PC: Personal Computer o Process Control?

Micaela Caserza Magro, Valerio Malerba, Paolo Pinceti

Nelle architetture di automazione PC-based il fulcro dell'applicazione si sposta dall'hardware al software. Un PC con un sistema operativo general-purpose è in grado di emulare il funzionamento di un PLC utilizzando un programma SoftPLC. Tutti i sistemi operativi general-purpose sono multi-tasking, e questo non garantisce a priori il rispetto dei limiti temporali necessari per soddisfare i requisiti di un'applicazione di controllo di processo. La memoria presenta una metodologia di misura per valutare le prestazioni delle architetture PC-based in termini di tempo di esecuzione del ciclo e ripetitività del ciclo stesso.

L'introduzione degli Intelligent Field Device (IFD) ha modificato intimamente la struttura del sistema di automazione che è oggi costituita da una rete di IFD e da uno o più controllori, dispositivi dedicati per eseguire funzioni di controllo. La rete può essere integrata con dispositivi aggiuntivi per l'interfaccia uomo-macchina (HMI), comunicazioni remote, immagazzinamento dati, controllo avanzato e così via. I controllori tradizionali sono costruiti su supporti hardware dedicati con a bordo un sistema operativo real-time (RTOS) che permette di implementare ed eseguire le funzioni strettamente necessarie per il controllo di processo rispettando le deadline temporali imposte dal processo controllato. Un tipico esempio di controllore è rappresentato dal PLC (Programmable Logic Controller) che non solo è costruito su hardware dedicato, ma viene anche programmato per mezzo di un linguaggio standard appositamente sviluppato [1].

Speciali pacchetti software per piattaforme PC rendono possibile la realizzazione dei cosiddetti "soft-PLC" che offrono diversi vantaggi se comparati con i PLC tradizionali:

- nessun obbligo di avere un hardware dedicato;
- integrazione di funzioni diverse all'interno di una stessa macchina (ad esempio funzioni di HMI e di controllo possono coesistere e girare in un unico PC);
- facilità di interfacciamento e di integrazione delle funzioni di controllo con le funzioni di più alto livello;
- le comunicazioni remote tramite Ethernet o Internet sono native nella macchina.

Dall'altro lato, ed è lo scopo di questo lavoro, deve essere verificata l'idoneità delle prestazioni real-time di personal computer con sistemi operativi general purpose. Il problema principale nasce dal fatto che un sistema operativo multitasking general purpose, come possono essere Windows o Linux, è intrinsecamente non deterministico. Un altro limite del "softPLC" è legato alla possibilità di realizzare architetture ridondanti per applicazioni critiche.

M. Caserza Magro, V. Malerba, P. Pinceti - Dipartimento di Ingegneria Elettrica, Università di Genova

Il presente lavoro offre una panoramica generale sulle diverse architetture hardware e software dei sistemi di controllo di processo che si possono incontrare oggi sul mercato. Un'attenzione particolare viene poi rivolta alle nuove architetture PC-based ed in particolare alla misura delle prestazioni temporali più significative: tempo di esecuzione e ripetibilità del ciclo di controllo. La caratteristica più importante di un controllore non è infatti soltanto la sua capacità di eseguire un certo task in un tempo prefissato, ma anche la sua capacità di eseguire task ciclici con adeguata coerenza temporale. Il benchmark che viene presentato in questo lavoro prende in considerazione entrambi gli aspetti: il tempo di esecuzione di un task e la sua ripetitività per task ciclici.

Architetture dei sistemi di controllo

L'applicazione estensiva degli IFD ha portato alla nascita di una nuova architettura di automazione ibrida tra PLC e DCS, chiamata genericamente Process Control System (PCS). Come verrà nel seguito evidenziato, le prestazioni attuali dei PC per il controllo sono ancora lontane da quelle ottenute con PLC o DCS per applicazioni critiche, ma del tutto soddisfacenti per applicazioni non critiche.

Distributed Control System (DCS)

I sistemi DCS (Distributed Control System) sono nati nel mondo dell'automazione per processi continui, come il petrolchimico o la produzione di energia elettrica. Il task principale richiesto ad un DCS è quello di eseguire svariati loop di controllo su singole parti dell'impianto, o funzioni batch, integrate in un'unica funzione di supervisione. L'architettura di un DCS nella figura 1 è costituita da uno o più moduli di interfaccia verso il campo, costituito da strumenti ed attuatori, facenti capo ad un controllore che svolge le funzioni di regolazione specifiche per quella parte di impianto. I controllori sono collegati al livello superiore di supervisione, coordinamento, e configurazione per mezzo di una rete di comunicazione, solitamente proprietaria.

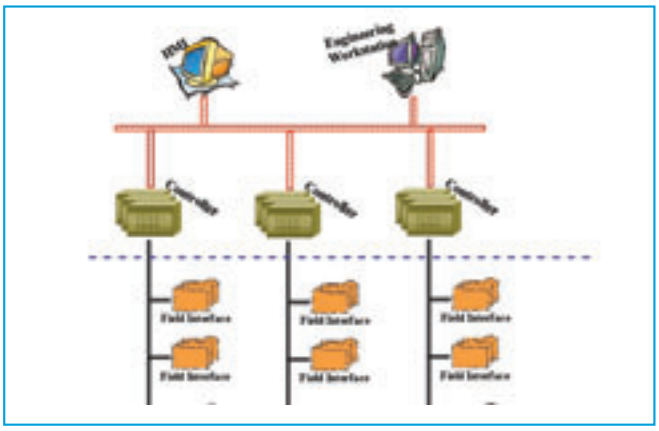

Figura 1 - Architettura DCS

L'intero sistema viene configurato attraverso una o più Engineering Workstation sulle quali gira il software per la programmazione, costituito da blocchi funzionali pre-definiti e da servizi di alto livello (gestione allarmi, template funzionali, storicizzazione ecc.). Altro punto distintivo di un DCS è rappresentato dal suo database real-time, che contiene gli indirizzi fisici dei dispositivi collegati al sistema ed i parametri di configurazioni assegnati. La gestione del database consente una comunicazione peer-to-peer verso i singoli controllori per poter scaricare la configurazione e la logica di controllo e per gestire i dati real-time per la supervisione o le altre funzioni di control room (trend, storici, advanced control, ecc.). Con l'impiego degli IFD su fieldbus l'architettura del DCS si è modificata come riportato nella figura 2.

