1 Introdução aos sistemas de tempo real

1.1 Definição de sistema de tempo real

Sistemas de tempo real são sistemas que envolvem um ou mais computadores, nos quais a correcção do sistema depende não só dos **resultados da computação, mas também do instante de tempo em que são produzidos os resultados**.

Pelo contrário, em sistemas de tempo não real, os *sistemas batch*, (por vezes referidos também como sistemas de tempo partilhado) **não é importante o instante no qual os resultados da computação estão disponíveis, mas sim apenas a sua correcção**.

Como mostra a figura seguinte, o **computador** é, um elemento fundamental de um sistema de tempo real, mas está normalmente inserido num sistema maior, envolvendo **sensores** e **actuadores** que recebem e fornecem informação de e para o ambiente.

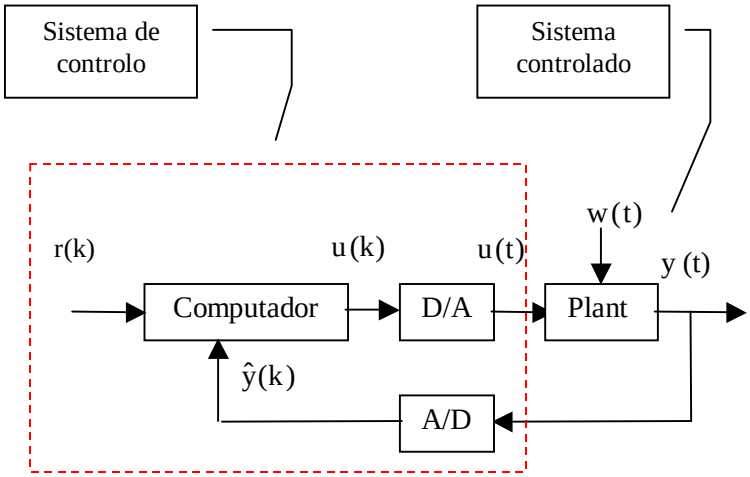


Fig. 1-1: Sistema de controlo digital típico (Sistema de tempo real)

r(k) – sinal de referencia (digital)
u(k) – sinal de controlo (digital)
u(t) – sinal de controlo (contínuo)
w(t) – sinal de perturbação (contínuo)
y(t) – sinal de saída (contínuo)
$\hat{y}(k)$ - sinal de saída (amostrado)

Como mostra a figura acima, um sistema de tempo real consiste de 2 grandes blocos: o **sistema de controlo**, e o **sistema controlado**. De notar que o sistema controlado é parte integrante do sistema de tempo real. Por exemplo, numa fábrica automatizada, o sistema de controlo consiste no(s) computador(es) e interfaces homem-máquina que gerem e coordenam as actividades na fábrica. O sistema controlado consiste nos robots, linhas de montagem e as várias peças.

Exemplos de sistemas de tempo real familiares são: o **microcomputador que controla o motor dos carros** modernos, **o sistema de controlo de processos nas refinarias de gasolina**, os sistemas de controlo de voo num avião (ex.: simulink f14), o sistema de reserva de bilhetes numa agência de turismo e o sistema de pagamento multibanco.

Um concepção errada normalmente associada com sistemas de tempo real é a de que os sistemas de tempo real devem ser apenas rápidos. **Mas o que caracteriza essencialmente um sistema de tempo real é a de que ele deve cumprir metas temporais (*deadlines*) explícitas.** Se se conseguir provar que um sistema de tempo real cumpre as suas deadlines (usando uma análise de pior caso, e não uma análise de média), então diz-se que o sistema é *previsível*. Neste contexto, previsibilidade significa que, quando uma tarefa ou um conjunto de tarefas é activada, é possível determinar quando essa tarefa ou grupo de tarefas terminam, dentro das metas temporais associadas.

