Sistemas Embarcados (embutidos)

Paulo C. Masiero

Caracterização

- São usados para controlar sistemas de diferentes tipos: máquinas domésticas, fábricas, carros, jogos etc.
- O software é embutido no hardware do sistema e com ele interage
- O tempo de resposta o diferencia de outros sistemas.
- É tipo de sistema mais comum, com grande impacto econômico.

Sistema de Tempo real

- O funcionamento depende dos resultados produzidos pelo sistema e do tempo em que esses resultados são produzidos
- STR Leve: se a operação for degradada caso os resultados não sejam produzidos de acordo com os requisitos de tempo/desempenho.
- STR Rígido: se a operação for incorreta caso os resultados não forem produzidos de acordo com os requisitos de tempo/desempenho.

Geralmente:

- O software que executa em um computador e controla outras máquinas é um sistema embarcado de tempo real.
- Recebe eventos (sinais) gerados pelo hardware e emite sinais de controle para o hardware em resposta a esses eventos.
- A resposta pode estar condicionada a restrições de tempo.
- Nem todo sistema embarcado é de tempo real e vice-versa.

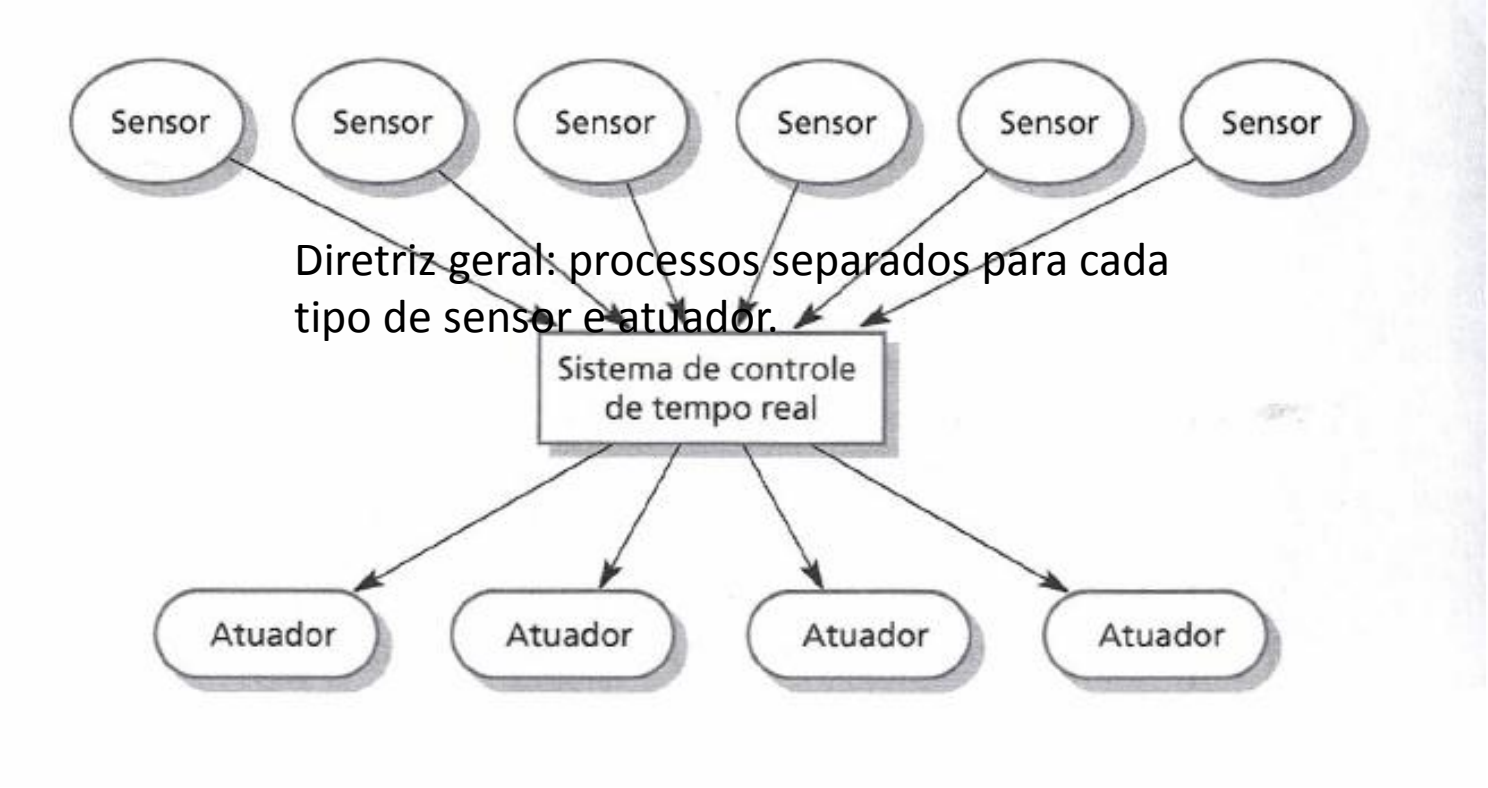
Engenharia de Sistemas Embarcados

- Deve considerar em detalhes o projeto e o desempenho do hardware do sistema.
- É preciso decidir que recursos deve ser implementado em software e que parte de hardware
- Custos e consumo de energia são críticos.
- É necessário decidir se serão usados processadores de prateleira (ex. fpga) ou se deverá ser projetado e construído um novo hardware.

Projeto de Sistemas Embarcados

- Um processo top down pode ser difícil de usar.
- Decisões de baixo nível relacionadas ao hardware precisam ser tomadas no início do projeto, o que diminui a flexibilidade do projeto.
- Os sistemas são geralmente reativos e baseados numa abordagem estímulo-resposta.
- O estímulo é uma entrada e deve produzir uma resposta, que é uma saída e geralmente é dirigido a atuadores
- Os estímulos podem ser de duas categorias:
  - Periódicos
  - Aperiódicos

Um modelo sensor-sistema-atuador



Diretriz geral: processos separados para cada tipo de sensor e atuador.

Arquitetura genérica abstrata

- Contém 3 tipos de processos
- Para cada atuador existe um processo gerenciador
- Para cada atuador há um processo gerenciador
- Há um ou mais processos controladores/processadores.
- Essa arquitetura genérica pode ser instanciada para arquiteturas específicas. Exs: sistemas de monitoração e de aquisição de dados.

