Prototipazione rapida di controllori real-time per robot industriali

Michele Bongiovanni, Basilio Bona

Questo articolo presenta un setup sperimentale per il controllo in tempo reale di un manipolatore industriale utilizzando Matlab/Simulink e il toolbox di generazione automatica di codice Real-Time Workshop. In questo lavoro si vuole sviluppare uno schema a blocchi in ambiente Simulink che gestisca tutte funzionalità di attivazione, pianificazione e sicurezza necessarie da tradurre in codice real-time per piattaforma Intel/VxWorks e che ne permetta il controllo da Pc remoto.

Le prestazioni sono il primo requisito che viene richiesto ai manipolatori e ciò fa sì che l'affidabilità del modello di riferimento sia la specifica principale nel progetto di controllori numerici. La ricerca di una descrizione fisica coerente con la realtà ha dato origine a diverse strategie di identificazione come punto di partenza su cui basare il progetto degli algoritmi di controllo ma in questo modo la dipendenza del modello di simulazione è sempre stata molto profonda.
L'obiettivo che ci si pone nella prototipazione rapida è abbandonare al più presto il modello matematico del robot per inserire direttamente nello schema di controllo il manipolatore: ciò permette al progettista di avere immediatamente a sua disposizione le risposte del sistema reale, regolando di conseguenza i parametri del controllore, e contemporaneamente di sviluppare l'architettura (interfacce comprese) dello schema controllo. Il Real Time Workshop supporta questo tipo di approccio e riduce la durata del ciclo di sviluppo: progetto, valutazione delle prestazioni, modifica dei parametri e infine revisione del progetto. La chiave di volta nella prototipazione rapida è la scrittura automatica di codice real time di basso livello a partire da modelli e strutture che al contrario descrivono il sistema ad alto livello, più quindi vicino agli schemi mentali del progettista. Dopo la creazione, le applicazioni real-time vengono compilate e messe in esecuzione sulla macchina target; grazie alla modalità "remoto" (*external-mode*) che permette di dotare il codice generato di funzioni per lo scambio di dati attraverso la rete con il modello di origine, possiamo agire direttamente sul modello Simulink per gestire il comportamento o modificare i parametri del sistema di controllo. A questo punto, il processo di prototipazione rapida è completo: il modello Simulink riveste contemporaneamente la funzione di interfaccia di controllo e di sistema controllato dove intervenire con soluzioni e strategie diverse. Dall'altro lato, sulla macchina target, abbiamo Wind River VxWorks, il sistema operativo real-time per cui il codice è stato generato e compilato. Si tratta di un ambiente ad alte prestazioni progettato per applicazioni real-time critiche (*hard* real-time) è ed supportato per piattaforme Intel x86 dal Real Time Workshop di Matlab. Di seguito verrà descritta, attraverso una sua applicazione, la metodologia da noi utilizzata per approfondire la conoscenza dei sistemi real-time e come abbiamo integrato il Real Time Workshop con le nostre conoscenze pregresse del sistema operativo VxWorks per ottimizzare la generazione di codice. Il passo successivo sarà acquisire e adattare l'architettura del codice generato ad ambienti di tipo open-source, e quindi ridurre fino a eliminare i costi legati agli strumenti di sviluppo per applicazioni industriali.

M. Bongiovanni, ricercatore in Ingegneria Informatica e dei Sistemi; B. Bona, professore ordinario di Automatica - Dipartimento di Automatica e Informatica Politecnico di Torino.

