

Real Time Linux nell'Automazione ed in Robotica

Alessandro Macchelli
CASY-DEIS, LAR-DEIS
Università di Bologna
Assegnista di Ricerca

Claudio Melchiorri
CASY-DEIS, LAR-DEIS
Università di Bologna
Professore Ordinario

Sommario

Flessibilità, brevi tempi di sviluppo e ridotti investimenti sono probabilmente le più importanti caratteristiche richieste in un'applicazione industriale e, certamente, in un setup sperimentale per la robotica ed applicazioni di controllo in generale. La crescita in termini di performance di piattaforme hardware a basso costo basate su PC e la diffusione di pacchetti software distribuiti liberamente ha permesso di sviluppare questo tipo di sistemi tenendo in considerazione tutte queste richieste.

In questo lavoro verranno descritte due delle attività di ricerca nell'ambito dello sviluppo di software real time per l'automazione portate avanti presso il Laboratorio di Automazione e Robotica (L.A.R.) dell'Università di Bologna: il controllo real time di un manipolatore industriale, il Comau SMART 3-S, ed una applicazione di controllo assi per motori DC di piccolo taglio. In entrambi i casi, l'idea di base è la realizzazione di setup sperimentali altamente flessibili e configurabili che consentano di testare velocemente nuovi algoritmi di controllo ed utilizzare il sistema in diverse applicazioni.

1. Introduzione

La diffusione delle varianti real time di Linux, il popolare sistema operativo per desktop e server, potrebbe portare ad una svolta nel modo in cui le applicazioni real time vengono sviluppate. Infatti, questo tipo di sistema fornisce notevoli performance che, insieme alla disponibilità dei codici sorgenti, di potenti tool di sviluppo e di documentazione generalmente ben realizzata, possono essere il punto di partenza per la realizzazione di validi ambienti di sviluppo. Questi sistemi operativi sono distribuiti sotto la GNU Public Licence e, quindi, sono liberamente disponibili e configurabili per soddisfare esigenze specifiche. I progetti RT-Linux [1,13,10] ed RTAI-Linux [3,9] si sono evoluti all'interno di questo modello di sviluppo, entrambi con lo scopo di permettere l'implementazione di applicazioni *hard real time* in ambiente Linux.

In questo lavoro, verranno descritte due delle attività di ricerca portate avanti presso il Laboratorio di Automazione e Robotica (L.A.R.) dell'Università di Bologna [15] nel campo della robotica e dell'automazione industriale.

La prima, descritta nella Sez. 2, riguarda il controllo real time in ambiente Linux di un robot industriale (il Comau SMART 3-S) e la realizzazione di un setup sperimentale flessibile per l'utilizzo del manipolatore in diverse applicazioni e con diversi tool (sistemi di visione e pinza robotica), [4,7,8].

Riguardo la seconda attività, l'idea di base è mostrare come, attraverso una semplice applicazione di *motion control*, sia possibile passare agevolmente da Matlab/Simulink, dove gli schemi di controllo sono simulati prima utilizzando un modello matematico del plant e successivamente il sistema reale, ad una applicazione *hard real time* e *stand alone*, caratterizzata da un proprio insieme di comandi e da, eventualmente, una propria interfaccia grafica, [5]. Questa attività è discussa nella Sez. 3.

2. Un setup sperimentale per la robotica basato su real time Linux

2.1. Motivazioni

Grazie ad una continua ricerca nel campo della robotica e, più in generale, dell'automazione, sono attualmente disponibili macchine affidabili e relativamente a basso costo che possono sostituire o assistere l'operatore umano in attività ripetitive o pericolose. In robotica, tutto questo è vero in particolar modo per applicazioni in ambiente strutturato, ovvero quando lo spazio di lavoro in cui il robot si trova ad operare è noto con grande accuratezza e precisione. E' ovviamente di grande interesse anche lo sviluppo di applicazioni in ambiente non-strutturato o sconosciuto. In questo caso, l'algoritmo di controllo deve fornire una sorta di "comportamento umano" alla macchina: in altre parole, la macchina deve essere caratterizzata da una autonomia funzionale, ovvero deve essere in grado di modificare il proprio comportamento sulla base delle informazioni acquisite in tempo reale dall'ambiente.

