Università degli Studi di Siena

Prof. A. Bemporad, Ing. S. Di Cairano 26 Maggio 2005

Esercitazione di Tecnologia dei Sistemi di Controllo

Implementazione di controllori con `xPC target`

Questa esercitazione mostra in maniera pratica come implementare un controllore utilizzando lo `xPC Target` in ambiente `Matlab`.

Cosa è lo xPC target

Lo xPC target è un metodo di implementazione del controllo che sfrutta un calcolatore standard. Sul calcolatore viene eseguito un kernel Real-Time a singolo task: il task è il programma che implementa il controllore e che attraverso le system-calls si interfaccia con eventuali schede di acquisizione e dispositivi esterni. Il kernel ed il task risiedono in memoria RAM ed utilizzano I/O mappato in memoria.

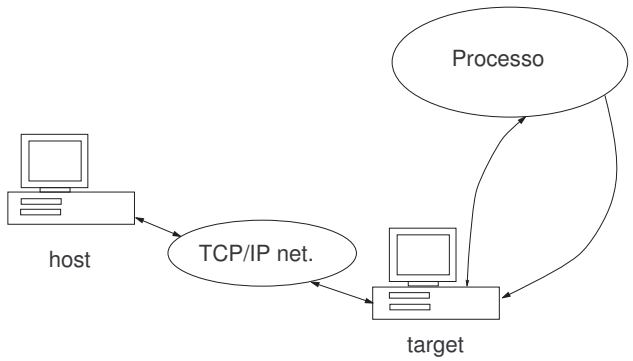

Figura 1: Architettura di un sistema basato su xPC target

Il sistema complessivo richiede due calcolatori connessi via cavo seriale o via rete TCP/IP: il *target* su cui eseguire il task di controllo e *lo host* utilizzato per definire il controllore e programmare il target.

Gli steps di implementazione del controllore sono i seguenti:

1. setup dell'architettura e della connessione host-target
2. progetto del controllore con `Simulink`
3. generazione automatica di codice `C` tramite `Real Time Workshop`
4. esecuzione del task real-time sul target
5. eventuale update online dei parametri del controllore sul target tramite lo host

Il maggior vantaggio di questo approccio è che il progetto viene eseguito in `Matlab` e l'implementazione è praticamente automatica dato che il codice `C` per architettura standard viene prodotto dal

`Real-Time Workshop` . Un vantaggio ulteriore è che i parametri fondamentali del controllore possono essere cambiati "on-the-loop" dal PC host, tramite apposite GUI o tramite Simulink, mantenendo su questo il controllo remoto.

Il caso di studio

Il caso di studio che prenderemo in considerazione è il controllo di posizione di un motore DC, lo stesso che avrete visto ad altri corsi quali *Progetto di Sistemi di Controllo*, e *Telelaboratorio di Automatica*.

Modello fisico

Il modello fisico di questo sistema è riportato in Figura 2: il circuito elettrico rappresenta il circuito di armatura del motore controllato in tensione mentre l'asse rappresenta il sottosistema meccanico corrispondente all'asse di trasmissione.

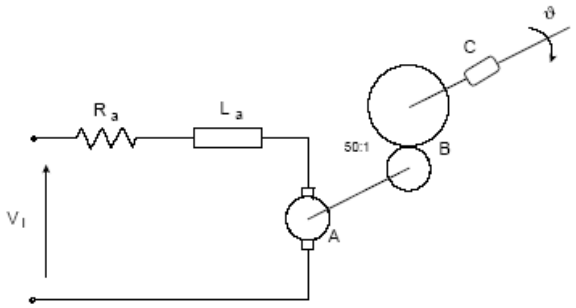

Figura 2: Modello del sistema di posizionamento

Le equazioni che descrivono il sistema sono le seguenti:

$$\begin{aligned} E &= V_I - L_a \frac{d\,i}{dt} - R_a\, i \\ E &= k \frac{d\,\theta}{dt} \\ \tau &= k\, i \\ j \frac{d^2\,\theta}{dt^2} &= r\, C - \beta \frac{d\theta}{dt} \end{aligned}$$

dove k è la costante del motore, E è la f.e.m, τ la coppia e r il rapporto di riduzione; β è l'attrito viscoso che agisce sull'asta e J è l'inerzia.

Questo modello è già stato tradotto (thanks Ing. M. Casini) nella relazione

$$\Theta(s) = \frac{K}{s(1+T_f s)(1+T_m s)} V(s)$$

in cui la funzione di trasferimento presenta due poli (uno meccanico ed uno elettrico) a sinistra dell'asse immaginario ed un polo nell'origine. I valori delle costanti sono riportate in Tabella 1.

