

INAF-OAA Gruppo Strumentazione Infrarossa

Progetto Giano

I sistemi embedded Porteus per Giano

Revision : 1.12, Firenze, Giugno 2012

E.Giani, C.Baffa

Rapporto Interno dell'Osservatorio di Arcetri N° 5-2011

Sommario

*La complessa struttura di uno strumento astronomico moderno come **Giano** richiede una notevole quantità di intelligenza distribuita per poter funzionare affidabilmente.*

*La struttura che è stata costruita per **Giano** si basa su di un gruppo di demoni software[3] operanti sul calcolatore centrale di controllo e su due sistemi embedded, dotati anch'essi di demoni dedicati alle funzioni di inizializzazione, monitoring e controllo delle varie componenti hardware dello strumento.*

Il presente documento descrive la struttura e le possibili azioni di configurazione dei sottosistemi embedded.

1 Hardware dei sistemi embedded

Il sistema di controllo di **Giano** include due sistemi embedded, il primo, **pfm**, destinato al controllo dell'elettronica di acquisizione, il secondo, **xill**, con lo scopo di gestire la porzione di pre-slit.

I due sistemi hanno la caratteristica di fornire bus e interfacce di tipo standard, requisito che riteniamo fondamentale nella realizzazione di un sistema embedded.

Lo hardware di **pfm**^[7] è costituito da una scheda PC104 PFM-620S della AAEON con CPU Celeron 400/650MHz a basso consumo, controller video VIA VT8606, controller IDE, interfaccia di rete, due porte seriali RS232, porta parallela (non collegata nel nostro montaggio), due porte USB 1.1 e il supporto per moduli di memoria Type I CompactFlash. Tale sistema offre tutte le funzionalità di un classico PC, ma con un fattore di forma più compatto.

Lo hardware di **xill** è costituito da un sistema *Koala MicroPc*^[8] comprendente una scheda mini-itx, con processore VIA C7 ULV a 1,2 GHz a basso consumo, controller video VIA CX700, controller SATA, due interfacce di rete, una porta seriale RS232, USB 2.0 e il supporto per moduli di memoria Type I CompactFlash.

Per la locazione (sistema di preslit) cui è destinato, abbiamo preferito l'acquisto di un sistema completo di contenitore dissipante della ditta Koala.

Entrambi i sistemi embedded di **Giano** non prevedono l'uso di dischi con parti mobili, utilizzando invece una cartuccia di memoria tipo CompactFlash sulla quale è stato installato il sistema operativo e il software. Questo ci ha permesso di utilizzare un software molto simile per entrambi i sistemi

2 Il sistema operativo

La scelta di sistemi operativi per il mondo embedded è molto ampia, ma la tendenza è verso l'affermazione di sistemi *Open Source* tra cui spicca Linux. L'alternativa di un sistema proprietario o commerciale potrebbe offrire il vantaggio di migliori prestazioni per il real-time e/o una migliore armonizzazione delle applicazioni con dell'hardware. Spesso però tale vantaggio è più che compensato da una maggiore possibilità di innovazione oppure dal crosso vantaggio di un patrimonio di competenze tecniche già acquisite.

Grazie al libero accesso dei sorgenti, alla sua robustezza, affidabilità, flessibilità, personalizzazione e configurazione come pure alla disponibilità di molti protocolli di comunicazione, Linux è entrato a come giocatore di primo piano nel mondo embedded.

In ambito GNU/Linux vi sono molte distribuzioni adattate per lavorare con sistemi embedded ed è possibile, inoltre, crearne una personalizzata come già fatto dagli autori in altra occasione^[5].

Abbiamo comunque ritenuto più semplice, soprattutto per motivi di manutenzione, appoggiarsi ad un sistema già sviluppato e provato.

La nostra scelta è originariamente caduta sulla distribuzione GNU/Linux **Slax**^[12], di piccole dimensioni e basata sulla storica e intramontabile distribuzione Slackware.

Nata come distribuzione Linux Live CD, **Slax** era pensata non per essere installata sul disco rigido ma per lavorare da CD o da Pendrive USB. È altresì possibile

installare **Slax** ed i suoi derivati su di una partizione di un disco rigido o di una scheda di memoria (come SD o CompactFlash).

L'opzione di lavorare completamente in RAM lasciando libero l'accesso al disco, rende **Slax** particolarmente adatta ai sistemi embedded.

Un'altra caratteristica che ha contribuito a rendere la distribuzione **Slax** più interessante ai nostri occhi è la sua capacità di essere estesa tramite moduli aggiuntivi che arricchiscono il sistema di ulteriore software oltre a quello fornito di default.

Nel frattempo, sfortunatamente, il progetto **Slax** è stato abbandonato, ma da un suo *fork*, noto con il nome di *Slax-remix*, è nata la distribuzione **Porteus** un sistema portabile che unisce alla solida struttura di base un kernel e dei drivers aggiornati.

3 Configurazione dei sistema embedded

Porteus, come **Slax**, si appoggia a un sistema di *moduli* invece dei tradizionali pacchetti per installare o aggiungere funzionalità al sistema.

I moduli di **Porteus** hanno estensione *xzm* e sono file compressi con l'algoritmo di compressione LZMA, contenenti un *filesystem* di sola lettura di tipo *squashfs*^[10].

La distribuzione **Porteus** genera il proprio sistema di moduli a partire dai pacchetti **tgz** della distribuzione Slackware da cui eredita anche la struttura e le funzionalità.

Con alcuni script messi a disposizione dal sistema, è possibile trasformare pacchetti del formato **Slackware**, in moduli *xzm*.

In modo analogo alle distribuzioni classiche che prevedono un'ampia gamma di pacchetti, **Porteus** fornisce una lista di moduli disponibili e scaricabili dal sito internet della distribuzione, altri possono essere facilmente creati e configurati dagli sviluppatori a seconda delle loro necessità.

3.1 Organizzazione del filesystem di Porteus

Nella sua versione standard, la distribuzione **Porteus** contiene due directory: **/boot** e **/porteus**: per installare la distribuzione il loro contenuto deve essere copiato sul disco o sulla memoria flash.

Il folder **/boot** contiene i file di configurazioni del bootloader, gli script di installazione, alcune utility, il kernel Linux e l'immagine compressa del filesystem di partenza *initrd.gz*.

Al boot del sistema il *bootloader* avvia il *kernel* che esegue il mount dell'immagine *initrd*: a questo punto il sistema è in grado di montare i moduli *xzm* con tutti i file del sistema base **Porteus**.

Le directory con i moduli contenenti i file che popolano il *filesystem* del sistema *live* sono invece presenti nel *folder* **porteus**.