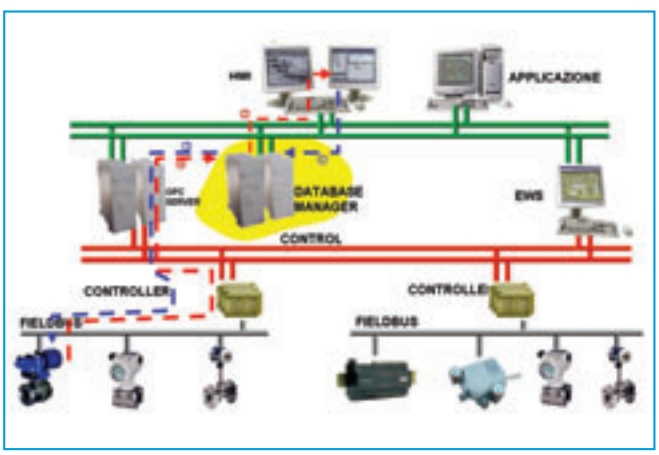

Figura 2: Architettura DCS moderno

Anche in queste nuove configurazioni il cuore del sistema resta il gestore del database, che svolge diversi compiti:
- definizione delle variabili (tag, tipo, tempo di refresh, etc.);
- update ciclico delle variabili (valori variabili di processo misurati);
- gestione delle chiamate acicliche (valori delle variabili aggiuntive rese disponibili dagli IFD);
- coordinamento con le applicazioni di più alto livello (HMI; interfacce esterne, controllo avanzato ecc.).

Solitamente i tempi di ciclo che riescono ad essere ottenuti e rispettati con un DCS sono nell'ordine delle centinaia di ms, senz'altro idonei per il controllo dei processi continui.

PLC-based

Per soddisfare le esigenze dei processi discreti (tipico il caso del manifatturiero) sono nati i Programmable Logic Controller (PLC). Il task primario per tali processi è l'esecuzione di sequenze di operazioni veloci e spesso cicliche, ed i segnali che si devono gestire sono in prevalenza di tipo discreto (on/off). I tempi di ciclo tipici che debbono essere soddisfatti dai PLC variano da qualche decina a pochi millisecondi. Ogni PLC deve essere configurato separatamente facendo riferimento a linguaggi standard (IEC 61131-3). È possibile realizzare un controllo distribuito anche con i PLC, realizzando una rete facente capo ad un sistema di HMI e supervisione, come rappresentato nella figura 3.

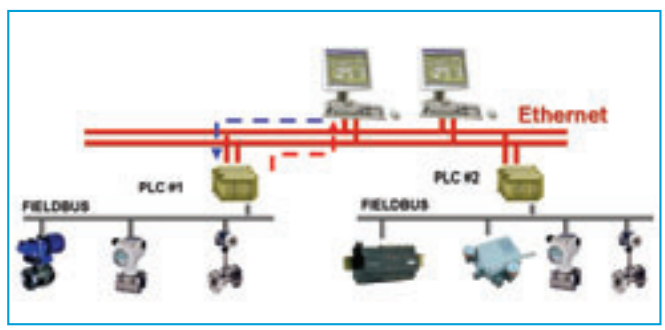

Figura 3 - Architettura PLC-based moderna

Process Control System (PCS): un'architettura ibrida

Come si vede confrontando le architetture nelle figura 2 e nella figura 3, la soluzione DCS e PLC in rete risultano molto simili dal punto di vista hardware, e vengono sempre più spesso chiamati PCS.

La differenza tra le architetture risiede in massima parte nella gestione del database real-time di sistema. In un'architettura PLC-based il database dell'impianto è distribuito; ogni singolo PLC contiene le informazioni e le variabili che si riferiscono unicamente agli strumenti ad esso collegati. Dall'altra parte un PCS derivante dall'evoluzione di un DCS ha un database real-time di sistema centralizzato [3] gestito da uno o più server ridondanti, come ben evidenzia lo schema nella figura 2. Un altro tratto distintivo dei moderni sistemi PCS è la loro interfaccia verso applicativi per la gestione di più alto livello (HMI, supervisione, manutenzione, gestione storici, etc.) e con il mondo ERP (Enterprise Resource Planning).

PC-based

È possibile l'impiego di PC general purporse a livello di controllo. Nella soluzione convenzionale, il controllo è realizzato impiegando dispositivi hardware dedicati con a bordo sistemi operativi Real-Time (RTOS) proprietari e sviluppati ad hoc, mentre i PC general-purpose vengono impiegati solo per le funzioni di alto livello. Oggigiorno è possibile integrare le funzioni di controllo all'interno dello stesso PC impiegato per le funzioni

di alto livello utilizzando un software che simuli il comportamento di un PLC, chiamato Soft-PLC. Nella figura 4 viene riportato uno schema di massima di un'architettura PC-based.

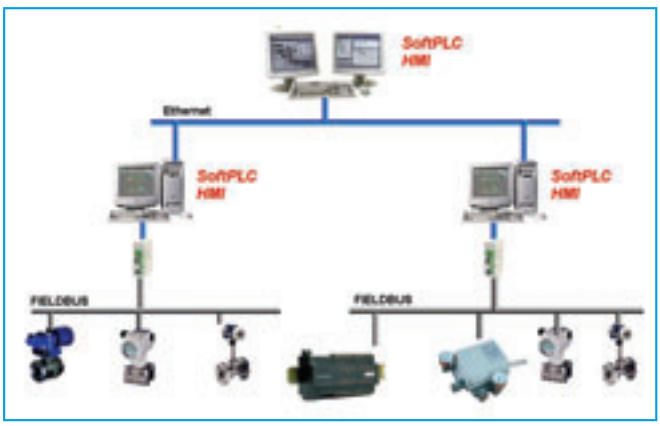

Figura 4 - Architettura PCS realizzata con l'utilizzo di SoftPLC

I vantaggi ottenuti una soluzione per il controllo PC-based possono così essere riassunti:
- impiego di una piattaforma standard di tipo general purpose;
- facilità di integrazione delle applicazioni;
- capacità di comunicazione nativa;
- semplice integrazione con periferiche di HMI.