Risulta di notevole interesse acquisire queste prestazioni ad esempio nelle applicazioni spaziali, dove sistemi robotici autonomi o semi-autonomi possono sostituire gli astronauti nelle attività di routine per liberarli dai compiti ripetitivi e diminuire i costi delle missioni.

In questa sezione, verrà descritto un sistema sperimentale basato su di un robot industriale a 6 gradi di libertà ed una pinza a 3 gradi di libertà espressamente progettata per una possibile applicazione all'interno del PaT, il "Payload Tutor" proposto dall'A.S.I. (Agenzia Spaziale Italiana), Fig. 1. Questo sistema consiste di un braccio robotico, di un sistema di visione e di una pinza robotica evoluta. Dal momento che la pinza deve interagire con oggetti irregolari ed in assenza di gravità, il sistema di visione fornisce tutte le informazioni necessarie per afferrare oggetti qualunque in modo ottimale.

2.2. Descrizione del setup sperimentale

Il braccio robotico è un robot Comau SMART 3-S, manipolatore industriale antropomorfo a 6 gradi di libertà con polso non sferico, equipaggiato con il controllore standard C3G-9000. Ogni giunto è attuato da un motore DC brushless la cui posizione angolare viene misurata tramite un resolver. Il controllore viene utilizzato unicamente come interfaccia tra resolver e driver sul robot ed un PC basato su RTAI-Linux. In ogni periodo di campionamento, il sistema di controllo real time in esecuzione sul PC acquisisce i dati dai resolver, calcola i nuovi ingressi di controllo per gli attuatori ed invia tali valori al C3G-9000.

Questa procedura è resa possibile dal fatto che il controllore C3G-9000 è *aperto*. Infatti, il suo bus interno (VME) è connesso al bus ISA-PCI tramite una coppia di schede Bit3, una all'interno del controllore e l'altra nel PC che esegue gli algoritmi di controllo. Lo scambio di dati tra PC e C3G-9000 avviene attraverso un'area di memoria condivisa all'interno del controllore e la sincronizzazione attraverso un segnale di interrupt generato dal controllore stesso. In questa configurazione, i loop di posizione e velocità normalmente gestiti dal C3G-9000 sono aperti e tutte le protezioni di sicurezza disabilitate.

Gli altri due elementi che costituiscono il setup sono il sistema di visione e la pinza robotica (gripper). Il sistema di visione riconosce la forma degli oggetti e consente di calcolare i punti di presa migliori (*target points*). L'oggetto è afferrato in questi punti e le forze di contatto

sono opportunamente controllate. Anche se generalmente tali punti sono calcolati in base ad una analisi cinematica del primo ordine e risultano indipendenti dalla forma dell'oggetto e dalle caratteristiche geometriche del gripper, in questo caso i target points sono calcolati tramite un'analisi cinematica del secondo ordine, che consente di tenere in considerazione la forma degli oggetti (ovvero la loro curvatura) e delle dita. La configurazione di presa è, quindi, più stabile, [11,12].

