaricare la configurazione, mentre il server Asterisk è in esecuzione.

Asterisk utilizza inoltre un protocollo particolare, chiamato IAX (inter-Asterisk Exchange), che permette di collegare vari uffici attraverso uno o più server. Questo protocollo supporta molte connessioni simultanee con un basso carico su una connessioni NAT. In questo modo, si possono costruire dei routing per le chiamate, ed avere un carico di lavoro bilanciato tra i server della rete.

Il server Asterisk viene configurato attraverso un insieme dei file testo, che sono localizzati nella directory /etc/asterisk, su una installazione standard. Tra tutti i possibili file che possono essere configurati, questi possono essere quelle essenziali per un sviluppo medio:

- **Extensions.conf**: file dove viene configurato il piano di lavoro, o dialplan;
- **Sip.conf**: file dove vengono configurati i telefoni e i provider SIP
- **Voicemail.conf**: file dove vengono configurate le voicemail per ogni telefono
- **MusicOnHold.conf:** file dove vengono configurati le musiche d'attesa
- **Manager.conf**: file dove vengono configurati gli utenti che possono accedere al server attraverso le API
- **Features.conf**: file dove vengono configurati dei tasti speciali

Il server Asterisk mette a disposizione varie funzionalità, per esempio:

1. Le applicazioni del dialplan, che danno la possibilità di costruire soluzioni avanzate del PBX: in particolare Asterisk mette a disposizione dei comandi come GotoIf, variabili da testare e impostare, e stringhe da manipolare, che permettono di controllare cosa succede quando un utente chiama una particolare estensione;

2. Le AGI (Asterisk Gateway Interface), applicazioni di interfaccia che estendono il dialplan con delle proprie funzionalità che possono essere scritte in vari linguaggi di programmazione, come: PHP, Perl, C, Java, Unix e altri ancora;

3. Le API, cioè un'interfaccia di comunicazione tra il server e un software client.

Infine è da notare che Asterisk viene distribuito come open source con licenza GPL, quindi significa che chiunque è libero di modificare il codice e distribuire la sua soluzione personalizzata, sempre distribuendo però tale soluzione modificata sotto licenza GPL.

4.2 SIP.CONF

In Internet, ci sono molte applicazioni che richiedono la creazione e la gestione di una sessione, dove per sessione si intende uno scambio di dati tra un'associazione di partecipanti. SIP (Session Initiation Protocol) è un protocollo di livello applicativo che può stabilire, modificare e terminare sessioni multimediali, quali, ad esempio, le telefonate su IP. SIP è un protocollo standard della IETF (RFC 3261), indipendente dal protocollo di trasporto (UDP o TCP) e programmabile, quindi facilmente estendibile. A differenza di altri protocolli, SIP dà supporto a servizi avanzati come il controllo di presenza, la ricerca del chiamato e il reinstradamento della comunicazione. SIP consente all'utente di decidere come e dove desidera essere raggiunto, aggiornando sul sistema, in modo dinamico, un registro con lo stato di tutti gli utenti. Le applicazioni basate su SIP rendono inutili gli elenchi di numeri e contatti da tenere costantemente aggiornati, la cui gestione è sempre onerosa nelle grandi aziende. Queste funzioni sono, infatti, demandate ad appositi Proxy Server, che sono in grado di localizzare una persona indipendentemente dal dispositivo di comunicazione che usa e dal luogo in cui si trova, in base ovviamente alla specifica volontà del corrispondente di voler ricevere o meno le chiamate. Per questi motivi il protocollo SIP è il più diffuso per quanto riguarda la telefonia su IP.

Il file di configurazione SIP.CONF contiene tutte le configurazioni dei telefoni o provider SIP per la comunicazione. Il file ha l'aspetto di tutti i

file di configurazione del server Asterisk, cioè è suddiviso in sezioni: normalmente è presente una sezione generale, chiamata *general*, dove vengono definiti dei parametri standard utilizzabili per tutti i terminali SIP, inoltre sono presenti varie sezioni, una per ogni telefono o provider SIP, in cui vengono configurati i parametri necessari a quel terminale.

Nella sezione [general] possono essere presenti i seguenti parametri:

- **allow=<codec>** : attiva quel determinato codec per la comunicazione. I codec disponibili sono: g723, gsm, ulaw, alaw, g726, adpcm, slin, lpc10, g729, speec, ilbc. Normalmente per le comunicazioni sono sufficienti i codec: gsm, ulaw e alaw.
- **disallow=all**: disattiva tutti i codec. Di solito viene messo prima di attivare una serie di codec, per disattivare tutti quelli precedenti.
- **bindaddr=0.0.0.0**: è l'indirizzo IP usato dal server per mettersi in ascolto delle richieste, per default viene impostato a 0.0.0.0 che rappresenta tutte le interfacce disponibili ed è utile quando una macchina ha indirizzi IP multipli.
- **bindport=5060**: è la porta utilizzata dal server Asterisk per mettersi in ascolto delle connessioni SIP. Di default viene utilizzata la porta 5060.
- **context=<nomecontesto>**: è il nome del contesto di default che viene usato quando nella definizione di un terminale SIP non viene specificato il contesto da usare. Questo tipo di contesto dovrà essere presente nel dialplan, cioè in 'extensions.conf', e verrà usato quando un terminale SIP effettuerà una chiamata, cioè è il punto di partenza di tutte le chiamate.
- **language=<tipo>**: definisce quale lingua viene usata per le comunicazioni
- **register => username:password@host:porta/estensione**: questo parametro viene usato per registrarsi a un provider SIP. Di solito la porta utilizzata è la 5060 mentre l'estensione definisce quale estensione viene chiamata quando arriva una chiamata da quel provider. Username e password sono i parametri usati per la connessione al provider.

Il file prosegue con la definizione dei terminali SIP, cioè di telefoni o provider. Ogni sezione viene rappresentata da un nome racchiuso tra parentesi quadre, e definisce i parametri utilizzati per quel terminale. I parametri che si possono trovare sono:

- **username**: è la username che viene usata per identificare il terminale connesso al server Asterisk;

- **secret**: è la password associata alla username per identificare il terminale connesso al server Asterisk;
- **callerid="nome" <numtel>**: definisce il nome utente e il numero di telefono che viene visualizzato quando viene effettua una chiamata;
- **type**: definisce quale relazione deve avere il terminale con il server. I valori possibili sono: *peer* che permette all'entità di ricevere solo le chiamate, *user* che permette di eseguire solo le chiamate e *friend* che permette sia di ricevere che di effettuare le chiamate dall'entità;
- **context=<nomecontesto>**: se viene specificato durante la configurazione di un telefono, significa che per effettuare una chiamata si vuole usare un contesto diverso da quello specificato nella sezione *general*, quindi bisogna inserire in questo campo quale contesto usare, in caso contrario verrà usato il contesto di default. Se invece si usa questo parametro nella definizione di un provider SIP, allora si definisce quale contesto viene usato quando arriva una chiamata esterna da quel provider.
- **defaultip=<indirizzoIP>**: viene specificato l'indirizzo IP di default del telefono.
- **host**: in questo campo si possono mettere o l'indirizzo IP del terminale oppure il nome dell'host. Normalmente quando si fa riferimento a un telefono viene definito con la parola *dynamic*, in modo tale che si possono assegnare diversi indirizzi IP al telefono che si sta configurando, ma Asterisk si riferisce sempre allo stesso, facendo riferimento alla username e password.
- **mailbox**: indica quale mailbox utilizzare per questo telefono. La mailbox definita in questo campo deve essere configurata nel file "voicemail.conf".
- **allow=<codec>** : ha la stessa funzione che svolge nella sezione *general*, quindi attiva un determinato codec per la comunicazione.
- **disallow=all**: ha la stessa funziona che svolge nella sezione general, quindi disattiva tutti i codec.
- **fromuser**: viene usato per definire un provider SIP e sostituisce il parametro callerid. In questo campo normalmente viene inserito il numero di telefono assegnato dal provider.
- **fromdomain**: come il campo *fromuser* anche questo viene usato quando si definisce un provider SIP. Questo parametro definisce il campo "From:" nei messaggi SIP scambiati.

- **dtmfmode**: definisce in quale modo vengono gestiti i toni DTMF. Si possono definire quattro tipi di toni DTMF: inband, rfc2833, info, auto.

Quelli appena elencati sono i parametri più comunemente usati per una configurazione base del server Asterisk. Nel seguito viene riportato un esempio in cui sono stati configurati tre telefoni SIP, rispettivamente con numeri di telefono interni, 200, 201. Inoltre è stato configurato un provider di esempio.

```
[general]
context=local
bindport=5060
bindaddr=0.0.0.0
language=it
disallow=all
allow=gsm
allow=ulaw
allow=alaw

register => usernameprovider:passwordprovider
@ipserver:numporta/200

[200]
username=200
secret=200
callerid="200" <200>
type=friend
host=dynamic
mailbox=200

[nomeprovider]
username=usernameprovider
secret=passwordprovider
fromuser=numerotelefonoprovider
fromdomain=ipserver
host=ipserver
context=local
type=friend
dtmfmode=rfc2833
```

4.3 VOICEMAIL.CONF

In questo file di configurazione vengono impostate le Voicemail di ogni telefono SIP configurato nel punto precedente. Il file può essere visto suddiviso in tre sezioni principali: una sezione *general*, in cui vengono definite delle caratteristiche generali per tutti i telefoni SIP, una sezione definita *zonemessages* in cui vengono definiti i fusi orari e i formati di notifica dell'orario, e infine troviamo una o più sezioni in cui vengono definiti le Voicemail per ogni telefono SIP.

- **Nuovazona**: è il nome che viene usato per definire una nuova zona.
- **Paese**: è il nome del paese in cui viene definita la nuova zona.
- **Città**: definisce la città all'interno del paese definito
- **Opzioni**: è una lista di opzioni che possono essere aggiunte per personalizzare l'annuncio dell'ora.

Le definizioni di Paese/Città sono definite nell'installazione di Linux e sono definite nel file /usr/share/zoneinfo.

Uno o più file audio possono essere inseriti nella variabile *Opzioni* ed inoltre possono essere aggiunti anche altri caratteri speciali come:

- A o a: giorno della settimana (Lunedì, Martedì, etc.)
- B o b: nome del mese (Gennaio, Febbraio, etc.)
- D o e: numero del giorno del mese (1,2,3,4,5, etc.)
- Y: anno
- I o i: ora, espressa in 12 ore
- H: ora espressa nelle 24 ore
- M: minuti
- P o p: AM o PM
- Q: oggi, ieri
- R: ora espressa in 24 ore, contenente anche i minuti.

Infine analizziamo i contesti dove vengono definite le voicemail. Questi nomi sono arbitrari e verranno usati ovunque negli altri file di configurazione. Per ogni contesto definito possono essere presenti più righe in cui si definiscono le voicemail. Ogni riga ha la seguente forma:
numero_estensione => password, username, emailutente,pageemail,opzioni

- **numero_estensione**: rappresenta il numero dell'estensione che definisce questa voicemail e che verrà utilizzata nel dialplan;
- **password**: è la password che l'utente dovrà immettere per accedere alla sua voicemail;
- **username**: definisce la username della voicemail;
- **emailutente**: definisce l'indirizzo e-mail in cui verrà spedito il messaggio di notifica;
- **pageemail**: definisce un indirizzo e-mail di una pagina, in modo tale che quando viene lasciato un messaggio dal chiamante, una pagina viene spedita all'indirizzo e-mail definito in questo campo;
- **opzioni**: è un campo in cui possono essere definite delle opzioni che sovrascrivono le impostazioni di default definite nella sezione general. Ogni opzione viene suddivisa dalle altre attraverso una barra verticale "|".
  Esempio:
  123=>2048,Joe User,juser@somewhere.net,,tz=san-diego|attach=yes

Riportiamo di seguito un esempio in cui sono stati configurate due voicemail, in riferimento ai telefoni sip configurati precedentemente.

```
[general]
attach=yes
delete=no
maxsilence=10
silencethreshold=128
serveremail=asterisk
maxmessage=180
maxmsg=100
minmessage=3
format=wav49
maxgreet=60
skipms=3000
maxlogins=3
review=no

[zonemessages]
italia=Europe/Rome|'vm-received' Q 'digits/at' IMp

[default]
200 => 200,200,fabio.montemaggiore@email.it
201 => 201,201,fabio.montemaggiore@studenti.unipd.it
```

4.4 MANAGER.CONF

Questo file contiene le informazioni necessarie per stabilire una connessione TCP/IP, tramite Telnet, per gestire tramite le API il server Asterisk. Il Manager di Asterisk permette, quindi, ad un programma client di connettersi ad Asterisk e di inviare comandi, o leggere degli eventi dal PBX, su un flusso dati TCP/IP. Possiamo quindi creare qualsiasi programma che sia in grado di gestire da remoto il server Asterisk, tramite una connessione TCP/IP.

Per la comunicazione viene usato un semplice protocollo, definito come una riga di testo corrispondente ad una coppia di valori "chiave:valore". Ogni pacchetto è rappresentato da un insieme di righe "chiave:valore" che terminano con un CRLF.

Ci sono tre tipi di pacchetti che possono essere trasmessi e che ora si andranno ad analizzare:

- **ACTION**: è il pacchetto che viene trasmesso dal client al server. Questo pacchetto contiene la richiesta da parte del client di soddisfare una sua richiesta. Ad ogni pacchetto viene aggiunto un ID, in modo tale che ad ogni richiesta viene associata una risposta univoca;
- **RESPONSE**: è il pacchetto che il server invia al client dopo aver ricevuto un pacchetto ACTION;
- **EVENT**: è il pacchetto che viene spedito dal server al client quando avviene un determinato evento all'interno di Asterisk. Per esempio quando avviene una chiamata viene spedito una serie di pacchetti al client per comunicargli cosa sta avvenendo sul server.

Anche questo file di configurazione contiene una sezione **general** in cui vengono impostati dei parametri standard, ed inoltre vengono aggiunti una o più sezioni che definisco i parametri per una connessione TCP/IP.

La sezione general contiene:

- **enable**= yes/no: attiva o disattiva la possibilità di effettuare la connessione TCP/IP;
- **portno**=5038: definisce su quale porta Asterisk deve rimanere in ascolto per soddisfare le richieste. Di solito viene utilizzata la porta 5038;
- **bindaddr**: definisce su quali indirizzi IP deve rimanere in ascolto. Per default viene utilizzato l'indirizzo 0.0.0.0 in modo tale che siano possibili tutti gli indirizzi IP.

Di seguito vengono definite tutte le possibili connessioni che il server Asterisk può accettare, in particolare si andranno a definire Username, Password e indirizzi IP. Ogni sezione viene nominata con la Username utilizzata per la connessione. I parametri che possono esserci sono:

- **secret**: è la password di questa connessione;
- **deny=0.0.0.0/0.0.0.0**: viene definito un range di indirizzi IP che non possono connettersi al server. Di solito viene inserito prima del parametro "permit", in modo tale che vengono vietati tutti gli indirizzi IP e successivamente si attivano solo quelli di nostro interesse;
- **permit=192.168.100.1/255.255.255.0**: viene definito quali indirizzi IP possono collegarsi al server;
- **read=system,call,log,verbose,comand,agent,user**: definisce quali sono gli eventi che la connessione può leggere;
- **write=system,call,log,verbose,comand,agent,user**: definisce quali sono le azioni che la connessione può effettuare.

Vediamo un esempio di configurazione in cui possiamo impostare un solo utente che si colleghi al server Asterisk.

```
[general]
enabled = yes
port = 5038
bindaddr = 0.0.0.0

[asterisk-gestione]
secret=asterisk-gestione
deny=0.0.0.0
permit=192.168.1.1/255.255.255.0
read = system,call,log,verbose,command,agent,user
write = system,call,log,verbose,command,agent,user
```

Tra le varie azioni che il Manager mette a disposizione, viene preso in esame solo quelle più utilizzate:

- **LOGIN**: utilizzato per effettuare la connessione del client al server.

```
Action: login
Username: tuaUsername
Secret: tuaPassword
```

- **LOGOFF**: utilizzato per chiudere la connessione del client al server.

```
Action: logoff
```

- **ORIGINATE**: utilizzato per effettuare le chiamate.

```
Action: originate
Channel: canale (es.SIP/200)
Context: contesto (contesto usato per effettuare
         le chiamate. Es. local)
Exten: numeroTelefono (numero da chiamare)
Priority: 1 (è la priorità che viene usata nel
             contesto, di solito è 1)
```

- **HANGUP**: utilizzato per chiudere una conversione. In particolare chiude un determinato canale attivo.

```
Action: hangup
Channel: canaleAttivo (es. SIP/200-44d335d0)
```

- **REDIRECT**: utilizzato per trasferire una chiamata verso un altro numero.

```
Action: redirect
Channel: canaleAttivo (es. SIP/200-44d335d0)
Context: contesto (contesto usato per chiamare
         l'altro numero. Es. local)
Exten: numeroTelefono (numero di telefono
       verso cui viene trasferita la chiamata
Priority: priorità (priorità usata nel contesto.
          Di solito viene usata la priorità 1)
```

- **MONITOR**: utilizzato per registrare una chiamata. L'azione Monitor effettua una registrazione su due file, un file per il chiamante e un file per il chiamato. Se si vuole unire i due file si deve aggiungere la riga Mix con parametro uguale a 1.

```
Action: monitor
Channel: canaleAttivo (es. SIP/200-44d335d0)
File: nomeFile
Mix: 1
```

- **COMMAND**: utilizzato quando vengono inoltrati dei comandi al server. Questi comandi sono quelli utilizzati nella console di Asterisk. In particolare questa funzione diviene utile per manipolare le informazioni sul database di Asterisk.

  ```
  Action: command
  Command: comando
  ```

  Il comando è quello che verrebbe usato tramite console di Asterisk. Es. <database put x y 1>; questo imposta un valore 1 nel database appartenente alla famiglia x e con chiave y.