Quando **Porteus** è installato per la prima volta, vengono montati solo i moduli nel folder **/porteus/base**. Questi sono inseriti in ordine alfanumerico, così il modulo *000-kernel.xzm* è elaborato per primo, poi *001-core.xzm*, etc.

Oltre alla directory **/base**, in **/porteus** ci sono alcune directory, inizialmente vuote, che possono essere usate per configurare ed estendere il sistema:

`/porteus/modules`: contiene i moduli che devono essere attivati automaticamente all'avvio e che personalizzano il sistema. Ciascun modulo di questa directory viene inserito nel filesystem successivamente a quelli base.

`/porteus/optional`: contiene i moduli opzionali non inseriti automaticamente all'avvio del sistema, ma che possono essere attivati al volo dall'operatore a sistema già funzionante.

`/porteus/changes`¹: contiene le modifiche apportate al sistema se si sceglie di salvare queste in modo permanente.

Le modifiche sono applicate al sistema dopo l'attivazione dei moduli presenti in `base` e `modules`.

`/porteus/rootcopy/`: è una directory speciale in cui possono essere copiati quei file che vogliamo aggiungere al filesystem *live*, senza la necessità di convertirli in moduli. Questi files sono gli ultimi ad essere inseriti e sovrascrivono qualunque file già presente che abbia stesso *path* e stesso *nome*.

Il file estratti dai moduli compressi sono accessibili, in sola lettura, nella directory `/mnt/live/memory`. In particolare:

`/mnt/live/memory/images`: contiene una directory per ogni modulo attivato

`/mnt/live/memory/changes`: contiene tutti i file che sono stati modificati all'interno del filesystem.

4 Il software del progetto Giano

La struttura software di **Giano** rispecchia la complessità dello strumento. La figura 1 riporta uno schema di principio di tale struttura.

La filosofia generale impiegata è stata di creare alcuni demoni con compiti specifici, modellati sulla struttura hardware. Tali demoni comunicano tra di loro tramite canali realizzati con socket unix o di rete che utilizzano il paradigma di comunicazione client/server.

La maggior parte dei demoni sono vincolati ai sottosistemi hardware, mentre il demone **Gbridge/Balor** agisce da *middleware*, agendo come un ponte tra l'interfaccia utente ed in generale il mondo esterno e lo strumento.

4.1 Le librerie del progetto Giano

Le librerie sviluppate per il progetto **Giano** sono installate nella directory del progetto `/opt/softir/lib`.

Queste sono librerie dinamiche e, ad ogni modifica, la *cache* usata dal linker al run-time deve essere aggiornata con il comando `ldconfig` in modo da includere il path di ricerca sopra indicato, aggiunto nel file di configurazione `/etc/ld.so.conf`.

¹Se il disco è stato formattato con FAT16/32, le modifiche sono salvate nel file `/porteus/save.dat`.

Giano Acquisition System

Software layout: Structural View

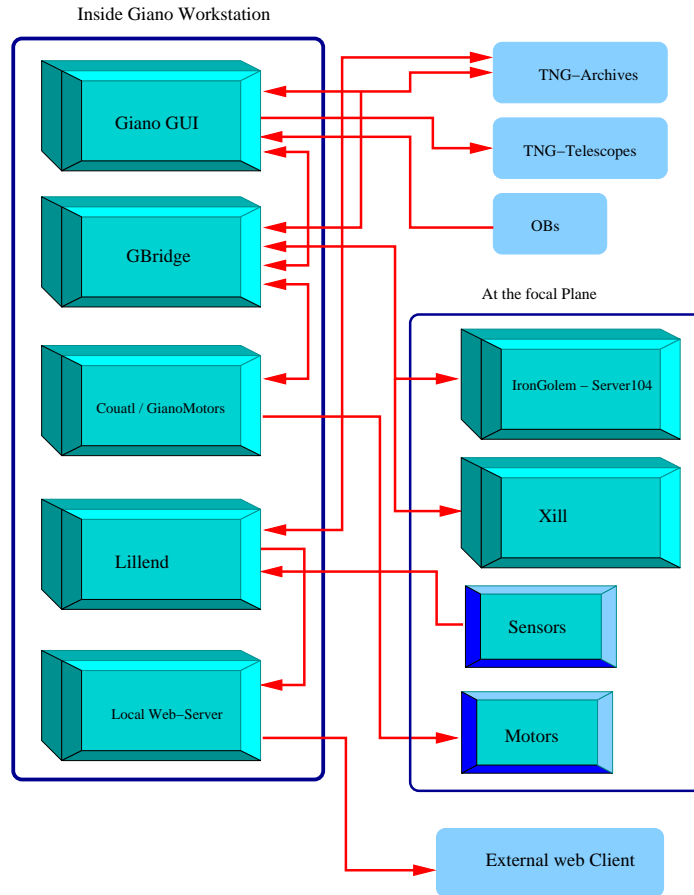


Figura 1: *Struttura del software di Giano*

Nei sistemi come *Slackware* questo comando viene eseguito durante la fase di *boot* da uno dei primi script di sistema (*rc.S*). Nel caso specifico questa operazione avviene troppo presto rispetto al momento in cui viene inserito il modulo *xzm* con le librerie di progetto e risulta indispensabile richiamare l'esecuzione del comando *ldconfig* successivamente all'inserimento del modulo in questione. *ldconfig* viene ri-eseguito all'interno del file di sistema *rc.local*, lo script di inizializzazione specifico del sistema locale, in cui vengono generalmente aggiunti, come ultima fase del processo di inizializzazione, tutti i comandi da eseguire al momento dello startup del sistema.

In questo script sono state inserite anche le righe di comando che avviano i server in esecuzione sui due sistemi embedded: *IronGolem/server104* su **pfm**, *xill* su **xill**.

4.2 I programmi server e le *utilities* minori

Sui due sistemi embedded **pfm** e **xill** sono attivi due applicazioni che controllano il funzionamento di parte dello hardware di **Giano**.

Il programma installato su **pfm** che permette il controllo dell'elettronica di **Giano**, sia in fase di test sia durante l'utilizzo regolare è *IronGolem/server104*[2]. *IronGolem/server104* è disegnato per operare in due modalità distinte: come demone per il controllo del sistema al telescopio, e in modo interattivo per lo sviluppo ed il test sia in laboratorio sia al telescopio.

