

Robotica di servizio su piattaforma Linux

relatore: Alessandro Budai

con la collaborazione dello

“SmartLab”

(<http://smartlab.univ.trieste.it>)

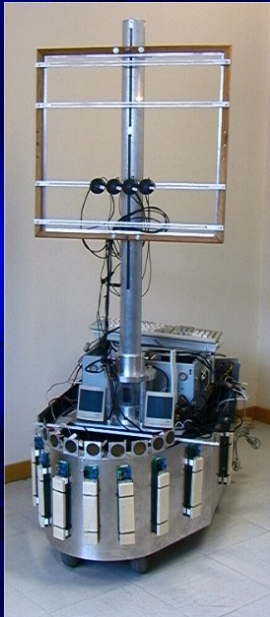
Robotica di servizio

- Utilizzo di robot in ambienti civili
- Cooperazione altri robot e con operatori umani
- Svolgimento di azioni finalizzate a scopi di servizio:
 1. Trasporto di cose o persone
 2. Sorveglianza
 3. Ausilio ai disabili

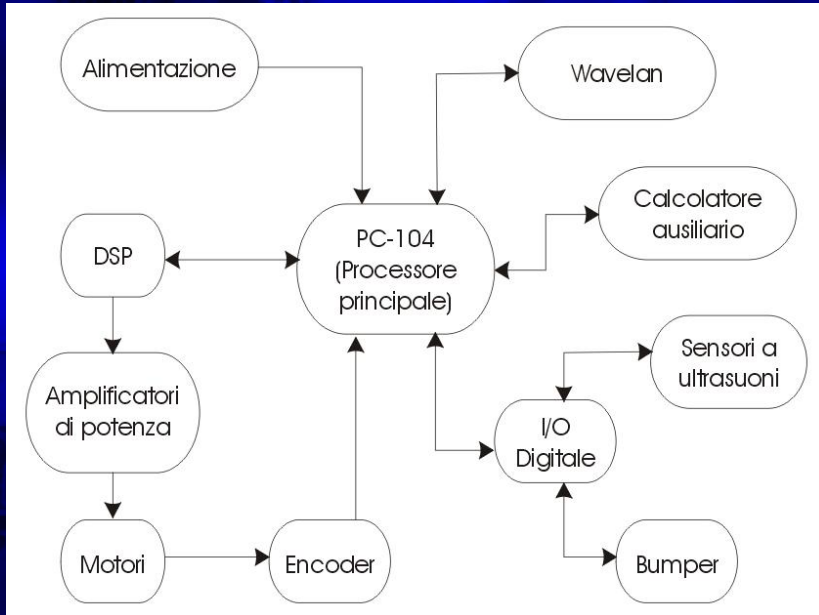
Il robot Snoopy

- Robot mobile autonomo
- Buona capacità di carico (100kg)
- Munito di sensori ultrasonici, telecamera, bumper, microfoni, altoparlanti
- Sistema di controllo basato su architettura x86
- Sistema distribuito basato su S.O. Linux

Snoopy dal vivo



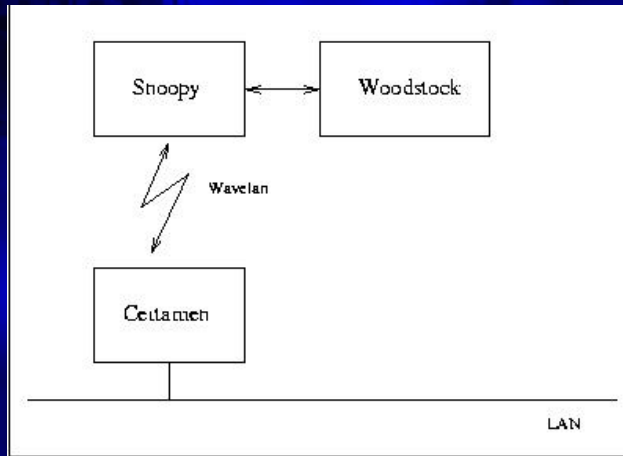
Architettura HW



Architettura Software

- Kernel per lo scheduling di task in realtime (tipo rate monotonic) Ethnos
- Pianificatore di percorso
- Generatore di traiettorie smussate (utilizza Xi model)
- Gestore sensori ultrasonici
- Gestore attuatori (motori)
- Gestore microfoni
- Demone trasmissione stream video (con compressione H.263)

Architettura distribuita



Il robot è costituito da due elementi Snoopy e Woodstock ed è connesso ad Internet tramite un gateway (Certamen). Il collegamento tra Snoopy e Certamen è realizzato tramite una rete privata wireless a 2MB/s