Sistemas de tempo real

- As linguagens de programação precisam incluir recursos de baixo nível. Ex. C.
- Java vem sendo atualizada para incluir vários mecanismos que permitam o uso em sistemas de tempo real.
  - O mecanismo básico são as threads

Projeto de Sistemas

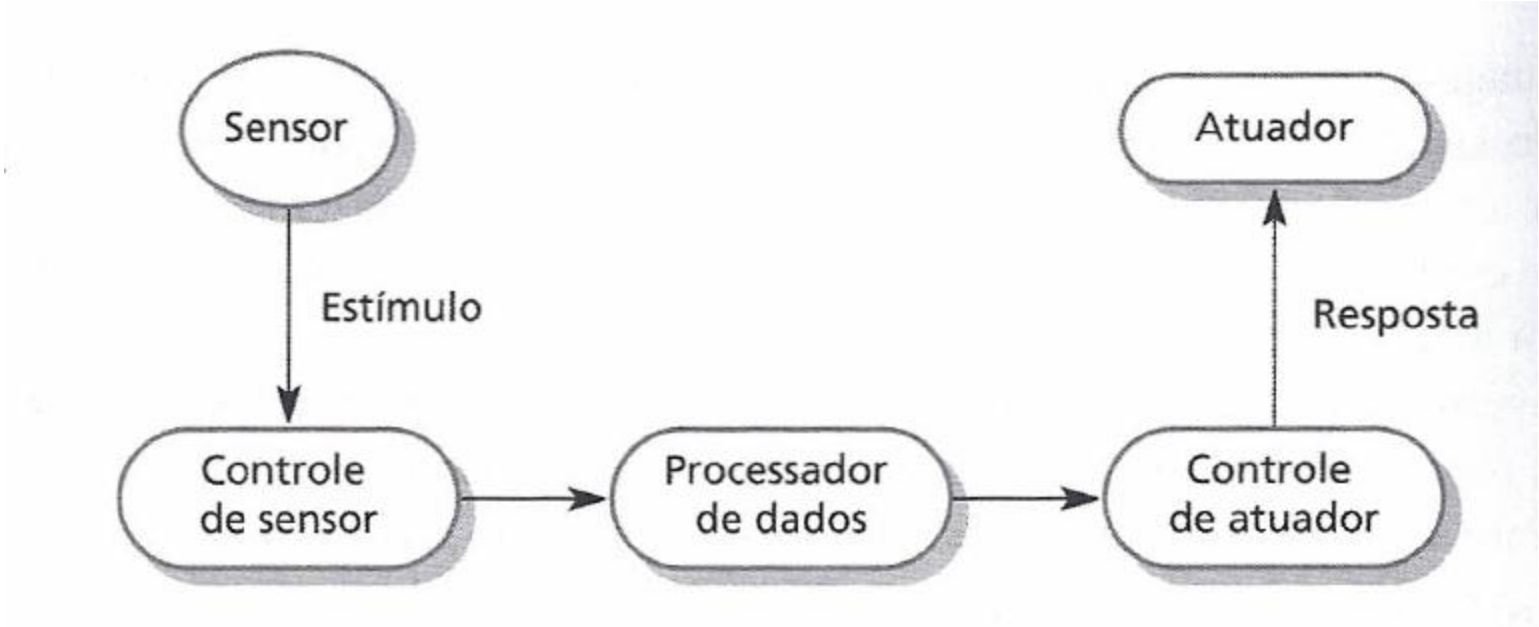
- Decisão importante: quais partes serão implantadas como hardware e quais partes serão implantadas como software.
- Para muitos sistemas de TR embutidos, os custos, consumo de energia e espaço são críticos.
- Muitas vezes, um processo top-down não é prático, pois há decisões de baixo nível que precisam ser tomadas.

Projeto de Sistemas: Atividades Principais

- Seleção da plataforma (Hardware e SO)
- Identificar os estímulos/respostas
- Analisar restrições temporais (timing) para cada estímulo (restrições de tempo)
- Alocar estímulo e processamento a processos concorrentes
- Projeto de processos (concorrentes), de acordo com a arquitetura do sistema.

Projeto de Sistemas: Atividades Principais

- Projetar os algoritmos para fazer o processamento necessário
- Projeto de dados
- Programação de processo: projetar um sistema de alocação (scheduling) que permita atender ás restrições temporais.



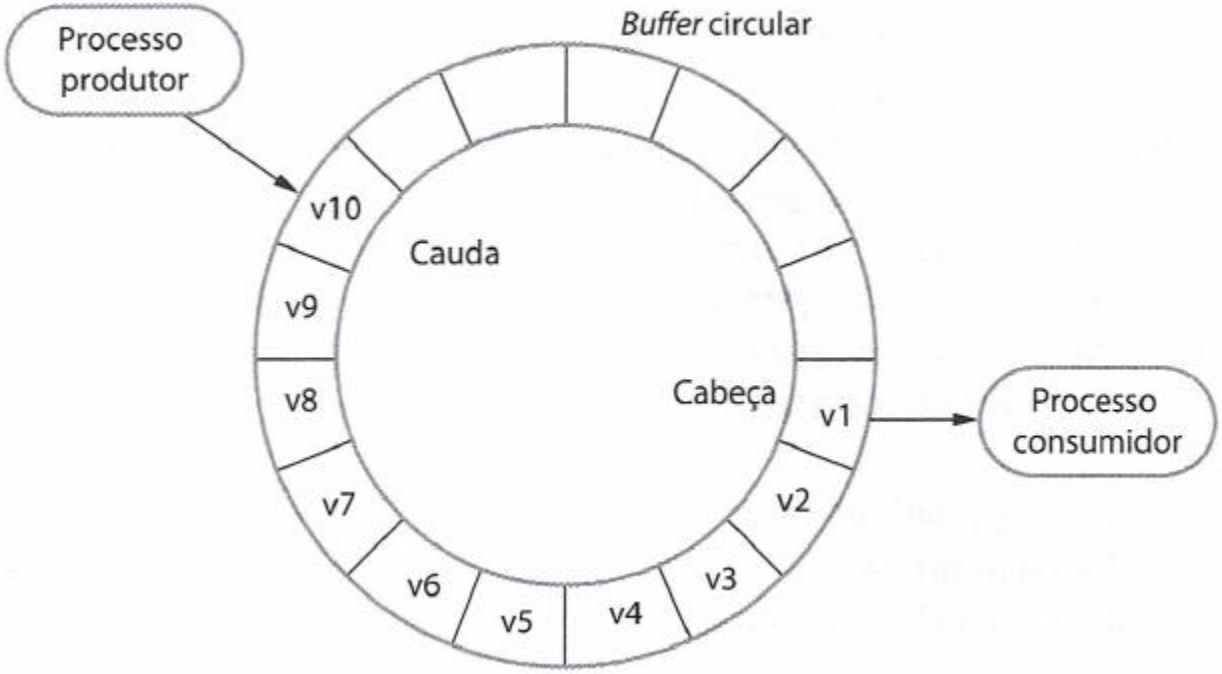
Processos de sensores e atuadores

Algumas considerações

- Coordenação de processos (semáforos, regiões críticas.
- Análises teóricas para avaliar se as restrições temporais serão atendidas. Isso pode ser difícil
- Linguagens OO podem não ser eficientes para implementar um STR.