4.5 MUSICONHOLD.CONF

Questo file contiene le impostazioni per la musica d'attesa. Asterisk utilizza un pacchetto aggiuntivo per eseguire i file MP3. Tale pacchetto deve essere scaricato dal seguente indirizzo:

```
http://www.mpg123.de/cgi-bin/sitexplorer.cgi?/mpg123
```

A questo punto si deve decomprimere tale pacchetto con il seguente comando digitato nella console di Linux:

```
tar -zxvf mpg123.tar.gx
```

Successivamente si compila il modulo e se lo installa, con i seguenti comandi:

```
make linux
make install
```

Infine si crea un link nella directory /usr/bin:

```
ln -s /usr/local/bin/mpg123 /usr/bin/mpg123
```

Una volta installato il pacchetto, si passa alla configurazione del file "musiconhold.conf". Questo file presenta varie sezioni in cui vengono configurati le possibili musiche d'attesa. Per ogni sezione è possibile configurare i seguenti parametri:

- **mode**: definisce il modo in cui viene eseguito le musiche di attesa. Normalmente si usa il modo quietmp3.

- **directory**: viene specificata la directory in cui vengono cercati i file da eseguire. Normalmente Asterisk esegue i file presenti nella directory /var/lib/asterisk/mohmp3

- **application**: se non viene usato il programma standard di Asterisk per l'esecuzione dei file MP3, cioè il programma MPG123, è possibile definire in questo campo la directory del programma da utilizzare

Esempio del file di configurazione installato in un server Asterisk:

```
[default]
mode=quietmp3
directory=/var/lib/asterisk/mohmp3
```

4.6 FEATURES.CONF

In questo file, oltre a definire le impostazioni per gestire il parcheggio di una chiamata, è possibile definire un insieme di tasti che digitati durante una conversazione permettono di far eseguire determinate operazioni. In particolar modo con questo file è possibile definire quali tasti permettono di trasferire una chiamata, registrare una chiamata, parcheggiare una chiamata e altre funzioni.

In questo file sono presenti tre sezioni particolari: una sezione **general** in cui vengono impostati dei parametri standard per il parcheggio di una chiamata e i tempi per il trasferimento di una chiamata; una sezione **featuremap** in cui vengono definiti alcuni tasti particolari standard di Asterisk; ed infine una sezione **applicationmap** che definisce i tasti personalizzabili dall'utente.

La sezione **general** definisce:

- **Parkext => 700**: definisce quale estensione digitare per parcheggiare una chiamata, cioè definisce quale numero di telefono chiamare per parcheggiare una chiamata;
- **Parkpos => 701-720**: definisce il range delle estensioni in cui possono essere parcheggiate le chiamate. Con questa configurazione possono essere parcheggiate fino a 20 chiamate;
- **Context => parkedcalls**: definisce in quale contesto vengono parcheggiate le chiamate;
- **Parkingtime => 45**: definisce quanti secondi una chiamata può essere parcheggiata;
- **Transferdigittimeout => 20**: definisce il tempo massimo di attesa, espresso in secondi, per digitare il numero di telefono a cui trasferire la chiamata;
- **Featuredigittimeout = 500**: definisce quanti millisecondi si possono attendere tra la digitazioni di due tasti, per determinare quando un numero è completo e quindi è possibile trasferire la chiamata.

La sezione **featuremap** definisce invece:

- **Blindxfer => *** : definisce quale tasto viene usato per trasferire direttamente una chiamata;
- **Atxfer => #** : definisce quale tasto usare per trasferire una chiamata, però in questo caso il cliente viene messo in attesa e solo dopo che l'utente ha chiuso la comunicazione, la chiamata viene trasferita;
- **Automon => 0** : definisce quale tasto usare per registrare (o monitorare) una chiamata.

L'ultima sezione, applicationmap, definisce delle impostazioni personalizzabili, per esempio si può definire un tasto (es. tasto 8) che durante una conversazione annunci un messaggio. Esempio:

```
[applicationmap]
nome => 8,Playback,messaggio
```

Per utilizzare questa sezione si deve impostare, nel file "extensions.conf", la variabile DYNAMIC_FEATURES con il nome che è stato usato nella configurazione. Nel caso dell'esempio precedente si deve definire DYNAMIC_FEATURE=nome.

Quanto segue è un esempio del file di configurazione presente in un server Asterisk:

```
[general]
transferdigittimeout => 20
featuredigittimeout => 500

[featuresmap]
blindxfer => *
atxfer => #
automon => 0
```

4.7 Database

Asterisk usa al suo interno un database DB1, in modo tale che sia possibile eseguire in realtime delle modifiche al DialPlan senza la necessità di ricaricare tutti i moduli. Il database di Asterisk è strutturato in famiglie e ogni famiglia contiene delle chiavi.

Asterisk mette a disposizione 4 applicazioni per gestire il database:

- **Del(famiglia,chiave)**: cancella la chiave contenuta nella famiglia
- **Deltree(famiglia)**: cancella l'intera famiglia
- **Get(famiglia,chiave)**: restituisce il valore della chiave che appartiene ad una determinata famiglia
- **Put(famiglia,chiave,valore)**: imposta un valore nella chiave, appartenente a una determinata famiglia.

Questi comandi possono essere eseguiti tramite console di Asterisk usando questa sintassi:

- **database del** *famiglia chiave*
- **database deltree** *famiglia*
- **database get** *famiglia chiave*
- **database put** *famiglia chiave valore*

Oppure è possibile intervenire direttamente nel DialPlan con la seguente sintassi:

```
Set(variabile=${DB(famiglia/chiave)}) per leggere dal database
Set(DB(famiglia/chiave)=valore) per scrivere sul database
```

Asterisk oltre ad avere un database interno può essere configurato per funzionare anche con database esterni, per esempio con MySQL. Per usare questo tipo di database bisogna però compilare e installare un modulo aggiuntivo chiamato "Asterisk Add-on", che viene sempre fornito da Asterisk, ma che non viene installato con l'installazione standard.

4.8 Variabili

Asterisk può fare uso al suo interno di variabili globali, di variabili di canale e di variabili di ambiente, utilizzabili come argomenti nei comandi. Le variabili vengono riferite nel DialPlan usando la seguente sintassi:

```
${variabile}
```

Un nome della variabile può essere una qualsiasi stringa alfanumerica che inizia con una lettera. Le variabili globali, che vengono definite dall'utente non sono case-sensitive, per esempio la variabile $FOO e $foo sono la stessa variabile. Mentre le variabili di Asterisk, definite come variabili di canale, sono case-sensitive, per esempio la variabile $EXTEN funziona, mentre $exten non funziona.

In Asterisk vengono definiti tre tipi di variabili:

- **Global**: variabili che possono essere impostate nella sezione [global] del file "extensions.conf", oppure impostate usando il comando Set();
- **Channel**: variabile di canale, cioè ogni canale ottiene il suo spazio di variabili, così non ci sono collisioni tra differenti chiamate e le variabili vengono automaticamente cancellate quando il canale viene chiuso. Anche queste variabili vengono impostate con il comando Set();
- **Enviroment**: sono variabili che forniscono l'accesso alle variabili di ambiente Unix dall'interno di Asterisk.

Se viene usata una variabile global con lo stesso nome della variabile channel, il riferimento sarà quella della variabile channel.

Le principali variabili channel che vengono utilizzate sono:

- **${CALLERID}**: contiene l'ID del chiamante
- **${CALLERIDNAME}**: contiene il nome del chiamante
- **${CALLERIDNUM}**: contiene il numero di telefono del chiamante
- **${CHANNEL}**: contiene il nome del canale attivo
- **${CONTEXT}**: contiene il contesto in cui viene generata l'estensione
- **${DATETIME}**: contiene l'ora corrente nel formato DDMMYYYY-HH:MM:SS
- **${DIALSTATUS}**: contiene lo stato della chiamata
- **${EXTEN}**: contiene l'estensione della chiamata
- **${LANGUAGE}**: contiene la lingua utilizzata

- **${PRIORITY}**: contiene la priorità corrente
- **${TIMESTAMP}**: contiene la data e l'ora corrente nel formato YYYYMMDD-HH:MM:SS
- **${TOUCH_MONITOR}**: contiene il nome del file registrato durante una registrazione di una chiamata.

Inoltre vengono definite alcune variabili utilizzabili con le macro:

- **${ARG1}**: è il primo argomento passato alla macro
- **${ARG2}**: è il secondo argomento passato alla macro
- **${ARGn}**: è l'n-simo argomento passato alla macro
- **${MACRO_CONTEXT}**: contiene il contesto che ha richiamato la macro
- **${MACRO_EXTEN}**: contiene l'estensione che ha richiamato la macro
- **${MACRO_OFFSET}**: viene impostato dalla macro per influenzare la priorità di ritorno dalla macro all'esecuzione normale
- **${MACRO_PRIORITY}**: contiene la priorità dell'estensione che ha richiamato la macro.

Infine troviamo le variabili enviroment:

- **${ENV(ASTERISK_PROMPT)}**: contiene la console (CLI) di Asterisk
- **${ENV(RECORD_FILE)}**: contiene l'ultimo nome del file salvato con il comando Record.

Asterisk oltre a definire un insieme di variabili, dà anche la possibilità di manipolarle, in particolare troviamo:

- **${LEN(foo)}**: restituisce la lunghezza della stringa "foo"
- **${foo:offset:length}**: restituisce una sottostringa di "foo" che inizia dall'offset e length caratteri successivi. Se offset è negativo, inizia dalla parte di destra della stringa. Se length viene omesso, viene presa tutta la sottostringa rimanente
- **${foo}${bar}**: concatena due stringhe

4.9 EXTENSIONS.CONF

Il file di configurazione "extensions.conf" contiene il dialplan di Asterisk, cioè il piano di controllo e il flusso di esecuzione per tutte le sue operazioni. Questo file controlla come le chiamate in ingresso e quelle in uscita vengono gestite e instradate. Questo file è dove viene configurato il comportamento di tutte le connessioni del PBX.

Il contenuto di "extensions.conf" è organizzato in sezioni, di cui due sono essenziali: la sezione **general** e la sezione **global**, in cui troviamo le impostazioni generali per tutto il DialPlan. Oltre a queste due sezioni che non possono mancare troviamo i contesti, cioè gruppi di lavoro che definiscono i comportamenti del DialPlan e oltre ai contesti troviamo anche dei contesti particolari che sono le macro. Le macro possono essere paragonate ad una procedure in un linguaggio di programmazione.

La sezione **general** è la sezione al più alto livello del file ed è dove vengono configurate le poche opzioni riguardanti il DialPlan:

- **static=yes/no**: attiva o meno la possibilità di salvare il dialplan modificato, utilizzando il comando da console "save dialplan".

- **writeprotect=yes/no**: se writeprotect=no e static=yes allora si può salvare il dialplan direttamente da console con il comando "save dialplan".

Il comando "save dialplan" sovrascrive il file esistente di "extensions.conf" con uno di nuovo generato dal DialPlan in esecuzione.

La sezione **globals** è la sezione successiva alla sezione **general** e definisce le variabili globali utilizzabili in tutto il DialPlan. Le variabili globali di Asterisk vengono utilizzate come delle costanti, quindi vengono inizializzate in questa sezione e mantengono il loro valore per tutto il DialPlan. Esse possono essere modificate durante l'esecuzione del DialPlan, ma una volta terminata l'operazione, il valore ritorna quello definito nella sezione. Le variabili globali non sono case-sensitive.

Dopo la sezione "general" e "global", il DialPlan definisce un insieme di contesti. Ogni contesto fornisce un insieme di estensioni. I contesti possono essere usati per realizzare molte caratteristiche importanti del PBX, tra cui:

- **sicurezza**: permette di chiamare lunghe distanze solo da certi telefoni;

- **instradamento**: instradare le chiamate basandosi sulle estensioni;

- **ricevitore automatico**: ricevere le richieste e chiedere a chi chiama di immettere le estensioni;

- **giorno/notte**: permette di avere un comportamento diverso per il giorno e la notte;

- **macro**: creare delle comuni macro da riutilizzare, come se fossero delle procedure in un linguaggio di programmazione.

Quando Asterisk riceve o effettua una connessione di chiamata da parte di un telefono, la chiamata viene indirizzata al contesto specificato nel file di configurazione del telefono. Per esempio se un telefono interno SIP esegue una chiamata, la chiamata viene indirizzata al contesto specificato nel file di configurazione "sip.conf", mentre se arriva una chiamata dall'esterno, la chiamata viene indirizzata al contesto specificato nel file "zapata.conf". Tutti i contesti specificati nei file di configurazione dei canali (es. sip, zapata, etc.) devono essere definiti nel file "extensions.conf".

Quando si definisce un'estensione all'interno di un contesto, si possono usare dei modelli di estensione,come:

- **X** : qualsiasi numero da 0 a 9
- **Z** : qualsiasi numero da 1 a 9
- **N** : qualsiasi numero da 2 a 9
- **[12-5]** : qualsiasi numero appartenente alla lista 1,2,3,4,5
- **.** : è un carattere jolly, che può essere uno o più caratteri. Tali modelli devono essere preceduti dal carattere "_".

Esempio:
_61XX qualsiasi numero che inizia con 61 e che segue con due numeri compresi tra 0 e 9

Un contesto inoltre può includerne un altro con la seguente sintassi:

```
include => "nome contesto da includere"
```

In questo caso il contesto che include un altro contesto eredita tutte le proprietà di quel contesto.

In alcuni casi si possono utilizzare delle estensioni predefinite che Asterisk mette a disposizione, tra le quali:

- **i** : invalid, usata quando viene digitata una estensione non valida
- **s** : start, usata da sola in un contesto accetta qualsiasi tipo di estensione. Normalmente viene usata nelle macro.
- **h** : hangup, usata quando un canale viene chiuso
- **t** : timeout, usata quando scade il tempo in un menu (IVR)
- **T** : timeout assoluto, usata quando scade il tempo di una chiamata

- **: operator**, usata nella voicemail quando si digita lo 0 per contattare l'operatore.

Un'estensione viene definita come una riga di comando che viene eseguita con una certa priorità, espressa dal tag **priority**. Alcuni comandi però, come Dial, Goto, GotoIf, hanno la possibilità di saltare ovunque, basandosi su alcune condizioni.

Quando un'estensione viene chiamata, il comando con priorità 1 viene eseguito, seguito dalla comando di priorità 2, e così via. Questo procede fino a che, o:

- la chiamata chiude;
- un comando restituisce un risultato di -1 (indica che è fallito);
- un comando con priorità superiore non esiste;
- la chiamata viene instradata su una nuova estensione.

La sintassi usata per ogni comando è la seguente:

```
exten => estensione,priorità,comando(parametri)

<estensione>: definisce l'etichetta dell'estensione;
<priorità>: è un intero che definisce la priorità.
Il primo comando che viene eseguito ha priorità 1;
se tale comando non esiste, l'esecuzione non procede;
<comando>: è il nome del comando che viene eseguito;
<parametri>: dipende dal comando. Alcuni comandi non
prendono nessun parametro mentre altri utilizzano
più parametri. Se il comando non ha parametri,
le parentesi possono essere omesse.
```

La macro può considerarsi un speciale contesto in cui si ha la possibilità di passare dei parametri, paragonabile ad una procedura in un linguaggio di programmazione. La macro viene definita allo stesso modo di un contesto, tranne che per il fatto che viene anteposto al nome della macro la parola "macro" ed inoltre l'estensioni all'interno della macro vengono determinate con l'estensione "s". Esempio:

```
[macro-prova]
exten => s,1,Wait(1)
exten => s,2,Playback(messaggio)
```

Questa macro genera il file audio "messaggio".

Nel DialPlan si trovano moltissime applicazioni che servono per costruire e gestire il PBX, nel seguito verranno prese in esame solo alcune applicazioni che vengono usate più comunemente:

- **Answer()**: risponde a un canale che squilla
- **DateTime()**: annuncia la data e l'ora corrente
- **Dial(type/group/identifier,timeout,options,URL)**: effettua una chiamata sul canale *type* (es.SIP), nel gruppo *group* (es. g2) e con estensione *identifier* (es.200). Se dopo timeout secondi il canale non risponde, allora la chiamata viene chiusa. Es.

```
Dial(SIP/200,20,tTwW)
Dial(ZAP/g2/0445308018,20,tTwW)
```

  - *Type*: specifica il tipo di canale.
  - *Gruppo*: è opzionale. Determina con quale gruppo chiamare.
  - *Identifier*: specifica il numero di telefono da chiamare.
  - *Timeout*: è opzionale. Determina quanti secondi si può attendere fino a che la chiamata non venga chiusa.
  - *URL*: è opzionale. Determina un indirizzo URL che può essere spedito al chiamato quando la connessione è avvenuta. Questo succede solo se la tecnologia lo permette.
  - *Options*: è una lista di opzioni che si possono inserire nel comando Dial per controllare la chiamata. In particolare si trovano:
    - t: permette all'utente chiamato di trasferire la chiamata;
    - T: permette all'utente chiamante di trasferire la chiamata;
    - m: genera una musica di sottofondo fino a che il chiamato non risponde;
    - f: permette al chiamante di fare solo chiamate dalla linea che gli è stato assegnato;
    - w: permette al chiamato di iniziare a registrare la chiamata dopo aver premuto un particolare tasto definito nel file "features.conf";
    - W: permette al chiamante di iniziare a registrare la chiamata dopo aver premuto un particolare tasto definito nel file "features.conf".

  Con questo comando è utile gestire i valori di ritorno, in particolare quando il timeout scade, la priorità che viene eseguita è **(n+1)**, dove "n" è la priorità in cui viene eseguito il comando. Quando invece il canale è occupato la priorità che viene eseguita è **(n+101)**, in questo modo è possibile gestire il comportamento da seguire in questo caso. La stessa cosa succede quando il canale non è raggiungibile, cioè quando non è collegato al server oppure quando riscontra qualche problema di connessione e a questo punto l'esecuzione prosegue alla priorità **(n+201)**.

