Modello di una centralina multi attività con Modelica

Luigi Glielmo, Oreste Riccardo Natale, Stefania Santini

Lo sviluppo di sistemi di controllo richiede non soltanto la progettazione delle strategie ma anche la ricerca di compromessi con le logiche di supervisione, la pianificazione delle attività in tempo reale e le problematiche legate alle comunicazioni su rete.
Si propone un modello funzionale per un processore con sistema operativo multi attività descritto secondo il paradigma orientato agli oggetti sulla base dello standard Iec 1131, il cui scopo è la verifica degli effetti della pianificazione delle attività sulle prestazioni delle leggi di controllo.
Il linguaggio utilizzato per la descrizione del modello è Modelica e le simulazioni sono state effettuate con Dymola.

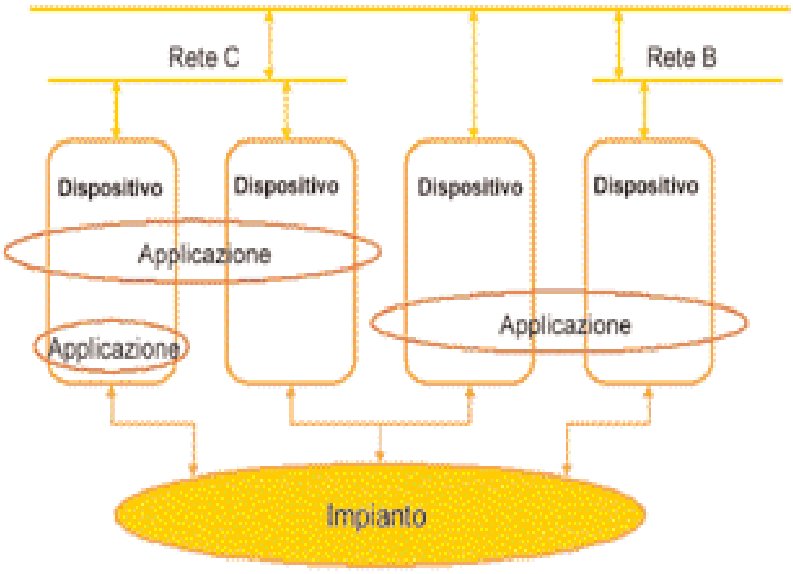

Figura 1 - Architettura di un sistema di controllo secondo lo standard Iec 1131

I programmi redatti per sistemi integrati in tempo reale sono molto diversi dalla corrispondente controparte per sistemi di elaborazione più tradizionali; infatti, i primi, a differenza dei secondi, sono causa ed effetto di una stretta interazione con una parte del mondo reale, e soprattutto questa interazione avviene alla velocità propria dell'ambiente. D'altra parte tali algoritmi sono spesso progettati sulla base di notevoli assunzioni sulle caratteristiche delle piattaforme di calcolo; spesso, inoltre, la necessità di utilizzare hardware ampiamente testato, e quindi non al passo con i tempi, ma sufficientemente affidabile per applicazioni critiche, si scontra spesso con la complessità del software progettato. Questo porta, dopo la progettazione degli algoritmi di controllo e supervisione secondo metodologie analitiche assestate, ad una complessa fase di sperimentazioni durante le quali le leggi originarie sono spesso modificate per venire incontro a esigenze più pratiche. Gli strumenti di simulazione più diffusi mancano della capacità di modellare le caratteristiche temporali del sistema operativo della centralina od i ritardi dovuti ai protocolli di comunicazione, e la comunità scientifica si sta adoperando per colmare questa lacuna [1, 2].
L'inclusione di questi aspetti all'interno di una campagna di simulazioni estensive permetterebbe l'identificazione dei possibili problemi e delle corrispondenti soluzioni prima della campagna di sperimentazioni hardware-in-the-loop.
L'introduzione di un modello del comportamento del microcontrollore, che porti in conto ad esempio la pianificazione dell'esecuzione dei task, o il non determinismo causato dalle interruzioni, aiuta l'ingegnere dei controlli a simulare e a validare più realisticamente le prestazioni e l'efficienza delle strategie progettate, anche nell'ottica dell'emergenza di metodologie orientate al trattamento delle problematiche implementative [3]. L'obiettivo è, dunque, quello di dotarsi di uno strumento per una prima validazione in simulazione delle leggi di con-