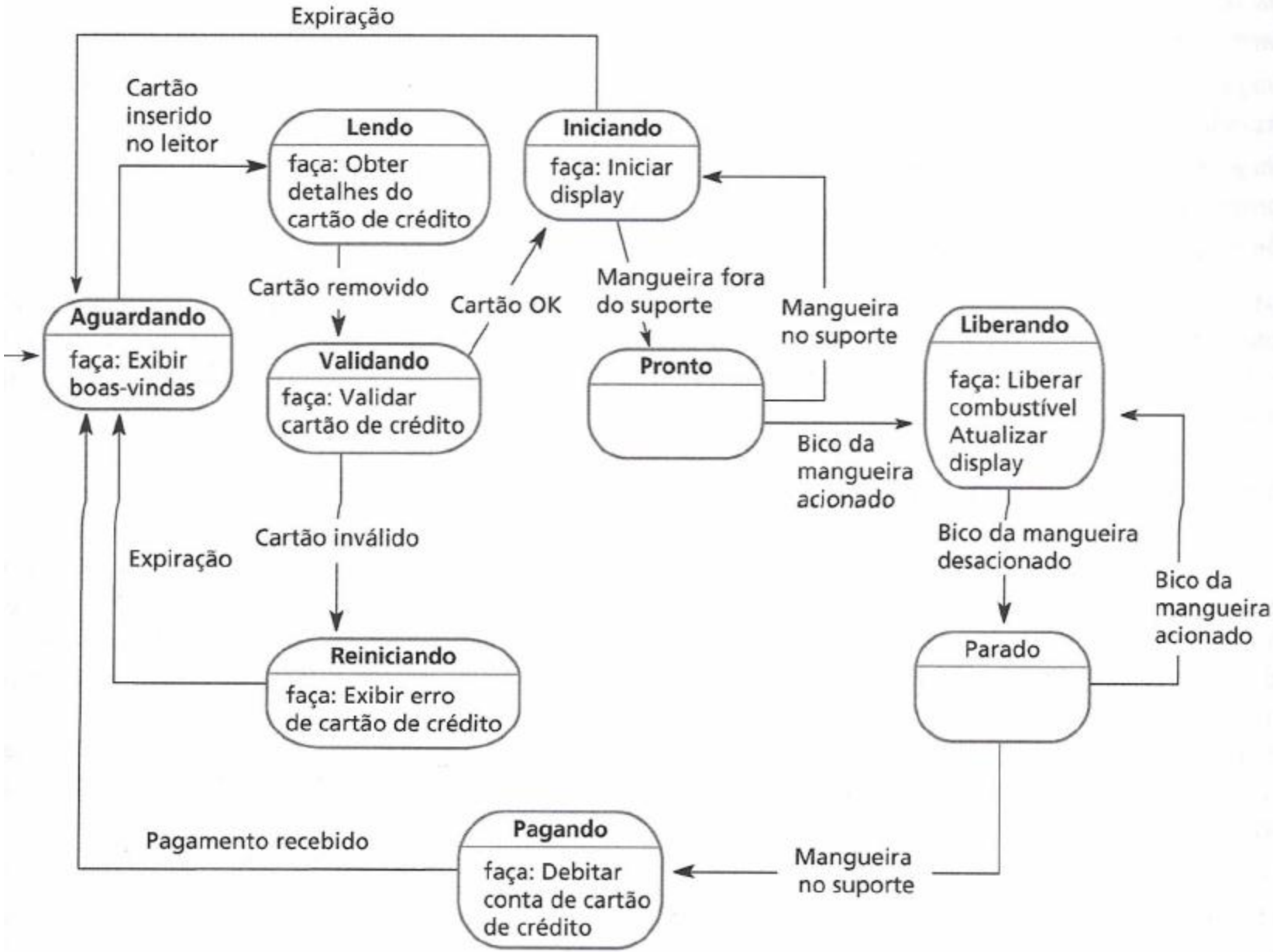
Algumas considerações

- Os processos podem executar com diferentes velocidades.
- Isso pode ser resolvido implementando trocas de informações por meio de “buffers” compartilhados e uso de exclusão mútua para controlar o acesso ao buffer.


Processo produtor
Buffer circular
v10
Cauda
v9
v8
Cabeça
v1
Processo consumidor
v7
v2
v6
v3
v5
v4

Modelagem de STR

- A resposta aos estímulos muitas vezes dependem do estado do sistema → uso de diagramas de estado
- UML: statecharts
- Um modelo de estado considera que em qualquer momento o sistema está em um dos vários estados possíveis. Os estímulos causam a transição para outros estados.
- Exemplo: uma bomba de gasolina automática


Expiração
Cartão inserido no leitor
Lendo
faça: Obter detalhes do cartão de crédito
Iniciando
faça: Iniciar display
Cartão removido
Cartão OK
Mangueira fora do suporte
Mangueira no suporte
Aguardando
faça: Exibir boas-vindas
Validando
faça: Validar cartão de crédito
Pronto
Liberando
faça: Liberar combustível
Atualizar display
Bico da mangueira acionado
Bico da mangueira desacionado
Cartão inválido
Expiração
Bico da mangueira acionado
Reiniciando
faça: Exibir erro de cartão de crédito
Parado
Pagamento recebido
Pagando
faça: Debitar conta de cartão de crédito
Mangueira no suporte

Padrões de arquitetura

- Um padrão de arquitetura pode ser pensado como um projeto genérico para ser instanciado.
- Os padrões de SE são orientados a processos, em vez de orientados a objetos e componentes.
- Três padrões:
  - Observar e reagir
  - Controle de ambiente
  - Pipeline de processo

Estrutura de processo 'Observar e Reagir'