Architettura del sistema

Hardware. Il manipolatore utilizzato è un semplice robot planare a due gradi di libertà con giunti rotoidali azionati da motori *brushless*; è inoltre dotato di resolver calettati direttamente sul motore e infine possiede quattro interruttori magnetici sui fine corsa. L'apertura degli attuatori è completa e questo ci ha permesso di accedere a bassissimo livello sul controllo del robot escludendo il controllore installato inizialmente. Ciò è possibile attraverso la programmazione via cavo seriale dell'attuatore stesso, trasmettendo la modalità operativa desiderata. I segnali scambiati, digitali e analogici, vengono inviato alla scheda di acquisizione montata sul target attraverso un'interfaccia hardware appositamente progettata per portare i cavi dell'attuatore su una serie di terminali da laboratorio. La scheda di acquisizione è montata su bus Pci nel target e offre tutte le funzioni necessarie a cooperare con il robot: I/O di segnali digitali e analogici, contatori per i segnali encoder e in caso di necessità, generazione di interrupts. Il target è una macchina Intel PIII-Mmx collegata alla rete con una scheda ethernet 10/100 Mbps. L'host computer, il sistema di sviluppo che ospita Matlab e Simulink, è connesso in rete al target attraverso il protocollo Tcp/Ip.
Software. In un sistema basato sull'architettura host-target il software è diviso in due tipologie: real-time per il computer target e asincrono per il computer host. VxWorks, come anti-

cipato, è il sistema operativo real-time che sovrintende l'esecuzione di tutte le operazioni necessarie alla chiusura dell'anello di controllo e grazie ad un apposito *makefile* il Real Time Workshop genera automaticamente il codice adatto. Ogni volta che il computer target si avvia, scarica dalla rete l'immagine binaria del sistema operativo e lo mette in esecuzione sotto forma di processi kernel. Successivamente attende l'operatore per l'invio di ulteriori moduli e l'esecuzione di nuovi processi. Nelle nostre applicazioni, tra i diversi moduli

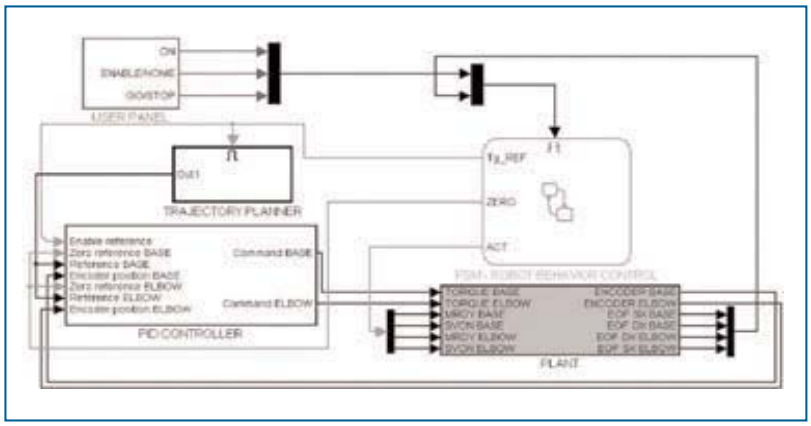

Figura 1 - I blocchi funzionali implementati nel modello Simulink del manipolatore

aggiunti al kernel di VxWorks, il più importante è proprio quello che si ottiene dalla compilazione del codice generato a partire dal modello Simulink. La messa in esecuzione di quest'ultimo pone il target sotto il controllo del computer host.
Simulink e Matlab vengono utilizzati sia per la costruzione del modello e la generazione di codice che per la gestione degli azionamenti e la modifica dei parametri durante la simulazione. Tornado, l'ambiente di sviluppo di VxWorks, permette invece la supervisione dei processi real-time, la configurazione del kernel e tutto ciò che è necessario al computer target per accogliere moduli derivanti da altre applicazioni. Inoltre offre diversi tipi di strumenti per una analisi approfondita delle attività del target tra cui WindView e Rti-Stethoscope. Il primo crea grafici temporali delle transizioni tra processi e degli eventi che si susseguono mentre il secondo acquisisce in real time qualsivoglia variabile o segnale per disegnarlo dinamicamente in un grafico. La connessione Tcp/Ip tra host e target ha quindi due funzioni principali: mantiene la comunicazione per l'*external mode*[1] e trasmette i dati acquisiti in real-time dal computer target verso il computer host. La modalità standard dell'*external mode* permette all'utente di disegnare le curve negli *Scopes* di Simulink ma non di utilizzare Rti-Stethoscope, strumento più potente soprattutto per l'analisi dei dati acquisiti. Il problema è legato alla gestione simultanea di più socket, troppo onerosa per il processo dedicato alle comunicazioni di rete. Per utilizzare Rti-Stethoscope in modalità *external* è stato quindi necessario separare il controllo dall'acquisizione, lasciando il compito di memorizzare i segnali alle Api di Rti inserite direttamente nell'immagine di VxWorks e richiamate nel codice dei moduli custom presenti nel modello. A tal proposito è interessante considerare un'altra caratteristica che rende Rti-Stethoscope la scelta migliore: esso si inserisce nel sistema real time con impatto quasi nullo essendo progettato per girare sotto forma di demone a bassa priorità.