Su **xill** operano simultaneamente due demoni distinti: *xill* e *fli_ccd*.

xill è il programma di basso livello che si occupa della gestione della pre-slit, mentre *fli_ccd* permette l'utilizzo della camera ottica di autoguida **MaxCam CM9** della *Finger Lakes Instrumentation*.

xill opera sotto il controllo della GUI che si occupa di riconfigurare lo stato dell'ottica di preslit e delle apparecchiature di calibrazione in base alle esigenze osservative. Lo scambio dei comandi avviene attraverso il demone middle-ware **Gbridge/Balor**[6].

Gli eseguibili del progetto **Giano**, come specificato nel documento [4], devono essere nella directory di sistema `/opt/softir/bin`.

Alcune delle applicazioni, come ad esempio *xill*, si appoggiano a file di configurazione che devono essere installati nella directory contenuta nella gerarchia `/opt/softir/share` e relativa al progetto stesso.

Nella directory `/opt/softir/bin` sono inoltre presenti alcune *utilities* sviluppate per accedere direttamente, a basso livello, ai sotto-sistemi della preslit:

powerboard: questa applicazione comunica attraverso seriale con il firmware della scheda **PowerBoard**: agisce su una serie di sub-devices e ne restituisce lo stato. Il comando '?' restituisce la lista dei comandi riconosciuti dal firmware della **PowerBoard**.

thorlabs: questo programma comunica attraverso bus USB con il controller della *Thorlabs* che gestisce il movimento della ruota con le ottiche di pre-slit. Digitando il carattere '?', viene restituita la lista dei comandi e della loro funzionalità.

freddy: questo applicativo comunica attraverso bridge RS232 - ethernet di **Giano**, con il controller Mercury della PI che supervisiona il posizionamento in altezza della fibra ottica. Il comando `init` inizializza il controller per lavorare con lo stage M111.1DG ad esso collegato. A parte quello citato, gli altri comandi accettati dal programma sono quelli appartenenti al set standard **GCS** di comandi della PI.[9]

startup_flicam: è uno script *bash* che avvia l'applicazione *fli_ccd* (vedi § 4.2.1).

4.2.1 L'applicazione *fli_ccd* e il driver linux **fliusb**

Il manager *fli_ccd* permette l'inizializzazione ed il controllo della camera **FLI MaxCam** utilizzata per l'autoguida .

fli_ccd opera sotto il controllo diretto di un apposito programma di autoguida in uso al TNG, producendo scatti singoli che sono poi trasferiti via *rsync* al computer che esegue il computo delle correzioni da apportare.

L'eseguibile *fli_ccd* non viene avviato dallo script `rc.local` come demone di sistema poichè la camera di autoguidera, per motivi di funzionalità, viene mantenuta spenta quando il sistema di acquisizione non lavora.

L'accensione della camera, pilotata dalla **PowerBoard** del sistema di preslit, per mezzo del server *xill*, manda in esecuzione lo *script bash startup_flicam* che controlla se il driver della camera FLI MaxCam è attivo nel kernel e, in caso positivo, lancia l'applicazione *fli_ccd*.

La funzionalità di basso livello della camera FLI MaxCam è gestita dal driver Linux `fliusb` il cui codice sorgente, fornito dalla ditta produttrice della camera, non risulta aggiornato per le versioni del kernel Linux più recenti della 2.6.24. L'aggiornamento delle API relative alle operazioni *DMA scatter-gather*, cioè operazioni di accesso alla memoria con trasferimento DMA di dati tra regioni di memoria *fisica* non contigue, a partire da questa versione del kernel in poi, ha provocato la incompatibilità con un numero elevato di drivers Linux, compreso quello sviluppato per la camera MaxCam.

La possibilità di passare a un kernel più datato per avere compatibilità tra questo e il driver `fliusb` è stata scartata per motivi legati a un supporto non corretto del chip FTDI o del chipset USB del **Koala microPC** in kernel più vecchi di quello in uso.

Abbiamo dunque lavorato sul codice del driver `fliusb` per implementare le modifiche necessarie: test eseguiti durante i mesi scorsi hanno mostrato il corretto funzionamento del nuovo driver, entrato a far parte del sistema finale della preslit. I sorgenti aggiornati fanno parte del modulo *softir_pfm_xill.xzm* insieme al resto del codice di **Giano**.

5 I moduli custom per la distribuzione Porteus

La distribuzione **Porteus** fornisce la possibilità di creare moduli adattati alle esigenze del progetto, al suo aggiornamento e manutenzione.

Di seguito è riportato l'elenco dei moduli **Porteus** prodotti per il progetto **Giano**. I primi due, *cvs* e *cfitsio*, non disponibili nel database della distribuzione **Porteus**, sono stati creati a partire dai corrispondenti pacchetti *tgz* della distribuzione *Slackware*², gli altri sono stati generati con la tecnica descritta nella sezione § 5.1.

`003_cfitsio-3.030-i386-1.xzm`: include le librerie di sviluppo `cfitsio` per il supporto completo dei file FITS.

Il file è stato copiato nella directory `/porteus/module` per essere caricato automaticamente all'avvio del sistema.

`cvs-1.11.23-i486-1.xzm`: implementa l'applicazione per l'accesso al server CVS ai fini di mantenere aggiornato il codice del progetto. Questo file viene copiato nella directory `/porteus/optional` per essere inserito solo su richiesta dello sviluppatore.

`011_porteus_xill_config.xzm`: implementa gli script e i file di configurazione del PC **xill** con l'indirizzo IP, il nome dell' host, server DNS, NTP etc.

²Inizialmente generati per la distribuzione **Slax**, i moduli originari *lzm* sono stati trasformati nel formato *xzm* con gli script messi a disposizione dalla distribuzione.

Include anche il driver e la libreria di sviluppo per la gestione della camera di autoguida FLI MaxCam. Questo modulo è specifico del sistema embedded **xill**.

Il file è stato copiato nella directory `/porteus/modules` di **xill** per essere inserito al boot del PC.

Una copia di backup di questo modulo è presente in `porteus/optional` del sistema embedded **pfm**.

`011_porteus_pfm_config.xzm`: fornisce la configurazione del PC **pfm** come l'indirizzo IP, il nome dell' host, server DNS, NTP etc.

Questo modulo è specifico del sistema embedded **pfm**. Analogo al precedente, comprende i file e gli script di configurazione del PC.

Una copia di backup di questo modulo è presente in `porteus/optional` del sistema embedded **xill**.

`010_softtir_pfm_xill.xzm`: questo modulo è comune a entrambi i sistemi embedded. Comprende gli eseguibili delle applicazioni destinate al controllo dei sotto-sistemi di **Giano** (pre-slit, elettronica di acquisizione, camera di autoguida), le librerie e gli include file comuni alle applicazioni (librerie `cfitsio`, `Irsock`, ...).