Na figura anterior o computador gera um sinal de **controlo digital,** u(k), que antes de ser aplicado ao processo ou sistema a controlar *(plant)*, tem que ser convertido para analógico, u(t), através de um conversor digital-para-analógico (A/D). Por outro lado, o sinal de saída contínuo, y(t), deve ser amostrado, antes de ser lido pelo computador, pelo conversor analógico-para-digital (A/D).

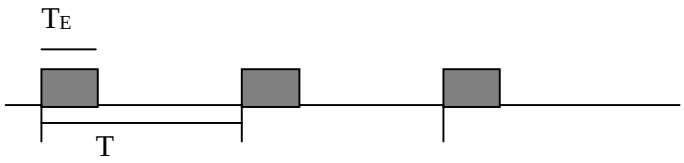


Fig. 1-2: Diagrama temporal para uma única tarefa síncrona

Vamos admitir que, de T em T segundos, o computador lê o valor da saída do processo, executa um algoritmo de controlo, aplicando assim o sinal de controlo ao **processo** (*plant*) T_E segundos depois. Se esta fôr apenas a única tarefa desempenhada pelo computador, temos então o diagrama temporal, acima.

Consideremos agora que neste sistema existe uma entrada de um alarme, possivelmente relacionado com um valor demasiado elevado da perturbação. Poderíamos então ter o seguinte diagrama temporal:

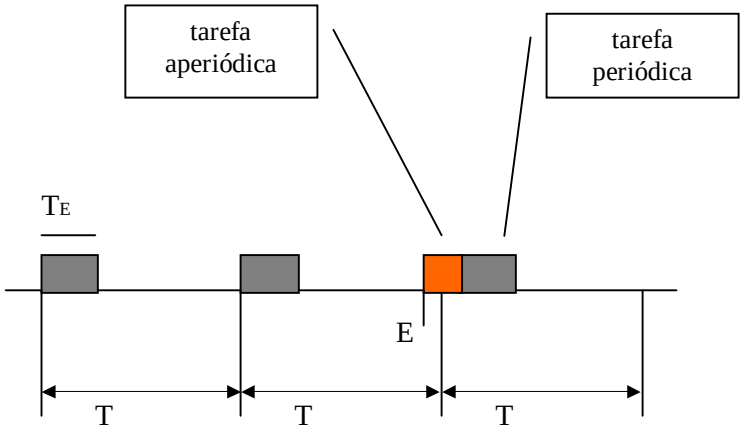


Fig. 1-3: Diagrama temporal para uma tarefa sincrona e outra assíncrona

Repare que, a ocorrência do evento **assíncrono** E, imediatamente antes de ser iniciada a tarefa **síncrona**, irá implicar (caso esta última seja de menor prioridade do que a tarefa assíncrona), um atraso na inicialização da tarefa **síncrona** e consequentemente também na produção do sinal de controlo.

Assim, um sistema de tempo real pode ser classificado quanto à **periodocidade**, dos estímulos a que deve responder, e consequentemente, à execução periódica ou não dos processos ou tarefas associadas. Estes podem então ser de dois tipos:

- **aperiódicos**, ou *inicializados por eventos (event-driven* ou *interrupt driven*) - processos que são desencadeados por ocorrência de acontecimentos externos assíncronos. Exemplo: alarmes

- **periódicos**, ou *inicializados por tempo* (*time-triggered*) - processos que são executados ciclicamente. Normalmente o processamento de sensores, ou a execução de um loop de controlo, são actividades periódicas.

Num mesmo sistema de tempo real, podem coexistir os dois tipos de processos acima, situação essa onde é mais difícil de garantir previsibilidade.