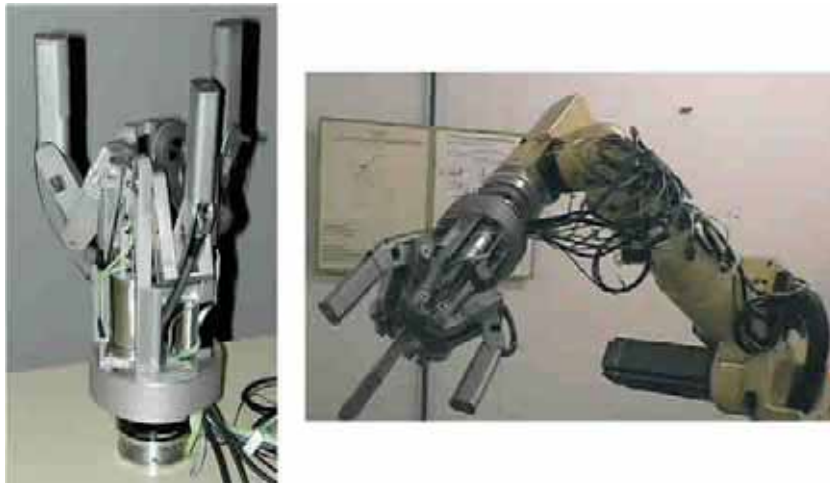


Fig. 1: La pinza robotica e il manipolatore Comau SMART 3-S

Il terzo elemento del setup è il Gripper A.S.I. [2], caratterizzato da 3 gradi di libertà ed adatto ad operazioni di manipolazione in assenza di gravità come, per esempio, le applicazioni spaziali. Gli algoritmi di controllo vengono eseguiti su una scheda DSP custom, basata sul chip TMS320C32. Per l'utilizzo in ambiente Linux di questa scheda, sono stati sviluppati un loader ed un DSP-monitor, insieme ad alcuni driver. Appena calcolata la distanza dell'oggetto ed i punti di contatto, il gripper viene posizionato in modo opportuno nello spazio di lavoro. Quindi, le dita vengono chiuse e l'oggetto afferrato nei punti pre-calcolati. A questo punto, la scheda DSP inizia ad eseguire gli algoritmi di controllo che forniscono le corrette forze di contatto.

2.3. Controllo del robot con RTAI-Linux

Il software sviluppato per il controllo real time del robot Comau SMART 3-S è diviso in due moduli distinti: un modulo real time, che esegue gli algoritmi di controllo e comunica direttamente con il plant (robot), ed un insieme di applicazioni non real time che sfruttano le funzionalità fornite dal primo. Chiaramente, si è reso necessario implementare una serie di meccanismi per consentire lo scambio di informazioni tra le due parti. Il modulo real time è periodicamente attivato da un segnale esterno di interrupt generato dal controllore C3G-9000. La variante real time di Linux adottata è RTAI, già utilizzata al L.A.R. con successo in altre applicazioni robotiche nel passato, [6].

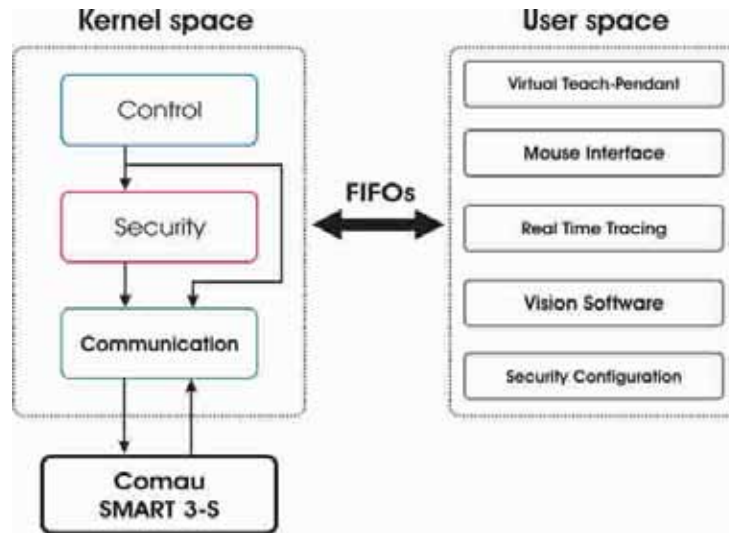


Fig. 2: Modulo real time ed applicazioni user space. Organizzazione e canali di comunicazione.

Per realizzare un software con struttura modulare e flessibile, che consentisse di testare rapidamente algoritmi di controllo ed utilizzare il robot in diverse applicazioni, il codice, la cui struttura è descritta in Fig. 2, è stato diviso in tre sotto-moduli principali che forniscono:

- la comunicazione con il robot SMART 3-S;
- i test di sicurezza;
- l'implementazione degli algoritmi di controllo.

Il *modulo di comunicazione* è utilizzato per leggere e scrivere dati sull'area di memoria condivisa presente all'interno del controllore C3G-9000. In particolare, questo modulo legge le sei posizioni angolari dei giunti e scrive i sei riferimenti di corrente per gli attuatori. Inoltre, implementa le funzioni di accensione e spegnimento. Questo è l'unico modulo che ha accesso all'area di memoria condivisa.