Il modulo include anche la *home* dell'utente `softtir` con parte del codice di sviluppo.

Il modulo è copiato nella directory `module` e, insieme agli altri, è inserito automaticamente all'avvio del sistema, ma prima di quelli di configurazione (`011_XX`).

5.1 Creazione dei moduli custom

Per generare i moduli *custom* dei due sistemi embedded, creiamo nel *folder* di lavoro, generalmente `/root`, l'albero delle directory in cui, nelle opportune locazioni, andiamo a copiare il codice ed i files ausiliari^[1] che andranno a completare il filesystem finale.

Ad esempio per il modulo `softtir_pfm_xill.xzm` procediamo come segue:

Creiamo la directory ausiliaria

```
# mkdir softtir_pfm_xill
```

Generiamo l'albero delle directory

```
# mkdir -p softtir_pfm_xill/opt/softtir/bin
```

Copiamo i file nelle locazioni di destinazione

```
# cp sever104 /softtir_pfm_xill/opt/softtir/bin
```

```
# cp xill /softtir_pfm_xill/opt/softtir/bin
```

```
# cp libIrsock.so.1 /softtir_pfm_xill/opt/softtir/lib
```

...

Nella Appendice A è riportata la lista dei file del modulo `softtir_pfm_xill.xzm`.

I moduli `porteus_xill_config.xzm` e `porteus_pfm_config.xzm`, sono creati in modo analogo:

```
# mkdir porteus_xill_config
```

```
# mkdir -p porteus_xill_config/etc/rc.d
```

```
# cp hosts porteus_xill_config/etc
```

```
# cp rc.local porteus_xill_config/etc/rc.d
```

```
# cp rc.intel.conf porteus_xill_config/etc/rc.d
# mkdir -p porteus_xill_config/usr
# mkdir -p porteus_xill_config/lib/modules/2.6.38.8-porteus/misc
# cp fliusb.ko /porteus_xill_config/lib/modules/2.6.38.8-porteus/misc
...
```

I file che fanno parte dei moduli *porteus_xill_config.xzm* e *porteus_pfm_config.xzm* sono listati nella appendice B.

L'esecuzione dello script *dir2xzm* permette di generare un pacchetto *xzm* a partire dal contenuto della directory specificata in argomento.

```
# dir2xzm softir_pfm_xill /mnt/sda1/porteus/optional/softir_pfm_xill.xzm
# dir2xzm /porteus_xill_config /mnt/sda1/porteus/optional/porteus_xill_config.xzm
# dir2xzm /porteus_pfm_config /mnt/sda1/porteus/optional/porteus_pfm_config.xzm
```

dove */mnt/sda1* è, nel caso specifico, il punto di montaggio del disco compact flash.

Gli script *bash extract_local* e *compress_local* in */root/src* contengono le procedure di estrazione e compressione dei moduli descritte più sopra.

5.2 Aggiornamento del contenuto di un modulo custom

È altresì possibile aggiornare solo un eseguibile o un file presente in uno dei moduli *custom*. La procedura è leggermente più complessa: dobbiamo prima estrarre il suo contenuto in una directory ausiliaria, apportare la modifica e ripetere la procedura illustrata al paragrafo precedente.

```
# mkdir softir_pfm_xill
# xzm2dir softir_pfm_xill.xzm softir_pfm_xill
# cp server104 softir_pfm_xill/opt/softir/bin
# dir2xzm softir_pfm_xill /mnt/sda1/porteus/optional/softir_pfm_xill.xzm
```

Per aggiornare ad esempio la configurazione di rete del PC embedded *xill*, procediamo come segue:

- creiamo la directory di appoggio


```
# mkdir porteus_xill_config
```
- estraiamo il contenuto del modulo


```
# xzm2dir porteus_xill_config.xzm porteus_xill_config
```
- relativamente a questa directory ci spostiamo in */etc/rc.d* ed editiamo il file di sistema *rc.inet1.conf* modificando gli indirizzi IP dell'host e del gateway con quelli forniti dall'amministratore della rete locale:


```
# cd porteus_xill_config/etc/rc.d
# vim rc.inet1.conf
```
- modifichiamo infine i file *hosts*, *HOSTNAME* e *resolv.conf* presenti in */etc* in modo da aggiornare il nome dell'host, il dominio di ricerca di default e il server DNS per la configurazione della rete locale in cui ci troviamo a operare


```
...
```

Alla fine viene rigenerato il modulo con il comando:

```
# dir2xzm /porteus_xill_config /mnt/sda1/porteus/optional/porteus_xill_config.xzm
```