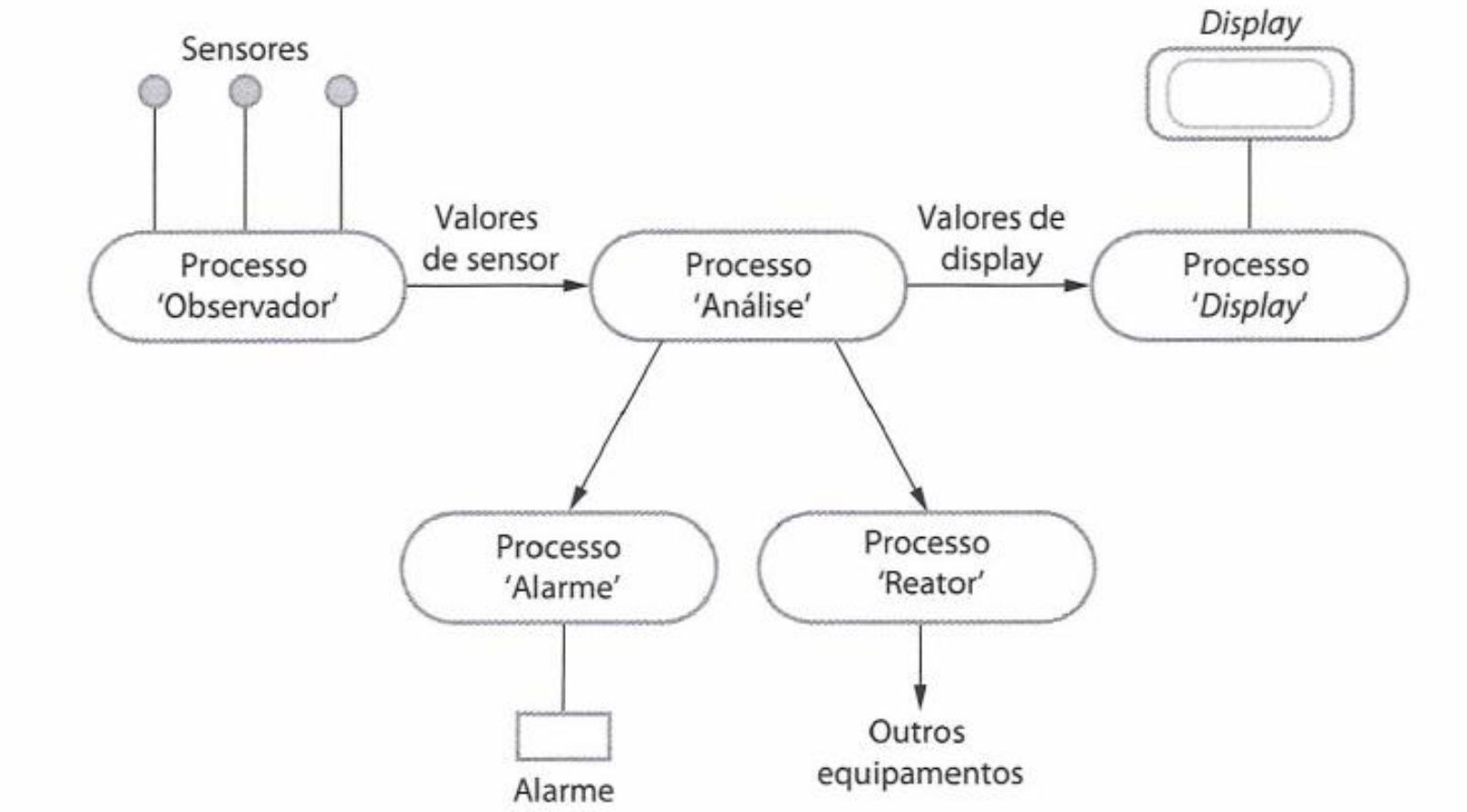
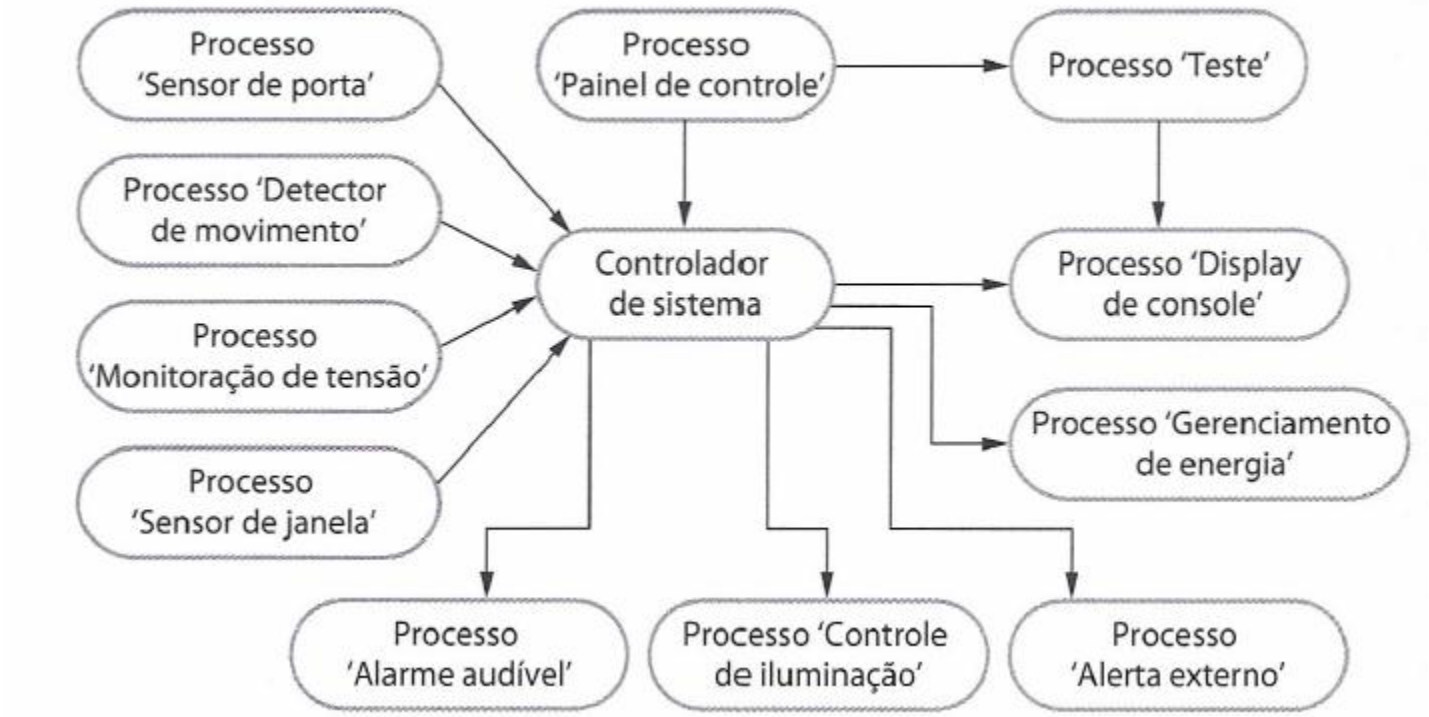


