Lezione R1

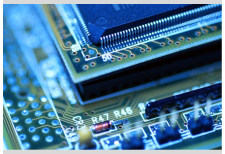
Introduzione ai sistemi real-time

Sistemi embedded e real-time

25 settembre 2020

Marco Cesati

Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università degli Studi di Roma Tor Vergata

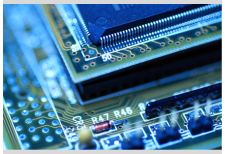


Di cosa parliamo in questa lezione?

processore in ambito RT è un termine 'generalizzato', mentre (solo) quello del PC

Questa è una lezione introduttiva sui sistemi real-time.
In particolare:

1. discuteremo del rapporto tra sistemi embedded e sistemi real-time
2. descriveremo alcuni esempi tipici di sistemi real-time
3. daremo una definizione formale di cosa si intende per sistema real-time e di quali sono i suoi componenti fondamentali
4. discuteremo sulla distinzione tra sistemi soft real-time e sistemi hard real-time



anche aprire Word, ma non è un tempo "fisso", non ha vincoli. I RT sì.

Sistemi real-time

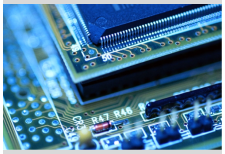
Definizione informale di sistema real-time: è un sistema progettato per operare entro parametri temporali ben definiti

In pratica, un sistema real-time opera correttamente solo se per ogni configurazione degli ingressi (input) viene prodotto la giusta uscita rispettando vincoli temporali ben determinati

↑ fuori tempo => risultato scorretto

Perché in questo corso parliamo sia di sistemi embedded che di sistemi real-time? Spesso, RT => embedded, e viceversa.

1. La maggior parte dei sistemi embedded è anche real-time, e viceversa
2. Nella progettazione di un sistema embedded non si può trascurare quale sistema operativo e applicazione verrà eseguita
3. La tipologia di sistema operativo real-time è un fattore critico da considerare in tutte le applicazioni safety-critical
4. La teoria della schedulazione real-time studia cosa è possibile ottenere avendo a disposizione risorse hardware *limitate*



Sistemi embedded come sistemi real-time

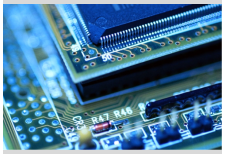
- I sistemi embedded (immersi nell'ambiente) interagiscono in modo significativo con l'ambiente in cui sono "immersi"
- Nella maggior parte dei casi hanno il compito di reagire "rapidamente" a segnali provenienti dall'ambiente

Esempio: consideriamo il sistema embedded che controlla il rilascio degli *air-bag* di una automobile in conseguenza ad una collisione:

↳ si gonfia e resta gonfio per Δt secondi

- Se il sistema embedded rilascia l'air-bag in ritardo, l'essere umano subisce l'urto della collisione, poi viene colpito con violenza dal cuscino in espansione
- Se il sistema embedded rilascia l'air-bag in anticipo, l'essere umano è ancora lontano dal cuscino ed il suo movimento non viene frenato; poiché il cuscino si affloscia subito, l'essere umano subisce l'urto della collisione

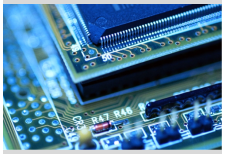
Il sistema embedded che controlla l'air-bag deve operare con vincoli temporali estremamente rigorosi, altrimenti l'air-bag è non solo inutile ma anche pericoloso



Organizzazione del software nei sistemi embedded

- La progettazione dei sistemi embedded deve tenere in considerazione aspetti quali la rapidità di sviluppo, l'economia di scala, la manutenibilità, ecc.
- Di conseguenza, non è possibile sviluppare l'hardware senza considerare anche il progetto del software
- Ad esempio, in un sistema embedded di fascia alta e con caratteristiche real-time si deve stabilire fin da subito quale sistema operativo real-time dovrà essere utilizzato
- Anche nei sistemi embedded in cui non è utile inserire un sistema operativo completo è necessario stabilire fin dall'inizio l'organizzazione del software di controllo
- Se il sistema embedded è safety-critical, la scelta dell'organizzazione del software svolge un ruolo cruciale nella possibilità di certificare il sistema per l'uso a cui è destinato