A Appendice: Il modulo `softir_pfm_xill`

Lista dei file principali presenti nel modulo `softir_pfm_xill.xzm`.

```
drwxr-xr-x 5 root root 67 Jun 21 07:36 etc/
drwxr-xr-x 3 root root 19 Sep 21 2007 home/
drwxr-xr-x 3 root root 19 May 16 2011 opt/
drwxr-xr-x 4 root root 57 Oct 14 2011 root/
drwxr-xr-x 2 root root 22 Oct 13 2011 sbin/
drwxr-xr-x 3 root root 17 Oct 13 2011 usr/
drwxr-xr-x 4 root root 28 Oct 13 2011 var/

./etc:
drwxr-xr-x 2 root root 20 Oct 13 2011 cron.daily/
-rw-r--r-- 1 root root 61 May 16 2011 ld.so.conf
drwxr-xr-x 2 root root 22 Oct 14 2011 profile.d/
drwxr-xr-x 2 root root 26 Jun 20 14:40 rc.d/

./etc/cron.daily:
-rwxr-xr-x 1 root root 67 Oct 13 2011 timeset*

./etc/profile.d:
-rwxr-xr-x 1 root root 448 Oct 14 2011 softir.sh*

./etc/rc.d:
-rwxr-xr-x 1 root root 999 Oct 14 2011 rc.local.skel*

./home:
drwxr-xr-x 3 guest users 46 Jun 21 07:13 softir/

./home/softir:
drwxr-xr-x 6 guest users 56 Jun 20 14:42 src/

./home/softir/src:
drwxrwxrwx 5 root root 40 Jun 20 14:12 flicamd/
drwxr-xr-x 8 root root 89 Jun 21 07:24 giano/
drwxr-xr-x 3 root root 152 Oct 12 2011 include/

./home/softir/src/flicamd:
drwxrwxrwx 5 root root 42 Sep 13 2011 driver/
drwxrwxrwx 3 root root 142 Sep 13 2011 lodi/

./home/softir/src/flicamd/driver:
drwxrwxrwx 3 root root 4096 Jun 20 14:35 fliusb/
drwxrwxrwx 5 root root 4096 Jun 20 14:35 libfli/

./home/softir/src/flicamd/driver/fliusb:
-rwxrwxrwx 1 root root 434 Jul 5 2011 Makefile*
-rwxrwxrwx 1 root root 0 Sep 13 2011 Module.symvers*
-rwxrwxrwx 1 root root 3124 Jul 5 2011 README*
-rwxr-xr-x 1 root root 26358 Jun 20 14:14 fliusb.c*
-rwxrwxrwx 1 root root 3985 Jul 5 2011 fliusb.h*
-rwxrwxrwx 1 root root 2996 Jul 5 2011 fliusb_ioctl.h*
-rwxrwxrwx 1 root root 77 Jun 20 14:17 modules.order*

./home/softir/src/flicamd/driver/libfli:
-rwxrwxrwx 1 root root 1204 Jul 5 2011 Makefile*
-rwxrwxrwx 1 root root 32439 Jul 5 2011 docxx.sty*
drwxrwxrwx 3 root root 49 Sep 13 2011 flifilter/
-rwxrwxrwx 1 root root 20046 Jul 5 2011 libfli-camera-parport.c*
-rwxrwxrwx 1 root root 4533 Jul 5 2011 libfli-camera-parport.h*
```

```

-rwxrwxrwx 1 root root 45490 Jul 5 2011 libfli-camera-usb.c*
-rwxrwxrwx 1 root root 5720 Jul 5 2011 libfli-camera-usb.h*
-rwxrwxrwx 1 root root 20151 Jul 5 2011 libfli-camera.c*
-rwxrwxrwx 1 root root 3347 Jul 5 2011 libfli-camera.h*
-rwxrwxrwx 1 root root 2055 Jul 5 2011 libfli-debug.h*
-rwxrwxrwx 1 root root 24827 Jul 5 2011 libfli-filter-focuser.c*
-rwxrwxrwx 1 root root 2732 Jul 5 2011 libfli-filter-focuser.h*
-rwxrwxrwx 1 root root 7051 Jul 5 2011 libfli-libfli.h*
-rwxrwxrwx 1 root root 6441 Jul 5 2011 libfli-mem.c*
-rwxrwxrwx 1 root root 2260 Jul 5 2011 libfli-mem.h*
-rwxrwxrwx 1 root root 44160 Jul 5 2011 libfli.c*
-rwxrwxrwx 1 root root 9491 Jul 5 2011 libfli.h*
drwxrwxrwx 4 root root 4096 Sep 13 2011 unix/

./home/softtir/src/flicamd/driver/libfli/flifilter:
-rwxrwxrwx 1 root root 393 Jul 5 2011 Makefile*
-rwxrwxrwx 1 root root 5243 Jul 5 2011 flifilter.c*

./home/softtir/src/flicamd/driver/libfli/unix:
-rwxrwxrwx 1 root root 3272 Jul 5 2011 libfli-debug.c*
-rwxrwxrwx 1 root root 2168 Jul 5 2011 libfli-parport.h*
-rwxrwxrwx 1 root root 4976 Jul 5 2011 libfli-serial.c*
-rwxrwxrwx 1 root root 1995 Jul 5 2011 libfli-serial.h*
-rwxrwxrwx 1 root root 9827 Jul 5 2011 libfli-sys.c*
-rwxrwxrwx 1 root root 3060 Jul 5 2011 libfli-sys.h*
-rwxrwxrwx 1 root root 2751 Jul 5 2011 libfli-usb.c*
-rwxrwxrwx 1 root root 2838 Jul 5 2011 libfli-usb.h*
drwxrwxrwx 3 root root 101 Sep 13 2011 linux/

./home/softtir/src/flicamd/driver/libfli/unix/linux:
-rwxrwxrwx 1 root root 4415 Jul 5 2011 fli_ioctl.h*
-rwxrwxrwx 1 root root 2996 Jul 5 2011 fliusb_ioctl.h*
-rwxrwxrwx 1 root root 3390 Jul 5 2011 libfli-parport.c*
-rwxrwxrwx 1 root root 5863 Jul 5 2011 libfli-usb-sys.c*

./home/softtir/src/flicamd/lodi:
-rwxrwxrwx 1 root root 157 Sep 13 2011 Makefile*
-rwxrwxrwx 1 root root 206529 Sep 13 2011 ccd*
-rwxrwxrwx 1 root root 31413 Jun 15 2010 ccd.c*
-rwxrwxrwx 1 root root 26228 Sep 13 2011 ccd.o*
-rwxrwxrwx 1 root root 33132 Jun 15 2010 ccd_tng_original.c*
-rwxrwxrwx 1 root root 26228 Sep 13 2011 ccd_tng_original.o*
drwxrwxrwx 2 root root 6 Sep 13 2011 flicam/
-rwxrwxrwx 1 root root 16149 Sep 13 2011 flicam_tng.tar.gz*

./home/softtir/src/flicamd/lodi/flicam:

./home/softtir/src/giano:
drwxr-xr-x 4 root root 26 Jun 21 07:13 PI_Mercury/
drwxr-xr-x 3 root root 152 Jun 20 14:11 include/
drwxr-xr-x 4 root root 4096 Jun 20 14:34 server104/
drwxr-xr-x 3 root root 4096 Jun 21 07:24 thorlabs/
drwxr-xr-x 7 root root 4096 Jun 21 07:36 xill/

./home/softtir/src/giano/PI_Mercury:
drwxr-xr-x 3 root root 82 Jun 21 07:17 src/

./home/softtir/src/giano/PI_Mercury/src:
-rwxr-xr-x 1 root root 79 May 22 13:01 compile_freddy*
-rwxr-xr-x 1 root root 10998 Jun 21 07:17 freddy*
-rw-r--r-- 1 root root 13847 May 22 13:01 freddy.c
-rw-r--r-- 1 root root 2469 May 22 12:49 freddy.txt