L. Glielmo, Professore Ordinario di Automatica, O.R. Natale, dottorando di Ricerca in Ingegneria dell'Informazione Dipartimento di Ingegneria dell'Università degli Studi del Sannio in Benevento; S. Santini, ricercatore, Dipartimento di Informatica e Sistemistica dell'Università degli Studi di Napoli Federico II.

trollo progettate ed in questa memoria si propone, quale primo caso di studio, un modello di centralina multi attività, descritto secondo il paradigma orientato agli oggetti. Il linguaggio utilizzato per la descrizione del modello è Modelica.
La memoria si articola come segue: dopo una breve introduzione al linguaggio utilizzato, è presentato l'approccio modellistico basato sullo standard Iec 1131, successivamente si introduce il modello proposto completato da alcuni esempi di utilizzo.

Modelica

Modelica è un linguaggio per la descrizione della dinamica dei sistemi fisici, ed è stato progettato per essere un valido strumento per lo sviluppo di librerie che permettano la condivisione dei modelli. Questo formalismo, sviluppato dalla organizzazione senza fini di lucro Modelica Association con sede a Linköping in Svezia, è basato sulla modellistica non-causale ed è dotato di costrutti lessicali per la descrizione orientata agli oggetti dei sistemi dinamici, il cui scopo è facilitare il riutilizzo e la particolarizzazione dei modelli già disponibili [4].
Modelica è dotato di costrutti per la descrizione di modelli che siano in parte guidati da dinamiche tempo continue ed in parte da dinamiche tempo discrete. In particolare è possibile definire insiemi di equazioni che possono essere valutate condizionalmente al verificarsi di talune condizioni logiche: senza essere esaustivi si cita il caso di equazioni alle ricorrenze che hanno senso soltanto in istanti di tempo equispaziati, oppure il caso di sistemi dinamici tempo continui il cui comportamento si diversifica al variare della traiettoria all'interno dello spazio di stato.
Il paradigma di programmazione orientato agli oggetti ha letteralmente rivoluzionato nell'ultima decade le metodologie per lo sviluppo del software, facendo leva sulle esigenze di riutilizzo e facilità di modifica per particolarizzazione del codice già sviluppato. Le metodologie ideate possono essere applicate con successo alla modellistica dei sistemi dinamici [5].
Modelica permette sia la costruzione di modelli per composizione, facendo uso di librerie di modelli già disponibili, sia la definizione di nuovi modelli descrivendone il comportamento facendo uso di equazioni differenziali, alle ricorrenze o algebriche [6].