Il modello Simulink

Il primo passo di questo lavoro è la costruzione del modello Simulink del manipolatore. La figura 1 mostra i blocchi funzionali implementati nel modello e segue una breve descrizione della funzione di ciascuno. Il sistema è diviso nelle sue parti funzionali: un'interfaccia per catturare i comandi dell'operatore e una macchina a stati che li gestisce, un interpolatore per la pianificazione delle traiettorie, il controllore vero e proprio che chiude l'anello e infine il blocco che rappresenta i dispositivi hardware per l'I/O dei segnali. L'interfaccia utente (user panel) riceve tre comandi dall'operatore (interpretati dalla macchina a stati come eventi): *on*, *enable/home* e *go/stop*. Trasmessi in sequenza, questi comandi attivano il modello, abilitano gli azionamenti e portano i giunti del manipolatore in posizione di home; infine il comando *go* attiva i canali analogici dove viene inviata sotto forma di tensione la coppia da esercitare su ciascun giunto e il robot è quindi guidato dal segnale di riferimento.
La macchina a stati (*robot behavioral control*) riceve anche la lettura dei fine-corsa oltre agli eventi utente per gestire situazioni di emergenza o per controllare la procedura di homing. Esistono tre tipi di stato: Setup, Homing e Moving. Gli stati Setup si hanno all'inizio (o alla fine visto che la macchina a stati è "circolare") di una movimentazione e mantengono fermo il manipolatore, azzerando anche tutti i flag che normalmente vengono utilizzati per registrare gli eventi.
Gli stati Homing controllano invece il posizionamento iniziale utilizzando i segnali di fine-corsa per determinare l'area di lavoro; gli stati Moving infine inviano i comandi agli attuatori per ottenere la movimentazione. Il Pid controller calcola il comando di catena chiusa a partire dalla posizione di home, dalla posizione corrente e dal riferimento da seguire. Questo blocco è abilitato dalla macchina a stati insieme al blocco *trajectory planner*, deputato a creare il riferimento stesso. Infine il blocco *plant* rappresenta il robot come viene visto dal punto di vista del modello Simulink e quindi anche dall'operatore. Si tratta infatti del gruppo di porte di I/O necessarie a controllare il robot e contemporaneamente ottenere la lettura dell'uscita prima di tutto. La scheda di acquisizione dati assume quindi la funzione di interfaccia robot-controllore e le Api necessarie al suo utilizzo sono state direttamente inserite nei blocchi di codice custom (vedi 3.1) e di qui nel codice creato dal Real Time Workshop. Questa è la strategia scelta per utilizzare la scheda vista la sua facilità d'impiego.

[1] L'*external mode* viene utilizzato per modificare i parametri del modello Simulink mentre il controllo è in esecuzione. Per il modello si tratta sempre di numeri che automaticamente vengono aggiornati all'interno del modulo in esecuzione sul target mentre per noi è utile distinguere i parametri a seconda del loro significato: possono essere comandi, interruttori o nuovi parametri di controllo.

Codice custom
Il Real Time Workshop fornisce numerosi blocchi per inserire righe di codice direttamente in quello generato (tra i numerosi file creati il principale è nome_modello_Simulink.c) e ciascuno di essi fa riferimento ad una particolare routine all'interno di esso: dichiarazione, esecuzione e terminazione. Esistono poi altri blocchi che si riferiscono invece agli *header* file del codice ed è possibile utilizzarli per inserire le proprie inclusioni di file. L'inclusione di questi blocchi custom nel nostro modello realizza tre componenti principali del sistema: il driver per la scheda di acquisizione dati, l'interfaccia tra la macchina a stati e la scheda stessa e il modulo di comunicazione seriale.