```

```

./home/softir/src/giano/include:
-rw-r--r-- 1 root root 12494 May 22 07:44 commanddef.h
-rw-r--r-- 1 root root 9080 Mar 31 2011 gerrno.h
-rw-r--r-- 1 root root 1651 May 10 11:54 giano_struct.h
-rw-r--r-- 1 root root 7911 May 28 12:42 gianodef.h
-rw-r--r-- 1 root root 4341 May 22 07:44 info_struct.h
-rw-r--r-- 1 root root 2985 Mar 23 2011 lill_struct.h
-rw-r--r-- 1 root root 23466 Mar 11 2011 thorlabs_struct.h

./home/softir/src/giano/server104:
-rw-r--r-- 1 root root 1632 Mar 28 2011 Makefile.am
-rw-r--r-- 1 root root 159707 Apr 5 06:58 acq104.c
-rwxr-xr-x 1 root root 202 Jul 23 2009 autogen.sh*
-rw-r--r-- 1 root root 4218 Oct 20 2010 command.def
-rwxr-xr-x 1 root root 1011 Mar 28 2011 compile_server104*
-rw-r--r-- 1 root root 6953 Mar 28 2011 configure.in
-rw-r--r-- 1 root root 5966 Mar 28 2011 data104.c
-rw-r--r-- 1 root root 17106 Nov 21 2011 define.h
drwxr-xr-x 3 root root 112 Jun 20 14:34 docs/
-rw-r--r-- 1 root root 106997 Mar 5 10:23 exec104.c
-rw-r--r-- 1 root root 12787 Jan 18 2011 list104.c
-rwxr-xr-x 1 root root 242 Feb 14 2011 lookcvsh*
-rw-r--r-- 1 root root 18147 Mar 28 2011 lpt104.c
-rw-r--r-- 1 root root 6210 Sep 30 2010 lpt104.h
-rw-r--r-- 1 root root 87523 Apr 14 2011 mem104.c
-rwxr-xr-x 1 root root 7643 Feb 7 2011 mk_version.pl*
-rw-r--r-- 1 root root 1057 Nov 19 2009 multi.txt
-rw-r--r-- 1 root root 10695 Mar 28 2011 packetio.c
-rw-r--r-- 1 root root 94 Jan 14 2008 sequenza.txt
-rw-r--r-- 1 root root 18330 Feb 13 14:07 serial.c
-rw-r--r-- 1 root root 14920 Apr 5 2011 server104.c
-rw-r--r-- 1 root root 23818 Aug 31 2011 server104.h
-rw-r--r-- 1 root root 3715 Sep 30 2010 slist.h
-rw-r--r-- 1 root root 20724 Mar 31 2011 sock104.c
-rw-r--r-- 1 root root 24269 May 4 2011 socket.c
-rw-r--r-- 1 root root 4558 Mar 28 2011 socket.h
-rw-r--r-- 1 root root 27273 Oct 1 2010 spooler.c
-rw-r--r-- 1 root root 7636 Jan 18 2011 string104.c
-rw-r--r-- 1 root root 69370 May 9 2011 util104.c
-rwxr-xr-x 1 root root 587 Oct 22 2009 version.sh*
-rw-r--r-- 1 root root 43447 Nov 21 2011 win104.c

./home/softir/src/giano/server104/docs:
-rw-r--r-- 1 root root 54375 Oct 20 2010 Doxyfile.in
-rw-r--r-- 1 root root 572 Mar 27 2008 Makefile.am
-rw-r--r-- 1 root root 31545 Oct 21 2010 accuracy.pdf
-rwxr-xr-x 1 root root 143 Sep 27 2010 cvs_version.sh*
-rw-r--r-- 1 root root 4991 Oct 29 2010 frontpage.tex

./home/softir/src/giano/thorlabs:
-rw-r--r-- 1 root root 317 Jul 12 2011 Makefile.am
-rwxr-xr-x 1 root root 202 Sep 14 2010 autogen.sh*
-rw-r--r-- 1 root root 21966 Apr 22 2011 command.h
-rwxr-xr-x 1 root root 805 Jun 21 07:24 compile_thorlabs*
-rw-r--r-- 1 root root 3192 Sep 14 2010 configure.in
-rw-r--r-- 1 root root 28174 Feb 14 14:28 packetio.c
-rw-r--r-- 1 root root 21655 Feb 14 14:28 rotator.c
-rw-r--r-- 1 root root 1873 Jul 12 2011 rotator.h
-rw-r--r-- 1 root root 21340 Feb 14 14:28 thorlabs.c

./home/softir/src/giano/xill:

```

```

-rw-r--r-- 1 root root 24779 Jun 20 14:26 Makefile
-rw-r--r-- 1 root root 2640 May 10 12:27 Makefile.am
-rw-r--r-- 1 root root 25006 Jun 20 14:26 Makefile.in
-rw-r--r-- 1 root root 34611 Jun 20 14:25 aclocal.m4
-rwxr-xr-x 1 root root 202 Mar 11 2011 autogen.sh*
drwxr-xr-x 2 root root 81 Jun 20 14:25 autom4te.cache/
-rwxr-xr-x 1 root root 193 May 30 11:01 compile_fli*
-rwxr-xr-x 1 root root 218 May 30 11:02 compile_fli4arcetri*
-rwxr-xr-x 1 root root 794 Mar 28 2011 compile_xill*
-rw-r--r-- 1 root root 3389 Jun 20 14:26 config.h
-rw-r--r-- 1 root root 2958 Jun 20 14:25 config.h.in
-rw-r--r-- 1 root root 18565 Jun 20 14:26 config.log
-rwxr-xr-x 1 root root 34289 Jun 20 14:26 config.status*
-rwxr-xr-x 1 root root 171766 Jun 20 14:26 configure*
-rw-r--r-- 1 root root 5821 Apr 26 2011 configure.in
-rw-r--r-- 1 root root 7260 May 28 12:22 define.h
lrwxrwxrwx 1 root root 32 Jun 22 12:59 depcomp ->
/usr/share/automake-1.11/depcomp*
-rwxr-xr-x 1 root root 206754 Jun 20 14:24 fli_ccd*
-rw-r--r-- 1 root root 34765 May 30 11:33 fli_ccd.c
drwxr-xr-x 3 root root 110 Jun 20 14:11 flicam/
-rw-r--r-- 1 root root 51679 May 15 12:41 gmain.c
-rw-r--r-- 1 root root 6846 May 15 12:41 gmain.h
-rw-r--r-- 1 root root 1863 Jun 20 14:35 gversion.h
lrwxrwxrwx 1 root root 35 Jun 22 12:59 install-sh ->
/usr/share/automake-1.11/install-sh*
-rwxr-xr-x 1 root root 242 Mar 11 2011 lookcvcs*
-rw-r--r-- 1 root root 44382 May 17 10:13 mercury.c
-rw-r--r-- 1 root root 3970 May 14 11:52 mercury.h
lrwxrwxrwx 1 root root 32 Jun 22 12:59 missing ->
/usr/share/automake-1.11/missing*
-rwxr-xr-x 1 root root 6714 Mar 11 2011 mk_version.pl*
-rw-r--r-- 1 root root 14791 Jun 13 2011 packetio.c
-rw-r--r-- 1 root root 34021 May 28 12:22 powerboard.c
drwxr-xr-x 3 root root 126 Jun 21 07:36 serial/
-rw-r--r-- 1 root root 2315 Mar 11 2011 signal.c
-rw-r--r-- 1 root root 10125 May 26 2011 slist.c
-rw-r--r-- 1 root root 1923 Mar 11 2011 slist.h
-rw-r--r-- 1 root root 23 Jun 20 14:26 stamp-h1
-rw-r--r-- 1 root root 2488 May 24 10:34 startup_flicam
-rw-r--r-- 1 root root 6574 Mar 11 2011 string.c
-rw-r--r-- 1 root root 88692 Jun 4 08:27 thorlabs.c
-rw-r--r-- 1 root root 7050 May 17 10:13 thorlabs.h
-rw-r--r-- 1 root root 317 Mar 13 10:11 thorlabs.pos
-rwxr-xr-x 1 root root 587 Mar 11 2011 version.sh*
-rw-r--r-- 1 root root 27210 May 28 12:22 xill.c
-rw-r--r-- 1 root root 12193 May 28 12:22 xill.h
-rw-r--r-- 1 root root 14032 Nov 21 2011 xilllog.c
-rw-r--r-- 1 root root 59904 Jun 4 08:27 xillsock.c

./home/softir/src/giano/xill/flicam:
-rw-r--r-- 1 root root 310 May 30 09:31 README
-rw-r--r-- 1 root root 40648 Jun 1 06:51 flicam.cpp
-rw-r--r-- 1 root root 5827 Jun 1 06:51 flicam.h
-rw-r--r-- 1 root root 146082 Jun 1 06:51 flicam.ui
-rw-r--r-- 1 root root 13515 Jun 1 06:51 flicam.ui.h
-rw-r--r-- 1 root root 340 Jun 1 06:51 main.cpp

./home/softir/src/giano/xill/serial:
-rw-r--r-- 1 root root 241 Jun 23 2011 Makefile.am
-rwxr-xr-x 1 root root 202 Jun 23 2011 autogen.sh*
-rwxr-xr-x 1 root root 799 Jun 21 07:33 compile_pwb*