Il *modulo di sicurezza* implementa i fine corsa software, limita la velocità di giunto, controlla se uno dei giunti è bloccato e, se necessario, satura la corrente di riferimento al massimo valore ammissibile. In particolare, gli ultimi due test sono necessari per evitare di danneggiare gli attuatori.

Il *modulo di controllo* implementa gli algoritmi di controllo ed è responsabile della pianificazione della traiettoria sia nello spazio di lavoro che in quello di giunto. Il sotto-sistema che implementa gli algoritmi di regolazione può cambiare in relazione alle tecniche di controllo che si intendono utilizzare (per esempio, controllo decentralizzato o multi-variabile centralizzato). Dal momento che nella maggior parte delle applicazioni la traiettoria desiderata non è nota a priori, è stato implementato un generatore di traiettoria che utilizza un filtro non-lineare con vincoli sulla massima velocità e sulla massima accelerazione, [14].

Tutte queste funzioni sono compilate in un modulo che viene dinamicamente "*linkato*" al kernel real time del sistema operativo. Dal momento che l'utente deve essere in grado di interagire con il sistema, è necessario implementare una serie di canali di comunicazione tra *kernel space* e *user space*. Ogni modulo può scambiare dati con applicazioni nello spazio utente tramite il proprio canale di comunicazione. Siccome il quantitativo di informazioni che scambiamo con il modulo caricato nel kernel non è rilevante, è stato deciso di implementare tutti i canali di comunicazione utilizzando delle FIFO. Questa soluzione fornisce robustezza ed un meccanismo di coordinazione/sincronizzazione tra chi trasmette e chi riceve i dati.

Dallo spazio utente, è possibile trasmettere il comando di *drive ON/OFF* al modulo di comunicazione del modulo real time, ed è possibile modificare *run-time* alcuni parametri nel modulo di sicurezza, funzione da disabilitare nel momento in cui si testano nuovi algoritmi di

controllo. Per quanto riguarda il modulo di controllo, può ricevere comandi dalle applicazioni dello spazio utente e restituire informazioni sullo stato del robot (per esempio, velocità di giunto o riferimenti di corrente negli attuatori). In questo modo, è possibile muovere il robot con una tastiera, il mouse, il joystick o tramite un *Teach Pendant Virtuale*, oppure interfacciarlo con altre applicazioni per il tracing dei dati o per il movimento tramite un sistema di visione

2.4. Comau SMART 3-S con videocamera e gripper

Il sistema di visione è costituito da una videocamera monoculare montata sul polso del robot. Il software relativo viene eseguito nello spazio utente e comunica con il modulo real time attraverso una serie di comandi che consentono di fare eseguire al robot movimenti e compiti specifici. La Fig. 3 presenta uno *screenshot* del software di visione in cui **Robotic Vision** è la finestra principale. Inoltre, la finestra **TP** è il *Teach Pendant Virtuale* citato in precedenza, mentre **Pos:Giunti** è un altro strumento che consente di muovere il robot agendo su ogni giunto separatamente. La scena ripresa dalla videocamera è riprodotta nella finestra **COMAU's eye**.