Viceversa i principali svantaggi presentanti da questa soluzione sono:
- comportamento del sistema operativo a volte imprevedibile;
- un ambiente non deterministico di natura;
- difficoltà nella definizione di configurazioni ridondanti.

Nelle soluzioni PC-based il comportamento finale della macchina nasce dall'interazione dell'applicativo SoftPLC [4] con gli svariati componenti software che girano su un sistema operativo multi-tasking general purpose, come, ad esempio, Windows 2000 [5,6]. Il sistema operativo gestisce la contemporaneità di task concorrenti attraverso un sistema di gestione sequenziale (scheduler) dei tempi tra i diversi processi attivi [7]. Lo scheduler assegna ad ogni processo uno slot temporale in cui essere eseguito. Trascorso questo tempo, il processo viene interrotto e viene attivato il processo successivo identificato nello scheduler. La durata dello slot assegnato non è fissa, ma viene gestita a seconda del livello di priorità: più è alta e maggiore è lo slot temporale assegnato a quel processo. Se l'utente non interviene tutti i processi installati dall'utente hanno di default lo stesso livello di priorità (livello 8) che si traduce in un utilizzo uguale delle risorse computazionali della macchina.

Da questa prima e sommaria descrizione del funzionamento di un sistema operativo multi-tasking, emerge subito che il sistema non è intrinsecamente deterministico per le applicazioni che vi girano, in quanto il tempo di esecuzione di un applicativo dipende dagli altri applicativi attivi in quel momento. Ad un'analisi superficiale può sembrare che questo problema possa essere superato solo impiegando un RTOS [8,9,10]. Prima di proseguire occorre chiarire il concetto di applicazione real-time. Infatti real-time non è sinonimo di veloce, quanto piuttosto di deterministico o puntuale, cioè esistono dei limiti temporali entro cui un certo processo deve essere completato e questi limiti devono essere rispettati in tutte le condizioni di funzionamento. Le prestazioni real-time attese da un sistema possono essere suddivise in due tipologie:
- hard real-time: una specifica azione deve essere eseguita in un determinato lasso temporale e la perdita di questa scadenza comporta effetti disastrosi sulla prestazione del sistema;
- soft real-time: la perdita di una scadenza temporale da parte dell'applicativo si traduce in una prestazione degradata, ma comunque il sistema è sempre in grado di funzionare.

Per ottenere un comportamento hard real-time è oggi necessario utilizzare soluzioni hardware e software appositamente sviluppate: un sistema operativo real-time embedded in un hardware specifico, che è stato progettato e sviluppato per l'applicazione (per esempio sensori di macchine fotografiche, controllo di robot, etc.).

Per applicazioni meno critiche di tipo soft real-time è possibile adottare soluzioni diverse:
- utilizzare un PC general purpose che abbia un'estensione real-time per un sistema operativo multitasking. I tempi delle applicazioni real-time vengono gestiti dall'estensione real-time e sono completamente trasparenti al sistema operativo multitasking;
- utilizzare un PC general purpose con un sistema operativo multitasking e far girare le applicazioni real-time in questo ambiente, verificando che le prestazioni siano adeguate a quanto richiesto dall'applicazione stessa. Si accetta cioè che la risposta del PC sia controllata da un sistema operativo non deterministico, purché le prestazioni siano comunque sufficienti a garantire comportamenti efficaci dell'applicativo. Ciò significa che l'ambiente PC ha prestazioni così elevate che variazioni casuali del tempo di risposta rimangono all'interno dei limiti accettabili per l'applicazione in oggetto.

Per applicazioni di controllo di processo che ricadano nel campo del soft real-time e che non richiedano prestazioni temporali troppo stringenti è così possibile adottare una solu-

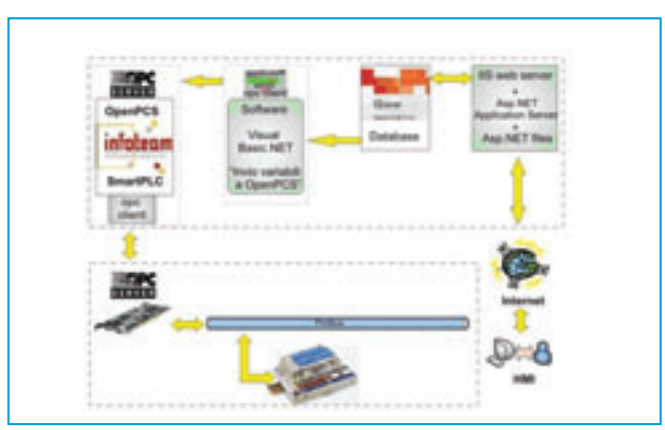

Figura 5 - Architettura software di una soluzione PC-based

zione PC- based senza estensioni real-time. Nel caso si scelga quest'ultima configurazione, il SoftPLC [11, 12] si comporta come un PLC virtuale, che deve interagire in modo fisico sia con il campo sia con l'ambiente PC su cui è installato.
Anche nelle architetture PC-based è necessario realizzare un'interfaccia fisica tra il campo ed il SoftPLC, cioè trasferire un dato trasmesso da un fieldbus ad un applicativo software su PC, e viceversa. La connessione è realizzata per mezzo di una scheda che comunica con i dispositivi su fieldbus e trasferisce i dati al PC utilizzando un software di interfaccia, solitamente un OPC server. OPC viene utilizzato anche come interfaccia tra il softPLC e le applicazioni di HMI o di più alto livello.
Come mostrato nella figura 5, un sistema PC-based può essere suddiviso nei seguenti sottosistemi:
- i moduli di I/O fisici su cui vengono cablati i segnali provenienti dal campo;
- il fieldbus per collegare il campo con la parte controllo;
- la scheda di interfaccia tra il fieldbus ed il PC su cui è installato il soft-PLC;
- OPC client/server per lo scambio dati tra la scheda di interfaccia ed il softPLC;
- il SoftPLC;
- OPC client/server per lo scambio dati tra il softPLC e gli applicativi di HMI;
- gli applicativi di HMI.