- **Goto([[contesto,]estensione,]priorità)**: salta l'esecuzione a una determinata priorità, estensione o contesto.
  Con questo comando è possibile saltare ad una determinata priorità, oppure saltare ad una determinata estensione e priorità, oppure ancora ad un determinato contesto, estensione e priorità.

  ```
  Goto(priorità)
  Goto(estensione,priorità)
  Goto(contesto,estensione,priorità)
  ```

- **GotoIf(condizione?label1[:label2])**: è simile al comando Goto, ma in questo caso il salto viene condizionato da una condizione. Se la condizione è vera (true) il salto viene fatto alla label1, in caso contrario alla label2. La label1, come la label2, hanno la stessa forma del salto in Goto, cioè può essere sia una priorità, oppure una estensione e una priorità, oppure un contesto, una estensione e una priorità.

  ```
  exten => 206,1,GotoIf($["${CALLERIDNUM}" = "303"]?3:2)
  exten => 206,2,GotoIf($["${CALLERIDNUM}" = "304"]?5:7)
  exten => 206,3,Dial(${SPHONE1},15,rt)
  exten => 206,4,Hangup
  exten => 206,5,Dial(${PHONE2},15,rt)
  exten => 206,6,Hangup
  exten => 206,7,MusicOnHold(default)
  ```

- **GotoIfTime(<times> | <weekdays> | <mdays> |<months> [[context,]extension,]priority]**: questo salto viene condizionato dall'ora e dal giorno in cui viene eseguito. Se l'esecuzione del comando appartiene al range di tempo definito nella condizione, allora la prossima esecuzione seguirà il contesto, l'estensione e la priorità definita, mentre se siamo al di fuori di quel range l'esecuzione prosegue alla priorità successiva a quella del comando GotoIfTime. Esempio:

  ```
  exten => 3000,1,GotoIfTime(08:30-12:30|mon-fri|*|*?
                  chiama,s,1)
  exten => 3000,2,....
  ```

  Se l'esecuzione di tale comando viene eseguito dalle 8.30 alle 12.30, dal lunedì al venerdì, di tutti i mesi e anni, allora viene eseguito un salto al contesto "chiama" di estensione 's' e di priorità 1. Altrimenti se viene eseguita al di fuori di questo orario l'esecuzione prosegue alla priorità 2.

- **Hangup**: chiude il canale attivo

- **Macro(nome,arg1,arg2,...)**: chiama la macro "nome". È possibile passare al comando più argomenti, in modo tale che possono essere gestiti all'interno della macro.

- **MixMonitor(nome.estensione)**: registra una chiamata e salva il file con un determinato nome e estensione. Quando Asterisk registra una chiamata genera due file, uno per il chiamante e uno per il chiamato, con questo comando il file risultante è uno solo dato dall'unione dei due file.

- **Monitor(estensione,nome)**: registra una chiamata. A differenza del comando precedente, questo comando registra la conversazione su due file differenti.

- **MP3Player(nomefile)**: riproduce il file audio MP3 specificato nel parametro.

- **MusicOnHold(classe)**: riproduce una musica di sottofondo specificando la classe usata. La classe viene configurata nel file di configurazione "musiconhold.conf"

- **NoOp(testo)**: con questo comando viene visualizzato nella console di Asterisk un testo definito, oppure può essere usato per visualizzare il contenuto di una variabile. Questo sistema può diventare utile per fare un debug della tracciabilità di esecuzione di una particolare estensione. Esempio:

  ```
  NoOp(${EXTEN})
  ```

  Viene visualizzata l'estensione che ha generato questo comando.

- **Playback(nomefile)**: viene eseguito il file audio specificato. Questo comando legge solo i file audio standard di Asterisk che sono in formato GSM.

- **Playtones(nome)**: viene eseguito il tono specificato. Per esempio se si vuole sentire il tono di occupato allora si deve passare come parametro "busy".

- **Record(nomefile:formato[—silenzio][—maxdurata])**: registra la voce in un file audio con nome 'nomefile' e formato 'formato'. I formati possibili sono: g723, g729, gsm, ulaw, alaw, vox, wav. Inoltre questo comando può avere le seguenti opzioni:

  - **Silenzio**: rappresenta il numero di secondi di silenzio prima che la registrazione si ferma

  - **Maxdurata**: rappresenta il numero di secondi di registrazione massima ammessa.

- **Set(nomevariabile=valore)**: imposta il valore di una variabile. Con questo comando è possibile anche leggere o scrivere un valore su database. Esempio:

```
Set(nomevar=${DB(famiglia/chiave)}) legge un valore
                                    da database
Set(DB(famiglia/chiave)=valore) scrive un valore
                                su database
```

- **SetLanguage(lingua)**: imposta il valore della lingua usata per i file audio.

- **SetMusicOnHold(classe)**: imposta la classe per la musica di sottofondo.

- **Transfer(exten)**: trasferisce una chiamata attiva ad un'altra estensione (numero di telefono).

- **Voicemail([flags],numerobox)**: registra nella casella vocale "numerobox" un messaggio. Questo comando può avere i seguenti flag associati:

  - *s*: se presente viene annunciato al chiamante: "Lascia il tuo messaggio dopo il segnale acustico. Quando finito, riaggancia oppure premi il tasto #".
  - *u*: se presente annuncia il messaggio di non disponibile: "La persona ... non è al momento disponibile".
  - *b*: se presente annuncia il messaggio di occupato: "La persona ... è al momento occupata".

- **VoiceMailMain([s]numerobox)**: entra nel menu della mailbox dell'utente. Se il numero della mailbox viene preceduto dalla lettera "s" allora la password non viene richiesta. Il menu della Voicemail è il seguente:

```
1. Leggi i nuovi messaggi
    3. Opzioni avanzate
      1. Rispondi
      2. Richiama
      3. Intestazione
      4. Chiama fuori
    4. Ascolta il messaggio precedente
    5. Ripeti questo messaggio
    6. Ascolta il prossimo messaggio
    7. Cancella questo messaggio
    8. Inoltra il messaggio ad un'altra cartella
    9. Salva il messaggio in un'altra cartella
    *. Help
    #. Exit
  2. Cambia cartella
  0. Opzioni della mailbox
    1. Registra il messaggio di non disponibile
    2. Registra il messaggio di occupato
    3. Registra il tuo nome
    4. Registra un tuo messaggio temporaneo
    5. Cambia la tua password
    *. Ritorna al menu principale
  *. Help
  #. Exit
```

- **Wait(secondi)**: sospende l'esecuzione per un determinato numero di secondi.

Capitolo 5

Linphone

5.1 Introduzione

Linphone è un'applicazione specifica per la telefonia via internet, Voip, basata sul protocollo SIP. L'applicazione permette di effettuare oltre alle chiamate anche videochiamate includendo al suo interno diversi codec video, oltre che audio, supporta i protocolli IPv4 e IPv6 per il trasferimento dei pacchetti dati, offre inoltre uno strumento per la gestione della rubrica dei contatti. Con Linphone è possibile comunicare liberamente con le persone attraverso internet, tramite voce, ma anche tramite video e messaggi (chat). Linphone usa il protocollo SIP, il protocollo standard per la telefonia tramite internet. E' possibile quindi usare Linphone con qualsiasi operatore che fornisca un provider VOIP SIP. Linphone è free ed inoltre è open-sorce, e quindi è possibile scaricare liberamente il software ma anche è possibile adattarlo alle nostre esigenze, dato che viene rilasciato come open-sorce. Linphone è disponibile per i PC, sia Linux che Windows, per MacOSX e per i telefoni mobili, come Android e iPhone.

5.2 Caratteristiche

Le caratteristiche del core di Linphone sono:

- È un User Agent SIP compliant dell'RFC3261
- Si sdoppia tra il motore liblinphone e la grafica utente (UI): permette cioè di integrare le funzionalità di Linphone in qualsiasi applicazione grafica
- L'audio può essere impostato con i seguenti codec: speex, G711 (ulaw, alaw), GSM. Grazie a dei plugin ulteriori, supporta anche AMR e iLBC.

- Il video può essere impostato con i seguenti codec: H263, H262-1998, MPEG4, theora e H262 (grazie ad un plugin basato su x264), con risoluzione da QCIF(176x144) a SVGA(800x600)
- Supporta l'IPv6
- Supporta qualsiasi webcam con driver V4L o V4L2 sotto Linux e driver Directshow sotto Windows
- Fornisce i messaggi istantanei e il rilevamento della presenza (usando lo standard SIMPLE)
- Fornisce una rubrica
- Fornisce il DTMF (i toni di telefonia) usando SIP INFO o RFC1833
- Supporta la cancellazione dell'eco
- Supporta più proxy SIP
- Supporta il NAT
- Supporto dell'audio
  - Linux: ALSA, OSS, PulseAudio
  - Windows: waveapi
  - MacOSX: audio queues
  - iPhone: AudioUnit
  - Android sound system
- Fornisce la gestione della banda in modo efficiente: i limiti di banda sono segnalati usando SDP, risultando la sessione dell'audio e del video stabiliti con il bitrate della capacità della rete dell'utente
- Si possono usare altri plugin: si possono aggiungere nuovi codec, o nuove funzionalità al core, come la ricerca di indirizzi sip in modo remoto, per esempio
- Infine fornisce tutte le caratteristiche del protocollo standard SIP

5.3 Interfaccia utente

Linphone ha una propria interfaccia grafica che può essere eseguita sia su Linux, sia su Windows che su MacOSX. Tutto questo grazie ad un'interfaccia grafica sviluppata con gtk2.

Linphone inoltre mette a disposizione degli utenti due tipi di interfacce grafiche o più precisamente due tool tramite console. Grazie a queste interfacce a linea di comando è possibile usare Linphone anche da riga di comando rendendo questo software più portabile.

Il primo tool è "linphonec" che è un'interfaccia grafica a linea di comando che può essere usata per gestire completamente Linphone attraverso dei comandi dal prompt dell'utente.

- **linphonec**: avvia linphone da linea di comando

```
$ linphonec
Ready
Warning: video is disable in linphonec, use -V or -C
         or -D to enable
```

- **call sip:num_telefono@ipserver**: per effettuare una chiamata

```
linphonec> call sip:594305005@someproxy.net
```

- **terminate**: per terminare una chiamata

```
linphonec: terminate
```

- **help**: per avere in qualsiasi momento la lista completa dei comandi disponibili

  linphonec> help
  Commands are:

help	Visualizzare i comandi disponibili
call	Chiamare un utente SIP
chat	Chattare con un uri SIP
terminate	Termina la chiamata corrente
answer	Risponde ad una chiamata
autoanswer	Mostra/imposta la modalità di auto-risposta
proxy	Gestisce i proxy
soundcard	Gestisce la scheda audio
webcam	Gestisce la webcam
staticpic	Gestisce l'immagine quando non c'è la webcam
ipv6	Usa l'IPV6
refer	Trasferire la chiamata corrente alla destinazione specificata
nat	Imposta l'indirizzo nat
stun	Imposta il server stun
firewall	Imposta la politica del firewall
call-logs	Visualizza i log delle chiamate
friend	Gestisce gli amici
play	Esegue un file wave
record	Registra in un file wave
quit	Esce da linphonec
register	Registra in una linea per un proxy
unregister	Si scollega dal proxy di default
duration	Visualizza la durata in secondi dell'ultima chiamata
status	Visualizza varie informazioni di stato
ports	Configura la porta delle rete
speak	Riproduce il testo di una frase, con voce, usando il sistema espeak
codec	Configura i codec audio
vcodec	Configura i codec video
ec	Attiva l'eco cancellazione
mute	Disattiva il microfono e sospende la trasmissione dell'audio
unmute	Riattiva il microfono e riprende la trasmissione dell'audio
nortp-on-a	Imposta i parametri di configurazione di rtp_no_xmit_on_audio_mute

  Digitare *help <ltcommand>* per avere ulteriori dettagli sui comandi.

"Linphonecsh" è un altro tool a linea di comando per controllare da remoto il demone di linphonec. Diversamente da linphonec, linphonecsh immediatamente esce dal comando una volta che è stato eseguito.

- **Linphonecsh init**
  Per attivare il demone di linphonec
- **Linphonecsh register —host myproxy.net —usename bill —password thisisasecret**
  Per registrarsi a un proxy
- **Linphonecsh dial "sip:alice@myproxy.net"**
  Per effettuare una chiamata in uscita
- **Linphonecsh hangup**
  Per terminare la chiamata
- **Linphonecsh generic "proxy list"**
  Linphonecsh può trasmettere qualsiasi comando di linphonec usando la parola chiave "generic"
- **Linphonecsh exit**
  Per terminare il demone linphonec

5.4 Portabilità

Linphone è eseguibile su varie piattaforme, grazie anche alla sua interfaccia tramite console. Vediamo ora in particolare su quali piattaforme Linux gira:

- Linux x86 e Linux x86-64
- Windows XP, Vista e 7
- Linux/ARM: senza interfaccia grafica. Linphonec o liblinphone sono dei buoni candidati per fornire lo stack software di un telefono hardware o di un sistema di comunicazione hardware
- Linux/Blackfin: il progetto uclinux.org mantiene un port di linphone per i processori blackfin, senza gui.
- MacOSX x86: per il momento abbiamo solo la versione con solo l'audio.
- Google Android: per il momento solo con la versione audio
- Iphone OS: per il momento solo con la versione audio
- Può anche lavorare su FreeBSD e OpenBSD con delle modifiche minori sul sistema build.

Capitolo 6

SipToSis

6.1 Introduzione

SipToSis (software di integrazione tra SIP e Skype) è un software Java che permette di effettuare e ricevere chiamate Skype da un telefono SIP/VOIP, o PBX Asterisk/SIP. È inoltre possibile effettuare o ricevere chiamate SIP da Skype. Questo sistema quindi permette di integrare Skype nel sistema VOIP SIP. Si tratta fondamentalmente di un software multifunzione Skype/SIPBridge/Gateway/Proxy/Adapter/Converter. SipToSis dispone quindi di un codec convertitore in grado di convertire l'audio SIP RTP nell'audio Skype PCM e viceversa, cioè convertire l'audio Skype PCM in audio SIP RTP. Il software esegue la signalizzazione SIP e la chiamata Skype gestisce la connessione con il dispositivo SIP, il server Asterisk, il PBX SIP o il provider VOIP SIP. Occorre precisare che questo prodotto utilizza le API Skype ma non è certificato o approvato in alcun modo da Skype.

6.2 Caratteristiche

Possiamo sommariamente riassumere alcune notevoli funzionalità e caratteristiche:

- Gli utenti skype chiamano usando il mapping/speed-dial o usando lo SkypeOut per effettuare chiamate nella PSTN
- Si effettuano chiamate SIP da Skype usando un SIP provider o un PBX SIP
- Le chiamate Skype in ingresso possono essere indirizzate ad un indirizzo SIP desiderato
- Le chiamate SIP in ingresso possono essere indirizzare ad un utente Skype desiderato

- L'autenticazione/rifiuto di SIP a Skype avviene tramite l'ID SIP e il blocco degli IP
- L'autenticazione/rifiuto di Skype a SIP avviene tramite lo user ID di Skype
- I toni SIP DTMF vengono decodificati/decodificati attraverso lo RFC2833, INFO o Inband
- I toni Skype DTMF vengono decodificati/decodificati attraverso Inband
- È possibile connettere Asterisk, FreePBX, Trixbox e altri PBX SIP a utenti Skype
- È possibile effettuare chiamate in conferenza
- È possibile effettuare il callback
- È possibile ricevere le Voicemail attraverso SIP
- Include i codec PCMU (ulaw)/PCMA (a-law)/G.711 e il GSM con una libreria aggiuntiva
- È possibile aggiungere altri codec grazie ad una interfaccia codec
- È possibile mettere in attesa sia SIP che Skype
- È possibile personalizzare le regole di chiamata in uscita di Skype
- È possibile personalizzare le regole di chiamata in uscita di SIP
- Fornisce il supporto per il server STUNT
- Fornisce la possibilità di limitare il tempo di chiamata
- Può venire impiegato (con Asterisk) come trunk Multi Channel Skype, per supportare chiamate multiple simultanee.
- È multipiattaforma
- In Windows può essere eseguito come servizio

6.3 Configurazione

Il programma SipTosis ha una serie di file di configurazione che vengono usati sia per impostare il client SIP e il client Skype da usare, sia per impostare altri parametri di configurazione che servono in caso non si ha la possibilità di gestire le chiamate SIP o Skype attraverso il server SIP Voip.