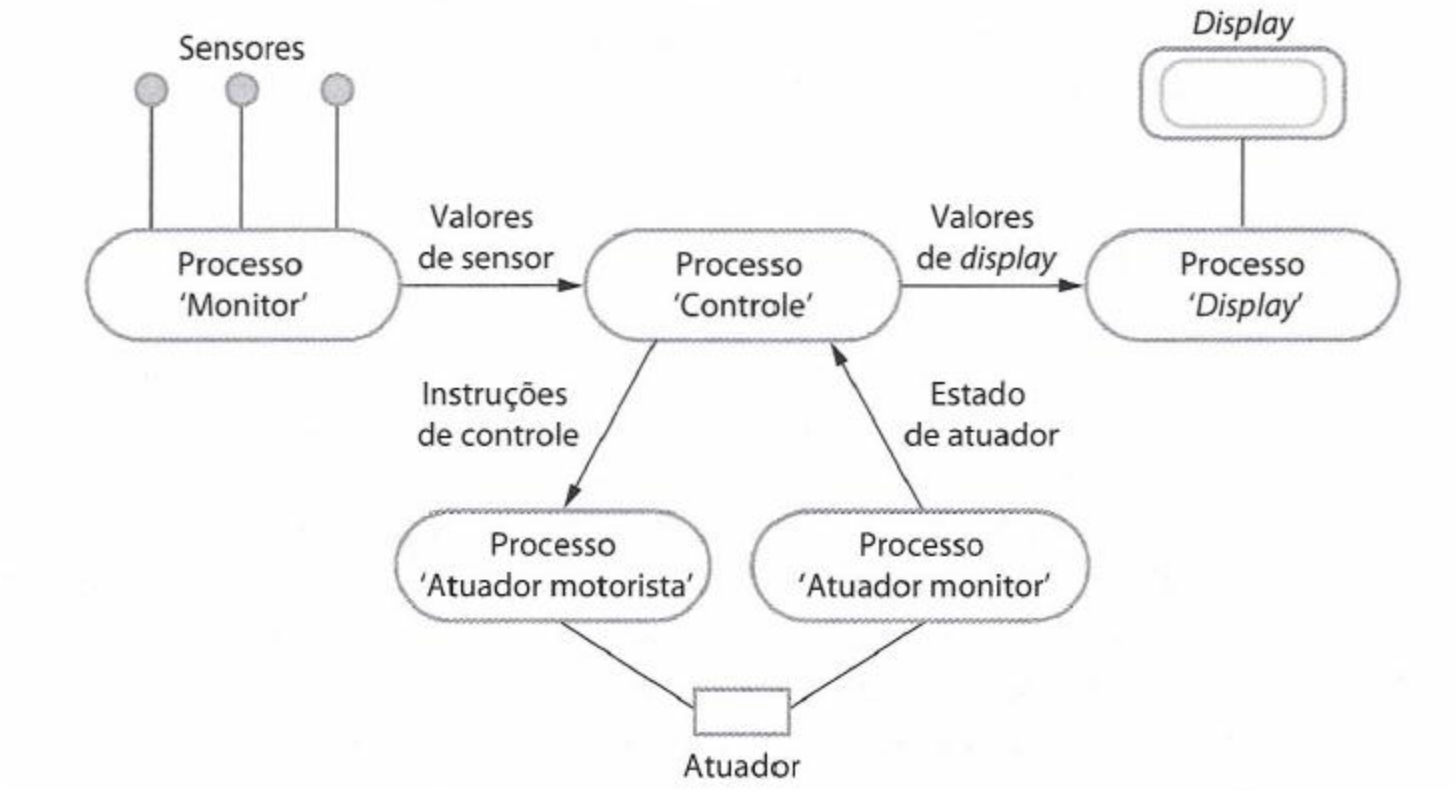
Tabela 20.3 O padrão 'Controle de ambiente'

Nome	**Controle de ambiente**
Descrição	O sistema analisa informações de um conjunto de sensores que coletam dados do ambiente do sistema. Mais informações também podem ser coletadas sobre o estado dos atuadores que também estão conectados. Com base nos dados dos sensores e atuadores, sinais de controle são enviados aos atuadores que, então, causam alterações no ambiente do sistema. Podem ser exibidas informações sobre os valores de sensor e o estado dos atuadores.
Estímulos	Valores de sensores ligados ao sistema e o estado dos atuadores de sistema.
Respostas	Sinais de controle para atuadores, informações de *display*.
Processos	Monitor, Controle, *Display*, Atuador motorista, Atuador monitor.
Usado em	Sistemas de controle.

Estrutura de processo de um sistema de alarme contra roubo



Estrutura de processo 'Controle de ambiente'



Arquitetura de sistema de controle de um sistema de frenagem antiderrapante

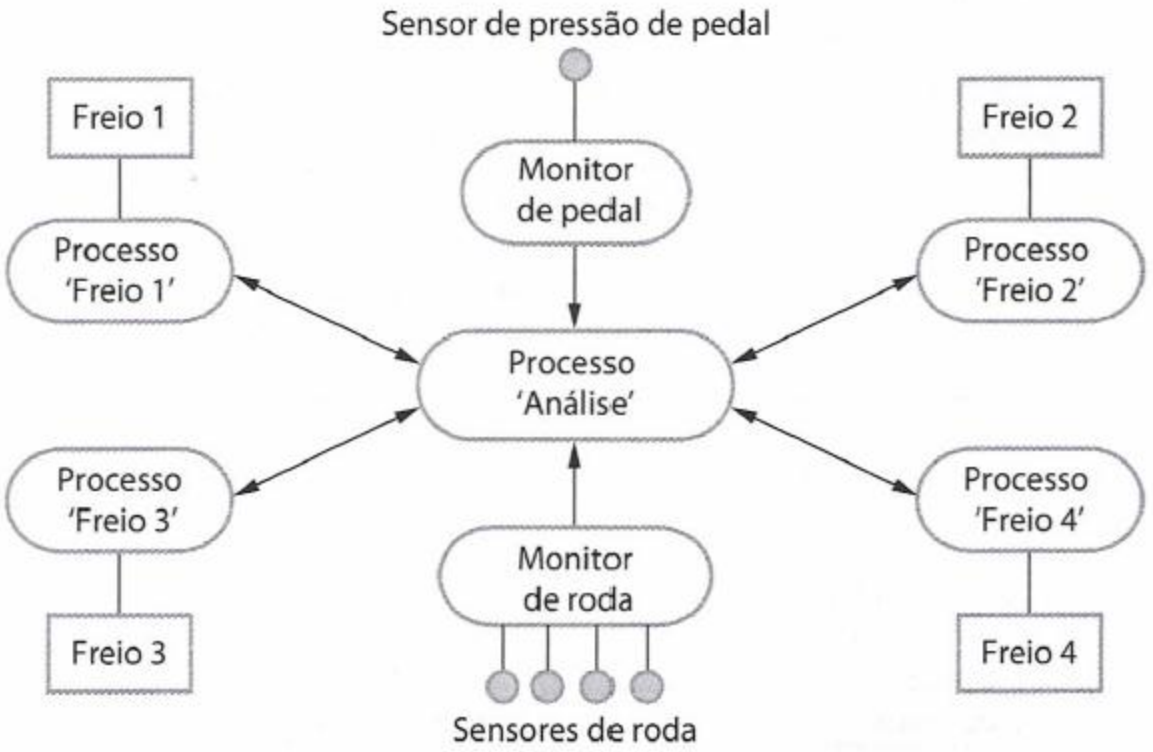
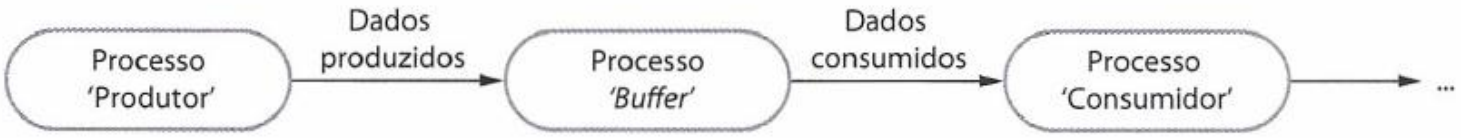