↑ bollino / certificazione sulla qualità

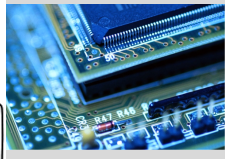


Teoria della schedulazione real-time

- Informalmente, molti risultati teorici della teoria della schedulazione real-time assicurano il rispetto dei vincoli temporali di un insieme di processi se l'utilizzo totale del dispositivo di calcolo è inferiore ad una certa soglia
- Dato un sistema real-time che deve assicurare l'esecuzione nei tempi previsti di un insieme di funzioni, aumentare le risorse di calcolo a disposizione in genere facilita l'effettivo rispetto dei vincoli temporali
- Lo sviluppo tecnologico delle piattaforme di calcolo ad uso generale continua ad accrescere la potenza di calcolo disponibile per il progettista del sistema
- Anche i sistemi embedded hanno un rapido sviluppo tecnologico, ma altri aspetti oltre alla potenza di calcolo possono predominare (consumo, costo, dimensione, . . .)

es: sistema carico al 40%, o potenza CPU se general purpose

Nei sistemi embedded la potenza di calcolo è generalmente ridotta, perciò il problema di cosa è possibile garantire rispettando predeterminati vincoli temporali è fondamentale

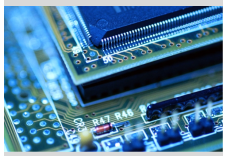


→ potenziare è limitato, oggi non riesco a superare una soglia. Negli embedded e RT ho limiti precisi (batteria, calore ...)

Una brutta storia

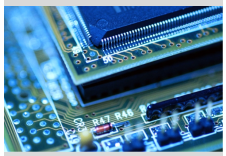
- Durante la prima Guerra del Golfo, il 25 febbraio 1991, un missile iracheno Scud è lanciato verso l'Arabia Saudita
- Il radar di un sistema missile anti-missile Patriot rileva la minaccia
- Il computer integrato nel sistema Patriot calcola la traiettoria e deduce la posizione futura dell'oggetto in un certo istante futuro
- In quell'istante però il radar non rileva il missile nella posizione corrispondente alla traiettoria calcolata, quindi il sistema classifica il segnale come un "falso allarme"
- Qualche minuto più tardi il missile colpisce la città di Dhahran causando 28 vittime e danni ingenti

Quale è stato il motivo del fallimento del sistema Patriot?



Introduzione ai sistemi real-time

Marco Cesati



Schema della lezione

Sistemi embedded come RT

Un caso reale

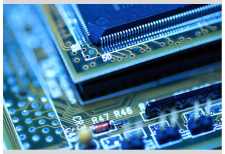
Tipiche applicazioni RT

Definizione di real-time

Hard e soft real-time

Una brutta storia (2)

- Il motivo dell'incidente: un bug nel sistema operativo del computer del Patriot causava un ritardo sistematico nell'aggiornamento dell'orologio di sistema (57 microsecondi al minuto)
- Il giorno dell'incidente, l'ultimo reboot del sistema Patriot era stato fatto circa 100 ore prima
- Di conseguenza, l'orologio di sistema aveva accumulato un ritardo di 343 millisecondi rispetto all'orologio del radar
- Alla velocità di crociera del missile Scud, il ritardo dell'orologio ha comportato un errore nel predire la posizione del missile di 687 metri
- Per la cronaca, gli israeliani avevano scoperto il bug l'11 febbraio, ma:
  - il 25 febbraio non era ancora disponibile un aggiornamento del software
  - Ai comandanti delle batterie di Patriot era stato detto di "fare il reboot", senza indicare però con quale frequenza...



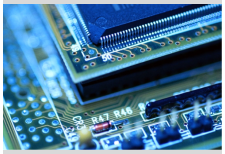
Una brutta storia (3)

Perché i test del sistema Patriot non hanno scoperto il bug?

Perché il sistema non è stato mai provato nella specifica condizione in cui si è verificato l'incidente (funzionamento ininterrotto da oltre 100 ore)

- Molto spesso, risorse preziose e vite umane sono affidate al corretto funzionamento di sistemi informatici
- In molti di questi casi uno dei fattori cruciali è il *tempo*: parliamo di sistemi real-time
- Non possiamo affidarci esclusivamente ai test di funzionamento per escludere la presenza di gravi difetti in questi sistemi informatici

Quando affidiamo le nostre vite ad un sistema, vogliamo essere certi che il sistema è progettato e implementato correttamente



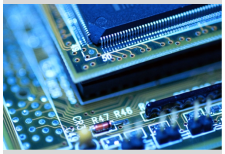
Cosa sono i sistemi real-time?