Os processos podem ainda ser:

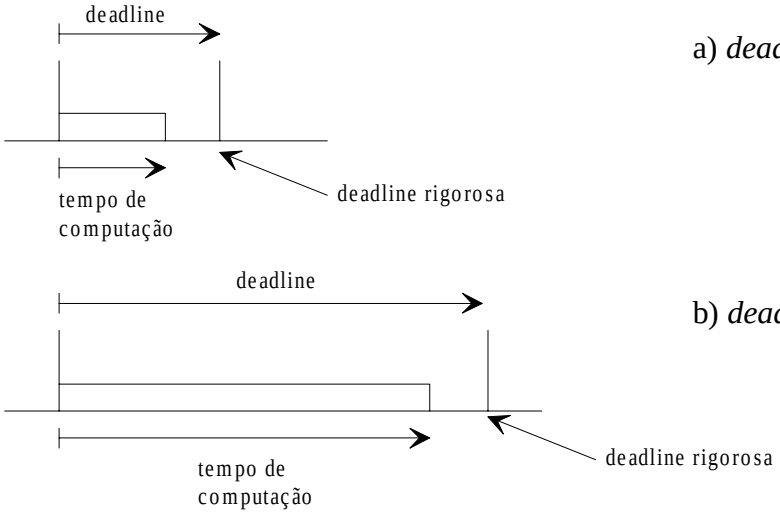
- **permanentes** - existem desde que o sistema foi inicializado. É o caso da maioria dos processos periódicos.
- **transitórios**, *ou criados dinamicamente* - Um avião entra no espaço aéreo de um computador de controlo de tráfego. É então criado um processo que periódicamente segue o voo até que o avião deixe o espaço aéreo.

1.2 Caracterização de sistemas de tempo real

1.2.1 Metas temporais *(deadlines)*

As deadlines podem ser classificadas por dois prismas:

a) Granularidade



a) *deadline de* granularidade fina

b) *deadline de* granularidade *coarse*

Fig. 1-4: Granularidade das deadlines

Num sistema de tempo real, algumas das tarefas têm metas temporais associadas. Se o intervalo de tempo que decorre desde que uma tarefa é completada até ser novamente activada (i.e., que deve começar a ser executada) é curto, então existe uma *meta temporal rigorosa (tight)*. A sua granularidade é fina se o tempo de computação é pequeno e grosseira (*coarse*) se é grande.

As *deadlines* rigorosas implicam que os projectistas devem desenvolver técnicas simples e rápidas para reagir a este tipo de activação de tarefas. De um modo geral, quanto mais apertada for a *deadline*, mais difícil será a tarefa de projecto.

b) Rigidez

A questão que se põe aqui é se vale ou não a pena executar uma tarefa se a sua meta fôr ultrapassada. No caso de tarefas de tempo real *hard*, então não vale a pena. No caso de tarefas de tempo real *soft*, elas devem ser executadas.

Exemplos:

Hard - O detector de proximidade de veículo automático detectou um objecto na sua trajectória, mas este acontecimento não foi processado no intervalo de tempo de ½ segundo. Infelizmente esse objecto era uma pessoa!

Soft - Um controlador de processos não conseguiu cumprir a sua meta temporal no intervalo de amostragem corrente. Provavelmente é aceitável esperar pela próxima amostra, devendo porém ser garantido que a próxima amostra será processada. Num sistema automático de pagamento, o tempo médio de atendimento é o factor temporal mais importante. Se uma determinada transacção demora muito tempo, continua a ser preferível continuar com a transacção do que interrompê-la.

Diferentes técnicas são usadas para tarefas **hard** e soft. Na maioria dos casos, as tarefas hard são **pré-alocadas** e pre-escalonadas (i.e., é determinada a altura de serem executadas antes de elas serem necessárias) para assegurar que cumprem as suas metas. As tarefas **soft** são muitas vezes escalonadas com **algoritmos de escalonamento (*scheduling*) não de tempo real**, onde a preocupação se centra no tempo de resposta médio.