Costante	Valore
T_f	0.0005
T_m	0.1
K	20

Tabella 1: Costanti della funzione di trasferimento

Analisi del sistema

Il sistema ad anello aperto presenta un polo in zero (sistema di tipo 1) e due poli reali e negativi. Il diagramma di Bode con evidenziato il margine di fase è riportato in Figura 3. Il sistema ad anello chiuso è stabile.

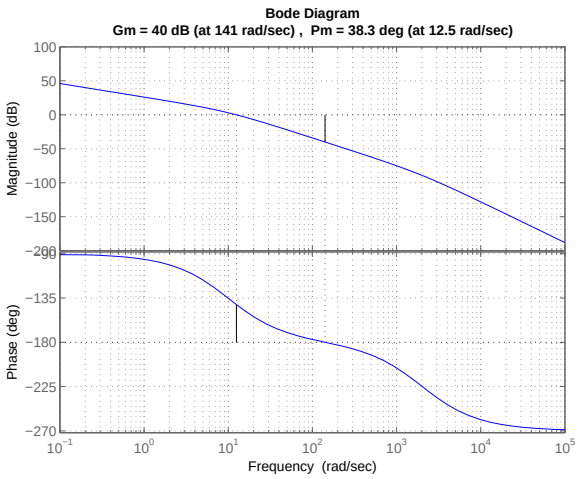

Figura 3: Diagramma di bode e margine di fase del sistema

Sistema di controllo

Vogliamo rispettare le seguenti specifiche: tempo di salita $t_s < 0.12$ secondi e sovraelongazione massima minore del 40%. Applicando le relazioni approssimate otteniamo le specifiche in frequenza sul margine di fase $\phi_m \geq 14$ e sulla frequenza di attraversamento $w_c \geq 30$ rad/sec.

Un controllore proporzionale sarebbe sufficiente ad ottenere il risultato desiderato ma per migliorare le prestazioni di inseguimento si decide di utilizzare un controllore PI con lo zero a frequenza molto inferiore di w_c.

La funzione che descrive il controllore è:

$$C(s) = K_P + \frac{K_I}{s}$$

in cui $K_P = 1.35$ e $K_I = 0.01$. Il digramma di Bode del controllore è riportato in Figura 4 ed il diagramma di bode del sistema controllato in Figura 5

Il progetto è terminato: adesso il controllore deve essere implementato tramite lo xPC target.

Setup dell'architettura xPC

La configurazione consiste principalmente nella configurazione della connessione fra host e target e nella generazione del kernel real time che la supporta. Il comando che permette di eseguire ciò è

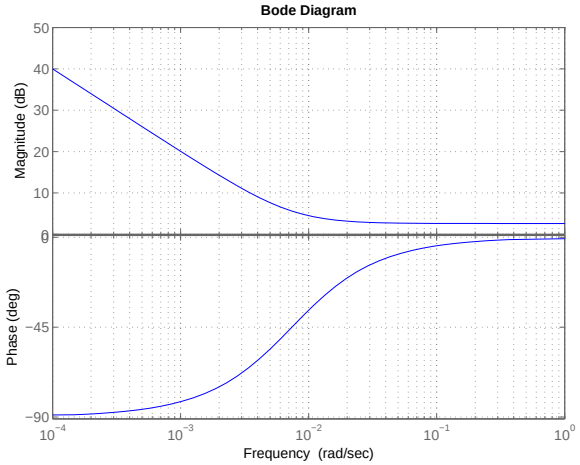

Figura 4: Diagramma di bode del controllore PI

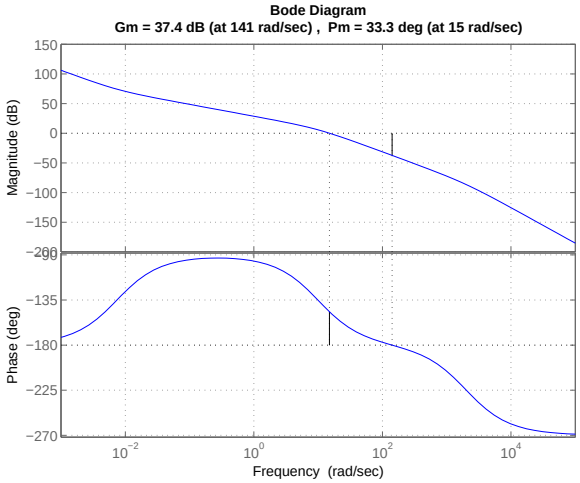

Figura 5: Diagramma di bode e margine di fase del sistema controllato

`xpcsetup`, che attiva la GUI di configurazione. Per verificare il funzionamento dell'architettura si può eseguire il comando `xpctest`.