I sistemi real-time sono sistemi progettati per operare entro parametri temporali ben definiti

Tale definizione è poco rigorosa: più avanti in questa lezione cercheremo di fornire una definizione migliore

È utile comunque avere innanzi tutto una idea di quali siano gli ambiti applicativi principali in cui vengono utilizzati i sistemi real-time

- Sistemi di controllo digitale
- Sistemi di controllo ad alto livello
- Processamento di segnali
- Basi di dati temporali
- Applicazioni multimediali



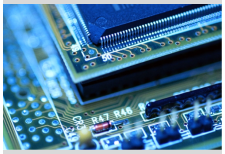
Sistemi di controllo digitale

Un gran numero di sistemi real-time sono integrati in sensori ed attuatori e operano come sistemi di controllo digitale

Tipicamente, un sistema di controllo digitale esegue un ciclo senza fine in cui:

- Legge i dati forniti da alcuni sensori, generalmente convertendo i segnali dal formato analogico a quello digitale
- Confronta lo stato del sistema sotto esame confrontandolo con uno stato obiettivo da raggiungere o mantenere e, tramite una opportuna legge di controllo, determina i segnali di uscita (output)
- Converte l'output in segnali da fornire ad opportuni attuatori per manipolare lo stato del sistema, generalmente convertendo dal formato digitale a quello analogico

Questo meccanismo è noto come *feedback control loop*



Sistemi di controllo digitale (2)

È possibile distinguere tra:

- Sistemi a frequenza di campionamento unica: le iterazioni del feedback control loop si ripetono con frequenza costante
- Sistemi multi-frequenza: differenti parti del sistema sono analizzate a frequenze diverse, spesso armoniche tra loro

In generale i sistemi con feedback control loop funzionano bene se:

1. I dati forniti dai sensori sono ragionevolmente accurati
2. I dati forniti dai sensori forniscono un quadro completo dello stato del sistema
3. La dinamica del sistema è conosciuta con sufficiente approssimazione

tutto con tempistiche ben precise

Se queste ipotesi non sono pienamente verificate, i sistemi di controllo digitale devono implementare logiche di controllo molto più complesse

come rimane in volo?

Esempio: schema di un "flight controller" per elicottero

In ogni iterazione di un ciclo ripetuta ogni 1/180 secondo vengono eseguiti i seguenti task: (o job)

T_1) Valida i dati dei sensori per selezionare i dati da acquisire; in caso di guasti, riconfigura il sistema ∀ $\frac{1}{180}$ s

T_2) Esegui una delle seguenti funzioni "avioniche" (ciascuna con frequenza 30 Hz):

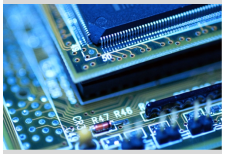
- campiona i controlli del pilota
- esegui normalizzazione di dati e trasformazione di coordinate
- aggiorna la posizione del velivolo

3 jobs

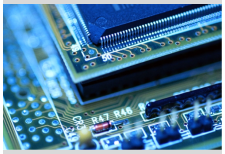
T_3) In alternativa a T_2, esegui una delle seguenti funzioni "di controllo" (ciascuna con frequenza 30 Hz): troppo dilatati nel tempo, non bastano per controllare

- calcolo della legge di controllo "esterna" per pitch
- calcolo della legge di controllo "esterna" per roll
- calcolo della legge di controllo "esterna" per yaw e movimento

("esatte")



Introduzione ai sistemi real-time
Marco Cesati



Schema della lezione
Sistemi embedded come RT
Un caso reale
Tipiche applicazioni RT
Definizione di real-time
Hard e soft real-time

Esempio: schema di un “flight controller” per elicottero (2)

T_4) Esegui una delle seguenti funzioni “di controllo” (ciascuna con freq. 90 Hz), utilizzando i risultati di T_2 e T_3:

- calcolo della legge di controllo “interna” per pitch
- calcolo delle leggi di controllo “interne” per roll e per il movimento

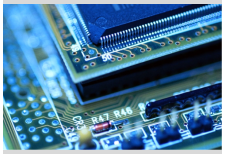
T_5) Esegue il calcolo della legge di controllo “interna” per yaw usando i risultati prodotti da T_4 (più complessa, controllo posizione elicottero)

T_6) Fornisce i segnali di output agli attuatori

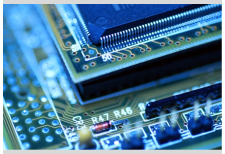
T_7) Esegue test interni di consistenza

Perché esistono leggi di controllo “interne” ed “esterne”?