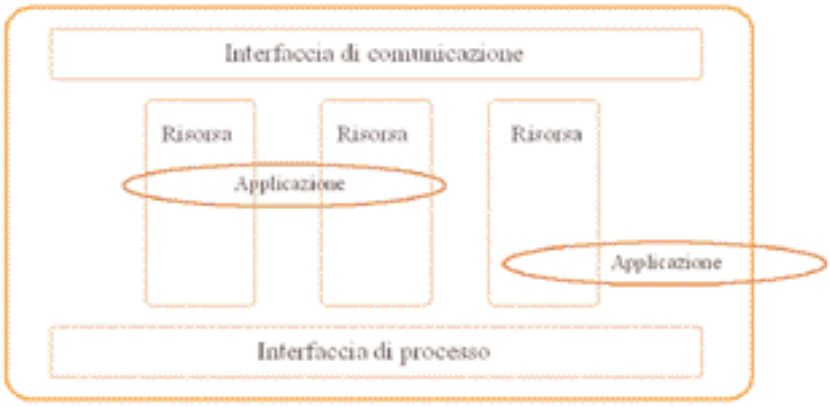

Figura 2 - Un dispositivo secondo lo standard Iec 1131

Approccio modellistico

Lo standard Iec 1131 definisce l'architettura di un sistema di controllo come in Figura 1 [7]. Ciascun dispositivo è dotato di un'interfaccia verso il processo, e può opzionalmente comunicare con altri dispositivi attraverso infrastrutture informatiche eventualmente strutturate gerarchicamente. Un'applicazione di controllo può utilizzare uno o più dispositivi (sensori, attuatori, microcontrollori, controllori a logica programmabile) condividendone alcuni con altre applicazioni.
Ciascun dispositivo è dotato inoltre di alcune risorse, logiche o fisiche, assegnate alle applicazioni secondo una logica che ne prevede l'utilizzo mutuamente esclusivo per quel che riguarda le risorse condivise (Figura 2).
Al fine di proporre un modello sufficientemente generale, i dettagli architetturali di un dispositivo possono essere convenientemente riassunti nello schema funzionale di Figura 3; l'esecuzione delle diverse funzionalità è amministrata da un *pianificatore*, il quale ne regola l'esecuzione per conto delle diverse applicazioni.
La descrizione funzionale di Figura 3 può essere meglio razionalizzata utilizzando il formalismo del linguaggio grafico Uml [8, 9]; in Figura 4, utilizzando un diagramma delle classi, un dispositivo *(device)* è messo in re-

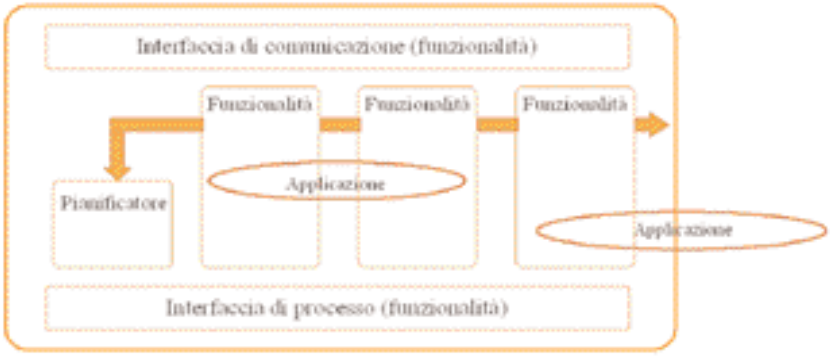

Figura 3 - Rappresentazione funzionale di un dispositivo

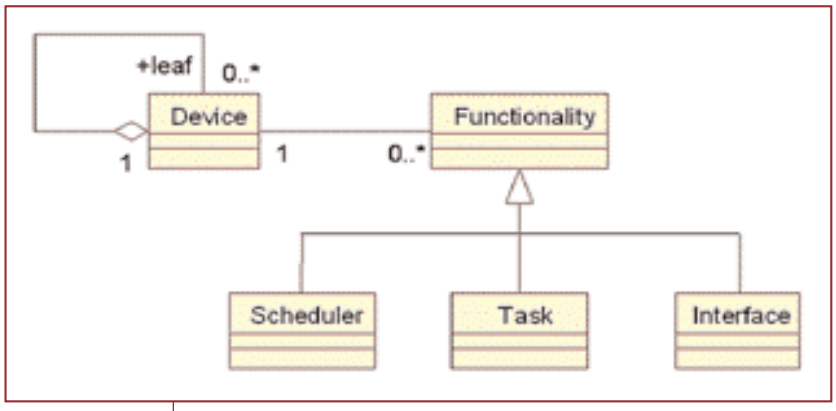

Figura 4 - Rappresentazione ricorsiva di un dispositivo mediante un diagramma delle classi