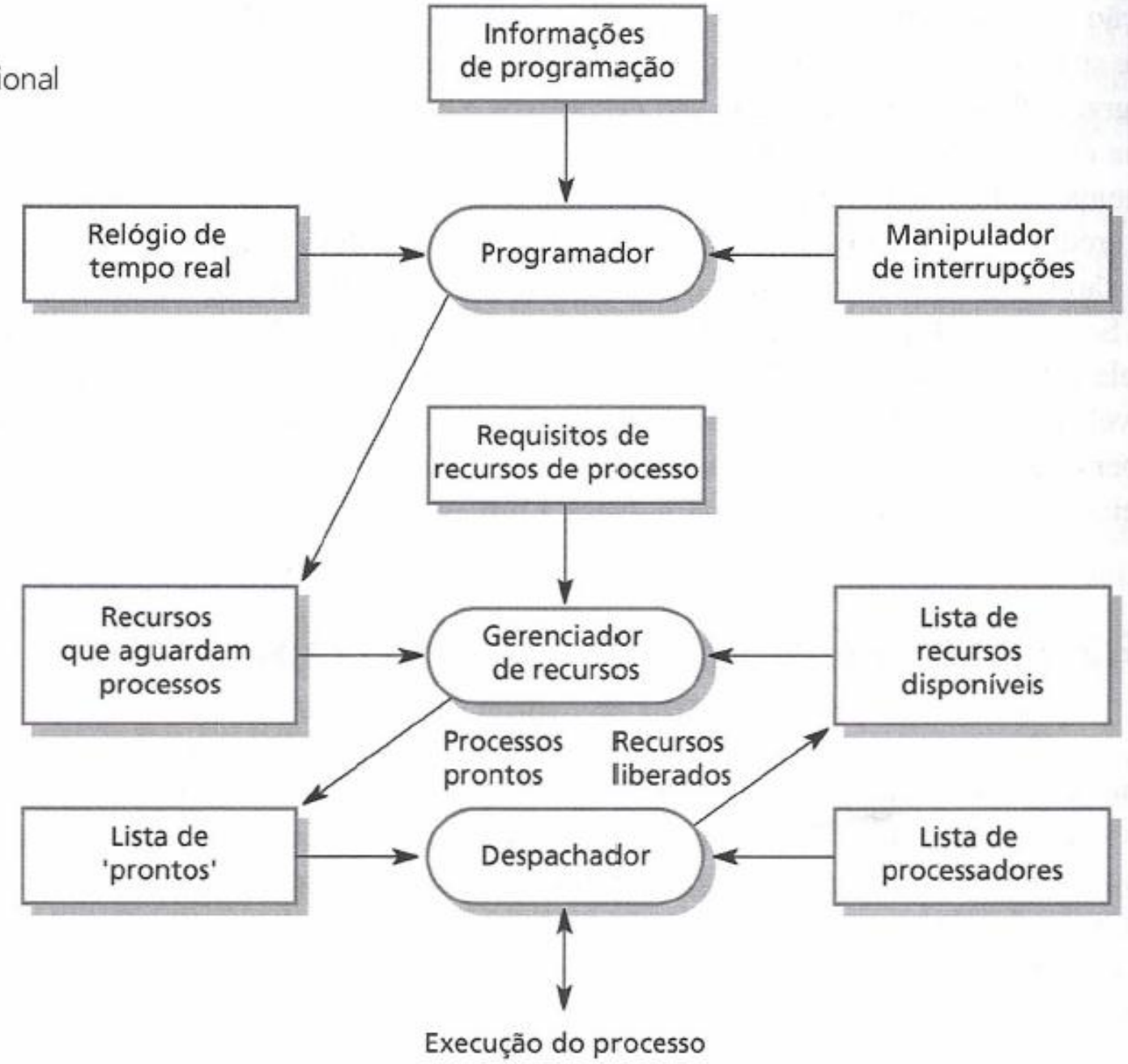
Tabela 20.4 O padrão Processo *Pipeline*

Nome	**Processo *Pipeline***
Descrição	Um *pipeline* de processos é criado com uma sequência de dados movendo-se de uma das extremidades do *pipeline* para outra. Muitas vezes, os processos são ligados por *buffers* sincronizados para permitir que os processos produtor e consumidor sejam executados em velocidades diferentes. O auge de um *pipeline* pode ser armazenamento ou exibição de dados ou o *pipeline* pode terminar em um atuador.
Estímulos	Valores de entrada de ambiente ou de outro processo.
Respostas	Valores de saída para o ambiente ou um *buffer* compartilhado.
Processos	Produtor, *Buffer*, Consumidor.
Usado em	Sistemas de aquisição de dados, sistemas multimídia.