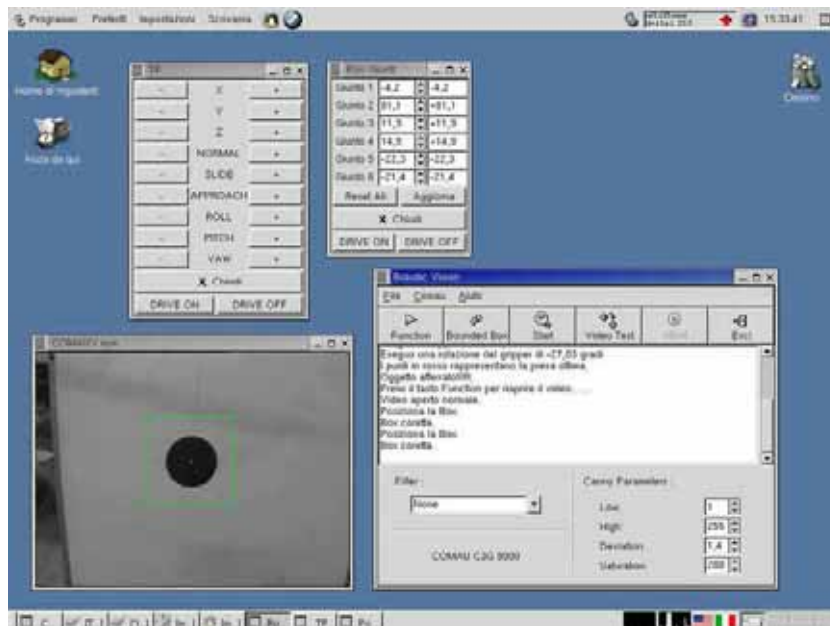


Fig. 3: Screenshot del software di visione..

Un tipico compito per questo sistema consiste nell'afferrare in modo automatico un oggetto selezionato dall'utente all'interno dello spazio inquadrato dalla videocamera tramite il gripper montato sul polso del robot. Inoltre, la presa deve essere *ottimale e stabile*. Questa procedura consiste di cinque passaggi intermedi:

1. L'utente muove il robot utilizzando tastiera, mouse o joystick fino a quando il sistema di visione riprende l'oggetto da afferrare.
2. Il sistema di visione automaticamente muove il robot per allineare l'*end-effector* (ovvero il gripper) con l'oggetto. Gli algoritmi di visione sono in grado di gestire anche oggetti in lento movimento: è il ritardo di elaborazione che riduce le performance in termini di *tracking*.
3. Dal momento che è necessaria una stima della distanza dell'oggetto ed il sistema di visione *non* è stereo, il robot viene mosso lungo la direzione oggetto-videocamera con lo scopo di acquisire due immagini dell'oggetto con cui calcolare la distanza incognita. I movimenti del robot lungo questa direzione sono scelti in modo tale da avere la *miglior* stima della distanza.

4. Utilizzando l'informazione sulla distanza dell'oggetto, il robot viene portato ad una distanza fissa dall'oggetto.
5. I punti di presa ottima vengono calcolati utilizzando le informazioni fornite dal software di visione. Per ulteriori dettagli al riguardo, si faccia riferimento a [7].

I movimenti del robot nei passi 2 e 4 sono gestiti dal software di visione che invia una serie di comandi di *roll-pitch-yaw* e *normal-slide-approach* al manipolatore. Lo scopo è quello di mantenere videocamera ed oggetto allineati.

3. Controllo assi con real time Linux

3.1. Motivazioni

Nello sviluppo di un algoritmo di controllo per un generico processo industriale, il primo passo consiste generalmente nel testare il controllore in un opportuno ambiente di simulazione: Matlab/Simulink è probabilmente il software più diffuso ed utilizzato. Una volta che i risultati simulativi siano soddisfacenti, si passa alla fase di sviluppo vero e proprio dell'applicazione, che verrà implementata su un'opportuna piattaforma hardware, in generale diversa da quella in cui la precedente fase di studio si è svolta.

Questa procedura può essere rivista se si decide di utilizzare una diversa combinazione di hardware e software e, in questa sezione, è fornito come esempio lo sviluppo di una semplice applicazione di controllo assi. L'idea consiste nell'utilizzare Matlab/Simulink su una architettura PC basata su real time Linux (RT-Linux) e fornita di una scheda I/O (prodotta dalla Quanser) capace di essere integrata facilmente all'interno dell'ambiente di simulazione. In questo modo, il personal computer usato nella fase di simulazione diventa anche il sistema che implementerà la versione definitiva del software o, almeno, un suo prototipo funzionante.

3.2. Descrizione dell'hardware

Il setup sperimentale è costituito da tre azionamenti di piccola potenza, ciascuno dei quali costituito da:

- un servomotore MiniMotor BLD 5606;
- un motore brushless MiniMotor 2444

entrambi realizzati dalla Faulhaber. Il loop di corrente è realizzato in hardware nell'azionamento (servomotore): il sistema, quindi, riceve in ingresso un set point di corrente e fornisce in uscita una coppia all'albero motore. Il software di controllo viene eseguito e sviluppato su un PC IBM compatibile che lavora ad una frequenza di 450MHz, mentre l'interfaccia verso l'azionamento è costituita da una scheda MultiQ-PCI di I/O analogico/digitale, non provvista di DSP, prodotta dalla Quanser. Tale scheda dispone di:

- 16 ingressi analogici, 8 dei quali differenziali ed i restanti configurabili come single-ended o differenziali;
- 4 uscite analogiche;
- 6 ingressi per encoder;
- 48 canali digitali di I/O.