Modello Simulink

L'applicazione di controllo deve essere codificata tramite un modello `Simulink`, perciò si riporta in un progetto `Simulink` il controllore PI che è stato progettato e si collegano gli ingressi e le uscite con gli appositi blocchi che rappresentano l'ingresso e l'uscita della scheda di acquisizione montata sullo host. Il modello `Simulink` è riportato in figura 6 in cui il blocco `controllore` definisce il controllore PI.

La scheda montata sul nostro target è una scheda National Instruments 6024E, alimentata con tensione fra -10 e 10 Volts.

Visto che vogliamo generare i riferimenti in gradi si aggiungono i blocchi che eliminano l'offset di tensione e definiscono la scala.

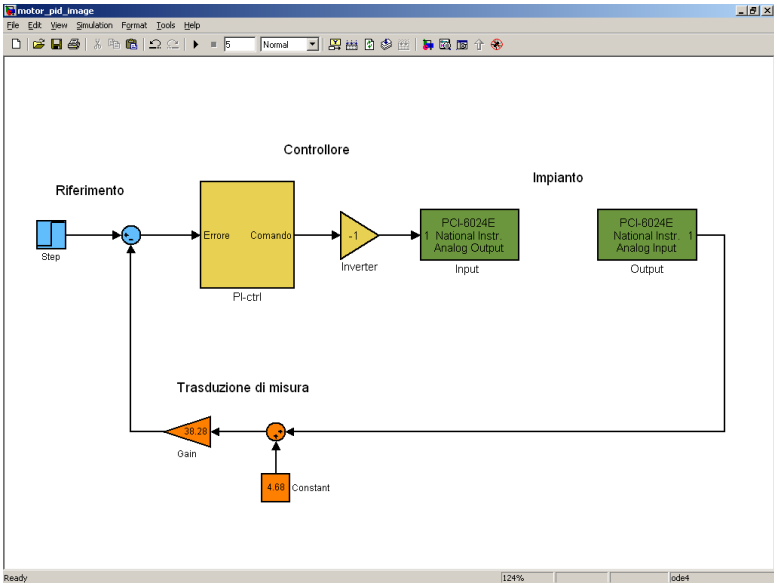

Figura 6: Modello Simulink del sistema

Compilazione dell'applicazione e Download

Il modello `Simulink` viene implementato tramite il `Real Time Workshop` che genera codice `C` eseguibile e lo compila. Il `Real Time Workshop` provvede anche all'inserimento nel codice dei driver per la scheda di acquisizione.

L'applicazione poi generata viene scaricata sul target e può essere eseguita attivandola e controllandola tramite lo host.

Esecuzione dell'applicazione

In seguito al download dell'applicazione viene creato un oggetto di classe `target`, di default con nome `tg`, che rappresenta la via di comunicazione con il target. Questo oggetto ha vari attributi e metodi: un elenco di comandi base è riportato in Tabella 2

Comando	Funzionamento
`+tg, start(tg)`	avvia l'applicazione sul target
`-tg, close(tg)`	ferma l'applicazione sul target
`tg.StopTime=<k>`	fissa la durata della simulazione a k secondi
`tg.SampleTime=<k>`	fissa il periodo di campionamento a k secondi

Tabella 2: Comandi di base dell'oggetto

Con questi semplici comandi si può eseguire l'applicazione e fermarla, ma un vantaggio dello xPC target è che si mantiene comunque il controllo remoto dell'applicazione attraverso lo host tramite un set di comandi avanzati.

Controllo remoto e monitoraggio

Quando l'applicazione è in esecuzione il target funge da controllore per il processo mentre lo host permette il monitoraggio dei segnali ed l'update di alcuni parametri.

Il monitoraggio è in tre forme: *signal monitoring*, *signal logging*, *signal tracing*.

Il signal monitoring consiste nell'acquisizione dei dati sui segnali senza riferimenti temporali (i.e. su richiesta).

Il signal logging consiste nell'acquisizione dei dati sui segnali riferiti al tempo per una *successiva* visualizzazione.

Il signal tracing consiste nell'acquisizione e visualizzazione online dei segnali.

L'update dei parametri consiste nel cambiare i valori delle costanti numeriche che definiscono il funzionamento dell'applicazione. Riferendosi al modello `Simulink` questo vuol dire poter cambiare qualsiasi costante numerica in qualsiasi blocco, senza però poter aggiungere nuovi blocchi o eliminare quelli presenti.

Vediamo ora i due principali ambienti in cui eseguire queste operazioni.

Controllo da Matlab

In `Matlab` il controllo dell'applicazione è mantenuto tramite l'oggetto target (N.B., di default ha nome `tg`). Cambiando le proprietà di questo oggetto si possono cambiare i parametri. Con il comando `tg.ShowParameters='on'` si rendono visibili gli indici dei segnali nell'oggetto `tg`. Con il comando `tg.setparam(<id>,<k>)` si ad esempio fissa il valore del parametro con indice `<id>` a `<k>`.