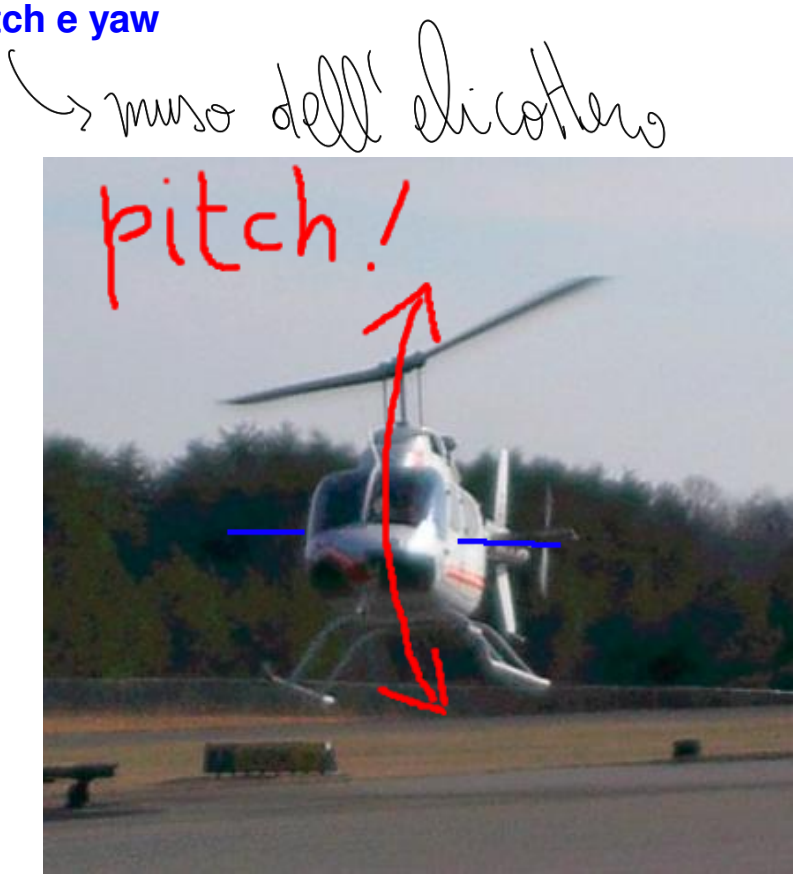
Le leggi di controllo “interne” sono computazionalmente leggere ma danno risultati approssimati, quelle “esterne” danno risultati esatti ma sono computazionalmente più pesanti