6.3.1 Siptosis.cfg

Questo file di configurazione ci permette di impostare dei parametri standard per il programma, è il file di base che contiene tutte le varie impostazioni per il corretto funzionamento di SipToSis. Tra tutti i parametri di configurazione ne vediamo solo alcuni:

- Impostare il file di log con le proprietà

```
logConfigFile=log.properties
```

- Impostare la registrazione della conversione tra Skype e SIP

```
recordSkypeIn=no
recordSkypeOut=no
recordSIPIn=no
recordSIPOut=no
```

- Impostare se viene eseguito il log delle API di Skype

```
skypeAPITrace=no
```

- Dopo quanto tempo (espresso in minuti) viene effettuato il test per verificare se i file di configurazione sono stati cambiati (0=disattivato)

```
configWatchInterval=0
```

- Dopo quanto tempo viene testato se il client Skype è connesso (0=disattivato)

```
connectorWatchDogMinutes=0
```

- Caratteristiche della durata della chiamata
  - Lunghezza massima della chiamata

    ```
    MaxCallTimeLimitMinutes=0
    ```

  - Minuti richiesti prima che venga generato un file di warning

    ```
    WarnMinutesBeforeCutoff=1
    ```

  - File da eseguire quando raggiungiamo il limite della chiamata

    ```
    OverLimitWarningFile=clips/overlimit.wav
    ```

  - Risposta SIP quando si ha raggiunto il limite della chiamata

    ```
    OverUsageLimitSipResponse=480
    ```

  - Minuti giornalieri massimi permessi per una chiamata PSTN

    ```
    dailyPstnLimitMinutes=350
    ```

  - Massimo numero di chiamate PSTN nella giornata

    ```
    dailyPstnUniqueNumberLimit=48
    ```

  - Minuti richiesti minimi per poter effettuare una nuova chiamata

    ```
    refuseNewPstnCallsWhenRemainingMinutesUnder=5
    ```

  - Minuti massimi permessi per una chiamata PSTN

    ```
    MaxPstnCallTimeLimitMinutes=0
    ```

  - Imposta di importare da Skype il log delle chiamate all'avvio

    ```
    loadSkypeClientCallHistory=yes
    ```

  - Elenco dei prefissi PSTN che non vengono inclusi nel calcolo del numero di chiamate e della durata

    ```
    tollFreeNumberPrefixes=1800,1888,1866,1877
    ```

- Impostazione di una notifica per email quando il bilanciamento di carico è sotto una determinate soglia (-1= disattivato)

```
emailWhenBalanceDropsTo=-1
emailHost=smtp.host.com
emailPort=25
emailUsername=
emailPassword=
emailRecipients=
emailFrom=
emailTest=no (se viene impostato a yes, viene inviato
una email di test all'avvio del programa per verificare
le corrette impostazioni)
```

- Intervallo in minuti per impostare lo stato di Skype a "In linea" (0=disattivato)

```
setSkypeOnlineStatusInterval=0
```

- Lo stato da visualizzare quando ha raggiunto l'intervallo di tempo (opzioni: ONLINE, DND, AWAY, INVISIBLE, OFFLINE)

```
skypeOnlineStatus=ONLINE
```

- Percorso dove vengono salvati i log delle chiamate

```
callLogPath=log/
```

- File in cui troviamo le regole di autorizzazione

```
siptoskypeauthfile=SipToSkypeAuth.props
skypetosipauthfile=SkypeToSipAuth.props
```

- File in cui troviamo le trasformazioni per le chiamat

```
SkypeOutDialingRulesFile=SkypeOutDialingRules.props
SipOutDialingRulesFile=SipOutDialingRules.props
```

- Parametro per aumentare la priorità del processo dell'audio (0-2; 0=normale)

```
audioPriorityIncrease=0
```

- Attivare/disattivare la connessione al client Skype (per scopo di test)

```
skype_connect=yes
```

- Impostare lo userid di Skype nel caso abbiamo più istanza di Skype in esecuzione nell'ambiente Windows

```
skypeUserId=
```

- Porta usata da Skype per trasferire e ricevere l'audio da SipToSis

```
skype_audioportbase=64432
```

- Impostazione per attivare il supporto dei toni DTFM di Skype

```
enableSkypeDtmfDetector=yes
SkypeDtmfGain=1
SkypeDtmfDetectorHitThreshold=30
SkypeDtmfDetectorSilenceThreshold=40
autoDisableSkypeDtmfDetectorSeconds=80
(disattivazione del detector dopo tot secondi.
0=nessuna disattivazione)
```

- Per impostare di rigenerare i SIP DTFM a Skype

```
sendSipDtmfToSkype=yes
```

- Per impostare di rigenerare i Skype DTFM a SIP

```
sendSkypeDtmfToSip=yes
```

- Per impostare cosa fare nel caso il canale Skype è occupato. Le possibili impostazioni sono: *refuse, voicemail, ignore, transferto:skypeid*

```
SkypeInboundAllChannelsBusyAction=refuse
```

- Tempo di attesa per la riuscita del trasferimento, espresso in millisecondi

```
SkypeTransferTimeoutMs=8000
```

- Per impostare cosa fare nel caso una chiamata di ingresso Skype o la destinazione SIP non sono disponibili. Le opzioni permesse sono: *ring* e *refuse*

```
SkypeInboundSipDestUnavailableAction=refuse
```

- Imposta in intervalli di 5 minuti, con 0=disattivato, l'auto shutdown di SipToSis quando rimane per X minuti inattivo

```
autoShutdownMinutes=0
```

- Numero di secondi che ha il chiamante per inserire il numero di destinazione

```
destinationtimeout=12
```

- Numero di tentativi prima che avvenga l'hangup

```
destinationretrylimit=3
```

- Spedire un tono di ring per le chiamate Skype in ingresso

```
sendRingToSkypeCaller=no
skypeRingFile=clips/skypeRing.wav
skypeRingInterval=8
```

- Sostituire l'indirizzo SIP From con lo UserID di Skype

```
replaceFromWithSkypeId=no
```

- Spedire un messaggio quando gli utenti Skype effettuano una chiamata (non usato per Skypeout)

```
sendSkypeIM=no
skypeimmessage=You are about to receive a Skype Voice
                  call from [callerid] [callernumber].
sendSkypeImDelay=2
(ritardo tra il messaggio e la chiamata, in secondi)
```

- Protocollo di trasporto. Opzioni possibili: udp o tcp

```
transport_protocols=udp
```

- Impostazioni di rete nel caso si usi un host SIP esterno oppure un proxy

```
outbound_proxy=proxy01.sipphone.com:5060
via_addr=192.168.0.8 (indirizzo IP del PC locale)
host_ifaddr=192.168.0.8 (nel caso si abbia più di
                         una interfaccia di rete)
publicIP=222.22.222.22
stunServer=stun.xten.net:3478,stunserver.org:3478
stunTestInterval=30 (minuti tra gli aggiornamenti
                 tra gli STUN. 0= solo all'avvio)
enableNatTranslate=yes
```

- Impostazione del client SIP

```
host_port=5070
contact_url=sip:skypetestuser@SipToSisIpAddress:
                                 SipToSisHostPort
from_url="skypetestuser" <sip:skypetestuser@
             asteriskIpAddress:asteriskHostPort>
username=skypetestuser
realm=asterisk
passwd=skypetest
expires=3600
do_register=yes
minregrenewtime=120
regfailretrytime=15
```

- Impostazioni audio

```
audio=yes
audio_port=63200
noRtpReceivedAutoHangupSeconds=30
  (dopo quanti secondi avviene l'hangup non caso
   non si riceva pacchetti rtp)

audio_codec=PCMU,PCMA,ILBC(inoltre c'è il GSM)
audio_frame_size=240,240,240

skype_audiooutgain=1,1,1 (sip->skype)
skype_audioingain=1.2,1.2,1.2 (skype->sip)
```

- Codici di errori restituiti da SIP

```
baseFailureResponse=403
       (in caso di errori non conosciuti)
skypeRefusedResponse=603
       (in caso l'utente skype rifiuta la chiamata)
skypeFailedResponse=408
       (in caso la chiamata skype fallisca)
skypeInvalidDestinationResponse=404
       (in caso il numero chiamato sia sbagliato)
skypeBusyResponse=600
       (in caso l'utente sia occupato)
```

- Impostazioni del buffer audio
(0=impostazioni dal sistema operativo)

```
TcpRxBufferSize=8192
TcpTxBufferSize=8192
RtpRxBufferSize=8192
RtpTxBufferSize=8192
```

- Impostazioni dei file audio della voicemail

```
vmNoMessagesClip=clips/vmpnomessages.wav
vmPlayingClip=clips/vmpplayingmessages.wav
vmEndOfMessageClip=clips/vmpendofmessageprompt.wav
vmMessageDeletedClip=clips/vmpmessagedeleted.wav
vmMessageDeleteAllClip=clips/vmpmessagedeleteall.wav
vmNoMoreMessagesClip=clips/vmpnomoremessages.wav
vmRecordingBeepClip=clips/vmprecordingbeep.wav
```

6.3.2 SipToSkypeAuth.props

Questo file di configurazione permette di impostare l'utente Skype su cui viene inoltrata la chiamata in arrivo da un utente SIP.

Questo file contiene un insieme di righe nel formato:

```
incomingSipCallerID, real, incomingCallersIP, skypeTarget
```

Parametri:

- *IncomingSipCallerID*: identificativo del client SIP che ha l'accesso ad inoltrare la chiamata. È possibile usare il simbolo * per identificare che qualsiasi client SIP può inoltrare la chiamata a Skype
- *realm*: indirizzo del server SIP. Si può usare il simbolo * per definire che vengono accettati tutti gli indirizzi IP
- *IncomingCallersIP*: indirizzo IP del client che può inoltrare la chiamata. È possibile usare il simbolo * per permettere qualsiasi indirizzo IP, oppure impostare degli indirizzi parziali, come 192.168.*.*. Si può inoltre usare la parola chiave "localnet" per identificare la rete privata locale.
- *skypeTarget*: viene definito la destinazione della chiamata SIP in ingresso
  - uno username di skype
  - uno speedDial di skype
  - un numero di skypeout
  - calleeid: viene usato lo userid della chiamata SIP in ingresso

Esempi:

```
15615555555,*,*,skypeuser1
15615555554,*,172.17.*.*,skypeuser1
*,*,192.168.*.*,calleeid
```

6.3.3 SkypeToSipAuth.props

Questo file di configurazione permette di impostare l'utente SIP su cui viene inoltrata la chiamata in arrivo da un utente Skype.

Questo file contiene un insieme di righe nel formato:

```
CallingSkypeUserID, SipTarget, SkypeClientUserID
```

Parametri:

- CallingSkypeUserID: l'utente skype chiamante che ha l'autorizzazione ad inoltrare la chiamata a SIP. Può essere * per permettere chiunque utente Skype
- SipTarget: può essere l'url del sip, oppure deny, oppure *transferto: skypeid1: skypeid2, play:filename*
- skypeClientUserId (opzionale): è possibile specificare un client skype allegato per il routing delle chiamate in ingresso

Esempi:

```
*,sip:15554443333@192.168.1.220:5060
*,sip:15554443333@192.168.1.220:5060,yourSkypeClientUserId1
*,play:clips/invalidDest.wav
```

6.3.4 SkypeOutDialingRules.props

Questo file di configurazione viene usato quando non abbiamo la possibilità di configurare il server SIP e quindi tramite questo file di configurazione possiamo impostare come vengono gestite le chiamate.

Il file lavora usando delle regole che trasforma il numero digitato da SIP in una chiamata Skype che modifica il numero digitato. Il lavoro svolto da SipToSis è:

- Ogni regola viene espressa come una riga del tipo:

  ```
  regex:replacement
  ```

- Se non ci sono regole che coincidono, la destinazione rimane invariata
- Una volta che la regola è stata trovata, non vengono fatti ulteriori processi

Questo diventa utile quando si sta usando un PBX ma non sia possibile configurarlo per eliminare il prefisso di selezione. In questo modo è possibile aggirare questa limitazione del PBX aggiungendo delle entry. Se la chiamata del PBX ha il prefisso 7 allora è possibile aggiungere la seguente riga:

```
^7([1-9][0-9]{10})$:+$1
```

Il simbolo $1 cattura quello che c'è nelle parentesi. Nell'esempio sopra, il numero 7 sarà lasciato fuori quando verrà effettuata la chiamata SkypeOut con 11 numeri. In questo caso se si effettua una chiamata con il numero 713051234567 al gateway SipToSis, allora il numero viene convertito in +13051234567. Ovviamente per altri prefissi, basta sostituire il numero 7 con il prefisso desiderato. Si deve porre attenzione che non ci siano altre regole che hanno lo stesso match.

Esempi:

```
^([1-9][0-9]{6})$:+1561$1
   3684111=+15613684111
^([1-9][0-9]{9})$:+1$1
   5613684111=+15613684111
^([0-9]{7,})$:+$1
   15613684111=+15613684111 (più di 7 numeri)
Per simulare una chiamata all' eco test di skype si può
inserire la seguente riga:
   ^55$:echo123
Per fare una conferenza con tre utenti Skype si può
inserire la seguente riga:
   ^56$:skypeuser1,skypeuser2,skypeuser3
Per fare una conferenza per la PSTN si può inserire la
seguente riga:
   ^57$:+18885555555,+18005555555
```

6.3.5 SipToDialingRules.props

Questo file di configurazione viene usato quando non abbiamo la possibilità di configurare il server SIP e quindi tramite questo file di configurazione possiamo impostare come vengono gestite le chiamate.

Anche in questo caso il file lavora usando delle regole che trasforma il numero digitato da Skype in una chiamata SIP che modifica il numero digitato. Il lavoro svolto da SipToSis è:

- Ogni regola viene espressa come una riga del tipo:

  ```
  regex:replacement
  ```

- Se non ci sono regole che coincidono, la destinazione rimane invariata
- Una volta che la regola è stata trovata, non vengono fatte ulteriori processi

Esempi :

```
^0([1-9][0-9]{6})$:sip:1561$1@proxy01.sipphone.com:5060
^0([1-9][0-9]{9})$:sip:1$1@proxy01.sipphone.com:5060
^0([1-9][0-9]{10})$:sip:$1@proxy01.sipphone.com:5060

^8([0-9]+)$,sip:$1@sip.voipbuster.com:5060;
usr=vbuserid;pwd=vbpassword;realm=sip.voipbuster.com;
from="caller name" sip:vbuserid@sip.voipbuster.com:5060
```

6.4 Installazione

Per installare SipToSis si ha necessità di avere:

- Client Skype
- Java 1.5 o superiore
- Un telefono SIP/VOIP registrato in un provider SIP o su un server PBX SIP
- Sufficiente banda per supportare tale configurazione
- Skype compatibile con la scheda audio

I passi da seguire per l'installazione e configurazione di SipToSis sono:

1. Installare il pacchetto "sun-java6-bin" e tutte le sue dipendenze
2. Disinstallare il pacchetto "gcj-4.3"
3. Scaricare il pacchetto SiptoSis-20100830 da:

   ```
   http://www.mhspot.com/sts/siptosis_download.php
   ```

4. Estrarre il file compresso nella cartella:

   ```
   /home/<username>/siptosis
   ```

5. Installare Skype e configurarlo con un account secondario. Questo skype serve per SipToSis ma poi sarà invisibile all'utente finale.
6. Avviare Skype con l'utente registrato
7. Accedere alla cartella "siptosis" e assegnare i permessi al file SipToSis_linux

   ```
   cd /home/<username>/siptosis
   chmod +x SipToSis_linux
   ```

8. Eseguire SipToSis con Skype avviato e accettare su Skype l'uso delle API da parte di SipToSis

   ```
   ./SipToSis_linux
   ```

   Il primo avvio di SipToSis permette di autorizzare SipToSis di usare le API di Skype ed inoltre di generare i file di configurazione standard del programma

9. Configurare SipToSis con in base alle esigenze:

   - Editare il file "siptosis.cfg" e apportare le seguenti modifiche:
     - Mettere il segno "#" nelle seguenti righe:

       ```
       #Sample AUTO config with NO registration
       #username and password not important in this mode
       #Set to available port to transport SIP
       #messages on siptosis computer
       #host_port=5070
       #username=skypests
       #passwd=unimportantpassword
       #do_register=no
       # --- end of NO registration example ---
       ```

– Configurazione del client SIP che è stato configurato sul server Asterisk, modificando le seguenti righe:

```
#Sample Asterisk registration example -
#comment out NO registration info above
#first and uncomment the following.
host_port=5070
contact_url=sip:202@192.168.0.100:5070
from_url="202" <sip:202@192.168.0.100:5060>
username=202
realm=192.168.0.100
passwd=202
expires=3600
do_register=yes
minregrenewtime=120
regfailretrytime=15
# --- end of Asterisk Reg example ---
```

- Editare il file "SipToSkypeAuth.props" e apportare le seguenti modifiche:
  – Commentare la riga di default, aggiungendo il simbolo "#"

    ```
    #*,*,localnet,calleeid
    ```

  – Aggiungere la seguente riga:

    ```
    200,192.168.0.100, localnet,fabio.montemaggiore
    ```

    Con questa riga si ha definito che la chiamata SIP indirizzata al numero 200, proveniente dall'indirizzo di server Asterisk 192.168.0.100, e che appartiene alla rete locale, viene inoltrata all'utente Skype fabio.montemaggiore

Capitolo 7

Linphone-Button

7.1 Introduzione

Linphone utilizza un'interfaccia grafica sviluppata in GTK+ con versione almeno 2.16. Lo sviluppo in particolare dell'interfaccia utente si basa su LibGlade, cioè una libreria che fornisce le interfacce descritte in un file xml glade. Per compilare e installare correttamente l'interfaccia grafica di Linphone si ha bisogno della versione di GTK+ 2.16 o superiore, inoltre si ha bisogno di installare le librerie di Libglade.

Con il sistema operativo installato nella scheda I.MX27-PDK è possibile installare al massimo la versione di GTK+ 2.12.12 quindi l'interfaccia grafica di Linphone non è possibile installarla. È da tenere presente inoltre, che tutte le funzionalità offerte da quell'interfaccia grafica non servirebbero per lo scopo da raggiungere, quindi conviene creare una nuova interfaccia grafica con il solo bottone di chiamata. Ecco quindi il motivo di creare una nuova applicazione che si interfaccia a Linphone, chiamata appunto Linphone-Button.

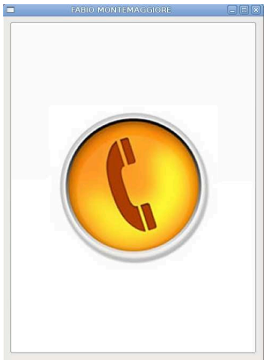
Questa applicazione infatti, presenta solo un bottone di chiamata, che una volta premuto fa eseguire la chiamata verso l'utente configurato. Linphone infatti mette a disposizione un'interfaccia che permette di interagire con il core di Linphone, questa interfaccia da console usa il comando "linphonecsh" il quale permette di inviare dei comandi al demone di Linphone.

Linphone-button sfrutta l'interfaccia da console di Linphone per avviare il suo demone e per inviare il comando di chiamata quando viene premuto il bottone del citofono.

7.2 Caratteristiche

Linphone è un'applicazione sviluppata in C che permette di avviare il softphone Linphone e di inviare il comando di chiamata grazie all'interfaccia di gestione di Linphone chiamata "linphonecsh".

Linphone-Button si presenta come una finestra di dimensioni 480x640, la dimensione del touch-screen della scheda i.mx27-pdk, con un bottone che riempie tale finestra. Si vede di seguito come si presenta il programma Linphone-Button.



La funzione svolta principale da Linphone-Button è di avviare il demone Linphone all'avvio della scheda i.mx27-pdk e di effettuare la chiamata ad un numero prefissato quando viene premuto il bottone.