lazione a un'insieme di funzionalità *(functionality),* dove il modello generale di funzionalità può essere specializzato alla descrizione di pianificatori *(scheduler),* attività *(task)* o interfacce *(interface)*. Qualora sia necessario modellare con maggior dettaglio una risorsa, è possibile descrivere un dispositivo come composto da altri sottodispositvi *(leaf)* ciascuno con le proprie funzionalità, mantenendo in questo modo la coerenza con il diagramma delle classi proposto; il modello ad oggetti risultante può dunque considerarsi una descrizione ricorsiva [11, 12].
La struttura di Figura 4 è sufficientemente generale per poter descrivere sia dispositivi di misura, sia dispositivi di attuazione, sia dispositivi per il controllo. Ad esempio un sensore può essere modellato come un dispositivo dotato di un'interfaccia per trasmettere la misura rilevata, e di almeno una funzionalità, per la misura della grandezza d'interesse; il pianificatore può mancare del tutto essendo unica la funzionalità del sensore. Nella prossima sezione è descritto l'utilizzo della metodologia precedentemente illustrata, applicata alla modellistica di una centralina di controllo monoprocessore dotata di una versione semplificata di un sistema operativo integrato multi attività con prerilascio.

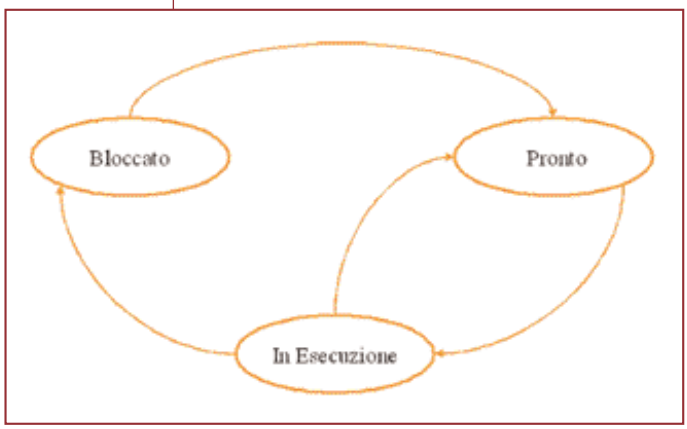

Figura 5 - Transizioni fra gli stati logici di un'attività

Modello della centralina

La descrizione del comportamento dinamico di una Ecu (Electronic Control Unit) può essere affrontata modellando le diverse funzionalità realizzate a tempo di esecuzione; infatti, il codice in esecuzione sulla centralina può pensarsi funzionalmente ripartito in uno o più attività eventualmente interagenti (controllo modulante di sistemi continui e supervisione di sistemi ad eventi discreti). Nell'ipotesi di centralina monoprocessore, un pianificatore, nelle veci di un sistema operativo integrato, ha il compito assegnare l'unica risorsa di calcolo alle diverse attività.

Modello di attività

Un'attività pianificata per l'esecuzione su una centralina può assumere i seguenti stati logici [13]:

- Stato *Pronto*. In questo stato l'attività è in attesa della disponibilità del processore.
- Stato *In esecuzione*. Un'attività in esecuzione occupa la risorsa processore, e permane in questo stato finché non termina la propria elaborazione oppure non le viene forzatamente sottratta la risorsa di calcolo per concederla ad un'altra attività. Un'attività in esecuzione a cui è forzatamente sottratta il processore commuta nello stato Pronto.
- Stato *Bloccato*. Un'attività in questo stato è in attesa di un evento che la faccia commutare nello stato Pronto. Concettualmente, questa stato rappresenta una condizione diversa da quella rappresentata dallo stato Pronto, poiché un'attività bloccata manca ancora di qualche informazione necessaria alla sua esecuzione, e non appena tali dati si rendono disponibili commuta nello stato Pronto.