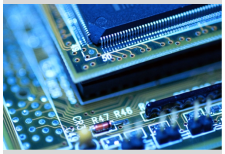


Roll, pitch e yaw

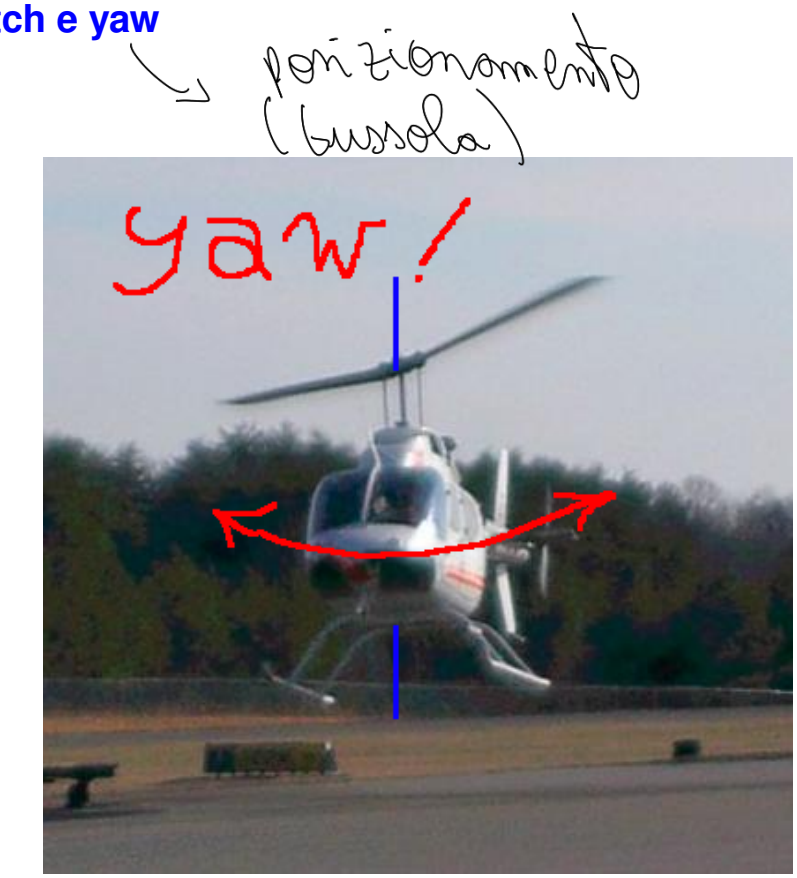


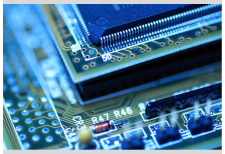
Roll, pitch e yaw





Roll, pitch e yaw





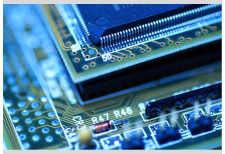
Esempio: schema di un “flight controller” per elicottero (3)

Job	Freq.	I_1	I_2	I_3	I_4	I_5	I_6
Validazione sensori	180 Hz	√	√	√	√	√	√
Controlli pilota	30 Hz	√					
Elaborazione dati	30 Hz		√				
Aggiornam. posizione	30 Hz			√			
Pitch esterno	30 Hz				√		
Roll esterno	30 Hz					√	
Yaw+movim. esterno	30 Hz						√
Pitch interno	90 Hz	√		√		√	
Roll+movim. interno	90 Hz		√		√		√
Yaw interno	180 Hz	√	√	√	√	√	√
Output attuatori	180 Hz	√	√	√	√	√	√
Self-test	180 Hz	√	√	√	√	√	√

Solo un intervallo

- Ogni “job” ha una propria frequenza di esecuzione
- All’interno di ciascuna iterazione vengono eseguiti esattamente sei “job”, processore tiene "il passo"

• NON è flessibile, ma funziona, non è "modificabile".



Sistemi di controllo ad alto livello

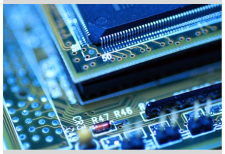
In genere i sistemi complessi sono controllati da *gerarchie di sistemi di controllo*

Ad esempio, in una sala di terapia intensiva di un ospedale potremmo trovare, procedendo dal basso verso l’alto:

- Sistemi di controllo digitale per l’impianto di controllo del battito cardiaco, della respirazione, e degli altri parametri vitali di ciascun paziente
- Sistemi di controllo digitale per gli impianti che somministrano ossigeno, medicinali, ecc.
- Un sistema di controllo generale per ciascun letto, che fornisce il quadro clinico di ciascun paziente
- Un sistema di controllo dell’intera sala che attiva allarmi per il personale sanitario nel caso il quadro clinico di un qualunque paziente cambi bruscamente

devono reagire e finire entro un tempo fisso (siano h, m, s ...)

I sistemi di controllo di livello più basso sono sistemi embedded mentre quelli di livello più alto sono computer general-purpose



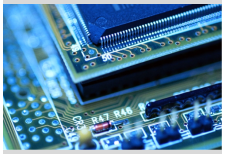
Processamento di segnali

La maggior parte delle applicazioni che processano segnali hanno requisiti temporali più o meno stringenti: l'elaborazione deve avvenire entro tempi prestabiliti (da qualche millisecondo a qualche secondo), altrimenti non è più di alcuna utilità

Ad esempio, si consideri un sistema radar:

- Una antenna viene puntata in una certa direzione ed emette una breve impulso elettromagnetico
- L'antenna viene posta in ricezione ed ascolta eventuali segnali di eco riflessi da un ostacolo a distanza
- Il segnale ricevuto viene elaborato per filtrare il rumore d'ambiente e misurare il ritardo dell'eco (distanza) e la sua frequenza (spostamento Doppler ossia velocità relativa)
- Il dato ottenuto viene confrontato con quelli ottenuti in precedenza in modo da associare l'oggetto come un nuovo rilevamento, oppure un rilevamento precedente in movimento lungo una data traiettoria

Tutto ciò deve essere compiuto in tempo reale prima di muovere l'antenna e ricominciare con un altro settore di spazio



Basi di dati temporali

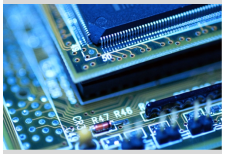
dati non in tempo reale possono essere inutili!