Figura 20.9 Estrutura de processo de Processo *'Pipeline'*



racional

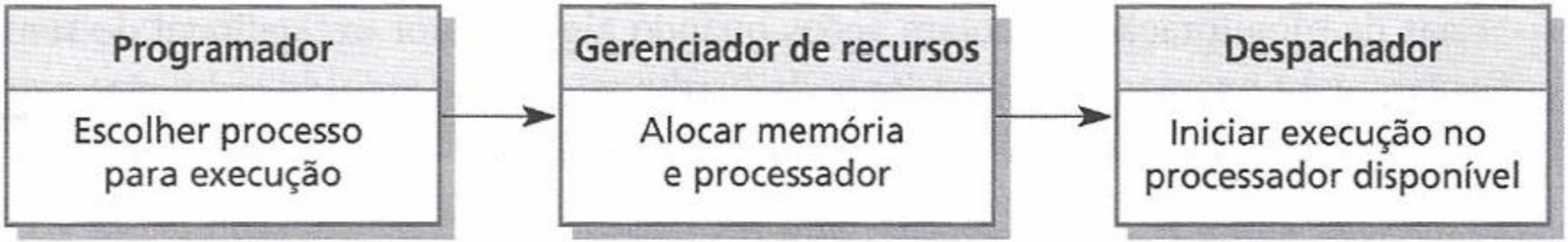

Informações de programação
Relógio de tempo real
Programador
Manipulador de interrupções
Requisitos de recursos de processo
Recursos que aguardam processos
Gerenciador de recursos
Lista de recursos disponíveis
Processos prontos
Recursos liberados
Lista de 'prontos'
Despachador
Lista de processadores
Execução do processo

Sist. Operacional de Tempo Real

- Muitos sistemas embutidos funcionam com um SOTR
- Os SOTR podem ser muito simples ou grandes e complexos.
- Em geral possuem: relógio de tempo real, tratador de interrupções, alocador (scheduler) gerenciador de recurso e despachador

Gerenciamento de Processos

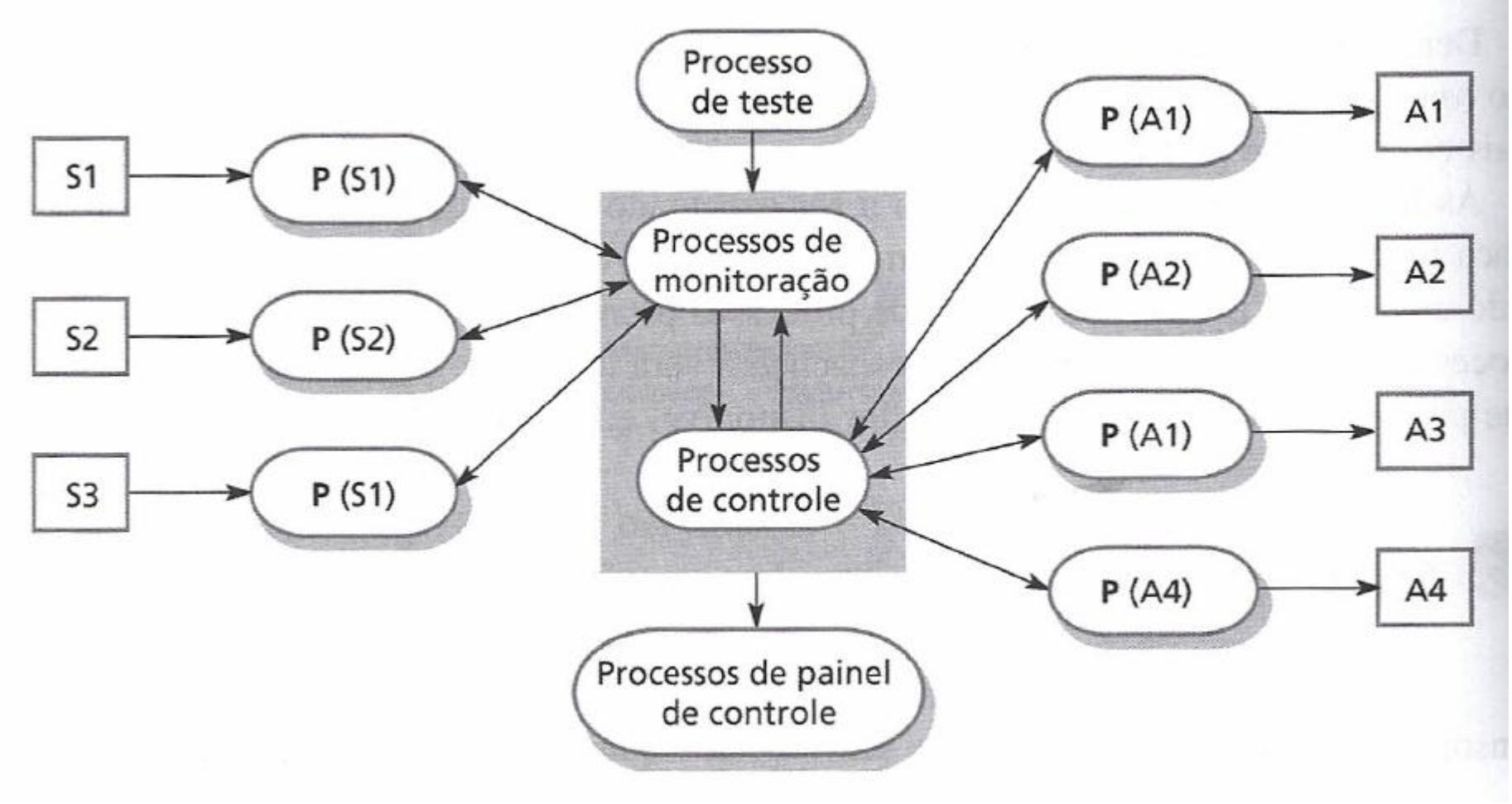
- É preciso tratar processos com prioridades
- Deve haver pelo menos dois níveis de prioridade:
  - Nível de interrupção, para os processos que precisam de resposta muito rápida
  - Nível de relógio, para os processos periódicos



Sistemas de monitoração e controle

- É uma categoria importante de STR
- Verificam sensores que fornecem informação sobre o ambiente do sistema e executam ações que dependem da leitura do sensor
- Sistema de monitoração executam ações de acordo com os valores/sinais dos sensores
- Os sistemas de monitoração controlam continuamente os atuadores de hardware de acordo com os valores dos sensores associados.