Misura delle prestazioni temporali di un soft-PLC

Per misurare le prestazioni di un componente software è necessario definire una metodologia specifica per il task richiesto. Esistono svariate tecniche adatte all'uso e queste spaziano dalle tecniche hardware-assisted a quelle completamente software [13].
Le tecniche hardware-assisted sono principalmente ottenute con dei Logic Analyzer e dei circuiti ICE (In-Circuit Emulators). Entrambe le tecniche monitorano il traffico sui bus interni del PC, e per questo si rivelano inutili per i PC general purpose, in quanto tutto lo scambio dati tra la CPU e la memoria cache non avviene attraverso un bus esterno e per questo non può essere rilevata dal circuito ICE.
I metodi software, invece, richiedono la presenza di un clock o un timer interno di riferimento per misurare il tempo trascorso tra due eventi ritenuti importanti. Anche se l'idea di base è piuttosto semplice, la sua implementazione non è del tutto immediata. Per questa ragione esistono diversi approcci per ottenere il risultato voluto. Le due principali filosofie seguite dai metodi software per la misura delle prestazioni sono:
- i tool di stack sampling;
- i metodi di data collection.

I tool di stack sampling lavorano facendo delle fotografie cicliche dello stack della CPU. A partire da questa informazione, l'utente può determinare dove il sistema sta spendendo il suo tempo, funzione per funzione. Questo approccio ha un grande svantaggio per quanto riguarda il possibile utilizzo nel nostro caso: non può determinare una funzione che abbia un tempo di ciclo più breve del tempo di acquisizione dello stack della CPU. Da questo si deduce che il tempo di ciclo del SoftPLC potrebbe non essere identificato dai risultati dell'analisi dello stack.
L'altro approccio, il metodo di data collection, coinvolge la manipolazione del codice da esaminare: istruzioni addizionali sono aggiunte in punti critici del codice, ad esempio quando sono richiamate le funzioni di start e stop. Durante l'esecuzione del programma, queste istruzioni addizionali inviano dati ad un data logger che memorizza questi dati insieme al loro time-stamp. Il lato negativo di questo approccio è che la risoluzione offerta dai timers è piuttosto bassa, nell'ordine dei millisecondi. Questo significa che l'incremento più piccolo che il benchmark è in grado di apprezzare è di un millisecondo. Una tale risoluzione risulta del tutto accettabile per gli scopi del presente lavoro. Infatti si è interessati a misurare le performance temporali di un'applicazione "lenta" come può essere il SoftPLC. Ovviamente il SoftPLC è un'applicazione lenta se riferita ai processi interni di una CPU. Inoltre la misura delle prestazioni temporali del SoftPLC non sono solo riferite alla sua velocità di esecuzione, ma anche e soprattutto alla ripetibilità del tempo di ciclo.
Per tutte queste ragioni, la metodologia di misura adottata è basata su un metodo di data collection.
Per valutare le prestazioni ottenute dalla soluzione PC-based è necessario definire anche un benchmark che deve includere:
- la definizione dell'ambiente PC che si impiega per far girare il software per il controllo;
- gli strumenti necessari per misurare il comportamento temporale del sistema sia in termini di tempo di risposta agli eventi per le funzioni basate su interrupt sia in termini di jitter per le funzioni cicliche.

I parametri temporali che entrano in gioco e che determinano la prestazione di un PLC e quindi anche un softPLC sono il tempo di ciclo (lettura degli ingressi, esecuzione della logica e scrittura delle uscite, si veda la figura 6) e la ripetitività dello stesso. Nel caso di un soft-PLC la misura delle prestazioni di traduce nel misurare delle prestazioni di un software.

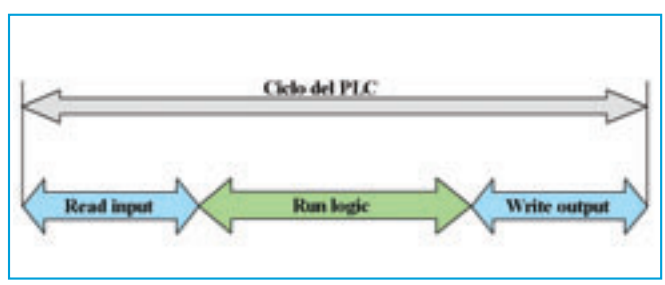

Figura 6 - Ciclo di un PLC

Il problema principale che si presenta nel misurare solamente il tempo di ciclo e la sua ripetitività è che il softPLC comunica con gli altri processi solo per mezzo di un server OPC. Da ciò ne consegue che si può accedere ad una variabile di uscita della logica solo utilizzando un client OPC. Un oggetto OPC

ha un formato standard, costituito dal valore del dato seguito dall'etichetta temporale (timestamp) e dallo stato della variabile trasmessa (cfr figura 7)

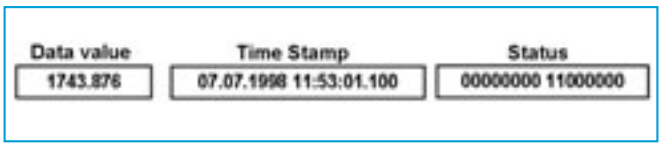

Figura 7 - Formato dati OPC

Il valore del dato è il valore dell'oggetto OPC. Il timestamp è aggiunto dal server e rappresenta il tempo in cui l'oggetto è stato processato dal server. L'ultimo campo, lo stato, rappresenta la qualità del dato trasmesso. Il timestamp associato al dato non è quindi indicativo del tempo di ciclo del softPLC, in quanto è la somma del tempo di ciclo e del tempo necessario al server per convertire il formato dati nel formato standard OPC.
Una tecnica che può essere utilizzata per misurare il solo tempo di ciclo sfrutta una funzione propria del soft-PLC, chiamata gettime, che memorizza il tempo di sistema quando viene richiamata. L'istruzione gettime è eseguita durante il ciclo del PLC, quindi il tempo di ciclo può essere facilmente calcolato come la differenza tra due tempi successivi ottenuti dal gettime, come mostrato nella figura 8.