Configurazione del robot

- RedHat Linux 6.0 (Snoopy): kernel 2.2.6
 1. Driver aggiuntivi per gestione ultrasuoni, bumper, display LCD
 2. Configurazione rete wireless (WaveLan su PCMCIA) con routing verso rete ethernet
 3. Condivisione filesystem con NFS
- RedHat Linux 7.1 (Woodstock): kernel 2.2.19
 1. Driver aggiunto per scheda di acquisizione
 2. Software di localizzazione acustica in tempo reale
 3. Sistema di acquisizione video e streaming

Scheda acquisizione audio

- Scheda di acquisizione a 4 canali microfoniche sincronizzati
- Capacità di elaborazione a bordo (DSP TMS320C50)
- Memoria condivisa con il PC ospitante
- Realizzazione di un driver specifico per Linux

Descrizione del device driver

- Driver a carattere (major number 25)
- Inizializzazione del modulo (probe della scheda)
- Il driver carica il codice sul DSP
- Ad ogni interrupt dei convertitori il DSP trasferisce i dati nella memoria condivisa
- Tramite l'interrupt del PC vengono trasferiti i dati a blocchi ("burst transfer")
- Il software legge dal corrispondente device (/dev/smart4[abcd].cod)
- Con una serie di ioctl si possono effettuare le operazioni di reset, inizializzazione, caricamento codice

Streaming video

- Telecamera con brandeggio (comandato via porta seriale)
- Frame grabber “commerciale”
- Utilizza Video4Linux (libreria+driver)
- Indipendenza dall'HW per la realizzazione del SW
- Demone di acquisizione e compressione dei dati, ed invio in rete (connessione TCP)

Display LCD

- Necessità di visualizzazione messaggi di vario tipo
- Interfacciato con porta parallela
- Modifiche al kernel (modifica printk)
- Driver per la gestione in spazio utente

Ma ci sono soluzioni migliori a questo problema. . .

LinuxBIOS: overview

- LinuxBIOS rimpiazza completamente il BIOS
- La fase di inizializzazione del BIOS viene eliminata e sostituita con una molto più semplice seguita dalle decompressione ed esecuzione di un kernel Linux
- Tende a superare le limitazioni dei BIOS tradizionali

LinuxBIOS: svantaggi del BIOS

- La fase di setup del BIOS è lunga
- Il BIOS contiene ancora supporto per il DOS (inutile!)
- Non sempre è in grado di gestire correttamente le periferiche non standard
- La gestione e la configurazione richiedono schermo e tastiera
- Il meccanismo di boot del sistema operativo non è flessibile

Tutto questo non è accettabile in sistemi embedded e cluster !

LinuxBIOS: architettura

Linux è in grado di gestire HW non completamente inizializzato, la fase di inizializzazione è relativamente semplice

- Il processore viene portato in modalità protetta (32 bit)
- Inizializzazione della RAM
- “Fixup“ della motherboard
- Decompressione del kernel e sua esecuzione

LinuxBIOS: dettagli

- Abilitazione della modalità protetta
- L'inizializzazione della RAM è la parte più complicata (dipende molto dal chipset)
- Abilitazione della cache (decompressione del kernel in tempi ragionevoli)
- Si rende necessario inizializzare il bus PCI (non gestibile dal kernel altrimenti)
- Piccole modifiche al kernel (per le periferiche non abilitate)

LinuxBIOS: conclusioni

- Struttura semplice (si fa il minimo necessario)
- Velocità di boot (3 secondi !)
- Configurato per la scheda madre (al momento della compilazione)
- Flessibilità: dopo il boot si può ad esempio
 1. Boot da rete (standard)
 2. Boot via ssh
 3. Nodo senza dischi (boot via rete, root filesystem in NFS)
 4. Boot su rete Myrinet
- Supporto limitato ad alcune schede madri

Maggiori informazioni su <http://www.acl.lanl.gov/linuxbios/>

Ulteriori funzionalità considerate

- Utilizzo di filesystem journaling sul robot (in particolare ReiserFS notail)
- Compilazione del kernel con estensioni RT (RTLinux,RTAI)
- Utilizzo di LinuxBIOS
- Porting dei driver sul kernel 2.4

Applicazioni pratiche realizzate

- Sorveglianza di ambienti
- Trasmissione video a distanza
- Localizzazione di una sorgente acustica (parlatore)
- Riconoscimento di parlatori autorizzati/non autorizzati
- Comando vocale del robot in presenza di disturbi ambientali

Conclusioni

Benefici di Linux

- Ampia disponibilità di documentazione
- Disponibilità dei sorgenti (modifiche al kernel !)
- Flessibilità e facilità di configurazione
- Disponibilità di numerose funzionalità
- Rapida eliminazione di banchi (modello di sviluppo OpenSource)

Controindicazioni

- in questo tipo di applicazione nessuna !