Per ogni motore viene utilizzata una uscita analogica per il dispositivo di potenza ed un ingresso encoder. Il sistema operativo sul quale è stato sviluppato tutto il progetto è Linux nella distribuzione 6.2 della Red Hat con il Kernel 2.2.14 + patch RTL 2.3. Oltre alla parte hardware, la Quanser fornisce anche librerie per poter direttamente interfacciare la scheda con l'ambiente Simulink.

3.3 Percorso di sviluppo

Il lavoro svolto può essere suddiviso in due fasi fondamentali: nella prima si è proceduto

alla realizzazione e alla verifica sperimentale dell'algoritmo di controllo, mentre nella seconda è stata realizzata una semplice applicazione *stand alone* per la gestione del sistema.

Nella prima parte del lavoro, si è operato all'interno dell'ambiente Matlab/Simulink. Il pacchetto fornito dalla Quanser mette a disposizione un set di blocchi Simulink con cui è possibile interfacciarsi con la scheda di I/O e, quindi, con il sistema reale. Una volta identificato il modello di motore+azionamento, si è proceduto alla realizzazione ed alla simulazione del controllore: vengono creati i due anelli di controllo di velocità e posizione che saranno eseguiti a frequenze di campionamento diverse. Una volta raggiunte prestazioni soddisfacenti in simulazione, il controllore viene testato sul sistema reale, senza abbandonare l'ambiente di simulazione. Attraverso il *real time Workshop* (RTW) di Matlab, infatti, è possibile generare del codice C (*hard*) *real time* a partire dallo schema di simulazione che, una volta compilato, consente di ottenere un modulo che può essere mandato in esecuzione se inserito *run-time* nel kernel real time di Linux. In questo modo, lo schema Simulink non viene ad interfacciarsi con il modello matematico del plant, ma con il sistema reale.

La possibilità di operare all'interno di un completo ambiente simulativo e con il supporto di un sistema operativo capace di buone prestazioni real time, consente di velocizzare lo sviluppo ed ottimizzare le prestazioni del controllore. Per esempio, risulta molto semplice studiare sperimentalmente quale possa essere la frequenza di campionamento ottimale da utilizzare: nel caso specifico, si sono effettuate una serie di prove a partire da 100Hz fino a 500Hz per l'anello di posizione, con l'anello di velocità operante a frequenza dieci volte superiore. Comparando prestazioni con carico computazionale richiesto, si sceglie la frequenza operativa che fornisce il miglior compromesso. In questo caso, si è deciso di operare a 500Hz per l'anello più lento e a 5kHz per quello più veloce. Per frequenze superiori, il sistema mostrava evidenti segni di rallentamento nell'esecuzione di applicativi non real time.

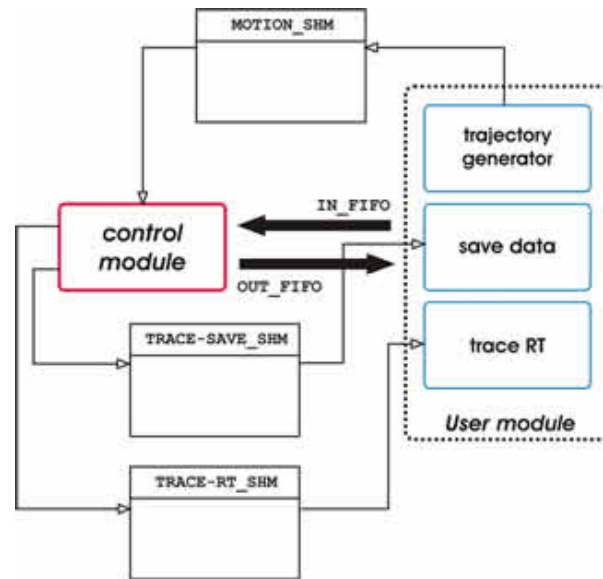


Fig. 4: Applicazione stand alone di motion control. Struttura generale..