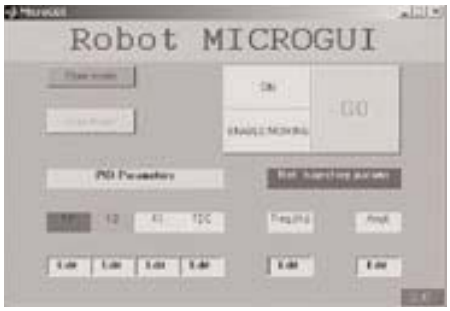

Figura 2 - Una semplice interfaccia sviluppata in Guide

Driver - La scheda di acquisizione utilizzata è una Sensoray S626 e possiede tutte le funzioni tipiche: input e output digitale e analogico, contatori e canale per gli interrupt. Il driver è codificato in un unico file C e incluso nel blocco *source file* con una semplice direttiva *#include*. In questo modo il generatore di codice porrà il file all'inizio di nome_modello_Simulink.c e saranno disponibili le chiamate alle funzioni per l'utilizzo della scheda durante l'esecuzione.
Modulo seriale - È utilizzato per la programmazione della modalità di lavoro degli attuatori durante l'avvio e anche in simulazione. Si possono scegliere tre modalità per il riferimento (coppia, velocità e posizione) e alcuni parametri collegati. Questo modulo è implementato nel blocco *model start function.*
Interfaccia - Questo blocco comunica con la macchina a stati per eseguire tutte le operazioni di I/O e per gestire gli eventi locali e esterni. Per ciascuno degli stati, attraverso di esso vengono attivati o disattivati i flag digitali degli attuatori, inviati i comandi sui canali analogici e acquisiti i fine corsa per completare la procedura di homing o garantire la sicurezza. Questo modulo viene pertanto utilizzato ogni passo di campionamento ed è realizzato quindi con il *blocco model output fuction.*

L'interfaccia grafica MicroGui
La modifica dei parametri in modalità *external* può talvolta risultare difficoltosa se questi parametri sono sparsi nel modello o nascosti all'interno di macro-blocchi (si pensi ad esempio al controllore, nel nostro caso un Pid, e ai suoi tipici parametri). La soluzione più efficace a questo problema viene offerta da Guide, l'ambiente di sviluppo di interfacce grafiche fornito con Matlab. Come mostrato in figura 2, è stata progettata una semplice interfaccia con pulsanti e campi testo appositamente collegati ai corrispettivi parametri all'interno del modello e quindi immediatamente disponibili per la modifica all'utente.

Il codice real-time

Dopo la preparazione del modello Simulink, il passo successivo è la generazione di codice e la compilazione per ottenere l'applicativo standalone. Questa procedura richiede la modifica di alcuni file di configurazione come descritto in [1] nella sezione dedicata ai target VxWorks. Il *template makefile* è l'oggetto più importante poiché definisce l'ambiente nel quale VxWorks verrà utilizzato e consente al Real Time Workshop di trovare i compilatori e le variabili di sistema necessarie. Occorre quindi configurare il generatore di codice: passo di campionamento, *external mode* e metodo di ottimizzazione del codice. Altre impostazioni riguardano invece VxWorks soltanto ma qui non verranno citate. Il codice generato è diviso in moduli ciascuno dei quali funzionante come uno strato (*layer*) operativo tra la struttura del modello e il sistema operativo specifico dove avviene la simulazione. Questi moduli sono codificati in diversi file come descritto di seguito (vedi figura 3).