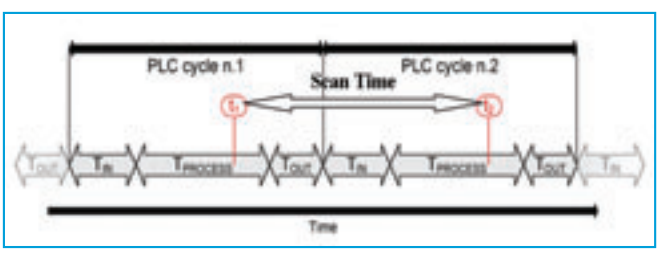

Figura 8 - Misura del tempo di ciclo

Facendo riferimento alla figura 8, T1 e T2 sono gli istanti in cui la funzione gettime è eseguita, rispettivamente durante il ciclo 1 ed il ciclo 2. I tempi per leggere gli ingressi e scrivere le uscite possono essere ritenuti costanti da un ciclo all'altro, quindi la differenza tra T2 e T1 rappresenta la durata del tempo di ciclo. La risoluzione del tempo misurato con la funzione gettime è pari a 1 ms.
Come già evidenziato in precedenza, l'interfaccia tra le uscite del SoftPLC e qualsiasi applicazione di più alto livello avviene per mezzo di una tecnica client/server OPC. Il client OPC legge i dati di uscita del SoftPLC che sono stati memorizzati nel server OPC, che a sua volta ha una propria dinamica e crea gli item OPC mediante interrogazioni periodiche al SoftPLC. Il ciclo di interrogazione da parte del server OPC al SoftPLC è impostabile dall'utente, con l'avvertenza che se il ciclo del PLC ha una durata inferiore al ciclo del server OPC verrebbero perse le variabili interne al ciclo, come mostrato schematicamente nella figura 9.

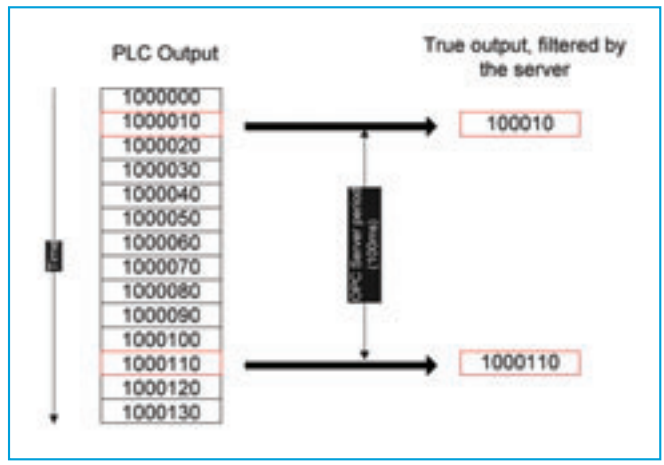

Figura 9 - Comportamento del server OPC

Per avere la certezza di non perdere alcun ciclo del PLC a causa del server OPC, tutte le variabili prodotte dall'istruzione gettime vengono memorizzate in un array, che viene ciclicamente memorizzato dal server OPC, come illustrato nella figura 10.

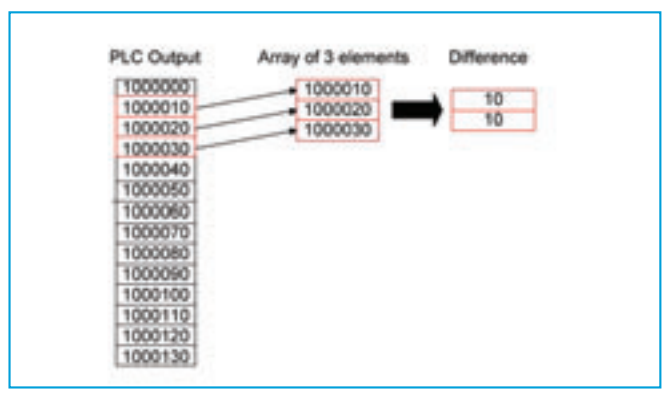

Figura 10 - Array di memorizzazione dei dati

Risultati sperimentali

Avendo definito un metodo di misura delle prestazioni temporali per un'architettura PC-based, si è passati ad eseguire una prova allestendo un sistema di prova, così da poter valutare i primi risultati in merito alle prestazioni che è ragionevole attendersi dal sistema. Altro scopo delle misure sul sistema di prova è quello di identificare i parametri di influenza sulle prestazioni del sistema.
Il semplice sistema di prova allestito è riportato nella figura 11 ed è composto dai seguenti componenti:
- moduli di I/O;
- rete fieldbus: Profibus DP;
- scheda di interfaccia;
- OPC client/server embedded sulla scheda di interfaccia;
- SoftPLC;
- OPC client/server per applicazioni HMI;
- HMI: sviluppato con pacchetti software freeware [14].

Le prove condotte sono state svolte considerando un carico

della CPU del PC variabile da circa zero al 100%, con step del 20%. La prima configurazione è ottenuta fermando tutti i processi che non sono necessari per far girare il sistema operativo ed il soft-PLC mediante uno specifico strumento software realizzato da Microsoft [15]. Al contrario, per creare un carico variabile sulla CPU è stato sviluppato un semplice programma (chiamato CPU-waster) che incrementa ciclicamente una variabile nella cache del sistema. Il SoftPLC è stato programmato con una logica che permetta di memorizzare in una tabella i successivi valori del tempo ottenuti con gettime, e la tabella viene esportata su Excel per mezzo dell'interfaccia OPC, e qui viene processata per ottenere i valori del tempo di ciclo. I dati sono richiamati dal server OPC per mezzo di un client OPC, che può avere due diversi metodi di accesso ai dati contenuti nel server:

- sincrono: il client spedisce periodicamente una richiesta al server e quest'ultimo risponde immediatamente inviando i dati richiesti.
- asincrono: il client spedisce periodicamente una richiesta al server, questo risponde con un feedback di ricevuta richiesta ed invia i dati solo dopo che è un certo intervallo temporale è trascorso.

L'approccio scelto per la campagna di misura è quello asincrono, in quanto ha un carico minore sulla CPU del sistema, come appare anche dal grafico riportato nella figura 12. Un altro parametro che influenza l'utilizzo di CPU è il tempo di ciclo del client OPC. È stato scelto un valore intermedio: 50 ms, che è esattamente metà del ciclo di scansione del server OPC selezionato. Un tempo di ciclo del client OPC inferiore si tradurrebbe in un carico elevato per la CPU privo di qualsiasi vantaggio in termini di risultati ottenuti perché sarebbero di fatto acquisiti più volte gli stessi dati.