Con il termine "basi di dati temporali" si fa riferimento a varie tipologie di sistemi informativi in cui la valenza ed il significato dei dati memorizzati è direttamente correlato al fattore "tempo"

Mentre in una base di dati tradizionale un dato non aggiornato rimane valido indefinitivamente, in una base di dati temporale un dato non aggiornato o non fornito in tempo utile non ha più valore o significato

Tipici esempi:

- Controllo del traffico aereo
- Controllo di missione di un velivolo
- Controllo di un satellite
- Controllo del mercato finanziario (1 ora è troppo)



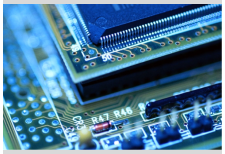
Applicazioni multimediali

Una classe di sistemi real-time molto vasta è quella delle applicazioni multimediali che processano flussi audio/video

I flussi senza compressione sono enormi ed ingestibili, perciò le applicazioni devono comprimere e decomprimere i flussi secondo standard sofisticati

L'elaborazione di ciascun frame deve avvenire in tempo utile!

Ad esempio, lo spettatore nota con fastidio uno sfasamento tra i movimenti della bocca e la voce maggiore di 160 millisecondi



Definizione di sistema real-time

Un *sistema* è una associazione (mapping) tra un insieme di ingressi (input) ed un insieme di uscite (output)

In informatica generalmente gli ingressi sono costituiti dai dati forniti ad un insieme di programmi per calcolatore, e le uscite sono i risultati prodotti da tali programmi

La correttezza logica di una applicazione informatica è definita come la capacità di produrre in modo deterministico le uscite corrette in funzione di ogni possibile configurazione di ingressi

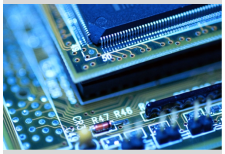
Un *sistema real-time* è un sistema la cui correttezza logica dipende non solo dal risultato fornito come uscita ma anche dall'istante temporale in cui tale risultato è reso disponibile

- *Applicazione real-time*: un programma (od un insieme di programmi) aventi vincoli temporali ben definiti (≠ prima possibile)
- *Sistema real-time*: l'insieme di dispositivi hardware e software che rendono possibile la corretta esecuzione di una applicazione real-time

se in anticipo: aspetto

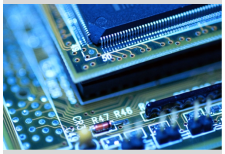
se in ritardo: è da gestire

deve essere un limite preciso



Tassonomia dei sistemi real-time

1. Puramente ciclici: ogni task nel sistema viene eseguito periodicamente; non esistono variazioni significative nell'uso di risorse nel tempo. La maggior parte dei sistemi di controllo digitali appartiene a questa categoria. (es: elicottero)
2. Perlopiù ciclici: la maggior parte dei task nel sistema viene eseguita periodicamente, ma esistono eventi esterni come guasti o comandi da operatore da gestire quando necessario. Esempio: sistemi avionici. (decollo, atterro, viro)
3. Asincroni ma grosso modo predicibili: La maggior parte dei task non sono periodici, ma le loro frequenze di esecuzione ricadono entro limiti prefissati o opportune statistiche. Fanno parte di questa categoria le applicazioni multimediali e l'elaborazione dei segnali radar. (spesso trova nulla, o stesse cose) se entro area posso grossomodo prevederlo!
4. Asincroni e impredicibili: Esistono casi di applicazioni che devono reagire ad eventi asincroni tramite task molto onerosi dal punto di vista computazionale. Un esempio è un sistema esperto che debba analizzare uno scenario e prendere una decisione in conseguenza di un evento straordinario. controllo sicurezza, non so quando, ma quando serve deve operare con vincoli (airbag, centrale nucleare).