VxWorks layer
Partendo dal livello più basso, abbiamo il modulo *Main Program* (rt_main.c). Qui si trova il codice per le modalità *Singletasking* e *Multitasking*; infatti i processi real-time sono costruiti a partire dal modello in numero uguale ai diversi valori di passi di campionamento che esistono al suo interno. Normalmente, con un unico valore, il modello è eseguito sotto forma di un unico processo, tSingleRate, temporizzato dal clock ausiliario di sistema: un periodo, un passo di simulazione. In teoria questa soluzione permette passi di campionamento fino a 250 µs, ma l'accesso alle periferiche rallenta il loop fino a 1 ms, comunque un buon compromesso. Con più passi di campionamento, il modello viene diviso in diversi processi e si passa in modalità *Multitasking*. Il generatore di codice crea un processo base ad alta priorità, TBaseRate, per le componenti più veloci del modello, e in sequenza altri processi con priorità via via inferiore man mano che il loro passo di campionamento è più ampio (*Rate monotonic scheduling policy*). L'esecuzione dei processi è gestita da semafori attraverso le funzioni semTake e semGive che mantengono la sincronizzazione necessaria tra essi servendosi degli interrupt del clock. Come processi di supporto rimangono da considerare i demoni di Rti-Stethoscope che acquisiscono i segnali in real-time e infine tExtern, il lato server della comunicazione tra Simulink e il codice real-time in esecuzione per l'aggiornamento dei parametri e la visualizzazione dello stato corrente (ad esempio le posizioni dei giunti) sull'interfaccia; questi processi hanno ovviamente una priorità molto bassa.

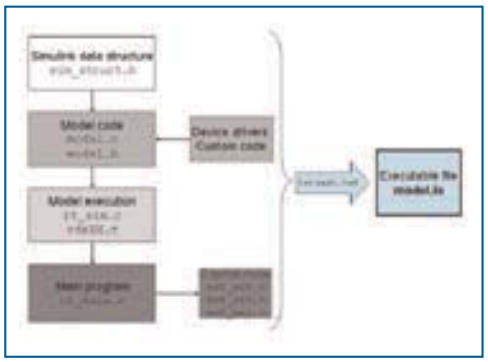
Figura 3 - La struttura dei diversi moduli sorgente

Common layer
Nel secondo strato troviamo istruzioni generali per l'esecuzione dell'applicativo. Il file rt_sim.c codifica le chiamate per l'ini-

zializzazione del modello, l'aggiornamento degli stati discreti, il calcolo degli output e infine per il passaggio al passo di campionamento successivo (vedi figura 4). Ad ogni passo il programma principale cede il controllo a questo modulo che indica a sua volta quali sono i blocchi del modello che devono essere processati. I blocchi continui vengono aggiornati ad ogni passo; i blocchi discreti invece vengono aggiornati solo quando il processo cui sono associati viene selezionato dallo *scheduler* per l'esecuzione secondo la loro priorità (vedi il paragrafo *VxWorks layer*).

Layer specifico

Il terzo strato contiene il codice generato direttamente dai blocchi del modello Simulink, compreso il codice custom tra essi inserito. Qui sono codificate tutte le routine che il modulo appena descritto (*Common layer*) deve chiamare durante la simulazione real-time: MdlInitialize, MdlOutput e MdlUpdate. La loro codifica si trova nei file nome_modello_Simulink.c e nome_modello_Simulink.h e negli stessi file viene inserito il codice C scritto per l'accesso alla scheda di acquisizione (vedi il paragrafo *Codice custom*). Una soluzione più rigorosa all'accesso a dispositivi hardware da parte di un modello Simulink è l'utilizzo di S-Function scritte in C; il risultato finale non è molto diverso dalla nostra implementazione ma è chiaro che inserire una S-Function permette una ottimizzazione del codice generato automaticamente. Tuttavia le nostre applicazioni non sono ancora così critiche dal punto di vista temporale e inoltre lo scopo del lavoro è stato proprio quello di mostrare come il sistema complessivo abbia avuto bisogno di un intervento relativamente contenuto da parte del programmatore. Ovviamente le S-Function rispondono al requisito di portabilità che nel nostro caso non è soddisfatto.