All'avvio del programma, Linphone-button esegue i seguenti passi:

- Recupera dal file "config.props" l'impostazione del numero da chiamare quando viene premuto il bottone del citofono
- Invia a Linphone il comando di avviare il demone grazie a questo comando:

```
linphonecsh init
```

Quando un utente preme il bottone del citofono, semplicemente il programma invia a Linphone il seguente comando:

```
linphonecsh dial <numero da chiamare>
```

Il numero da chiamare solitamente è nella forma:

```
sip:<numero>@indirizzoIpServer
```

7.3 Installazione

Per procedere con l'installazione di Linphone-Button sulla scheda i.mx27pdk si deve seguire i seguenti passi:

- Creare il pacchetto compresso:

```
tar zcvf buttons-1.0.tar.gz buttons-1.0
```

- Copiare il pacchetto nella cartelli dei pacchetti sorgenti

```
cp buttons-1.0.tar.gz /opt/freescale/pkgs/
```

- Creare il file dist per l'installazione:

```
mkdir /home/<username>/ltib/dist/lfs-5.1/linphoneButton
gedit /home/<username>/ltib/dist/lfs-5.1/linphoneButton/
        linphoneButton.spec
```

  Inserire le seguenti righe nel file spec:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: Some simple but meaningful text
Name: buttons
Version: 1.0
Release: 1
Packager: Fabio Montemaggiore
Source          : %{name}-%{version}.tar.gz
BuildRoot       : %{_tmppath}/%{name}
Prefix          : %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
make

%Install
rm -rf $RPM_BUILD_ROOT
mkdir -p $RPM_BUILD_ROOT/%{pfx}/usr/bin
cp buttons $RPM_BUILD_ROOT/%{pfx}/usr/bin
```

```
cp config.props $RPM_BUILD_ROOT/../../rootfs/
cp telefono.jpg $RPM_BUILD_ROOT/../../rootfs/

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

- Aggiungere il software nella lista dei pacchetti di LTIB

  ```
  gedit /home/<username>/ltib/config/userspace/packages.lkc
  ```

  Inserire le seguenti righe dopo il pacchetto PKG_LINPHONE

  ```
  config PKG_LINPHONE_BUTTO
    bool "linphone_button"
  ```

- Aggiungere il pacchetto nel file dove vengono mappati i pacchetti

  ```
  gedit /home/<username>/ltib/dist/lfs-5.1/common/pkg_map
  ```

  Aggiungere la seguente riga dopo il pacchetto PKG_LINPHONE

  ```
  PKG_LINPHONEBUTTON = linphoneButton
  ```

- Eseguire LTIB e selezionare il pacchetto Linphone_button

  ```
  ./ltib -c
  ```

Per far avviare Linphone-Button in modo automatico bisogna impostare alcune variabili d'ambiente e in più bisogna avviare la libreria gtk.
Per questo basta aggiungere le seguenti righe nel file */home/<username>/ltib/rootfs/etc/profile:*

```
/etc/rc.d/init.d/gtk2 start
/etc/rc.d/init.d/pango start
mknod /dev/input/event1 c 13 66
export TSLIB_TSDEVICE=/dev/input/event1
/usr/bin/buttons &
```

7.4 Configurazione

L'applicazione Linphone-Button non ha bisogno di molte configurazioni, infatti la sola configurazione che viene effettuata è l'impostazione del telefono da chiamare quando una persona preme il bottone del citofono. Per configurare il numero da chiamare basta solamente modificare il file "config.props" che si trova nella directory /home/<username>/ltib/rootfs/, inserendo il numero da chiamare e l'indirizzo IP del server Voip.

```
sip:<numero>@indirizzoIpServer

<numero>: è il numero SIP da chiamare
<indirizzoIpServer>: è l'indirizzo IP del server Voip
```

É possibile inoltre cambiare l'immagine del bottone della finestra. Per modificare tale immagine basta solamente cambiare l'immagine:

```
/home/<username>/ltib/rootfs/telefono.jpg
```

con un'immagine che desideriamo. L'immagine deve avere le dimensioni di 480x640 e deve avere lo stesso nome ed estensione del file precedente.

Capitolo 8

Conclusioni

Il lavoro affrontato in questa tesi ha avuto come oggetto la realizzazione di un audiocitofono SIP in un sistema embedded. Il vantaggio di utilizzare un citofono SIP apre la possibilità di creare molte funzionalità di gestione e di servizi forniti, che possono avvenire sia in locale ma anche da remoto e soprattutto a basso costo.

Per realizzare il citofono SIP è stato utilizzato la scheda di progettazione di Freescale I.MX27-PDK con sistema operativo Linux creato attraverso il tool di sviluppo LTIB. Per quanto riguarda la parte VOIP è stato utilizzato il server Asterisk installato su un computer che svolge il ruolo di server, in più è stato usato il client SIP Linphone installato sulla scheda I.MX27 e un telefono IP della Voismart. Per effettuare il gateway tra SIP e Skype è stato usato SipToSis installato sempre nel server.

Il sistema ottenuto risulta funzionante e buono: alla pressione del bottone del citofono avviene una chiamata verso il telefono SIP configurato, oppure verso un utente Skype. La scelta, se la chiamata del citofono avviene su SIP o Skype avviene tramite il telefono SIP. In particolare effettuando una chiamata al numero 0 si imposta che il citofono chiami l'utente SIP, mentre effettuando una chiamata al numero 1 si imposta che il citofono chiami l'utente Skype.

Un punto debole della realizzazione è dovuto al gateway per comunicare tra SIP e Skype. Il software SipToSis usato utilizza sia un utente SIP virtuale che un utente Skype virtuale. Avere quindi tutti questi utenti attivi richiede banda sprecata, soprattutto se si pensa al citofono inserito in un sistema più ampio.

8.1 Sviluppi futuri

La realizzazione di questo citofono si è soffermata alla comunicazione audio, ma una possibile ulteriore realizzazione dovrebbe comprendere anche il video per ottenere un videocitofono. Nella realizzazione del videocitofono però dobbiamo anche implementare la funzione del codec video di SipToSis, cioè la conversione del flusso video proveniente da SIP e convertirla al flusso video per Skype e viceversa. Attualmente questa funzione non è inclusa.
Dato che questo sistema di citofono è Always-On possiamo inserire questo sistema in un sistema di videosorveglianza con la possibilità di gestire l'intero sistema da remoto. Una realizzazione possibile sarebbe di ipotizzare di dover gestire la sicurezza di una casa, quindi prevedere di dover gestire delle videocamere motorizzate, le luci di ingresso, l'apertura del cancello. Inserendo nel citofono una gestione di autenticazione del proprietario, possiamo modificare l'interfaccia grafica del citofono: in un uso normale viene visualizzato nel monitor il bottone di chiamata, mentre quando il citofono riconosce il proprietario (per esempio tramite impronta digitale oppure sistema RFID) vengono visualizzati una serie di funzioni, come:

- La possibilità di aprire il cancello
- La possibilità di accendere le luci
- La possibilità di visualizzare l'immagine delle varie telecamere di sorveglianza
- La possibilità di effettuare il brandeggio delle telecamere e lo zoom
- La possibilità di effettuare una chiamata telefonica verso l'interno della casa oppure all'esterno
- La possibilità di accedere a dei servizi Web, Mail, etc.

Il citofono in più può gestire degli eventi, per esempio quando avviene un intrusion-detection, cioè quando il sistema di sorveglianza delle telecamere rileva una intrusione, il citofono può generare un evento, come l'invio di una email o sms, oppure di effettuare una chiamata al proprietario e di visualizzare in modo automatico l'immagine delle telecamere.

Tutte queste funzionalità possono avvenire anche da remoto ed è qui principalmente il punto di forza, perché questa gestione avviene sfruttando la rete Internet. Immaginate quindi alla possibilità di poter visualizzare le immagini di ingresso della vostra casa e di poter effettuare il brandeggio delle videocamere quando non siete a casa. La possibilità di aprire il cancello quando un vostro parente deve entrare in casa e voi non siete presenti per potergli aprire.

Tutto questo è a basso costo, perché sfrutta il collegamento Internet. Pensate quanto costerebbe se avvenisse anche tramite cellulare, soprattutto se vi trovate fuori all'estero: la differenza di costo è notevole.

Infine è da sottolineare che possiamo aumentare la sicurezza della rete implementando un sistema di crittografia. Infatti utilizzando questo sistema di comunicazione possiamo aumentare la protezione di tutte le comunicazioni e il traffico che avviene nella rete. Il SIP infatti fornisce un secure URI chiamato SIPS URI. Una chiamata fatta a un SIPS URI garantisce la sua sicurezza, usando un protocollo di trasporto criptato (chiamato TLS) per trasportare tutti i messaggi SIP dal chiamante al chiamato.
La stessa cosa vale per il flusso audio e video RTP: la protezione può avvenire grazie al protocollo ZRTP. ZRTP è un protocollo che consente di effettuare chiamate criptate sulla rete Internet. Il protocollo ZRTP intercetta e filtra tutti i pacchetti di dati VOIP che entrano ed escono dal telefono e mette in sicurezza la chiamata in tempo reale.

Appendice A

Configurazione di Linux

L'installazione di tutto il software è stato eseguito su Linux Debian 2.6.26-2.686.

Il sistema è stato aggiornato attraverso i seguenti server di pacchetti:

deb http://ftp.it.debian.org/debian/ lenny main non-free contrib
deb-src http://ftp.it.debian.org/debian/ lenny main non-free contrib
deb http://security.debian.org/ lenny/updates main
deb-src http://security.debian.org/ lenny/updates main
deb http://volatile.debian.org/debian-volatile lenny/volatile main
deb-src http://volatile.debian.org/debian-volatile lenny/volatile main
deb http://www.debian-multimedia.org experimental main
deb http://www.debian-multimedia.org etch main
deb http://download.skype.com/linux/repos/debian/ stable non-free

Sono stati usati questi server per poter installare Java 1.6 che servirà successivamente per installare SipToSis.

Appendice B

Configurazione di Linux

Per installare correttamente la versione di Asterisk 1.6.2.11 è necessario installare dei pacchetti di Linux. I pacchetti che devono essere installati sono:

- Linux-headers-2.6.26-2-686
- Bison
- Openssl
- Libssl0.9.8
- Libssl-dev
- Libreadline5
- Libreadline5-dev
- Libeditline0
- Libeditlive-dev
- Libedit-dev
- Libedit2
- Libncurses5
- Libncurses5-dev
- Zlib1g-dev
- gcc
- g++
- make

- libxml2-dev

Si procede con il download del pacchetto Asterisk dal seguente indirizzo: http://asterisk.org/downloads/asterisk/release/asterisk-1.6.2-current.tar.gz
I comandi che seguono vanno eseguiti come utente root:

- scompattare il pacchetto in /usr/src
- cd /usr/src/asterisk-1.6.2
- ./configure
- Make
- Make install
- Make samples
- Make config
  (solo se si desidera che Asterisk si avvii in modo automatico al boot)

Il passo successivo è quello di configurare i file di configurazione di Asterisk.

Si procede quindi con la configurazione dei client SIP attraverso il file /etc/asterisk/sip.conf. Sono stati configurati tre client SIP: un client è il citofono, un secondo client è il telefono SIP che possiede l'utente e l'ultimo SIP che deve essere configurato è un SIP virtuale che viene usato da SipToSis (software per far comunicare SIP con Skype). Le righe di questo file sono le seguenti:

```
[general]
context=local
bindport=5060
disallow=all
allow=alaw
allow=gsm
language=it
canreinvite=no

[campanello]
type=friend
defaultuser=campanello
secret=campanello
callerid="campanello" <campanello>
host=dynamic
qualify=no
```

```
[sipskype]
type=friend
defaultuser=sipskype
secret=sipskype
callerid="sipskype" <sipskype>
host=dynamic
qualify=no

[200]
type=friend
defaultuser=200
secret=200
callerid="200" <200>
host=dynamic
qualify=no
```

Si procede con la configurazione della gestione delle chiamate.
La configurazione impostata esegue i seguenti processi:

- Le chiamate locali tra telefoni SIP, nel caso ce ne fossero più di uno, avviene digitando 2xx, dove 2xx è il numero del telefono
- Quando una persona preme il pulsante del citofono, la chiamata può essere inoltrata ad un utente SIP (nel caso specifico al SIP 200), o ad un utente skype, o ad entrambi. La scelta su quale utente viene inoltrata la chiamata viene fatta grazie ad una variabile nel database di Asterisk (varibiale=campanello/deviazione)
- Per impostare la variabile l'utente deve effettuare una chiamata con il telefono SIP, digitando:
  - 0: per impostare che il citofono devia la chiamata verso l'utente SIP
  - 1: per impostare che il citofono devia la chiamata verso l'utente Skype
  - 2: per impostare che il citofono devia la chiamata sia verso SIP che Skype

Il file /etc/asterisk/extensions.conf è il seguente:

```
[general]
static=yes
writeprotect=no
clearglobalvars=no
```

```
[globals]

[local]

;CHIAMATA TRA TELEFONO SIP INTERNI
exten => _2xx,1,Dial(SIP/${EXTEN})
exten => _2xx,2,Hangup()

;IMPOSTA CHE IL CAMPANELLO CHIAMA IL TELEFONO SIP
exten => 0,1,Set(DB(campanello/deviazione)=0)
exten => 0,2,Hangup

;IMPOSTA CHE IL CAMPANELLO CHIAMA IL TELEFONO SKYPE
exten => 1,1,Set(DB(campanello/deviazione)=1)
exten => 1,2,Hangup

;IMPOSTA CHE IL CAMPANELLO CHIAMA SIP E SKYPE
exten => 2,1,Set(DB(campanello/deviazione)=2)
exten => 2,2,Hangup

;CHIAMATA EFFETTUATA DAL CAMPANELLO
exten => campanello,1,Set(valoreCampanello=${DB(campanello/
         deviazione)})
exten => campanello,2,GotoIf($["${valoreCampanello}" = "0"]?
         sip,1)
exten => campanello,3,GotoIf($["${valoreCampanello}" = "1"]?
         skype,1)
exten => campanello,4,GotoIf($["${valoreCampanello}" = "2"]?
         entrambi,1)

exten => sip,1,Dial(SIP/200)
exten => sip,2,Hangup


exten => skype,1,Dial(SIP/sipskype)
exten => skype,2,Hangup

exten => entrambi,1,Dial(SIP/200&SIP/sipskype)
exten => entrambi,2,Hangup
```

Una volta configurato questi file si può proseguire e avviare Asterisk con il seguente comando:

```
asterisk -vvvvvvvvvvvvvgc
```

Appendice C

Configurazione scheda I.MX27-PDK

C.1 Pacchetti da installare

Prima di procedere con l'installazione del software per la scheda I.MX27-pdk si ha la necessità di installare i seguenti pacchetti in Linux:

- patch
- g++
- rpm
- zlibg1g-dev
- m4
- bison
- libncurses5-dev
- libglib2.0-dev
- gettext
- Build-essential
- tcl
- intltool
- libxml2-dev
- liborbit2-dev
- libx11-dev

- ccache
- flex
- uuid-dev
- liblzo2-dev

Inoltre per installare l'ambiente grafico GTK+ servono i seguenti pacchetti:

- libdbus-glib-1-dev
- libgtk2.0-dev

Si prosegue con l'impostazione di Visudo eseguendo il seguente comando, con utente root:

```
/usr/sbin/visudo
```

Nel file Visudo aggiungo la seguente riga:

```
Username ALL = NOPASSWD: /usr/bin/rpm, /opt/freescale/
         ltib/usr/bin/rpm
```

Dove <username> è il nome utente dell'account con cui si sta lavorando

C.2 Installazione e configurazione di LTIB

Si procede con l'installazione e la configurazione di LTIB ed inoltre all'installazione dei vari pacchetti per ottenere l'ambiente Linux funzionante sulla scheda I.MX27-PDK.
I passi da seguire sono i seguenti:

- Scaricare dal sito di Freescale www.freescale.com, nella sezione i.mx27-pdk il pacchetto LPDK_iMX27_R1_1
- Scompattare il pacchetto all'interno della cartella /home/<username>
- Aprire la console in modalità non-root ed eseguire i seguenti comandi:

  ```
  cd /home/<username>/LPDK_IMX27_R1_1/ LPDK_IMX27_R1_1
  ./install
  ```

  Digitare "Y" per accettare di leggere la licenza e premere INVIO
  Leggere la licenza e proseguire velocemente premendo la BARRA SPAZIATRICE

Accettere la licenza digitando “yes” e premere INVIO
A questo punto viene chiesto dove installare la cartella di ltib, quindi inserire:

```
/home/<username>/
```

Con questo sistema si ha predisposto i file nella cartella

```
/home/<username>/ltib
```

- Entrare nella cartella principale di LTIB

```
cd /home/<username>/ltib
```

- Modificare il file bison.spec in quanto presenta un bug per la scheda i.mx27

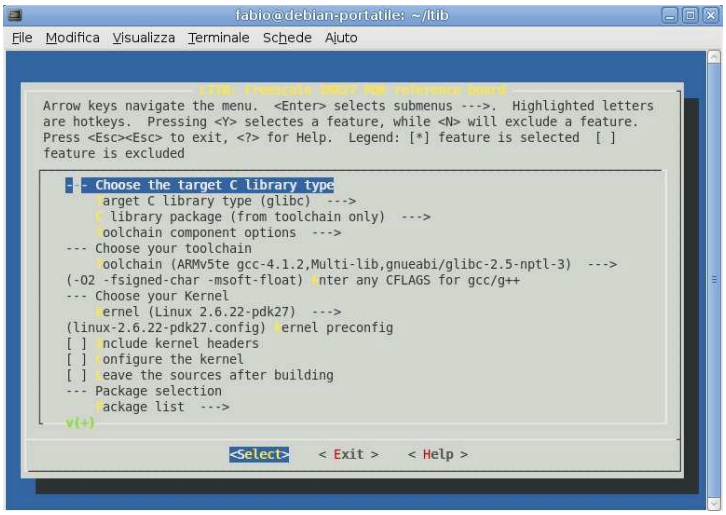
```
gedit dist/lfs-5.1/bison/bison.spec
```

Nella sezione

```
make CFLAGS=-O0 (ohhzero)
```

- Eseguire la prima installazione con il seguente comando:

```
./ltib -c
```



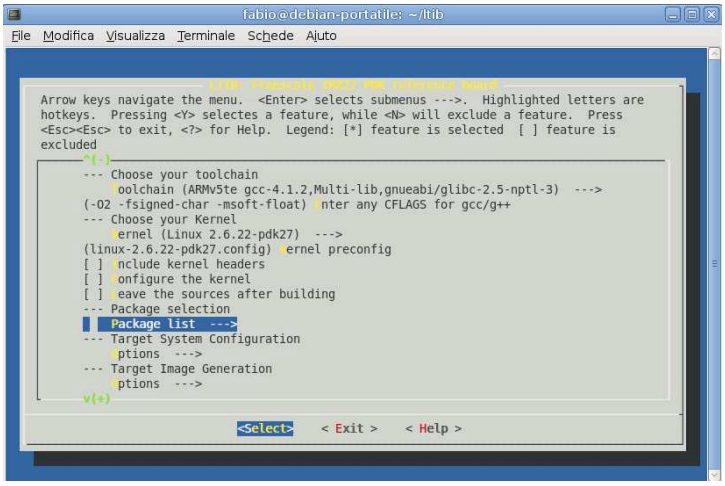
Viene visualizzata la schermata principale dei menu, quindi selezionare EXIT e accettare di salvare la configurazione dando conferma su "YES".