Job e task

Un *job* è una unità di lavoro che può essere schedulata ed eseguita da un sistema real-time

Esempi di job: (dipende dal contesto)

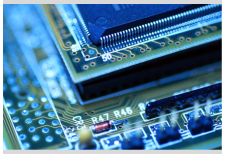
- un processo eseguito da una CPU
- la spedizione di un messaggio tramite un canale di comunicazione (in un certo tempo)
- la lettura di un file da un dispositivo di memoria di massa

Un *task* è un insieme di job tra loro correlati che insieme realizzano una determinata funzione del sistema

Esempio di task: (composto da 3 job separati ma correlati)

task ⊇
- job che acquisisce valori da alcuni sensori
- job che converte tali valori in formato appropriato
- job che aggiorna con tali valori opportune strutture di dati

Introduzione ai sistemi real-time
Marco Cesati



Schema della lezione
Sistemi embedded come RT
Un caso reale
Tipiche applicazioni RT
Definizione di real-time
Hard e soft real-time

Processori e risorse

Un *processore* è una componente attiva del sistema real-time in grado di "eseguire" un job

Esempi di processori: NON è solo CPU

- una CPU (i job sono i processi eseguiti)
- un canale di rete (i job sono i messaggi spediti)
- un disco rigido (i job sono i file acceduti)

Una *risorsa* è una componente passiva del sistema real-time la cui disponibilità è necessaria per eseguire alcuni job (es: RAM)

*Qual è la reale differenza tra **processori** e **risorse**?*

- Un processore è dotato di una intrinseca *velocità*: più è rapido, maggiore è il numero di job che può eseguire nello stesso tempo, incide sulle tempistiche, ATTIVA, se cambia, tempi job cambiano.
- Una risorsa non può incidere sulla velocità d'esecuzione dei job: un esempio di risorsa è un semaforo che protegge una struttura di dati condivisa, non incide sulle tempistiche

Noi vogliamo "astrarre", non pensiamo a RAM ddr2 o ddr5
Se tempi risposta piccoli => serve RAM veloce => è processore, no risorsa.

Vincoli temporali dei job

L'*istante di rilascio* (*release time*) di un job è l'istante di tempo in cui il job diventa disponibile per l'esecuzione

L'istante di rilascio di un job non coincide necessariamente con l'istante in cui esso inizia ad essere eseguito!

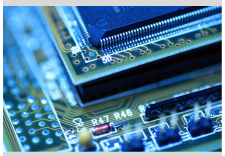
La *scadenza* (*deadline*) di un job è l'istante temporale entro cui il job deve aver completato l'esecuzione

Spesso è conveniente ragionare in termini di *tempo di risposta* di un job, ovvero l'intervallo di tempo trascorso tra il suo release time e l'istante in cui esso completa l'esecuzione

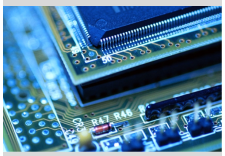
La *scadenza relativa* (*relative deadline*) di un job è il massimo tempo di risposta ammesso per esso

scadenza (assoluta) = istante di rilascio + scadenza relativa

deadline ↑



Introduzione ai sistemi real-time
Marco Cesati



Schema della lezione
Sistemi embedded come RT
Un caso reale
Tipiche applicazioni RT
Definizione di real-time
Hard e soft real-time

Sistemi hard e soft real-time

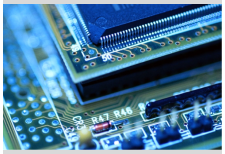
Si fa spesso riferimento in letteratura ad una distinzione tra applicazioni e sistemi hard real-time e soft real-time

Qual è la differenza tra i due termini?

In pratica, non esiste una definizione universale per il significato di questi termini

Esistono invece molte definizioni differenti e contraddittorie, la maggior parte basate alternativamente su:

- quanto è tollerabile violare i vincoli temporali di un job
- quanto è utile completare il job anche dopo la scadenza imposta
- quanto è statisticamente frequente la violazione dei vincoli temporali di un job



Tollerabilità della violazione dei vincoli temporali

Una diffusa definizione di applicazione hard real-time è quella per cui un fallimento nel rispettare i vincoli temporali di qualche job è un "guasto fatale" dagli effetti potenzialmente disastrosi

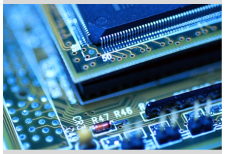
L'applicazione che controlla i freni di un treno è hard real-time:

- il conducente aziona il comando del freno perché il treno incontra un semaforo rosso: deve essere eseguito un job che arresta il treno entro un certo tempo
- violare il vincolo temporale di questo job è un guasto fatale: il treno potrebbe non fermarsi prima di arrivare ad una tratta di binario già occupata
- il mancato rispetto del vincolo temporale è assimilabile al guasto meccanico dell'impianto frenante del treno

Nei casi in cui la violazione di un vincolo temporale è indesiderabile ma non è assimilabile ad un guasto fatale si parla di applicazione soft real-time

Qual è il limite di questa definizione? NON OGGETTIVA!