I diversi stati e le possibili transizioni dall'uno all'altro sono rappresentati in Figura 5. Per poter portare in conto la complessità computazionale caratteristica di ciascuna attività, il modello prevede che lo stato permanga nella condizione di In esecuzione per un numero di cicli macchina ad essa proporzionale.
Il diagramma di Figura 5 può essere formalizzato in quello di Figura 6, ciascuno degli stati descritti è modellato come un posto di una classe di Rete di Petri nota come State

Machine [14]; in Tabella 1 il posto *StRunning2*, non corrispondente ad alcuno degli stati di Figura 6, è un posto instabile necessario per determinare la condizione di autosospensione dell'attività al termine della propria esecuzione in ciascun periodo. L'elaborazione dell'algoritmo assegnato all'attività è completata, in unico passo di integrazione, quando la transizione *T5* risulta superabile; i segnali di ingresso sono campionati all'attivazione dello stato *StReady* ed i risultati dell'algoritmo sono esposti contemporaneamente all'attivazione dello stato *StIdle*.

Modello di pianificatore con algoritmo Edf

L'algoritmo *Earliest deadline first* (Edf) assegna l'utilizzo del processore al processo la cui prossima deadline è più imminente fra

Tabella 1 - Significato dei posti e delle transizioni della rete in Figura 6

Posto	Significato
StIdle	Att. sospesa
StReady	Att. in attesa della CPU
StRunning1	Att. in esecuzione
StRunning2	Att. in esecuzione
$T1$	Istante di campionamento
$T2$	Abilitazione dal pianificatore
$T3$	Prerilascio
$T4$	Clock CPU
$T5$	Mancano ancora dei cicli
$T6$	Cicli necessari eseguiti

Tabella 2 - Significato dei posti e delle transizioni della rete in Figura 7

Posto	Significato
CPUIdle	CPU inattiva
PriorityTask1	Attività 1 più urgente
PriorityTask2	Attività 2 più urgente
$T1$	Scadenza più imminente
$T2$	$\overline{T1}$
$T3$	Scadenza più imminente
$T4$	$\overline{T2}$

quelli in attesa nella coda dei processi pronti [15]. Edf è un algoritmo con prerilascio, nel senso che il processore può essere forzatamente sottratto ad un'attività in esecuzione in favore di un'attività più urgente. A differenza di altri algoritmi di pianificazione off-line, quali per esempio il *Rate Monotonic*, Edf può essere utilizzato sia per la gestione di attività periodiche che aperiodiche (operanti su se-

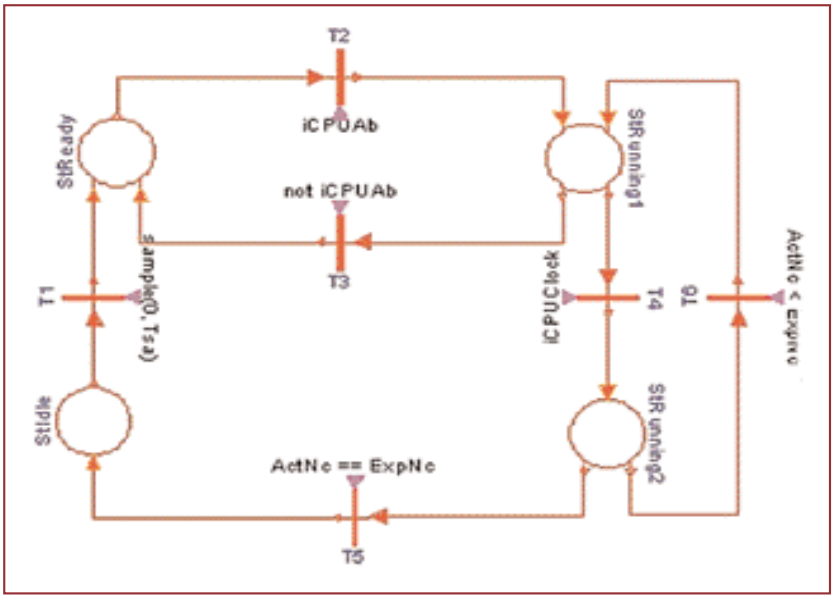

Figura 6 - Transizioni fra gli stati logici di un'attività; modello con Rete di Petri; descrizione in Modelica - Dymola