In questo modo LTIB scarica e installa tutti i pacchetti di default della scheda I.MX27-pdk.

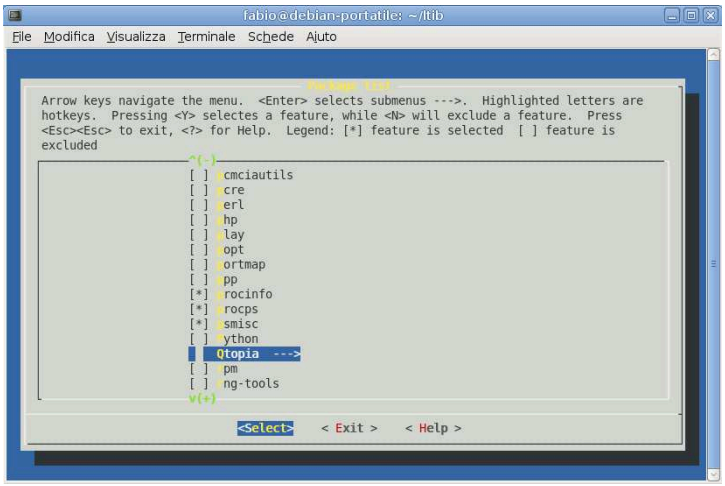
- Prima proseguire con le varie installazioni, si procede con la disinstallazione di Qtopia, in quanto non serve per lo sviluppo che si andrà a fare, e si imposta l'utilizzo della scheda i.MX tramite NFS. Per fare questo si procede come segue:

```
./ltib -c
```

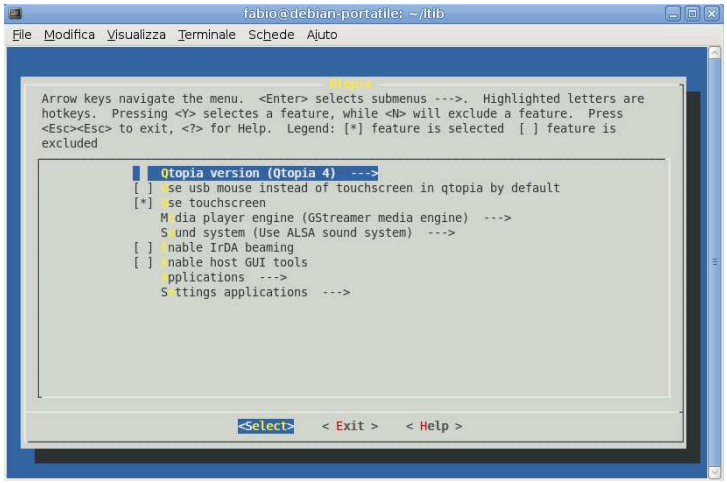
Selezionare la voce del Menu "Package List" per entrare nella lista dei programmi installati.



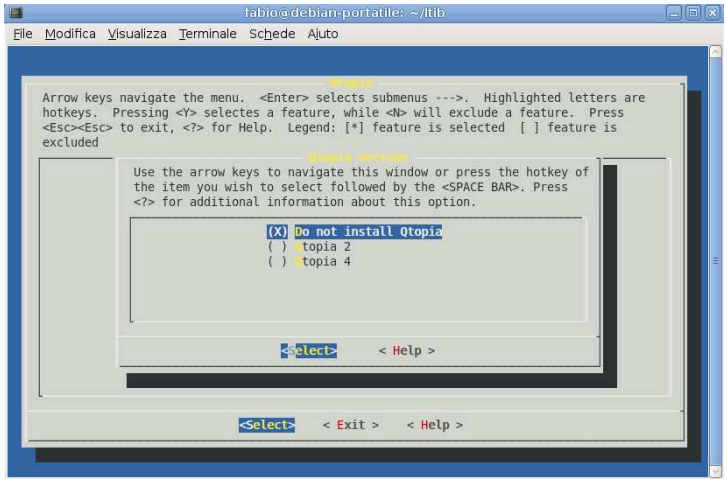
Selezionare "Qtopia"



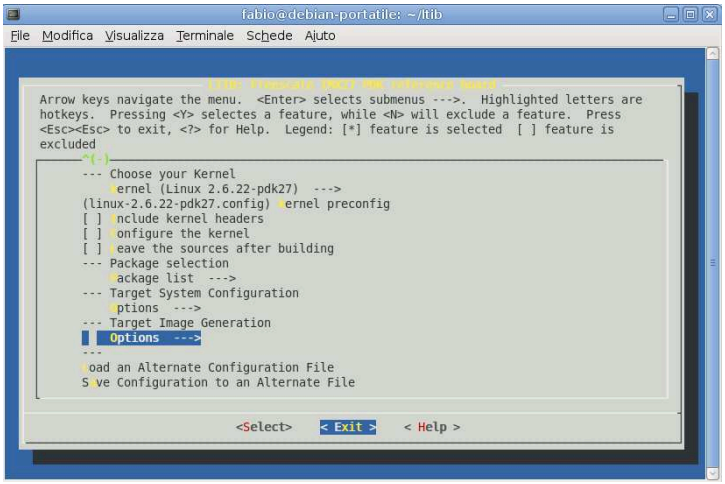
Selezionare la voce “Qtopia version”



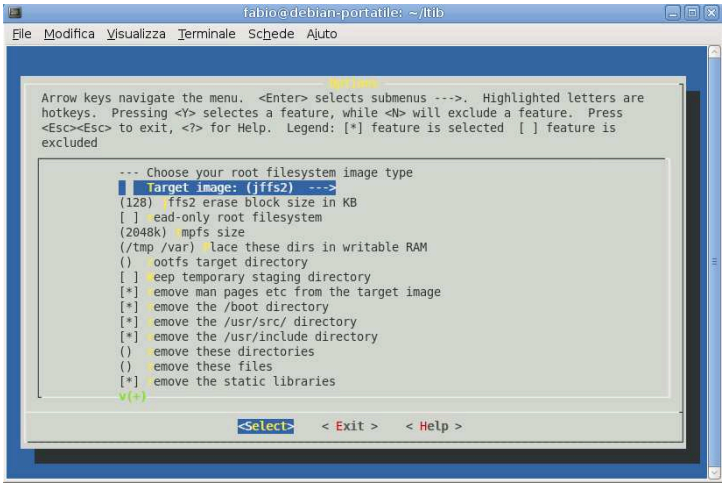
Selezionare “Do not install Qtopia”



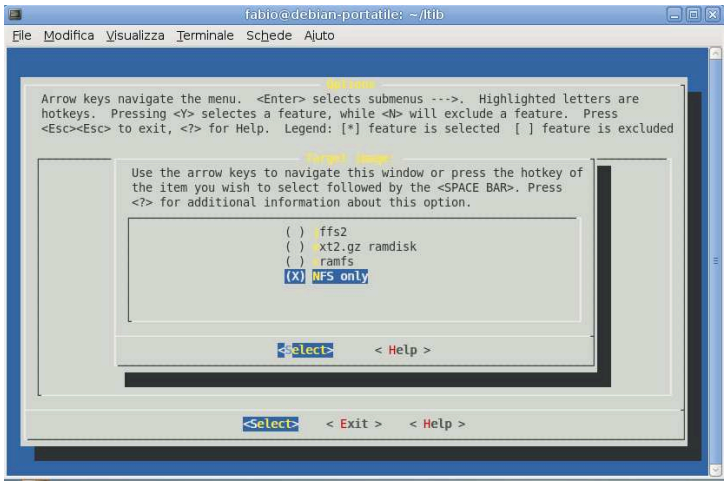
Proseguire quindi a impostare LTIB in modo tale che la scheda i.MX usi il kernel tramite NFS. Ritornare alla schermata principale di LTIB. Selezionare la voce “Options” della sezione “Target Image Generation”.



Selezionare la voce “Target image”



Selezionare la voce “NFS only”



Ritornare alla schermata iniziale, uscendo e salvando le modifiche.

- Proseguire con l’installazione e aggiornamento del pacchetto GTK. Prima di installare GTK però bisogna aggiornare e/o installare alcuni pacchetti:

  – **Tslib**: aggiungere la patch3 alla versione 1.0.
  Modificare il file
  /home/<username>/ltib/dist/lfs-5.1/tslib/tslib.spec
  come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: Abstraction layer for touchscreen panel
         events
Name: tslib
Version: 1.0
Release: 1
License: LGPL
Vendor: Freescale
Packager: Ross Wille
Group: System Environment/Libraries
Source: %{name}-%{version}.tar.bz2
Patch1: tslib-1.0-enable_input_events.patch
Patch2: tslib-1.0-mx-pre_gen-2.patch
Patch3: tslib-1.0-directfb_link_fix.patch
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}
```

```
%Description
%{summary}

%Prep
%setup
%patch1 -p1
%patch2 -p1
%patch3 -p1

touch aclocal.m4
sleep 1
find . -name Makefile.in | xargs touch
sleep 1
touch configure

%Build
#sed -i s/AS_HELP_STRING/AC_HELP_STRING/ configure.ac
#./autogen.sh
chmod +x ./configure
export ac_cv_func_malloc_0_nonnull=yes
./configure CC=${TOOLCHAIN_PREFIX}gcc --prefix=
%{_prefix} --host=$CFGHOST --build=%{_build}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/ts/*.la
# Remove unused platform binaries
for so in arctic2.so collie.so corgi.so h3600.so
mk712.so; do
rm -f$RPM_BUILD_ROOT/%{pfx}/%{_prefix}/usr/lib/ts/$so
done

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

– **Expat**: aggionare dalla versione 1.95.7 alla versione 1.95.8.
Modificare il file
/home/<username>/ltib/dist/lfs-5.1/expat/expat.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}
Summary: XML 1.0 parser
Name: expat
Version: 1.95.8
Release: 1
License: MIT/X Consortium License
Vendor: Freescale
Packager: Stuart Hughes
Group: Applications/Publishing/XML
Source0: %{name}-%{version}.tar.gz
Patch0: %{name}-DESTDIR-%{version}.patch
Patch1: %{name}-ac_fixes.patch
Patch2: expat-1.95.8-man1dir.patch
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup
%patch0 -p1
%patch1 -p1
%patch2 -p1

%Build
./configure --prefix=%{_prefix} --host=$CFGHOST
 --build=%{_build} --mandir=%{_mandir}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx} install
rm $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la
%Clean
rm -rf $RPM_BUILD_ROOT
%Files
%defattr(-,root,root)
%{pfx}/*
```

– **Libxml2**: aggiornare dalla versione 2.6.28 alla versione 2.7.2
Modificare il file
/home/<username>/ltib/dist/lfs-5.1/libxml2/libxml2.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: Libraries, includes, etc. to develop
         XML/HTML applications
Name: libxml2
Version: 2.7.2
Release: 0
Vendor: Freescale
Packager: Jason Jin, Stuart Hughes, Kurt Mahan
Group : Development/Libraries
URL: http://www.xmlsoft.org
Source: %{name}-%{version}.tar.gz
License: MIT
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
./configure --prefix=%{_prefix} --host=$CFGHOST
--with-history --without-python
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
find $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/ -name
"*.la" -exec rm -rf {} \;

%Clean
rm -rf $RPM_BUILD_ROOT
%Files
%defattr(-, root, root)
%{pfx}/*
```

– **Glib2**: aggiornare dalla versione 2.12.11 alla versione 2.17.6
Modificare il file
/home/<username>/ltib/dist/lfs-5.1/glib2/glib2.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: A library of functions used by GDK,
         GTK+, and many applications
Name: glib2
Version: 2.17.6
Release: 1
License: LGPL
Vendor: Freescale
Packager: Stuart Hughes/Kurt Mahan
Group: System Environment/Libraries
Source: glib-%{version}.tar.gz
Patch1: glib2-2.12.11-relink.patch
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup -n glib-%{version}
%patch1 -p1

%Build
# prevent configure from trying to compile and
# run test binaries for the target.
glib_cv_stack_grows=no \
glib_cv_uscore=no \
ac_cv_func_posix_getpwuid_r=yes \
ac_cv_func_posix_getgrgid_r=yes \
./configure --prefix=%{_prefix} --host=$CFGHOST
    --build=%{_build}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
find $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib
```

```
   -name "*.la" | xargs rm -f

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Scaricare il pacchetto:
http://ftp.gnome.org/pub/gnome/sources/glib/2.17/glib-2.17.5.tar.gz
Copiare il pacchetto nella cartella /opt/freescale/pkgs

– **Zlib**: aggiornare dalla versione 1.1.4 alla versione 1.2.3.
Modificare il file
/home/<username>/ltib/dist/lfs-5.1/zlib/zlib.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: zlib compression utilities and libraries
Name: zlib
Version: 1.2.3
Release: 2
License: zlib (Distributable)
Vendor: Freescale
Packager: Stuart Hughes
Group: Development/Libraries
Source: %{name}-%{version}.tar.bz2
Patch1: zlib-1.2.3-arflags-1.patch
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup
%patch1 -p1

%Build
./configure --prefix=%{_prefix} --shared
mv Makefile Makefile.shared
./configure --prefix=%{_prefix}
```

```
mv Makefile Makefile.static
make -f Makefile.shared
make -f Makefile.static

%Install
rm -rf $RPM_BUILD_ROOT
mkdir -p $RPM_BUILD_ROOT/%{pfx}/%{_prefix}
make -f Makefile.shared install prefix=
   ${RPM_BUILD_ROOT}/%{pfx}/%{_prefix}
make -f Makefile.static install prefix=
   ${RPM_BUILD_ROOT}/%{pfx}/%{_prefix}

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

– **Fontconfig**: aggiornare dalla versione 2.2.2 alla versione 2.4.2
Modificare il file
/home/<username>/ltib/dist/lfs-5.1/fontconfig/fontconfig.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: Font configuration and customization library
Name: fontconfig
Version: 2.4.2
Release: 1
License: MIT
Vendor: Freescale
Packager: Stuart Hughes
Group: System Environment/Libraries
Source: %{name}-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
```

```
# we don't have docbook-utils in the distribution
ac_cv_prog_HASDOCBOOK=no \
ac_cv_prog_CC_FOR_BUILD="${BUILDCC}" \
./configure --prefix=%{_prefix} --host=$CFGHOST
      --build=%{_build} \
      --with-arch=$GNUTARCH \
      --sysconfdir=%{_sysconfdir}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -rf $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la

perl -pi -e 's,^</fontconfig>,
<dir>%{_prefix}/X11R6/lib/X11/fonts/TTF</dir>
<dir>%{_prefix}/X11R6/lib/X11/fonts/Type1</dir>
<dir>%{_prefix}/X11R6/lib/X11/fonts/truetype</dir>
</fontconfig>
,;
 ' $RPM_BUILD_ROOT/%{pfx}/%{_sysconfdir}/
   fonts/local.conf

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file */home/<username>/ltib/config/userspace.lkc* e inserire le seguenti righe dopo il pacchetto Flex:

```
config PKG_FONTCONFIG
   depends CAP_HAS_MMU
   select PKG_EXPAT
   select PKG_FREETYPE
   select PKG_ZLIB
   bool "fontconfig"
   help
     Fontconfig is designed to locate fonts
     within the system and select them
     according to application requirements
```

– **Pixman**: andare ad aggiungere questo pacchetto con la versione 0.11.10
Creare il file
/home/<username>/ltib/dist/lfs-5.1/pixman/pixman.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: Pixman library
Name: pixman
Version: 0.11.10
Release: 1
License: X11
Vendor: X.Org Foundation
Packager: ProFUSION embedded systems
     <contact@profusion.mobi>
Group: System Environment/Libraries
URL: http://www.x.org/
Source: %{name}-%{version}.tar.bz2
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description

%Prep
%setup

%Build
./configure --prefix=%{_prefix} --host=$CFGHOST
    --build=%{_build}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
find $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/
    -name "*.la" | xargs rm -f

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file */home/<username>/ltib/config/userspace.lkc* e inserire le seguenti righe dopo il pacchetto Php:

```
config PKG_PIXMAN
   bool "pixman"
```

Aggiungere il pacchetto nel file /home/<username>/ltib/dist/lfs-5.1/common, dopo il pacchetto Fontconfig

```
PKG_PIXMAN = pixman
```