Quanto grave deve essere un guasto per definirsi "fatale"?
Ciascuno ha una propria opinione sulla gravità dei fallimenti



Utilità del job in ritardo

In base a questa definizione, la distinzione tra hard real-time e soft real-time dipende quantitativamente dall'*utilità* del risultato prodotto dal job in funzione del suo ritardo

Si definisce *tardività* (*tardiness*) di un job il ritardo con cui esso completa l'esecuzione rispetto alla sua scadenza

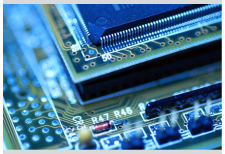
L'utilità del risultato di un job soft real-time decresce gradualmente in funzione della sua tardività, mentre l'utilità del risultato di un job hard real-time si abbatte bruscamente non appena la tardività cresce e può addirittura diventare negativa

Cosa si intende per "utilità negativa"?

Significa che completare il job dopo la sua scadenza è controproducente (esempio: job di sgancio di una bomba)

Qual è il limite di questa definizione?

Anche in questa definizione l'assegnazione della funzione di utilità ai job è arbitraria e frutto di opinioni personali



Vincoli temporali deterministici o probabilistici

In base a questo tipo di definizione, se un job non deve mai violare i propri vincoli temporali, si parla di hard real-time

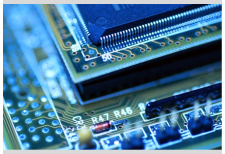
Viceversa, se un job può occasionalmente violare i vincoli temporali, ma la probabilità che ciò accada è accettabilmente bassa, si parla di soft real-time

Ad esempio, le specifiche di un certo job soft real-time potrebbero richiedere che esso completi la sua esecuzione entro un minuto il 99.999% delle volte, ossia con probabilità di superare ciascuna deadline inferiore a 10^{-5}

Qual è il limite di questa definizione?

Benché si asserisca che nel caso hard real-time un vincolo temporale non deve mai essere violato, in pratica ciò non può essere garantito in modo deterministico: esiste sempre una probabilità, per quanto piccola, che il sistema si guasti

Poiché non si considerano le conseguenze dei fallimenti, tutti i sistemi potrebbero essere definiti soft real-time

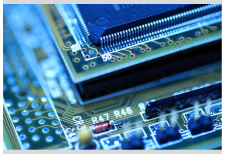


Definizione operativa di hard e soft real-time

Il vincolo temporale di un job è *hard real-time*, e di conseguenza anche il job ed il sistema che lo include lo sono, se è richiesta una validazione che i vincoli temporali sono sempre soddisfatti

La *validazione* è una dimostrazione formale, svolta tramite una procedura efficiente e *dimostrabilmente* corretta, oppure è una evidenza sperimentale prodotta da adeguate ed esaustive attività di simulazione e test

Il vincolo temporale di un job è soft real-time se non è richiesta una procedura di validazione, oppure se è sufficiente una dimostrazione che il job non fallisce entro precisi limiti statistici



Definizione operativa di hard e soft real-time (2)

Dal punto di vista di un libro di teoria la "definizione operativa" è adeguata, per altri versi è anch'essa insoddisfacente

"*Il sistema è hard real-time se deve essere validato. . . ma in quali condizioni il sistema deve essere validato?*"

Per il progettista la "definizione operativa" può essere inutile!

Concettualmente, in fase di sviluppo di un sistema real-time:

- si effettua una analisi dei rischi (ambiente di esecuzione, safety e security, implicazioni in caso di guasto, . . .)
- la classificazione dei rischi individuati diventa il modello di riferimento per valutare significato e gravità dei fallimenti
- i meccanismi di riduzione dei rischi portano alla definizione di vincoli temporali che il sistema deve soddisfare
- il rispetto dei vincoli temporali è quindi la condizione per minimizzare i rischi, e tale rispetto è tanto più importante quanto maggiore è la gravità del rischio associato

Non esiste una metrica universale di "gravità del rischio", ma esiste una metrica particolare per ogni singolo progetto