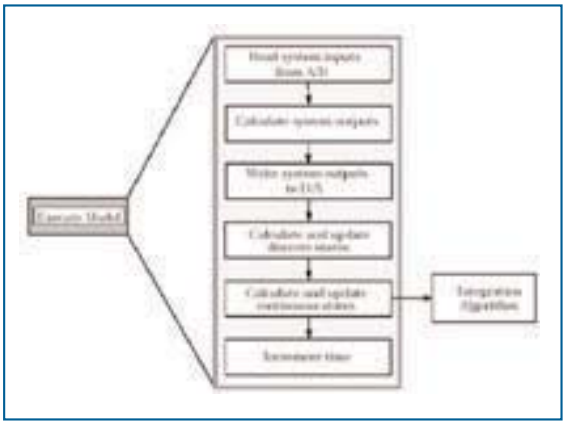

Figura 4 - Schema dell'aggiornamento degli stati discreti e del campionamento

Layer generico

L'ultimo strato definisce e crea SimStruct, la struttura dati di Simulink. A questo livello il codice è indipendente dall'applicazione e utilizza la struttura SimStruct per memorizzare tutte le informazioni relative al modello, compresi i parametri e le porte di ciascun blocco. Ogni istanza del modello crea la propria SimStruct attraverso la quale leggere e scrivere i dati. Tutte le funzioni nel codice generato sono pubbliche; per questa ragione può esistere un'unica istanza di un modello nel codice real-time e la funzione principale ad esso associata (che porta quindi lo stesso nome) viene chiamata in fase di inizializzazione per ottenere un puntatore alla SimStruct. Nella modalità *external,* il lato server della connessione accede attraverso quest'ultimo ai campi della SimStruct per aggiornare in real-time i valori con quelli provenienti dalla rete, ovvero dal modello Simulink.

Risultati

Il primo risultato importante che abbiamo ottenuto è la facilità con cui abbiamo realizzato uno strumento robusto per lavorare con il nostro manipolatore e per sperimentare diverse strategie di controllo. Ciò è utile per la ricerca ma anche per la didattica: studenti laureandi o utenti in generale non esperti possono concentrarsi sulle problematiche di controllo senza curarsi degli aspetti implementativi dell'architettura del sistema. Escluso il driver della scheda di acquisizione, non è stata necessaria una linea di codice e ogni tipo di modifica può essere fatta direttamente sul modello Simulink. Anche le procedure di avvio sono facilitate: il codice binario prodotto dalla compilazione (che richiede la semplice pressione di un pulsante) è pronto per essere scaricato e fatto girare sul computer target. Con semplici nozioni relative a Tornado e VxWorlks è inoltre possibile analizzare a fondo i processi e le loro interazioni. Con Rti-Stethoscope poi, le acquisizioni sono semplici e rapide: i demoni sono progettati per non perdere in prestazioni e ora che è possibile la cooperazione con la modalità *external* si possono vedere in tempo reale le risposte del sistema ai vari parametri che via via vengono inseriti. Pur essendo un setup sperimentale, il sistema generato può avere importanti utilizzi anche in campo teorico. Il più importante è nell'identificazione dei parametri delle strutture meccaniche attraverso l'acquisizione dei dati letti dal sistema reale. Lavori futuri puntano all'automatizzazione di queste procedure: muovi - acquisisci - stima, tutto all'interno dello stesso modello Simulink o per lo meno della stessa struttura software.

In questo modo l'*hardware-in-the-loop* offre uno strumento potente per la creazione di modelli affidabili. Ultime ma non meno importanti sono le possibilità di espandere questo ambiente: Simulink offre molti strumenti con in quali sviluppare nuove o migliori funzionalità. Particolare interesse è rivolto alla generazione di codice in quanto è una risorsa aperta (i file sono accessibili dopo la loro creazione) nonostante il software che li crea sia protetto da diritti. Ciò significa che possiamo trovare il modo di simulare quanto viene fatto dal Real Time Workshop per portare il codice su un ambiente libero, come in [3]. I sistemi operativi open-source possono garantire bassi costi, legati più che altro allo sviluppo, e proporre quindi soluzioni più economiche all'industria dei controlli automatici.

Riferimenti

[1] *Real-Time Workshop User's Guide.* The MathWorks, Inc., (September 2000).

[2] *VxWorks Reference Manual.* Wind River Systems, Inc., (May 1999).

[3] G. Quaranta, P. Mantegazza. "Using Matlab Simulink Rtw to build real time control applications". In: *III Linux RT workshop* (Nov 2001).