– **Libpng**: aggiungere la patch alla versione 2.2.2 alla versione 2.4.2 Modificare il file */home/<username>/ltib/dist/lfs-5.1/libpng/libpng.spec* come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: A library of functions for manipulating
         PNG image format files
Name: libpng
Version: 1.2.8
Release: 1
License: distributable, OSI approved
Vendor: Freescale
Packager: Stuart Hughes
Group: System Environment/Libraries
Source: %{name}-%{version}.tar.bz2
Patch1: libpng-1.2.8-link_to_proper_libs-1.patch
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup
%patch1 -p1

%Build
make prefix=%{_prefix} -f scripts/makefile.linux

%Install
rm -rf $RPM_BUILD_ROOT
```

```
mkdir -p $RPM_BUILD_ROOT/%{pfx}/%{_prefix}
make prefix=$RPM_BUILD_ROOT/%{pfx}/%{_prefix}
   install -f scripts/makefile.linux

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

– **DirectFB**: aggiornare dalla versione 0.9.24 alla versione 1.2.0
Modificare il file
/home/<username>/ltib/dist/lfs-5.1/DirectFB/DirectFB.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: DirectFB is a graphics library
         for embedded systems
Name: DirectFB
Version: 1.2.0
Release: 1
License: LGPL
Vendor: Freescale
Packager: WMSG
Group: System Environment/Base
Source: %{name}-%{version}.tar.gz
Patch1: DirectFB-1.1.0-ppcasm.patch
Patch4: DirectFB-0.9.24-relink.patch
Patch5: DirectFB-1.1.0-linux-config-h.patch
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup
%patch1 -p1
%patch4 -p1
%patch5 -p1

%Build
KHDR_DIR=$DEV_IMAGE/usr/src/linux/include
```

```
if [ ! -f $KHDR_DIR/linux/autoconf.h ]
then
    cat <<TXT

No file: $KHDR_DIR/linux/autoconf.h

You need to build the kernel and have
'Include kernel headers' set
to build this package

TXT
    exit 1
fi
export FREETYPE_CONFIG=${DEV_IMAGE}
       /usr/bin/freetype-config
export FREETYPE_CFLAGS="`${FREETYPE_CONFIG}
       --prefix=${DEV_IMAGE}/%{_prefix} --cflags`"
export FREETYPE_LIBS="`${FREETYPE_CONFIG}
       --prefix=${DEV_IMAGE}/%{_prefix} --libs`"

if [ -n "$PKG_DIRECTFB_WANT_TS" ]
then
    TSOPTS="--with-inputdrivers=tslib,
    keyboard,linuxinput"
fi
./configure --enable-shared --host=$CFGHOST
            --build=%{_build} \
            --prefix=%{_prefix}
            --with-gfxdrivers=none \
            --disable-x11 --enable-fbdev \
            --enable-video4linux2 --disable-sdl \
            --enable-zlib $TSOPTS

perl -p -i -e 's,^#define\s+HAVE_ASM_PAGE_H\s+1,/\*
   #define HAVE_ASM_PAGE_H 1 \*/,' config.h
make KHDR=$KHDR_DIR

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
find $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib
   -name \*.la | xargs rm -f

%Clean
```

```
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Scaricare il pacchetto:

```
http://www.directfb.org/downloads/Core/
DirectFB-1.2/DirectFB-1.2.0.tar.gz
```

Copiare il pacchetto nella cartella /opt/freescale/pkgs

– **Cairo**: aggiungere questo pacchetto con la versione 1.6.0
Creare il file
/home/<username>/ltib/dist/lfs-5.1/cairo/cairo.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: A 2D vectorial drawing library
         needed by GTK from 2.8.0
Name: cairo
Version: 1.6.0
Release: 1
License: LGPL or MPL 1.1
Vendor: Freescale
Packager: Stuart Hughes
Group: System Environment/Libraries
URL: http://cairographics.org/
Source: %{name}-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
XTRA_OPTS="--disable-win32"

if ! rpm --dbpath %{_dbpath} -q xorg-server &>/dev/null
then
        XTRA_OPTS='--disable-xlib';
fi
```

```
./configure --prefix=%{_prefix} --host=$CFGHOST
   --build=%{_build} $XTRA_OPTS --enable-directfb
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Bzip2:

```
config PKG_CAIRO
    depends CAP_HAS_MMU
    select PKG_LIBPNG
    select PKG_DIRECTFB
    bool "cairo"
    help
      A 2D vectorial drawing library
      needed by GTK from 2.8.0
```

Aggiungere il pacchetto nel file /home/<username>/ltib/dist/lfs-5.1/common, dopo il pacchetto DirectFB

```
     PKG_CAIRO = cairo
```

Scaricare il pacchetto:

```
http://cairographics.org/releases/cairo-1.6.0.tar.gz
```

Copiare il pacchetto nella cartella /opt/freescale/pkgs

– **Pango**: aggiornare dalla versione 1.4.0 alla versione 1.20.0
Modificare il file
/home/<username>/ltib/dist/lfs-5.1/pango/pango.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: Library for layout and rendering
         of internationalized text.
Name: pango
Version: 1.20.0
Release: 1
License: LGPL
Vendor: Freescale
Packager: Stuart Hughes
Group: System Environment/Libraries
Source: %{name}-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
if rpm --dbpath %{_dbpath} -q xorg-server &>/dev/null
then
        extra_opts='--with-x';
else
        extra_opts='--without-x';
fi

./configure --prefix=%{_prefix} --host=$CFGHOST
            --build=%{_build} $extra_opts
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/pango/
        */modules/*.la
```

```
install -d $RPM_BUILD_ROOT/%{pfx}/etc/rc.d/init.d
initscript=$RPM_BUILD_ROOT/%{pfx}/etc/rc.d/init.d/pango
cat > $initscript << EOF

if [ "\$1" = "start" ]; then
echo pango: creating module list
if [ ! -f /usr/etc/pango/pango.modules ]; then
mkdir -p /usr/etc/pango
pango-querymodules > /usr/etc/pango/pango.modules
fi
fi
EOF
chmod 744 $initscript

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Oprofile:

```
config PKG_PANGO
    depends CAP_HAS_MMU
    select PKG_FONTCONFIG
    select PKG_GLIB2
    select PKG_CAIRO
    bool "pango"
    help
       System for layout and rendering
       of internationalized text
```

Spostare il pacchetto, nel file /home/<username>/ltib/dist/lfs-5.1/common, dopo il pacchetto Cairo:

```
 PKG_PANGO = pango
```

Scaricare il pacchetto:

```
http://ftp.gnome.org/pub/gnome/sources/pango/
    1.20/pango-1.20.0.tar.gz
```

Copiare il pacchetto nella cartella /opt/freescale/pkgs

– **ATK**: aggiungere questo pacchetto con la versione 1.20.0
Creare il file
/home/<username>/ltib/dist/lfs-5.1/atk/atk.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: Interfaces for accessibility support
Name: atk
Version: 1.20.0
Release: 1
License: LGPL
Vendor: Freescale
Packager: Stuart Hughes
Group: System Environment/Libraries
Source: %{name}-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
./configure --prefix=%{_prefix} --host=$CFGHOST
             --build=%{_build}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -rf $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Apptrk_ppcbin:

```
config PKG_ATK
    depends CAP_HAS_MMU
    select PKG_GLIB2
    bool "atk"
    help
       Interfaces for accessibility support
```

– **LibTiff**: aggiungere questo pacchetto con la versione 3.8.2 Creare il file */home/<username>/ltib/dist/lfs-5.1/libtiff/libtiff.spec* come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: A library of functions for
         manipulating TIFF format image files
Name: libtiff
Version: 3.8.2
Release: 1
License: Distributable
Vendor: Freescale
Packager: Stuart Hughes
Group: System Environment/Libraries
Source: tiff-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup -n tiff-%{version}

%Build
./configure --prefix=%{_prefix}
   --host=$CFGHOST --build=%{_build}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
```

```
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Libtermcap:

```
config PKG_LIBTIFF
    depends CAP_HAS_MMU
    bool "libtiff"
    help
      A library of functions for
      manipulating TIFF format image files
```

– **Gtk**: aggiungere questo pacchetto con la versione 2.12.12 Creare il file
/home/<username>/ltib/dist/lfs-5.1/gtk2/gtk2.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: The GIMP ToolKit (GTK+),
         a library for creating GUIs for X
Name: gtk2
Version: 2.12.12
Release: 1
License: LGPL
Vendor: Freescale
Packager: Stuart Hughes
Group: System Environment/Daemons
Source: gtk+-%{version}.tar.gz
#Patch1: gtk2-2.12.1-builtin_icons.patch
Patch2: gtk2-2.12.1-no-update-icon-cache.patch
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
```

```
%setup -n gtk+-%{version}
#%patch1 -p1
%patch2 -p1

%Build
if rpm --dbpath %{_dbpath} -q xorg-server &>/dev/null
then
extra_opts='--with-x';
else
extra_opts='--without-x --with-gdktarget=directfb';
fi

./configure --prefix=%{_prefix} --host=$CFGHOST \
            --build=%{_build} \
            --without-x --with-gdktarget=directfb -C

perl -pi -e 's,^sys_lib_search_path_spec=.*,
     sys_lib_search_path_spec=,' libtool
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
find $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/
        -name "*.la" | xargs rm -f
install -d $RPM_BUILD_ROOT/%{pfx}/etc/rc.d/init.d
initscript=$RPM_BUILD_ROOT/%{pfx}/etc/rc.d/init.d/gtk2
cat > $initscript << EOF
#!/bin/sh
# Copyright 2008, Freescale Semiconductor Inc.
#
if [ "\$1" = "start" ]; then
echo gtk: creating gdk-pixbuf.loaders
if [ ! -f /usr/etc/gtk-2.0/gdk-pixbuf.loaders ] ; then
mkdir -p /usr/etc/gtk-2.0
/usr/bin/gdk-pixbuf-query-loaders >
/usr/etc/gtk-2.0/gdk-pixbuf.loaders
fi
fi
EOF
chmod 744 $initscript

#%Post
#export PATH=%{_prefix}/bin:$PATH
```

```
#mkdir -p %{_prefix}/etc/gtk-2.0
#gdk-pixbuf-query-loaders > %{_prefix}/etc/
     gtk-2.0/gdk-pixbuf.loaders

# Todo?  This may break certain workflows.
#%Postun
#echo Post uninstall: remove file
#generated by init script
#if [ -n "$RPM_INSTALL_PREFIX" -a -f
 $RPM_INSTALL_PREFIX/usr/etc/gtk-2.0/
 gdk-pixbuf.loaders ]; then
# rm -f $RPM_INSTALL_PREFIX/usr/etc/
gtk-2.0/gdk-pixbuf.loaders
#fi

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Gst_FslV4Lsink:

```
config PKG_GTK2
  depends CAP_HAS_MMU
  select PKG_PANGO
  select PKG_LIBTIFF
  select PKG_ATK
  select PKG_CAIRO
  bool "gtk+"
  help
   The GIMP ToolKit (GTK+), a library for creating GUIs
```

Spostare il pacchetto nel file /home/<username>/ltib/dist/lfs-5.1/common, dopo il pacchetto Pango

```
PKG_GTK2 = gtk2
```

Scaricare il pacchetto:

```
http://ftp.gnome.org/pub/gnome/sources/
       gtk+/2.12/gtk+-2.12.12.tar.gz
```

Copiare il pacchetto nella cartella /opt/freescale/pkgs

- A questo punto, applicare le modifiche effettuate. Entrare in ltib:

  ```
  ./ltib -c
  ```

  Selezionare i seguenti pacchetti nel menu “Package list” e procedere con l’installazione

  - Expat
  - Freetype
  - Fontconfig
  - Pixman
  - Libpng
  - DirectFB
  - Cairo
  - Pango
  - Atk
  - Libtiff
  - Gtk2

- Si prosegue per installare e aggiornare le librerie per installare Linphone, in particolare le librerie per il multimedia e il SIP.

  - **Alsa-lib**: aggiornare dalla versione 1.0.11rc2 alla versione 1.0.18
    Modificare il file
    /home/<username>/ltib/dist/lfs-5.1/alsa-lib/alsa-lib.spec
    come segue:

    ```
    %define pfx /opt/freescale/rootfs/%{_target_cpu}

    Summary: A libraries for ALSA
            (Advanced Linux Sound Architecture)
    Name: alsa-lib
    Version: 1.0.18
    Release: 0
    License: LGPL
    Vendor: Freescale
    Packager: Ross Wille/Wang Huan
    Group: System Environment/Libraries
    Source: alsa-lib-%{version}.tar.bz2
    Patch0: alsa-lib-1.0.18-nommu.patch
    Patch1: alsa-lib-1.0.18-relink.patch
    BuildRoot: %{_tmppath}/%{name}
    ```

```
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup
%patch0 -p1
%patch1 -p1

%Build
config_args="--disable-python"
if [ "$GNUTARCH" = m68knommu ]
then
    config_args="$config_args --disable-shared"
fi
./configure --prefix=%{_prefix} --host=$CFGHOST \
            --build=%{_build} $config_args
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/
      alsa-lib/*/*.la

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aprire il file
/home/<username>/ltib/config/platform/imx27pdk/pkg_map
e togliere la riga

```
PKG_ALSA_LIB = alsa-lib-1.0.11rc2
```

– **Readline**: aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file */home/<username>/ltib/config/userspace.lkc* e inserire le seguenti righe dopo il pacchetto Qtopia_Want_Cache_Host_Tools:

```
config PKG_READLINE
    depends CAP_HAS_MMU
    bool "readline"
    help
       Libraries to support command line
       editing functions
```

– **Libogg**: aggiungere questo pacchetto con la versione 1.1.3 Creare il file */home/<username>/ltib/dist/lfs-5.1/libogg/libogg.spec* come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary: The ogg lib
Name: libogg
Version: 1.2.0
Release: 1
License: BSD
Vendor: Maxtrack
Packager: Alan Carvalho de Assis
Group: System Environment/Libraries
Source: libogg-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
if [ `echo "${PLATFORM}" | grep "mpc85"` ]
then
    E500_BSP="yes"
else
    E500_BSP="no"
fi
case "${PLATFORM}" in
```

```
    mpc8323eisr | mpc832x_rdb | mpc832xemds |
     mpc860fads | qs875s | tqm823l | zen)
        HARD_FP="no" ;;
    *)
        HARD_FP="yes" ;;
esac
HAVE_E500=${E500_BSP} \
HAVE_FPU=${HARD_FP} \
./configure --prefix=%{_prefix} --host=$CFGHOST \
            --build=%{_build}
make -j1

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Libpng:

```
config PKG_LIBOGG
    depends CAP_HAS_MMU
    select PKG_GLIB2
    bool "libogg"
    help
      This is the library OGG from Xiph.Org Foundation.
```

Aggiungere il pacchetto nel file /home/<username>/ltib/dist/lfs-5.1/common, dopo il pacchetto Gtk2

```
     PKG_LIBOGG = libogg
```

Scaricare il pacchetto:

```
http://downloads.xiph.org/releases/
       ogg/libogg-1.2.0.tar.gz
```

Copiare il pacchetto nella cartella /opt/freescale/pkgs

– **Speex**: aggiungere questo pacchetto con la versione 1.2rc1
Creare il file
/home/<username>/ltib/dist/lfs-5.1/speex/speex.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}
Summary: The speex lib
Name: speex
Version: 1.2rc1
Release: 1
License: BSD
Vendor: Maxtrack
Packager: Alan Carvalho de Assis
Group: System Environment/Libraries
Source: %{name}-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
./configure --prefix=%{_prefix} --host=$CFGHOST \
            --build=%{_build}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Skell_Want_Terminfo:

```
config PKG_SPEEX
   bool "speex"
```

Aggiungere il pacchetto nel file /home/<username>/ltib/dist/lfs-5.1/common, dopo il pacchetto Libogg:

```
PKG_SPEEX = speex
```

Scaricare il pacchetto:

`http://downloads.xiph.org/releases/speex/speex-1.2rc1.tar.gz`

Copiare il pacchetto nella cartella /opt/freescale/pkgs

– **Libosip2**: aggiungere questo pacchetto con la versione 3.3.0
Creare il file
/home/<username>/ltib/dist/lfs-5.1/libosip2/libosip2.spec
come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}
Summary: The libosip2 lib
Name: libosip2
Version: 3.3.0
Release: 1
License: BSD
Vendor: Maxtrack
Packager: Alan Carvalho de Assis
Group: System Environment/Libraries
Source: %{name}-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
./configure --prefix=%{_prefix} --host=$CFGHOST \
            --build=%{_build}
make
```

```
%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Liboil:

```
config PKG_LIBOSIP2
   bool "libosip2"
```

Aggiungere il pacchetto nel file /home/<username>/ltib/dist/lfs-5.1/common, dopo il pacchetto Speex:

```
PKG_LIBOSIP2 = libosip2
```

Scaricare il pacchetto:

```
http://ftp.gnu.org/gnu/osip/libosip2-3.3.0.tar.gz
```

Copiare il pacchetto nella cartella /opt/freescale/pkgs

– **LibeXosip2**: aggiungere questo pacchetto con la versione 3.3.0 Creare il file */home/<username>/ltib/dist/lfs-5.1/libeXosip2/libeXosip2.spec* come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}
Summary: The libeXosip2 lib
Name: libeXosip2
Version: 3.3.0
Release: 1
License: BSD
Vendor: Maxtrack
Packager: Alan Carvalho de Assis
Group: System Environment/Libraries
Source: %{name}-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}
```

```
%Description
%{summary}
%Prep
%setup

%Build
./configure --prefix=%{_prefix} --host=$CFGHOST \
            --build=%{_build}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la

%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Libosip:

```
config PKG_LIBEXOSIP2
   bool "libosip2"
```

Aggiungere il pacchetto nel file /home/<username>/ltib/dist/lfs-5.1/common, dopo il pacchetto Libosip:

```
PKG_LIBEXOSIP2 = libeXosip2
```

Scaricare il pacchetto:

```
http://download.savannah.gnu.org/releases/exosip/
     libeXosip2-3.3.0.tar.gz
```

Copiare il pacchetto nella cartella /opt/freescale/pkgs

– **Linphone**: aggiungere questo pacchetto con la versione 3.3.0 Creare il file /home/<username>/ltib/dist/lfs-5.1/linphone/linphone.spec come segue:

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}
```

```
Summary: The Linphone softphone
Name: linphone
Version: 3.3.2
Release: 1
License: GPL
Source: %{name}-%{version}.tar.gz
BuildRoot: %{_tmppath}/%{name}
Prefix: %{pfx}

%Description
%{summary}
%Prep
%setup

%Build
./configure --prefix=%{_prefix} --host=$CFGHOST \
            --build=%{_build}
make

%Install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT/%{pfx}
rm -f $RPM_BUILD_ROOT/%{pfx}/%{_prefix}/lib/*.la

%Clean
rm -rf $RPM_BUILD_ROOT
%Files
%defattr(-,root,root)
%{pfx}/*
```

Aggiungere questo pacchetto nella lista dei pacchetti di LTIB. Aprire il file /home/<username>/ltib/config/userspace.lkc e inserire le seguenti righe dopo il pacchetto Libxml2:

```
config PKG_LINPHONE
   bool "linphone"
```

Aggiungere il pacchetto nel file /home/<username>/ltib/dist/lfs-5.1/common, dopo il pacchetto LibeXosip2:

```
PKG_LINPHONE = linphone
```

Scaricare il pacchetto:

```
http://download.savannah.gnu.org/releases/linphone/
    3.3.x/sources/linphone-3.3.2.tar.gz
```

Copiare il pacchetto nella cartella /opt/freescale/pkgs

- A questo punto, applicare le modifiche effettuate. Entrare in ltib:

  ```
  ./ltib -c
  ```

  Selezionare i seguenti pacchetti nel menu “Package list”

  - Readline
  - Libogg
  - Speex
  - Libosip2
  - LibeXosip2
  - Linphone

- Configurare Linphone inserendo il file “.linphonerc” nella directory */home/<username>/ltib/rootfs/*. Il file “.linphonerc” deve avere le seguenti righe:

```
[net]
download_bw=0
upload_bw=0
nat_address=192.168.0.100
firewall_policy=1
mtu=0

[sip]
sip_port=5060
guess_hostname=1
contact="campanello" <campanello:campanello@192.168.0.100>
inc_timeout=15
use_info=0
use_ipv6=0
register_only_when_network_is_up=0
default_proxy=0
use_rfc2833=0

[rtp]
audio_rtp_port=7078
video_rtp_port=9078
audio_jitt_comp=60
video_jitt_comp=0
nortp_timeout=30
```

```
[sound]
remote_ring=/usr/share/sounds/linphone/ringback.wav

[video]
enabled=0
size=cif
display=0
capture=0
show_local=0
self_view=0

[audio_codec_0]
mime=speex
rate=32000
enabled=1

[audio_codec_1]
mime=speex
rate=16000
enabled=1

[audio_codec_2]
mime=speex
rate=8000
enabled=1

[audio_codec_3]
mime=PCMU
rate=8000
enabled=1

[audio_codec_4]
mime=PCMA
rate=8000
enabled=1

[proxy_0]
reg_proxy=sip:campanello@192.168.0.100
reg_route=sip:
reg_identity=sip:campanello@192.168.0.100
reg_expires=3600
reg_sendregister=1
publish=0
dial_escape_plus=0
```

```
[auth_info_0]
username=campanello
userid=campanello
passwd=campanello
realm=192.168.0.100
```

C.3 Server NFS

Prima di importare tutti i file all'interno della scheda I.MX27-PDK, i vari test di funzionalità vengono fatti sfruttando il server NFS, in modo tale che la scheda I.MX27 carichi i vari file da un server NFS. Per installare il server NFS si eseguono i seguenti passi:

- Installare il pacchetto "nfs-kernel-server"
- Modificare il file /etc/export aggiungendo la seguente riga:

  ```
  /home/<username>/ltib/rootfs *(rw,no_root_squash,
         no_subtree_check,async)
  ```

  Dove <username> è il nome utente con cui stiamo lavorando
- Riavviare il server NFS con il seguente comando:

  ```
  /etc/init.d/nfs-kernel-server restart
  ```

C.4 TFTP

La comunicazione tra la scheda I.MX27 e il server NFS viene garantita grazie al protocollo TFTP (Trivial File Transfer Protocol). Il TFTP è un protocollo di trasferimento file molto semplice, con le funzionalità di base del FTP. Per installare TFTP si devono installare i seguenti pacchetti:

- tftp
- tftpd
- openbsd-inetd

Si prosegue con la creazione della cartella dove la scheda I.MX andrà a recuperare i file di boot all'avvio.

```
mkdir /tftpboot
```

Si imposta i giusti permessi:

```
chmod a+x /tftpboot
```

Si configura il file /etc/inetd.conf con la seguente riga:

```
tftp dgram udp wait nobody
   /usr/sbin/tcpd /usr/sbin/in.tftpd /tftpboot
```

Si riavvia il server:

```
/etc/init.d/openbsd-inetd restart
```

All'interno di questa directory viene copiato il file zImage creato:

```
cp /home/<username>/ltib/rootfs/boot/zImage /tftpboot
```

C.5 Importare il sistema operativo nella board

Prima di importare il sistema operativo nella scheda I.MX27 si deve formattare la flash NAND e importare il boot loader. Per eseguire queste operazioni ci viene in aiuto un'applicazione chiamata ATK (Advanced ToolKit). Questa applicazione è stata sviluppata per lavorare sotto Linux ma grazie all'applicazione Wine di Linux possiamo installare applicazioni Windows che lavorano anche in Linux.

I passi quindi da seguire sono:

- Installare il pacchetto "wine" e tutte le sue dipendenze da Linux
- Dal sito di Freescale scaricare l'applicazione IMX27_ATK_TOOLKIT_R160 (www.freescale.com, previa registrazione dell'utente)
- Scompattare il pacchetto in /home/<username>/
- Aprire la console come utente root ed eseguire il seguente comando per installare ATK

  ```
  wine /home/<username>/IMX_ATK_TOOLKIT_R160/
  FSL_ATK_TOOL_STD_1_60/
  FSL_ATK_TOOL_WINS_STD_INSTLL_1_60.exe
  ```

- Seguire le istruzioni di installazione.
- Alla fine deselezionare la voce di avviare ATK alla fine dell'installazione, in quanto non si avvia perché mancano dei file

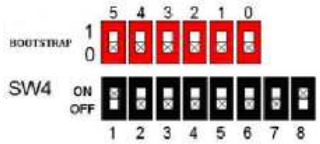
- Da un sistema operativo Windows prelevare i seguenti due file:

```
c:\windows\system32\mfc42.dll
c:\windows\system32\msvcp60.dll
```

- Copiare entrambi i file nella cartella */root/.wine/drive_c/windows/system32/*
- Avviare ATK con il seguente comando (come utente root)

```
wine /root/.wine/drive_c/Programmi/freescale/
AdvancedToolKit-STD/ADSToolkit_std.exe
```

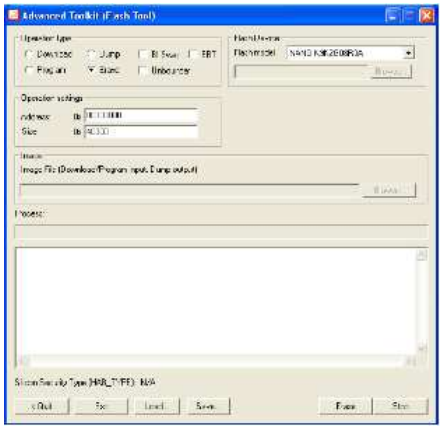
Prima di procedure con ATK si deve impostare correttamente i jumper i gli switch sulla board di debug. La figura sotto mostra come devono essere impostati per formattare e programma la NAND flash:



Una volta impostati correttamente i jumper e gli switch si può procedere con la formattazione e la programmazione della NAND flash, seguendo le seguenti procedure:

- Collegare il cavetto della porta seriale sia alla scheda PDK che al PC che ha in esecuzione il programma ATK
- Accendere la scheda PDK
- Avviare ATK
- Selezionare
- "i.MX27 TO2" sul campo "i.mx CPU"
- "COM1" sul campo "Serial Port"

- Premere il pulsante “Next”
- Selezionare il pulsante “Flash Tool”
- Premere il pulsante “GO”
- Selezionare
  - “Erase” nel campo “Operation type”
  - “NAND K9K2G08R0A” nel campo “Flash model”
  - 00000000 nel campo “Address”
  - 40000 nel campo “Size”

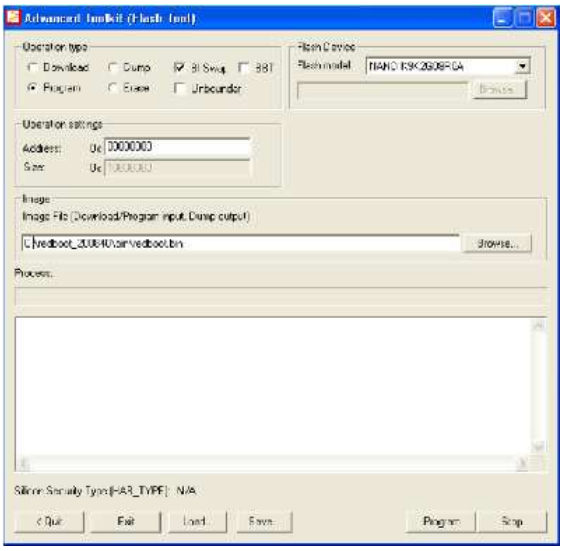


- Premere il pulsante “Erase”

Per caricare il boot loader nella scheda seguire i seguenti passi:

- Scompattare il file redboot_200840.zip che si trova all’interno del pacchetto scaricato di LTIB e che si dovrebbe trovare in */home/<username>/LPDK_iMX27_R1/*
- Eseguire ATK come prima

- Selezionare i passi precedenti fino ad arrivare alla schermata di “Flash Tool”



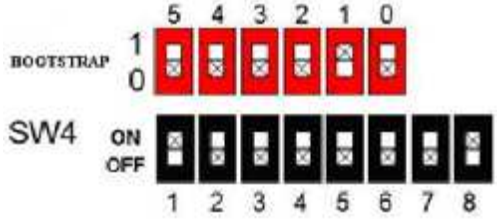
- Selezionare
  - “Program” nel campo “Operation type”
  - “BI Swap” nel campo “Operation type”
  - “NAND K9K2G08R0A” nel campo “Flash model”
- Nel campo “Image” selezionare con il bottone “Browse” il file del boot loader che si chiama “redboot.bin”. Tale file dovrebbe trovarsi in questo percorso:

```
/home/<username>/LPDK_iMX27_R1/redboot/redboot.bin
```

- Infine, premere il pulsante “Program”

Arrivati a questo punto si ha installato il boot loader nella scheda I.MX27-PDK, ora si deve programmare il kernel usando il boot loader caricato nella scheda. Per fare questo si deve utilizzare oltre alla connessione rete LAN, anche la connessione seriale della scheda, quindi si procede come segue:

- Con la scheda PDK spenta, impostare i jumper e gli switch sulla board debug in questo modo:



- Installare il pacchetto “GtkTerm” da Linux per effettuare la connessione seriale
- Collegare sia il cavo seriale che il cavo di rete LAN
- Avviare l’applicazione GtkTerm in Linux
- Dal menu di GtkTerm selezionare “Configuration” e poi l’opzione “Port”
- Impostare la velocità di bit per secondo a 115200 nella casella “Speed”
- Salvare l’impostazione
- Avviare la scheda PDK
- La scheda si avvia e al prompt di GtkTerm viene visualizzato dopo un po’ di secondi la seguente scritta:

```
Redboot>
```

- A questo punto si deve inizializzare la NAND, eseguendo questi comandi da RedBoot:

```
RedBoot> nand scan -o
RedBoot> fis init
```

Il sistema chiede se si è sicuri di continuare, e si risponde “y”

Per testare il corretto funzionamento di tutto il sistema si procede a lavorare tramite server NFS, cioè l’immagine del kernel e il rootfs vengono letti via rete LAN attraverso server NFS. Per fare questo eseguire i seguenti passi:

- Caricare l’immagine kernel con il seguente comando

```
RedBoot>load -r -b 0x100000 zImage -h<indirizzoServerNFS>
```

Dove *<indirizzoServerNFS>* è l’indirizzo del computer dove si è installato LTIB e si ha il server NFS installato

- Caricare il rootfs con il seguente comando

```
RedBoot> exec -b 0x100000 -l 0x200000
-c "noinitrd console=ttymxc0 root=/dev/nfsroot
rootfstype=nfsroot nfsroot=<indirizzoServerNFS>:
/home/<username>/ltib/rootfs rw ip=dhcp"
```

È possibile fare in modo che all'avvio della scheda venga eseguito in modo automatico il caricamento dell'immagine del kernel e del rootfs. Per fare questo occorre eseguire questi passi:

- Quando la scheda si avvia e si entra in modalità RedBoot eseguire il seguente comando:

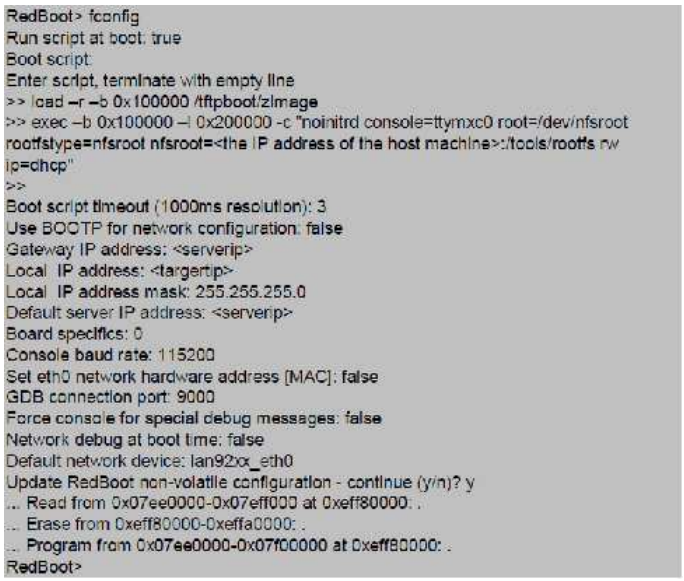
```
RedBoot> fconfig
```

- Alla richiesta iniziale se si vuole inserire uno script di avvio, basta rispondere "true" e inserire le righe che sono state inserite precedentemente, quindi:

```
load -r -b 0x100000 zImage -h <indirizzoServerNFS>
exec -b 0x100000 -l 0x200000 -c
"noinitrd console=ttymxc0 root=/dev/nfsroot
rootfstype=nfsroot nfsroot=<indirizzoServerNFS>:
/home/<username>/ltib/rootfs rw ip=dhcp"
```

- Premere due volte il tasto di INVIO per indicare a fconfig che si è concluso di inserire i script di avvio e procedere con le altre impostazioni

- Alla fine dare conferma di aggiornare la configurazione di RedBoot



Se tutta la configurazione funziona correttamente è possibile quindi caricare tutto il sistema operativo nella scheda, evitando così di sfruttare il server NFS. Per far questo bisogna seguire i seguenti passi:

- Modificare i permessi di alcuni file di LTIB

```
chmod 644 /home/<username>/ltib/rootfs/etc/pointercal
chmod 644 /home/<username>/ltib/rootfs/.linphonerc
chmod 644 /home/<username>/ltib/rootfs/etc/dropbear/
          dropbear_rsa_host_key
chmod 644 /home/<username>/ltib/rootfs/var/log/messages
chmod 644 /home/<username>/ltib/rootfs/var/log/messages.0
chmod 644 /home/<username>/ltib/rootfs/var/run/usb
chmod 644 /home/<username>/ltib/rootfs/.linphonec_history
```

- Entrare in LTIB e impostare la generazione del file JFFS2 con il menu:

  ```
  Target Image Generation->Options->Target image->jffs2
  ```

  Uscire da LTIB e salvare la nuova configurazione

- Copiare i nuovi file generati di zImage e rootfs nella cartella /tftpboot/

```
cp /home/<username>/ltib/rootfs.jffs2 /tftpboot/
cp /home/<username>/ltib/rootfs/boot/zImage
```

- Avviare la scheda I.MX
- Usando GtkTerm entrare in RedBoot (nel caso non si riesce ad entrare all'avvio a causa dell'avvio automatico premere subito CTRL+C)
- Da RedBoot digitare quanto segue per caricare zImage e rootfs:

```
load -r -b 0x100000 zImage -h <indirizzoServerNFS>
fis create -l 0xA00000 kernel

load -r -b 0x100000 rootfs.jffs2 -h <indirizzoServerNFS>
fis create -l 0x5000000 root
```

- Impostare l'avvio automatico:

  ```
  RedBoot> fconfig
  ```

  Impostare nei script le seguenti righe:

```
fis load kernel
exec -c "noinitrd console=ttymxc0,115200
root=/dev/mtdblock4 rw rootfstype=jffs2 ip=dhcp"
```

In questo modo la scheda si avvia in modo automatico senza prelevare le configurazioni dal server NFS.

Bibliografia

[1] **Scheda I.MX27-PDK.**
http://www.freescale.com/

[2] **Caratteristiche della board I.MX27.**
Nel CD allegato alla board, la cartella:
PDK1_Linux_Documentation/pdk10_imx27_Linux_RM.pdf

[3] **Manuale di configurazione della board I.MX27-PDK.**
Nel CD allegato alla board, la cartella:
PDK1_Linux_Documentation/pdk10_imx27_Linux_UG.pdf

[4] **Esempio di applicazione "Hello World".**
Nel CD allegato alla board, la cartella:
PDK1_Linux_Documentation/pdk10_imx27_Linux_HelloWorld_AN_
Applicazione Hello Word.pdf

[5] **Installazione ambiente di sviluppo**
http://www.imxdev.org/

[6] **LTIB**
http://www.bitshrine.org/ltib/

[7] **Linphone**
http://www.linphone.org/

[8] **SipToSis**
http://www.mhspot.com/sts/siptosis.html

[9] **Asterisk**
http://www.voip-info.org/wiki/view/Asterisk