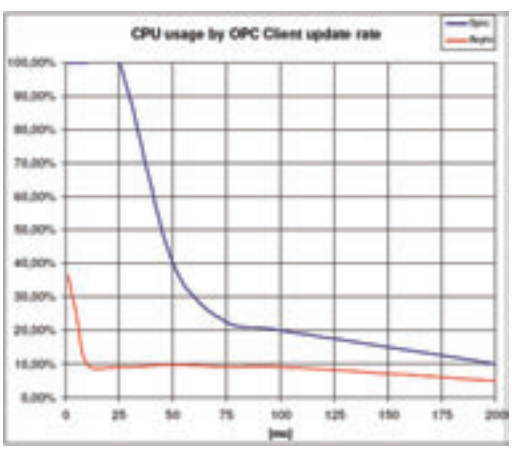

Figura 12 - Carico della CPU in funzione del tempo di ciclo del client OPC

Figura 11 - Set-up hardware del test-bed

L'ambiente in cui vengono eseguite le prove è caratterizzato non solo dal carico della CPU, ma anche dalla priorità assegnata ad ogni singolo processo che viene eseguito. La priorità assegnata influisce sulla durata dello slot temporale a disposizione del processo prima che questo venga interrotto e venga chiamato a lavorare un altro processo. Più la priorità è alta e maggiore è il tempo a disposizione. Per condurre le prove sono stati presi in considerazione diverse priorità associate ai diversi componenti:

- tutti i componenti software hanno la priorità di default (livello 8);
- la priorità dell'OPC server è al valore massimo (livello 13);
- la priorità dell'OPC client è al valore massimo (livello 13);
- la priorità del SoftPLC è al valore massimo (livello 13);
- la priorità del SoftPC e dell'OPC server è al valore massimo (livello 13)

I diversi ambienti che sono stati considerati per eseguire le prove sono riassunti nelle tabelle seguenti. Le colonne rappresentano il carico della CPU, mentre le righe definiscono la percentuale di CPU disponibile consumata da ogni processo attivo.

NORMAL	Priority	no	80	60	40	20	10	1	0
CPU Waster	8	0,00%	29,90%	36,87%	45,09%	56,63%	69,44%	77,86%	99,07%
Idle	N/A	90,00%	57,77%	48,60%	41,93%	25,17%	13,10%	16,90%	0,00%
OPC Client	8	9,17%	8,53%	11,37%	8,80%	12,63%	8,13%	1,73%	0,23%
Process Explorer	13	0,10%	0,10%	0,10%	0,10%	0,10%	0,13%	0,10%	0,10%
SmartPLC	8	0,03%	1,03%	0,30%	0,23%	0,33%	0,30%	0,13%	0,17%
SmartPLCDA	8	0,13%	1,93%	2,60%	3,47%	3,67%	8,23%	3,00%	0,13%
SmartPLCUI	8	0,57%	0,73%	0,17%	0,43%	1,47%	0,97%	0,27%	0,27%

Tabella 1 - Consumi di CPU con priorità di default

SmartPLC	Priority	no	80	60	40	20	10	1	0
CPU Waster	8	0,00%	32,07%	38,33%	47,70%	58,51%	71,67%	78,09%	98,90%
Idle	N/A	90,13%	56,70%	48,60%	41,37%	25,86%	15,03%	18,23%	0,00%
OPC Client	8	8,67%	9,33%	10,77%	8,97%	11,90%	8,77%	1,73%	0,63%
Process Explorer	13	0,10%	0,30%	0,10%	0,10%	0,13%	0,10%	0,10%	0,07%
SmartPLC	13	0,00%	0,00%	0,00%	0,00%	0,53%	0,03%	0,00%	0,00%
SmartPLCDA	8	0,10%	0,63%	1,63%	1,10%	1,77%	3,83%	1,87%	0,23%
SmartPLCUI	8	0,00%	0,97%	0,57%	0,77%	1,30%	0,57%	0,07%	0,17%

Tabella 2 - Consumi di CPU con SoftPLC con priorità massima

SmartPLCDA	Priority	no	80	60	40	20	10	1	0
CPU Waster	8	0,00%	31,44%	37,51%	45,77%	59,83%	71,13%	73,43%	91,39%
Idle	N/A	89,03%	57,69%	50,75%	42,53%	28,30%	16,40%	17,80%	0,00%
OPC Client	8	10,03%	8,20%	9,84%	9,10%	9,90%	9,27%	5,20%	7,24%
Process Explorer	13	0,10%	0,10%	0,10%	0,10%	0,10%	0,13%	0,10%	0,10%
SmartPLC	8	0,03%	0,37%	0,10%	0,50%	0,30%	0,50%	2,67%	0,10%
SmartPLCDA	13	0,13%	0,63%	0,70%	1,37%	1,40%	2,83%	0,73%	0,17%
SmartPLCUI	8	0,67%	0,57%	1,00%	0,60%	0,17%	0,03%	0,07%	0,97%

Tabella 3 - Consumi di CPU con Server OPC con priorità massima

OPC CLIENT	Priority	no	80	60	40	20	10	1	0
CPU Waster	8	0,00%	33,01%	38,40%	47,42%	61,34%	71,39%	77,79%	98,97%
Idle	N/A	89,23%	55,72%	47,47%	39,98%	24,35%	13,21%	18,24%	0,00%
OPC Client	13	10,00%	9,44%	11,27%	9,20%	10,67%	9,17%	1,20%	0,50%
Process Explorer	13	0,13%	0,10%	0,10%	0,10%	0,10%	0,10%	0,10%	0,07%
SmartPLC	8	0,07%	0,13%	0,07%	0,03%	0,07%	0,33%	0,07%	0,10%
SmartPLCDA	8	0,43%	1,43%	2,27%	2,53%	2,90%	4,97%	1,63%	0,10%
SmartPLCUI	8	0,13%	0,07%	0,43%	0,73%	0,53%	0,83%	0,07%	0,23%