Terminata la fase di sintesi e sperimentazione del controllo, si è passati alla fase di sviluppo di un semplice ambiente di controllo assi, la cui struttura è descritta dalla Fig. 4. In particolare, il controllore progettato nella fase precedente è stato inserito in un sistema di gestione assi sufficientemente ampio da consentire, oltre al controllo diretto dell'azionamento, anche una serie di servizi di supporto, come la generazione delle traiettorie o il *tracing* dei dati di interesse. Tale ambiente è strutturato in due moduli fondamentali:

- *modulo di controllo*, che si occupa della gestione dell'interazione con l'hardware;
- *modulo utente*, responsabile dell'interfaccia grafica e dell'interazione con l'operatore.

Il codice relativo al modulo di controllo è stato generato utilizzando *Real Time Workshop* direttamente dagli schemi Simulink. Per permettere l'interazione di questo modulo con altri processi operanti nel sistema, in particolare il modulo di comando, è stato necessario modificarne il codice generato automaticamente aggiungendo opportune funzionalità di scambio di dati (FIFO ed aree di memoria condivise) e di controllo dell'esecuzione. Per quanto riguarda il modulo utente, è stato sviluppato un ambiente grafico basato sulle librerie Gtk e dotato di una serie di funzionalità che permettono all'utente di gestire l'azionamento. Avviamento/arresto dei motori, generazione di vari tipi di traiettorie, modifica del legame di posizione esistente fra gli assi e visualizzazione grafica dei valori di interesse per il sistema sono le funzionalità previste.

3.4. Prestazioni ottenute

I processi real time di interesse nei controlli sono di natura periodica, la cui frequenza limite dipende dalla capacità computazionale del sistema che li esegue. Per ottenere un buon funzionamento dell'intero setup, in cui processi real time ed in spazio utente devono convivere, è necessario che le risorse vengano equamente divise tra le diverse tipologie di attività. Nel caso specifico, è evidente che tanto più è complesso l'algoritmo di controllo e tanto più è elevata la frequenza di campionamento, tanto meno risorse rimangono per le attività nello spazio utente.

Partendo da queste semplici considerazioni, sono state fatte delle prove per verificare i limiti del sistema. In particolare, utilizzando sempre lo stesso algoritmo di controllo sono state testate diverse frequenze di lavoro, a partire da 100Hz per l'anello di posizione e 1KHz per quello di velocità, fino ad arrivare a frequenze di 500Hz per il primo e 5KHz per il secondo. I risultati sperimentali hanno mostrato un comportamento facilmente prevedibile a priori: alle frequenze più basse il sistema ha un buon funzionamento per quanto riguarda la gestione della parte grafica, ma l'algoritmo di controllo non consente l'inseguimento di traiettorie in cui è richiesta una certa prontezza. Tale comportamento migliora, ovviamente, aumentando la frequenza operativa, con un corrispondente calo delle prestazioni nella gestione della parte grafica. La situazione 500Hz - 5kHz rappresenta la configurazione di miglior compromesso.

4. Conclusioni e lavoro futuro

In questo lavoro sono state descritte due delle attività di ricerca portate avanti al Laboratorio di Automazione e Robotica (L.A.R.) dell'Università di Bologna. Si è mostrato come sia possibile realizzare dei setup sperimentali altamente flessibili e configurabili per il controllo real time di un manipolatore industriale e di motori DC di piccola taglia utilizzando varianti real time del popolare sistema operativo per desktop e server Linux.

Per quanto riguarda l'applicazione robotica, il sistema consiste di un braccio robotico, il Comau SMART 3-S, un sistema monoculare di visione ed il Gripper A.S.I.. In questo setup, il gripper deve interagire con oggetti irregolari in moto in assenza di gravità e tutte le informazioni necessarie per afferrare gli oggetti sono fornite dal sistema di visione. Appena l'utente seleziona un oggetto all'interno del campo inquadrato dalla videocamera, il sistema inizia una procedura automatica che porta all'afferraggio ottimale dell'oggetto stesso.