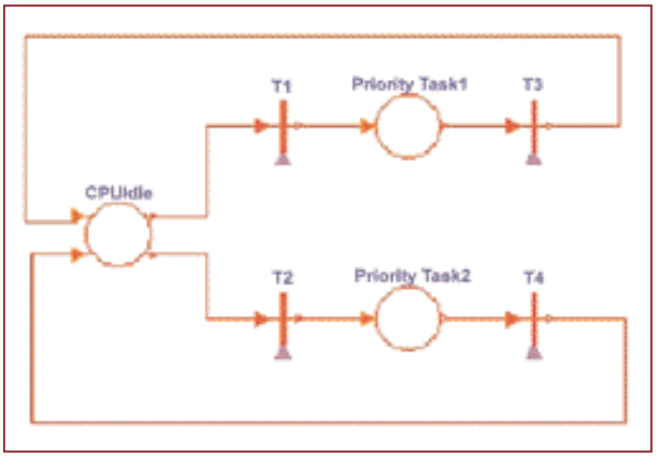

Figura 7 - Pianificazione dinamica di due attività secondo l'algoritmo Earliest deadline first

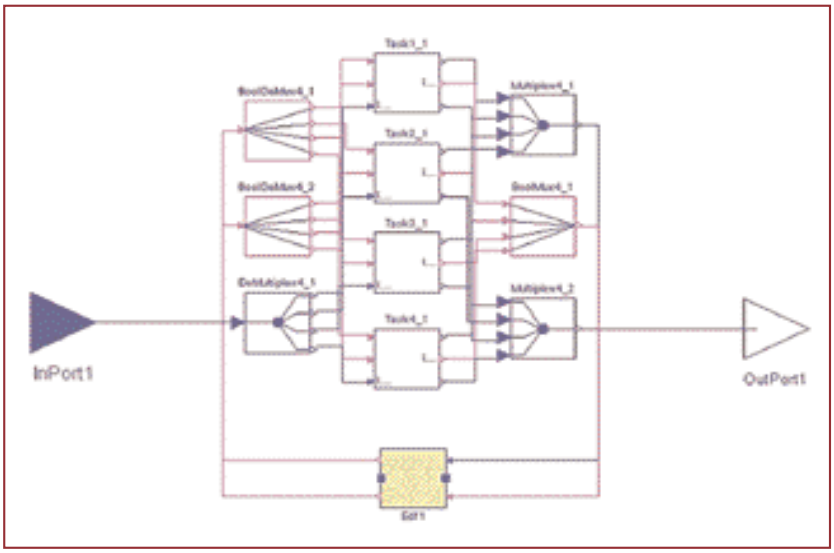

Figura 8 - Schema a blocchi di un'istanza del modello di centralina

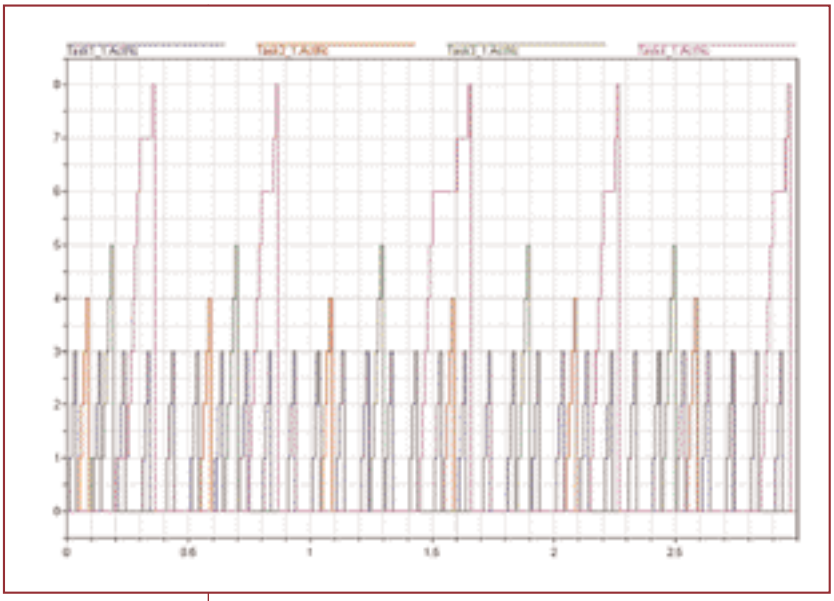

Figura 9 - Simulazione con pianificazione Edf; sono rappresentati il numero di cicli macchina eseguiti da ciascuna attività; si osservi come l'esecuzione delle attività meno frequenti sia sospesa in favore di quelle più urgenti

gnali di interruzione esterni e asincroni). In Figura 7 è rappresentato il pianificatore Edf, modellato come una Rete di Petri, per due sole attività per semplificarne l'esposizione; la condizione sulla transizione *T1* è che la scadenza relativa alla prima attività sia più imminente della scadenza relativa alla seconda qualora essa sia nello stato Pronto.

La centralina

La centralina, in accordo al diagramma di Figura 4, può essere dunque descritta da un pianificatore e da un'insieme di attività secondo lo schema a blocchi di Figura 8, dove sono rappresentate quattro attività periodiche. Sono state effettuate due tipologie di simulazioni. Nella prima si è ipotizzato che la relazione