Tabella 4 - Consumi di CPU con client OPC con priorità massima

PLC/PLCDA	Priority	no	80	60	40	20	10	1	0
CPU Waster	8	0,00%	33,90%	39,28%	48,52%	61,70%	73,03%	73,34%	89,22%
Idle	N/A	89,63%	55,67%	50,52%	39,95%	27,63%	16,47%	17,76%	0,00%
OPC Client	8	9,73%	9,87%	9,30%	10,23%	9,47%	9,40%	8,40%	9,00%
Process Explorer	13	0,10%	0,10%	0,10%	0,10%	0,10%	0,10%	0,13%	0,10%
SmartPLC	13	0,03%	0,03%	0,07%	0,10%	0,03%	0,10%	0,00%	0,03%
SmartPLCDA	13	0,13%	0,27%	0,13%	0,37%	0,30%	0,13%	0,30%	0,13%
SmartPLCUI	8	0,40%	0,17%	0,67%	0,77%	0,80%	0,80%	0,07%	0,93%

Tabella 5 - Consumi di CPU con SoftPLC e server OPC con priorità massima

I processi attivi sono i seguenti:
- CPU waster: è l'applicativo scritto in Visual Basic 6.0 che permette di generare un carico variabile e regolabile della CPU.
- Idle di sistema: è la funzione di ciclo del sistema e non è un vero e proprio processo, serve a mantenere il processore in stato attivo pur non impegnandone le funzioni di calcolo. L'idle di sistema assorbe le risorse del processore quando questo è scarico e le rilascia ai processi attivi concorrenti quando ve ne è bisogno. L'utilizzo di CPU dell'idle è alto quando il sistema è scarico, mentre diminuisce all'aumentare del carico.
- OPC client: è il client che si interfaccia con il server OPC per rendere fruibile all'utente l'array di tempi immagazzinato nel server OPC.
- Process Explorer: è un applicativo software fornito da Sysinternals che permette di visualizzare l'utilizzo di CPU di ogni singolo processo attivo.
- SmartPLC: è il programma di SoftPLC vero e proprio.
- SmartPLCDA: è il server OPC del programma di emulazione del PLC.
- SmartPLCUI: è l'interfaccia grafica che mette in relazione il SoftPLC ed il suo server OPC, viene avviata in automatica dal server OPC.

Per tutte le possibili combinazioni del carico della CPU e dei livelli di priorità i parametri misurati sono il tempo di ciclo medio del SoftPLC e la sua varianza, che rappresenta una misura della dispersione dei valori misurati rispetto al valore medio:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \qquad (1)$$

Dove:
- $\bar{x}$ è la media del tempo di ciclo calcolata
- n è il numero di elementi raccolti per calcolare la media.

Per avere delle buone prestazioni del SoftPLC è necessario che questo abbia una varianza molto bassa, e ciò implica che il tempo di ciclo rimane costante ed il suo comportamento può essere ritenuto di fatto deterministico. I risultati delle prove eseguite sono riportati nella tabella 6.

Se tutti i processi hanno ugual priorità, il tempo di ciclo medio non è influenzato in maniera significativa dal carico della CPU. Non così per la varianza, che può assumere valori elevati per un elevato carico della CPU, originando quindi problemi per quanto attiene il determinismo delle applicazioni di controllo.

Assegnando la priorità massima al server OPC, il tempo medio di ciclo diventa instabile e dipende dal carico della CPU. Probabilmente questo fenomeno dipende dal fatto che l'OPC server si impone sul SoftPLC che non riesce ad avere uno slot di CPU allocato in maniera adeguata alle sue esigenze. In questa condizione la varianza attorno al valore medio è piuttosto alta ed aumenta con l'aumentare del carico della CPU.

I risultati più interssanti sono ottenuti se la priorità più elevata è assegnata al SoftPLC. In questa condizione di lavoro, il tempo medio di ciclo è costante (10 ms) e le variazioni attorno a questo valore sono trascurabili. Il comportamento del SoftLPC può essere considerato di fatto deterministico sotto ogni aspetto pratico.

I risultati delle prove sono riportati graficamente nella figura 13: per ogni priorità la varianza è riportata in funzione del carico della CPU.

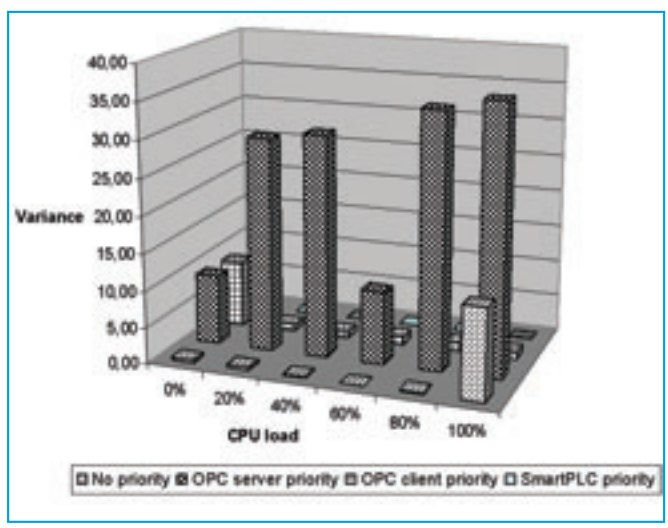

Figura 13 - Risultati delle prove

CPU load	No Priority		OPC server		OPC client		SoftPLC		OPC server + SoftPLC	
	Mean time [ms]	Variance [ms²]	Mean time [ms]	Variance [ms²]	Mean time [ms]	Variance [ms²]	Mean time [ms]	Variance [ms²]	Mean time [ms]	Variance [ms²]
Minimum	10	0,39	11	9,5	10	0,19	10	0,01	10	0,01
20%	10	0,40	12	29,52	10	1,02	10	0,01	10	0,01
40%	10	0,18	12	30,24	10	0,86	10	0,01	10	0,01
60%	10	0,06	11	9,97	10	1,12	10	0,01	10	0,01
80%	10	0,18	12	34,43	10	1,22	10	0,01	10	0,01
Maximum	10	12,39	12	36,30	10	0,96	10	0,01	10	0,01

Tabella 6 - Risultati delle prove eseguite sul ciclo di un SoftPLC

I risultati ottenuti sono riportati anche nelle figure 14 e 15 per una maggiore leggibilità

Nella figura 14 sono riportati gli andamenti del tempo medio di ciclo per diversi carichi di CPU e per diverse priorità assegnate ai processi che entrano in gioco. Il grafico evidenzia molto bene, come assegnando priorità elevata al softPLC il tempo medio di ciclo non sia influenzato dal carico della CPU, come invece accade negli altri casi (curva rossa costante a 10 ms).