```



```

-rw-r--r-- 1 root root 3196 Jun 23 2011 configure.in
-rw-r--r-- 1 root root 23968 Jun 21 07:33 powerboard.c
-rw-r--r-- 1 root root 1598 Jun 23 2011 powerboard.h

./home/softir/src/include:
-rw-r--r-- 1 root root 10767 Jun 30 2011 commanddef.h
-rw-r--r-- 1 root root 9080 Mar 31 2011 gerrno.h
-rw-r--r-- 1 root root 1292 Mar 24 2011 giano_struct.h
-rw-r--r-- 1 root root 6980 Aug 31 2011 gianodef.h
-rw-r--r-- 1 root root 3245 Sep 29 2011 info_struct.h
-rw-r--r-- 1 root root 2985 Mar 23 2011 lill_struct.h
-rw-r--r-- 1 root root 23466 Mar 11 2011 thorlabs_struct.h

./opt:
total 0
drwxr-xr-x 6 root root 52 May 16 2011 softir/

./opt/softir:
total 0
drwxr-xr-x 2 root root 127 Jun 20 14:27 bin/
drwxr-xr-x 2 root root 21 May 16 2011 include/
drwxr-xr-x 2 root root 88 May 16 2011 lib/
drwxr-xr-x 4 root root 28 May 16 2011 share/

./opt/softir/bin:
-rwxr-xr-x 1 root root 206754 Jun 20 14:27 fli_ccd*
-rwxr-xr-x 1 root root 10998 Jun 21 07:19 freddy*
-rwxr-xr-x 1 root root 21488 Jun 21 07:35 powerboard*
-rwxr-xr-x 1 root root 38580 Jun 21 07:21 rotator*
-rwxr-xr-x 1 root root 384276 Jun 20 14:31 server104*
-rwxr-xr-x 1 root root 2488 Jun 20 14:27 startup_flicam*
-rwxr-xr-x 1 root root 38760 Jun 21 07:21 thorlabs*
-rwxr-xr-x 1 root root 216759 Jun 20 14:27 xill*

./opt/softir/include:
-rw-r--r-- 1 root root 2105 May 16 2011 Irsock.h

./opt/softir/lib:
lrwxrwxrwx 1 root root 16 Jun 22 12:59 libIrsock.so ->
libIrsock.so.1.0*
lrwxrwxrwx 1 root root 16 Jun 22 12:59 libIrsock.so.1 ->
libIrsock.so.1.0*
-rwxr-xr-x 1 root root 11823 Jun 20 14:41 libIrsock.so.1.0*
-rwxr-xr-x 1 root root 43425 May 16 2011 libxccapi.so*

./opt/softir/share:
drwxr-xr-x 2 root root 4096 May 16 2011 xeva/
drwxr-xr-x 2 root root 62 Jun 21 07:03 xill/

./opt/softir/share/xeva:
-rw-r--r-- 1 root root 794 May 16 2011 compile_xeva
-rw-r--r-- 1 root root 29172 May 16 2011 ntsc_8631.ttb
-rw-r--r-- 1 root root 29259 May 16 2011 pal_8632.ttb
-rw-r--r-- 1 root root 72025 May 16 2011 xccfgh5_2.ttb
-rw-r--r-- 1 root root 76934 May 16 2011 xccvgh19_1.ttb
-rw-r--r-- 1 root root 84163 May 16 2011 xccvgh6_1.ttb
-rw-r--r-- 1 root root 77308 May 16 2011 xevaxs_ntsc.ttb
-rw-r--r-- 1 root root 76568 May 16 2011 xevaxs_pal.ttb

./opt/softir/share/xill:
-rw-r--r-- 1 root root 4520 Jun 21 07:00 command.def
-rw-r--r-- 1 root root 794 May 16 2011 compile_xill

```

```

-rw-r--r-- 1 root root 317 Jun 21 07:03 thorlabs.pos

./root:
drwxr-xr-x 2 root root 132 Jun 21 07:15 src/

./root/src:
-rwxr-xr-x 1 root root 804 Jun 21 06:58 compress_local*
-rwxr-xr-x 1 root root 978 Oct 13 2011 curses.mnt.slax*
-rwxr-xr-x 1 root root 463 Oct 14 2011 extract_local*
-rwxr-xr-x 1 root root 827 Oct 13 2011 giano.mnt.porteus*
-rwxr-xr-x 1 root root 978 Oct 13 2011 giano.mnt.slax*
-rwxrwxrwx 1 root root 157 Oct 14 2011 rsync.baffa*

./sbin:
-r-s--x--x 1 root root 92944 Mar 22 2011 mount.nfs*

./usr:
drwxr-xr-x 2 root root 20 Oct 13 2011 sbin/

./usr/sbin:
-rwxr-xr-x 1 root root 75840 Oct 13 2011 ntpdate*

./var:
drwxr-xr-x 3 root root 19 May 16 2011 log/
drwxr-xr-x 3 root root 17 Oct 13 2011 spool/

./var/log:
drwxr-xr-x 2 root root 6 May 16 2011 softir/

./var/log/softir:

./var/spool:
drwxr-xr-x 3 root root 21 Oct 13 2011 cron/

./var/spool/cron:
drwxr-xr-x 2 root root 17 Oct 13 2011 crontabs/

./var/spool/cron/crontabs:
-rw----- 1 root root 1100 Oct 13 2011 root