$$\sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1 \qquad (1)$$

fosse verificata (N è la cardinalità dell'insieme di attività periodiche considerate, C_i è la durata massima di ciascuna attività e T_i il corrispondente periodo) [15]; in questo caso è possibile eseguire tutte le attività rispettando le scadenze (simulazione in Figura 9, parametri in Tabella 3). Nella seconda, al contrario, si ipotizza che la configurazione delle attività sia tale che la (1) non sia rispettata; in tal caso con alcune di esse non riescono a completare la propria esecuzione entro le deadline assegnate (Figura 10, parametri in Tabella 4).

Tabella 3 - Configurazione attività per la simulazione in Figura 9

Attività	Complessità	Periodo
1	0.03s	0.1s
2	0.04s	0.5s
3	0.05s	0.6s
4	0.08s	0.7s

Tabella 4 - Configurazione attività per la simulazione in Figura 10

Attività	Complessità	Periodo
1	0.03s	0.05s
2	0.04s	0.5s
3	0.05s	0.6s
4	0.08s	0.7s

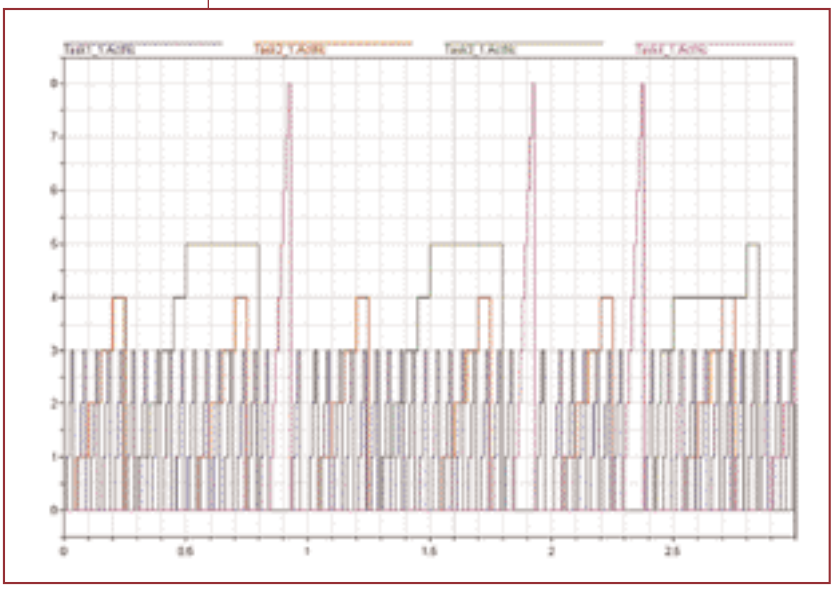

Figura 10 - Simulazione con pianificazione Edf; in questo caso la condizione (1) non è rispettata