Arquitetura genérica de um STR de monitoração e controle

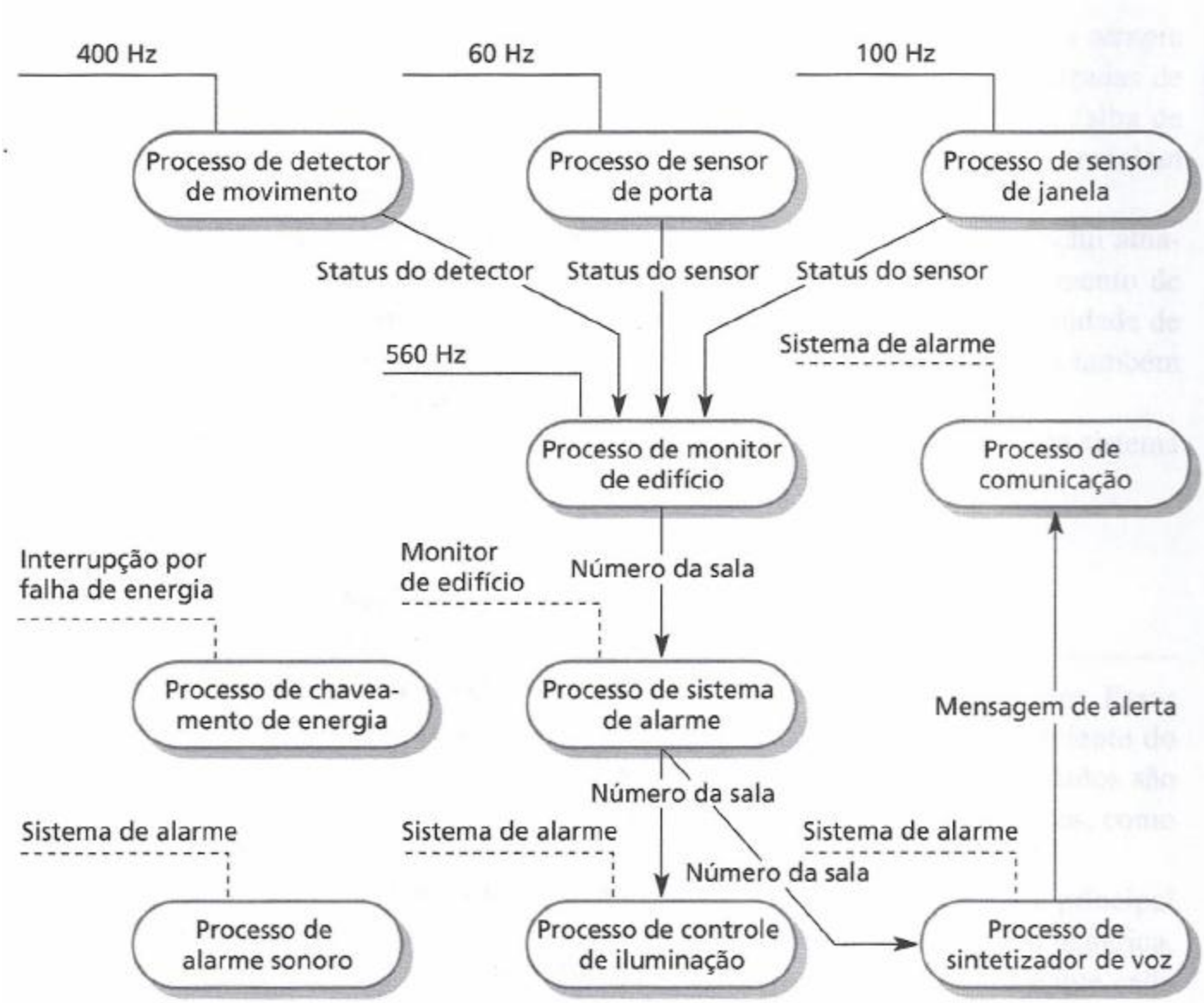


Exemplo: sistema de alarme contra roubos para edifícios comerciais

- Sensores de movimento em salas (200)
- Sensores em janelas (50)
- Sensores de porta (30)
- Chama um telefone da polícia e sintetiza voz para indicar onde foi ativado o alarme
- liga alarme sonoro
- Acende luzes ao redor do sensor ativado
- Tem back-up de energia e monitora a queda de energia, mudando para a conjunto de baterias.

Estímulos

- Há duas categorias: Falha de energia e alarme de intrusos
- Lista de estímulos-respostas
  - Interrupção por falha de energia (50ms)
  - Alarme de porta (2/s)
  - Alarme de janela (2/s)
  - Detector de movimentos (2/s)
  - Alarme sonoro (após 0.5s)
  - Interruptor de luzes (após 0.5s)
  - Comunicação (após 2s)
  - Sintetizar voz (após 4s)


400 Hz
60 Hz
100 Hz
Processo de detector de movimento
Processo de sensor de porta
Processo de sensor de janela
Status do detector
Status do sensor
Status do sensor
560 Hz
Sistema de alarme
Processo de monitor de edifício
Processo de comunicação
Interrupção por falha de energia
Monitor de edifício
Número da sala
Processo de chaveamento de energia
Processo de sistema de alarme
Mensagem de alerta
Número da sala
Sistema de alarme
Sistema de alarme
Sistema de alarme
Número da sala
Processo de alarme sonoro
Processo de controle de iluminação
Processo de sintetizador de voz

```
// Veja http://www.software-engin.com/ para obter os links do código completo
// exemplo
class BuildingMonitor extends Thread {
   BuildingSensor win, door, move ;
   Siren   siren = new Siren () ;
   Lights  lights = new Lights () ;
   Synthesizer synthesizer = new Synthesizer () ;
   DoorSensors doors = new DoorSensors (30) ;
   WindowSensors  windows = new WindowSensors (50) ;
   MovementSensors movements = new MovementSensors (200) ;
   PowerMonitor pm = new PowerMonitor () ;
   BuildingMonitor()
   {
      // ativar todos os sensores e iniciar os processos
      siren.start () ; lights.start () ;
      synthesizer.start () ; windows.start () ;
      doors.start () ; movements.start () ; pm.start () ;
   {}
```

```
{public void run ()
{{
    int room = 0 ;
    while (true)
    {
        // ler os sensores de movimento pelo menos duas vezes por segundo (400 H
        move = movements.getVal () ;
        // ler os sensores de janela pelo menos duas vezes por segundo (100 Hz)
        win = windows.getVal () ;
        // ler os sensores de porta pelo menos duas vezes por segundo (60 Hz)
        door = doors.getVal () ;
        if (move.sensorVal == 1 | door.sensorVal == 1 | win.sensorVal == 1)
            {
              // um sensor indicou um intruso
              if (move.sensorVal == 1)  room = move.room ;
              if (door.sensorVal == 1)  room = door.room ;
              if (win.sensorVal == 1 )  room = win.room ;
              lights.on (room) ; siren.on () ; synthesizer.on (room) ;
              break ;
            }
    }
    lights.shutdown () ; siren.shutdown () ; synthesizer.shutdown () ;
    windows.shutdown () ; doors.shutdown () ; movements.shutdown () ;
  } // run
} //BuildingMonitor
```