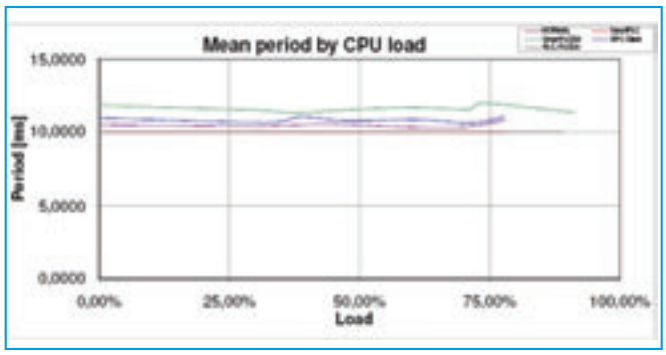

Figura 14 - Andamento del periodo di ciclo in funzione del carico della CPU

Anche la varianza rispetto al valor medio ha un comportamento analogo: rimane costante e con valori trascurabili se la priorità maggiore viene assegnata al SoftPLC.

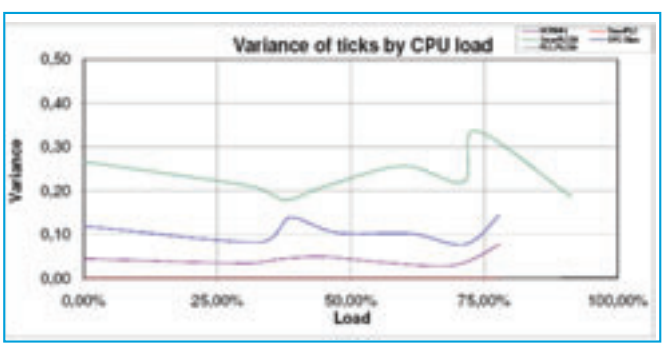

Figura 15 - Andamento della varianza in funzione del carico

Le prove sono state condotte anche su altri PC aventi caratteristiche diverse, in particolare sono state impiegate macchine con diversi processori, alcuni DualCore, ed i risultati ottenuti sono in linea con quanto riportato nel presente paragrafo.

Conclusioni

Gli sviluppi più recenti nel mondo del controllo di processo vedono il consolidarsi di architetture ibride tra DCS e PLC, chiamate Process Control Systems. Le architetture hardware sono basate su un campo intelligente collegato su fieldbus ed interfacciato ai controllori, deputati all'implementazione delle logiche di controllo. I livelli più alti per la supervisione, le funzioni di HMI e le funzioni più avanzate sono implementati su PC interconnessi attraverso una rete Ethernet.
Sono oggi disponibili applicativi software che consentono di emulare il comportamento dei PLC/controllori su piattaforma PC. In tal modo un normale PC dotato di un sistema operativo general purpose (ad es. Windows 2000 o XP) è in grado di riprodurre il comportamento di un PLC hardware. Le configurazioni PC-based devono essere verificate in termini di prestazioni temporali, in quanto il problema principale sembra essere quello di garantire prestazioni deterministiche utilizzando un sistema che per sua natura è intrinsecamente non deterministico in quanto multi-tasking. Le prove condotte in laboratorio su un semplice sistema di prova hanno mostrato che impostando a valori corretti le priorità assegnate al SoftPLC, questo ha comportamenti che possono essere del tutto paragonabili a quelli di un PLC convenzionale, sotto l'aspetto della stabilità temporale delle applicazioni. Le prestazioni però mostrano che un approccio PC-based può essere impiegato solo per processi aventi dinamiche nell'ordine delle decine di ms. Anche se i tempi di risposta possono essere soddisfacenti per alcune applicazioni che non abbiano dinamiche troppo spinte, rimane un problema legato all'affidabilità dell'intero sistema legato alla impossibilità di ridondare il controllo.
L'attività di ricerca in corso mira a identificare quali siano le condizioni al contorno che possono influenzare le prestazioni di un sistema PC-based: tipo di processore, frequenza, dimensioni della cache memory, periferiche, applicativi software diversi ecc.

Bibliografia

[1] IEC 61131-3, "Programmable controllers – Part 3: Programming languages", 2003
[2] D. Miklovic, "Integration of DCS, PLC, HMI and SCADA systems", in B. G. Liptak, *Process Software and Digital Networks, CRC Press*, 2002
[3] V. Barbero, A. Grimaudo, P. Pinceti, A. Russo, "Fully integrated control for a rotor over-speed testing facility", *2002 International Symposium on Industrial Electronics*, IEEE-ISIE 2002, July 8-11, 2002, L'Aquila
[4] F. Iwanitz, J. Lange, *OLE for Process Control: OPC. Fundamentals, Implementation and Application*, Huthig Verlag, 2001
[5] W. Stallings, *Operating systems internals and design principles*, Prentice Hall, 2004
[6] Microsoft Corporation, "Microsoft Windows 2000 Overview", MSDN Libray, www.msdn2.microsoft.com
[7] T.L. Casavant, J.G. Kuhl, "A taxonomy of scheduling in general purpose distributed computing systems", *IEEE Transactions on Software Engineering*, Vol. 14, Issue 2, Feb. 1998, pp. 141-154
[8] P.A. Laplante, *Real-Time systems design and analysis*, Wiley-IEEE Press, 2004
[9] S. Baskiyar, "A survey on Real-Time Operating Systems", *Proceedings on Networks, Parallel and Distributed Processing and Applications*, April 2002, Japan
[10] G. Bollella, K. Jeffay, "Support or real-time computing within general purpose operating systems supporting co-resident operating systems", *Proceeding of Real Time Technology and Applicatios Symposium*, 15-17 May 1995, pp. 4-14
[11] Infoteam, *OpenPCS 5.2.2 User's Guide*, 2006
[12] Infoteam, *Smart PLC/OPC 2006*, 2005
[13] D. J. Lilja, *Measuring Computer Performance., A practitioner Guide*, Cambridge University Press, 2000.
[14] Prosys OPC test client: www.prosys.fi
[15] TechNet, PSkill V1.12: www.sysinternals.com