SOFT	**HARD**
• Pouca previsibilidade - o escalonamento dinâmico de tarefas pode significar que algumas tarefas demoram mais do que o desejado • O desempenho deteriora-se em situações de pico de trabalho • Uso eficiente de recursos	• Boa previsibilidade - assegura-se que todas as metas temporais são cumpridas • desempenho uniforme independente do número de solicitações • Uso ineficiente de recursos

1.2.2 Manipulação de dados reais

Em alguns casos, o computador pode substituir, um controlador analógico. A sua implementação, pode ser efectuada recorrendo a equações diferença (o equivalente discreto de equações diferenciais) utilizando os dados de entrada, saída e de estado do sistema. A resolução das equações diferença consiste, em última análise, em somas e multiplicações de dados reais, normalmente em vírgula flutuante.

1.2.3 Fiabilidade

Muitos sistemas de tempo real operam em condições de fiabilidade extrema - por exemplo, o sistema de controlo de um reactor atómico, instrumentação médica, etc. Um factor que destingue um sistema de tempo real de um sistema não de tempo real é o facto que as consequências de uma falha são normalmente mais drásticas, dado os sistemas de tempo real interactuarem com o ambiente. Uma outra característica destes sistemas é a de que, no caso de uma ocorrência de uma falha, o sistema a detecte (detecção de falhas) e recupere dela (**sistema tolerante a falhas**). No caso de tal ser impossível o sistema deve acabar a sua execução de uma maneira controlada (**gracefull degradation**).
Em sistemas de segurança crítica (**critical safety systems**), alguns processos, denominados *processos críticos*, devem forçosamente cumprir as suas *deadlines*, senão alguma catástrofe pode ocorrer.

Existem algumas técnicas usadas para assegurar que os processos críticos asseguram as suas metas usando:

- Análise off-line - neste caso temos os métodos formais de especificação de software, que provam matematicamente a correcção de um programa através da sua especificação, usando teoria de conjuntos e lógica de predicados de 1ª ordem.
- Esquemas que reservam os recursos necessários aos processos críticos.
- Redundância em hardware e software.

Uma nota importante é a de que nem todos as tarefas *hard* são tarefas críticas, e, consequentemente, nem todos os sistemas de tempo real são sistemas críticos.

1.2.4 Tamanho e complexidade do sistema

De um modo geral, os sistemas de tempo real são sistemas complexos, dado interactuarem com o ambiente. Por este motivo, têm de prever as diferentes mudanças existentes ou passíveis de existir nesse ambiente, o que complica o projecto inicial do sistema, e irá implicar manutenção frequente e actualizações ao longo da vida do sistema.
Obviamente, os sistemas de tempo real variam em tamanho e complexidade. Em sistemas pequenos, todo o código reside normalmente em memória, ou, caso haja fases bem definidas, cada módulo de código é trazido para a memória antes de ser efectivamente necessário. (Pois o uso de memória virtual implica automaticamente imprevisibilidade.)
Em grandes sistemas, por vezes, não é possível todo o código residir em memória central. Então é necessário assegurar que pelo menos as tarefas *hard* residam permanentemente em memória central.

1.2.5 Coordenação de tarefas

Um sistema de tempo real normalmente desempenha várias funções, que para o utilizador parecem ser executadas simultaneamente. É o caso, por exemplo, de ler o relógio de tempo real, amostrar os dados dos sensores, calcular algoritmos de controlo, atender o operador, actualizar o monitor de saída, armazenar dados, etc. Em termos de software, a cada uma dessas tarefas está normalmente associado um processo, que executa concorrentemente com os outros processos, caso se trate de um sistema monoprocessador, ou em paralelo, no caso de um sistema multiprocessador ou distribuído.
O suporte para a coordenação dos diferentes processos é deixado ao sistema operativo, ou, mais recentemente, à linguagem que se utiliza para programar o sistema.

1.2.6 Interacção com o ambiente

Por definição, um sistema de tempo real responde a estímulos externos, e, normalmente, com base nestes, vai interactuar o ambiente. Assim, interfaces de entrada e saída de dados, normalmente de vários tipos, são sempre parte integrante de um sistema de tempo real.