L'applicazione di controllo assi è un semplice *benchmark* sviluppato per verificare le possibilità di Linux nelle sue varianti real time ed, in generale, del software Open Source, nell'ambito dell'automazione industriale. Si è mostrato come, con un supporto Hw/Sw opportuno, sia possibile passare dall'ambiente simulativo Matlab/Simulink, in cui gli schemi di controllo vengono simulati sul modello matematico del plant ad una applicazione real time *stand alone*, con una propria interfaccia grafica ed un set di comandi che consentano ad un operatore di configurarla opportunamente. Tutta la procedura può essere portata avanti su un semplice PC

dotato di opportune interfacce di I/O verso il sistema reale. La chiave risiede nell'utilizzo di un sistema basato su real time Linux, capace di prestazioni di ottimo livello nei processi di automazione non troppo spinti.

In entrambe queste applicazioni le prestazioni date da Linux real time sono state più che adeguate alle esigenze computazionali, ed anche i tempi di sviluppo e la robustezza del sistema si sono mostrati molto soddisfacenti. In conclusione si può senz'altro affermare che l'utilizzo di Linux e delle sue varianti real time sia di grande interesse per applicazioni, anche significative, di automazione industriale.

Bibliografia

1. M. Barabanov. *A Linux-based real time operating system*. Master's thesis, New Mexico Institute of Mining and Technology, Socorro, New Mexico, June 1997.
2. L. Biagiotti, C. Melchiorri, and G. Vassura. *Control of a robotic gripper for grasping objects in no-gravity conditions*. ICRA'01, IEEE Int. Conf. on Robotics and Automation, Seoul, Corea, 2000.
3. E. Bianchi, L. Dozio, G. L. Ghiringhelli, and P. Mantegazza. *Complex control systems, applications of DIAPM-RTAI at DIAPM*. In Proc. Realtime Linux Workshop, Vienna, 1999.
4. A. Macchelli and C. Melchiorri. *real time control system for industrial robots and control applications based on real time Linux*. In Proc. 15th IFAC World Congress, Barcelona, Spain, July 21-26 2002.
5. A. Macchelli and C. Melchiorri. *Controllo di azionamenti elettrici con real time linux: dalla simulazione all'implementazione*. In Proc. Motion Control 2003, 2003. In Italian.
6. A. Macchelli, C. Melchiorri, and D. Arduini. *real time Linux control of a haptic interface for visually impaired persons*. In Proc. IFAC Symposium on Robot Control, SYROCO'00, 2000.
7. A. Macchelli, C. Melchiorri, R. Carloni, and M. Guidetti. *Space robotics: an experimental setup based on RTAI-Linux*. In Proc. 4th real time Linux Workshop, Boston, USA, December 6-7 2002.
8. A. Macchelli, C. Melchiorri, and D. Pescoller. *An experimental setup for robotics and control systems research using real time Linux and Comau SMART 3-S robot*. In Proc. Real Time Linux Workshop 2001, Milan, Italy, November 26-29 2001.
9. RTAI-Linux web site. <http://www.rtai.org/>, 2003.
10. RTLinux web site. <http://www.rtlinux.org/>, 2003.
11. E. Rimon and J. W. Burdick. *Mobility of bodies in contact. I. A 2nd-order mobility index for multiple-finger grasps*. IEEE Transactions on Robotics and Automation, 14(5):696-708, Oct. 1998.
12. E. Rimon and J. W. Burdick. *Mobility of bodies in contact. II. How forces are generated by curvature effects*. IEEE Transactions on Robotics and Automation, 14(5):709-717, Oct. 1998.
13. V. Yodaiken. *The RTLinux Manifesto*. Department of Computer Science New Mexico Institute of Technology, Socorro, New Mexico, 1999.
14. R. Zanasi and R. Morselli. *Second order smooth trajectory generator with nonlinear constraints*. In Proc of European Control Conf. ECC'01, Oporto, Portugal, Sept. 2001.
15. Laboratorio di Automazione e Robotica - LAR, Università di Bologna, <http://www-lar.deis.unibo.it>