Il monitoring è ottenuto con il comando `tg.getsignal(<id>)` si legge il valore del segnale con indice `<id>`. Per rendere visibili gli indici dei segnali nell'oggetto `tg` utilizzare il comando `tg.ShowSignals='on'`.

Il logging è eseguito in maniera automatica: i valori dei segnali che finiscono in un blocco `out` nel modello simulink sono memorizzati. Per attivare il logging degli stati usare il menù di configurazione di `Simulink`. I valori vengono memorizzati nei vettori `tg.outputLog`, `tg.timeLog` e `tg.stateLog` (se la memorizzazione degli stati è attivata).

xPC scopes

Il tracing viene eseguito tramite gli *scopes*. Il comando `<nome>=tg.addscope(<tipo>,<id>)` crea un'oggetto `<nome>` di tipo scope. `<tipo>` può essere `'host'` oppure `'target'`: uno scope *host* è attivo sullo host mentre uno scope *target* è visibile sul monitor del target, se presente. Dopo aver creato lo scope è necessario attivarlo, se vogliamo che acquisisca i dati; il comando necessario è `start(<nome>)` oppure `+<nome>`, mentre l'acquisizione viene interrotta col comando `stop(<nome>)` oppure `-<nome>`.

xPC GUIs

In `Matlab` sono state introdotte varie GUIs per facilitare il controllo dello `xPC` dallo host. In Tabella 3 sono riportate le principali GUIs con il comando di attivazione e le funzioni.

Comando	Funzione
`xpcrctool`	controllo remoto dell'applicazione
`xpctargetspy`	visualizza lo schermo del target sullo host
`xpcscope`	controllo degli host scopes
`xpctgscope`	controllo dei target scopes

Tabella 3: GUIs per il controllo remoto

`xpcrctool` è una utility di controllo remoto che permette sia di avviare/fermare l'applicazione, sia di cambiare la durata di esecuzione ed il tempo di campionamento, sia di cambiare il valore dei parametri.

`xpctargetspy` permette di vedere sullo schermo dello host quello che si vede sullo schermo del target

`xpcscope` permette di aggiungere host scopes per il signal tracing e di monitorare sullo host i segnali di interesse.

`xpctgscope` permette di modificare i parametri dei target scopes.

Simulink external model

Un ulteriore modo per controllare l'applicazione sul target è quello di utilizzare `Simulink` in *external mode*. In questa modalità `Simulink` funziona come pannello di comando dell'applicazione in esecuzione sul target: cambiare il valore di un parametro nel modello comporta la variazione dello stesso parametro nel codice eseguito sul target. Ovviamente non è possibile cambiare la struttura del modello mentre si è in external mode.

Interfaccia web

L'ultima possibilità di controllo remoto dell'applicazione è rappresentata dall'interfaccia web. Per controllare l'applicazione via web è necessario chiudere il link fra `Matlab` ed il target, abilitare il browser (`xpcwwwenable`) e caricare la pagina web sull'indirizzo `http://<target address>:<target port>`. Indirizzo e porta possono essere ricavati dal menu di setup (N.B. la porta di default è 22222).

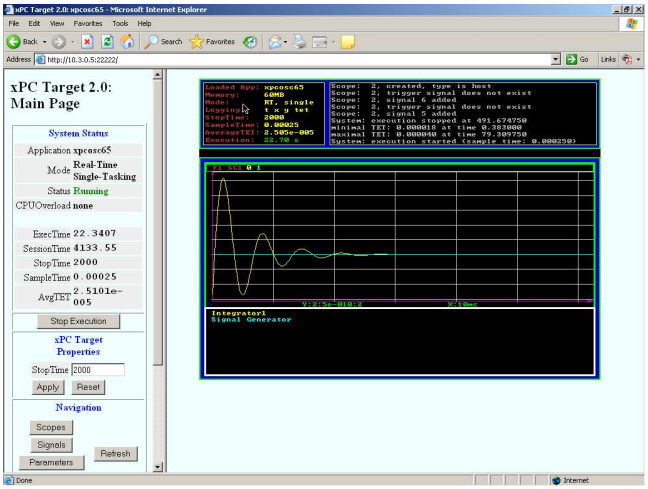

Figura 7: Interfaccia web di xPC-target

Conclusioni

In questa esercitazione avete visto i passi fondsamentali di sviluppo di un sistema di controllo con particolare enfasi sull' implementazione del controllore tramite l' architettura `xPC target`. In realtà xPC ha anche molte altre funzionalità. Per maggiori informazioni e progetti correlati contattate

Alberto Bemporad: `bemporad@dii.unisi.it`,
Stefano Di Cairano: `dicairano@dii.unisi.it`.