Conclusioni e sviluppi futuri

In questo articolo è stato proposto un modello di centralina multi attività descritto con il linguaggio Modelica; il modello può essere facilmente integrato nelle tradizionali simulazioni di schemi di controllo al fine di poterne validare le prestazioni in presenza degli effetti della pianificazione; attualmente è in corso di sviluppo la possibilità di trattare insiemi di attività anche non periodiche, per le quali la (1) non sia significativa se non ipotizzando la sporadicità dei processi aperiodici. Tale lavoro è inserito all'interno

di un progetto più ampio che prevede come prossimo passo lo sviluppo di modelli funzionali dei protocolli di comunicazione su rete, nell'obiettivo di aggiungere ulteriori dettagli implementativi già nelle prime fasi dello sviluppo di sistemi di controllo. L'obiettivo ultimo è il test congiunto con Modelica/Dymola di strategie di controllo automotive [16].

Bibliografia

[1] J. Liu, J. Eker, J.W. Janneck, E.A. Lee, *Realistic simulations of embedded control systems,* 15th Ifac World Congress, Barcelona, Spain, 2002.

[2] E.A. Leel, *Overview of the Ptolemy Project,* Technical Memorandum Ucb/Erl M01/11, University of California, Berkeley, March 6, 2001.

[3] J. Liu, E.A. Lee, *Timed Multitasking for Real-Time Embedded Software,* Ieee Control System Magazine, vol. 22, issue 6, 2002.

[4] M.M. Tiller, *Introduction to physical modeling with Modelica,* Kluwer Academic Publishers, 2002.

[5] C.P. Jobling, P.W. Grant, H.A. Barker, P. Townsend, *Object oriented programming in control system design: a survey,* Automatica, vol. 30, n. 8, pp.1221-1261, 1994.

[6] H. Elmqvist, S.E. Mattson, M. Otter, *Object oriented and hybrid modeling in Modelica,* Journal Europèen des systèmes automatisès, vol. 35, n. 1, pp.1 - 10, 2001.

[7] P. Chiacchio, *Plc e Automazione Industriale,* McGraw-Hill Italia, 1996.

[8] G. Booch, I. Jacobson, J. Rumbaugh, *The unified modeling language user guide,* Addison Wesley Technology Series, 1995.

[9] M. Fowler, K. Scott, *Uml Distilled, Second Edition,* Addison Wesley Technology Series, 1999.

[10] M. Fowler, K. Scott, *Uml Distilled, Second Edition,* Addison Wesley Technology Series, 1999.

[11] B. Selic, *An architectural pattern for real time control software,* Third Pattern Languages of Programming conference, 1996.

[12] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: elements of reusable object-oriented software,* Addison Wesley Longman Inc., 2000.

[13] A.S. Tanenbaum, *Modern Operating Systems*, Prentice Hall, 2001.

[14] J.O. Moody, P.J. Antsaklis, *Supervisory control of discrete event systems using Petri Nets,* Kluwer Academic Publishers, 1998.

[15] G. Buttazzo, *Sistemi realtime per il controllo automatico: problemi e nuove soluzioni,* Automazione e Strumentazione, pp. 107-116, Maggio, 2000.

[16] L. Glielmo, O.R. Natale, S. Santini, *Modellistica e simulazione di veicoli automobilistici con l'ausilio di Modelica,* 46 Convegno Nazionale Anipla, 2002.