```

B Appendice: il modulo `porteus_xill`

Lista dei file principali presenti nel modulo `porteus_xill.config.xzm`.

```
drwxr-xr-x 7 root root 4096 Jun 21 08:25 etc/
drwxr-xr-x 3 root root 20 Oct 11 2011 lib/
drwxr-xr-x 3 root root 18 Oct 11 2011 usr/

./etc:
-rw-r--r-- 1 root root 16 Jun 21 08:03 HOSTNAME
drwxr-xr-x 2 root root 20 Oct 14 2011 cron.daily/
-rw-r--r-- 1 root root 393 Sep 21 2007 fstab
-rw-r--r-- 1 root root 10 Apr 18 2007 hardwareclock
-rw-r--r-- 1 root root 666 Jun 21 08:25 hosts
-rw----- 1 root root 60 Sep 21 2007 ioctl.save
-rw-r--r-- 1 root root 1634 Oct 12 2011 issue
-rw-r--r-- 1 root root 23 Sep 14 2011 motd
-rw-r--r-- 1 root root 965 Apr 28 2011 passwd
-rw----- 1 root root 4096 Sep 21 2007 random-seed
drwxr-xr-x 2 root root 89 Jun 21 08:09 rc.d/
-rw-r--r-- 1 root root 41 Jun 21 08:09 resolv.conf
-rw----- 1 root shadow 510 Apr 28 2011 shadow
drwxr-xr-x 2 root root 148 Apr 13 2007 ssh/

./etc/cron.daily:
-rwxr-xr-x 1 root root 147 Oct 14 2011 savelog*

./etc/rc.d:
-rwxr-xr-x 1 root root 5774 Apr 13 2007 rc.inet1*
-rw-r--r-- 1 root root 3578 Jun 21 08:09 rc.inet1.conf
-rwxr-xr-x 1 root root 595 Oct 12 2011 rc.inet1.ori*
-rwxr-xr-x 1 root root 989 Jun 21 06:53 rc.local*
-rwxr-xr-x 1 root root 845 Apr 16 2007 rc.sshd*

./etc/ssh:
-rw----- 1 root root 668 Apr 13 2007 ssh_host_dsa_key
-rw-r--r-- 1 root root 599 Apr 13 2007 ssh_host_dsa_key.pub
-rw----- 1 root root 972 Apr 13 2007 ssh_host_key
-rw-r--r-- 1 root root 636 Apr 13 2007 ssh_host_key.pub
-rw----- 1 root root 1675 Apr 13 2007 ssh_host_rsa_key
-rw-r--r-- 1 root root 391 Apr 13 2007 ssh_host_rsa_key.pub

./lib:
drwxr-xr-x 3 root root 29 Oct 11 2011 modules/

./lib/modules:
drwxr-xr-x 3 root root 17 Oct 11 2011 2.6.38.8-porteus/

./lib/modules/2.6.38.8-porteus:
drwxr-xr-x 2 root root 22 Oct 11 2011 misc/

./lib/modules/2.6.38.8-porteus/misc:
-rw-r--r-- 1 root root 14486 Sep 13 2011 fliusb.ko

./usr:
drwxr-xr-x 4 root root 30 Oct 11 2011 local/

./usr/local:
drwxr-xr-x 2 root root 21 Oct 11 2011 include/
drwxr-xr-x 2 root root 21 Oct 11 2011 lib/
```

```
./usr/local/include:  
-rw-r--r-- 1 root root 9491 Sep 13 2011 libfli.h  
  
./usr/local/lib:  
-rw-r--r-- 1 root root 261990 Sep 13 2011 libfli.a
```

Riferimenti bibliografici

- [1] “Struttura delle directories per i progetti software del Laboratorio Infrarosso” C.Baffa, E.Giani Memo, Febbraio 2007
- [2] “Il programma server104 ed il sistema embedded”, C.Baffa, V.Biliotti, E.Giani, Arcetri Technical Report, 2-2010
- [3] “Nuova struttura del software di basso livello, C.Baffa, E.Giani, Memo, 1/2011.
- [4] “Protocollo per la comunicazione tra gbridge e Giano GUI”, C.Baffa, E.Giani, Arcetri Technical Report, 2-2012
- [5] “Fasti Global Controller, A.Checucci, C.Baffa, E.Giani, Arcetri Technical Report 3/2001.
- [6] “**Progetto Giano**. Gbridge/Balor: il software middle-ware di Giano”, E.Giani, C.Baffa, Arcetri Technical Report, 7-2010
- [7] Caratteristiche dello hardware di **pfm**:
http://www.tri-m.com/products/aaeon/files/manual/pfm620s_man.pdf
- [8] Caratteristiche dello hardware di **xill**,
<http://www.koala.it/it/micropc.htm>
- [9] Collezione dei manuali dei costituenti di Giano,
http://www.arcetri.astro.it/irlab/doc/giano/giano_manuals.tgz
- [10] Il filesystem Squashfs è documentato in:
<http://squashfs.sourceforge.net/>
- [11] La distribuzione Porteus
<http://www.porteus.org>
- [12] La distribuzione Slax, <http://www.slax.org/>

Indice

1	Hardware dei sistemi embedded	3
2	Il sistema operativo	3
3	Configurazione dei sistema embedded	4
3.1	Organizzazione del filesystem di Porteus	4
4	Il software del progetto Giano	5
4.1	Le librerie del progetto Giano	5
4.2	I programmi server e le <i>utilities</i> minori	7
4.2.1	L'applicazione <i>fli.ccd</i> e il driver linux <i>fliusb</i>	7
5	I moduli custom per la distribuzione Porteus	8
5.1	Creazione dei moduli custom	9
5.2	Aggiornamento del contenuto di un modulo custom	10
A	Appendice: Il modulo <i>softir_pfm_xill</i>	11
B	Appendice: il modulo <i>porteus_xill</i>	17