



# Sistemas de Tempo Real e Linguagens de Programação

Alan Burns e Andy Wellings



# Pre-requisitos

---

- Conhecimentos básicos de ADA e C
- Conhecimentos básicos de Arquitetura de Computadores
- Conhecimentos básicos de Sistemas Operacionais



# Objetivos

---

- Entendimento da visão geral do conceito de sistemas de tempo real
- Compreensão prática com foco na indústria
- Estimular o interesse nesta área de pesquisa



# Objetivos Técnicos Gerais

- Entender os requisitos básicos de sistemas de tempo real e como estes requisitos tem influenciado o projeto de linguagens de programação e sistemas operacionais de tempo real.
- Entender as técnicas de implementação e análise que permitem estes requisitos serem alcançados.



# O Que é um sistema de tempo-real?

---

- Um sistema de tempo-real é qualquer sistema de processamento de informação que tem que responder uma entrada gerada externamente dentro de um finito e determinado período
  - A “corretude” depende não somente do resultado lógico mas também do tempo em que foi respondido
  - Falha em responder é tão ruim quanto uma resposta errada!
- O computador é um componente em um sistema maior de engenharia => EMBEDDED COMPUTER SYSTEM
- 99% de todos os processadores são para o mercado de sistemas embarcados



# Terminologia

---

- **Hard real-time** — sistemas onde é absolutamente imperativo que as respostas ocorram dentro de um deadline preciso, ex.: Sistema de controle de voo.
- **Soft real-time** — Sistemas onde os deadlines são importantes, mas que ainda funcionarão corretamente se os deadlines são algumas vezes perdidos. Ex.: Sistema de Aquisição de Dados
- **Firm real-time** — sistemas que são soft real-time mas onde não há nenhum benefício de continuar o processamento daquele evento

Um único sistema pode ter sub-sistemas de tempo real hard, soft e firm. Na realidade muitos sistemas possuem uma função de custo associada com cada deadlines perdido.



# Terminologia

---

- **Time-aware** — sistema faz explícita referência ao tempo (ex.: abra a porta do cofre às 9:00)
- **Reativo** — sistema deve produzir uma saída dentro de um deadline (usando a entrada como referência)
  - Sistemas de Controle são sistemas reativos
  - Requisitos de definição de limites variabilidade de entrada e saída (tempo), **jitter de entrada** e **jitter de saída**



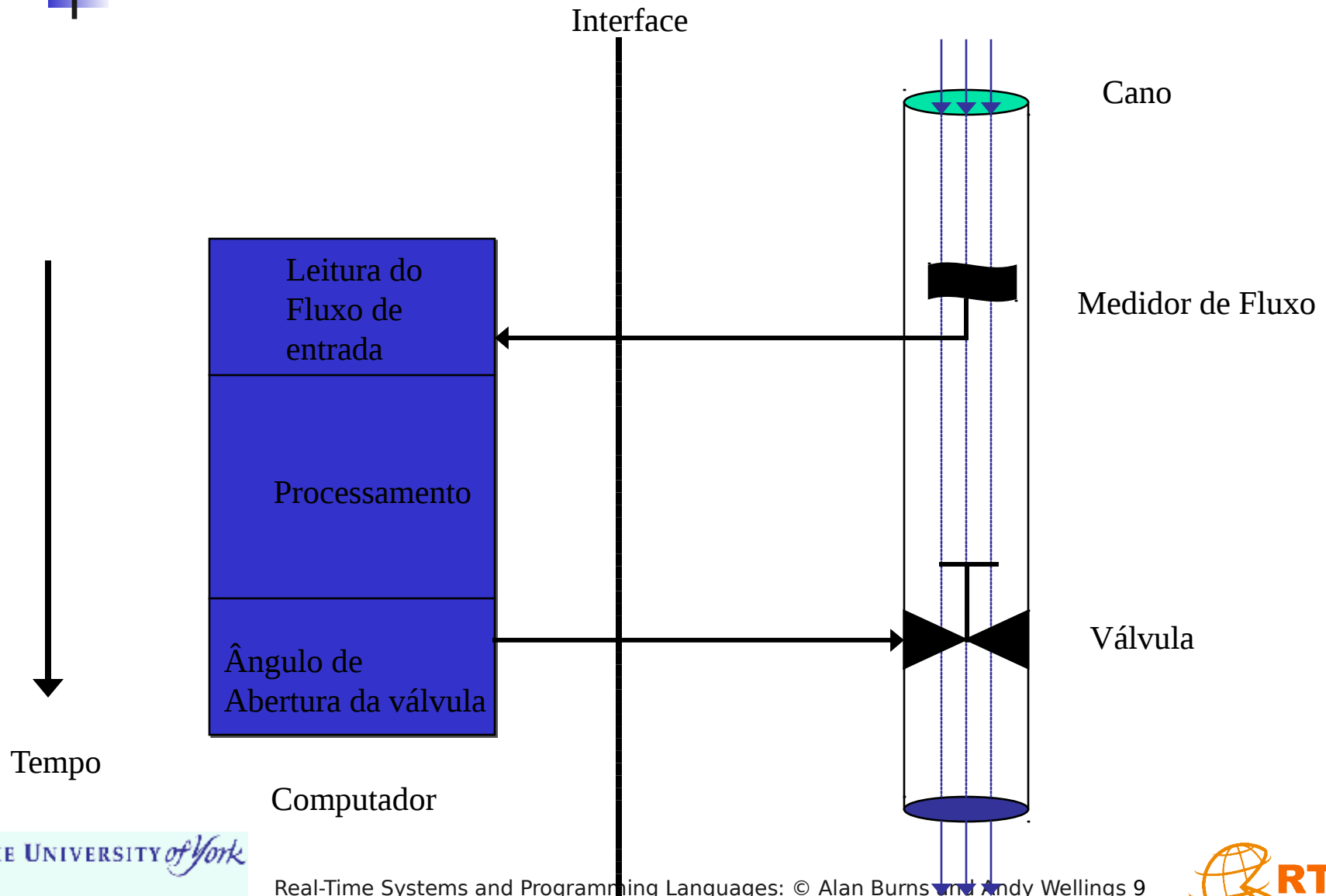
# Terminologia

---

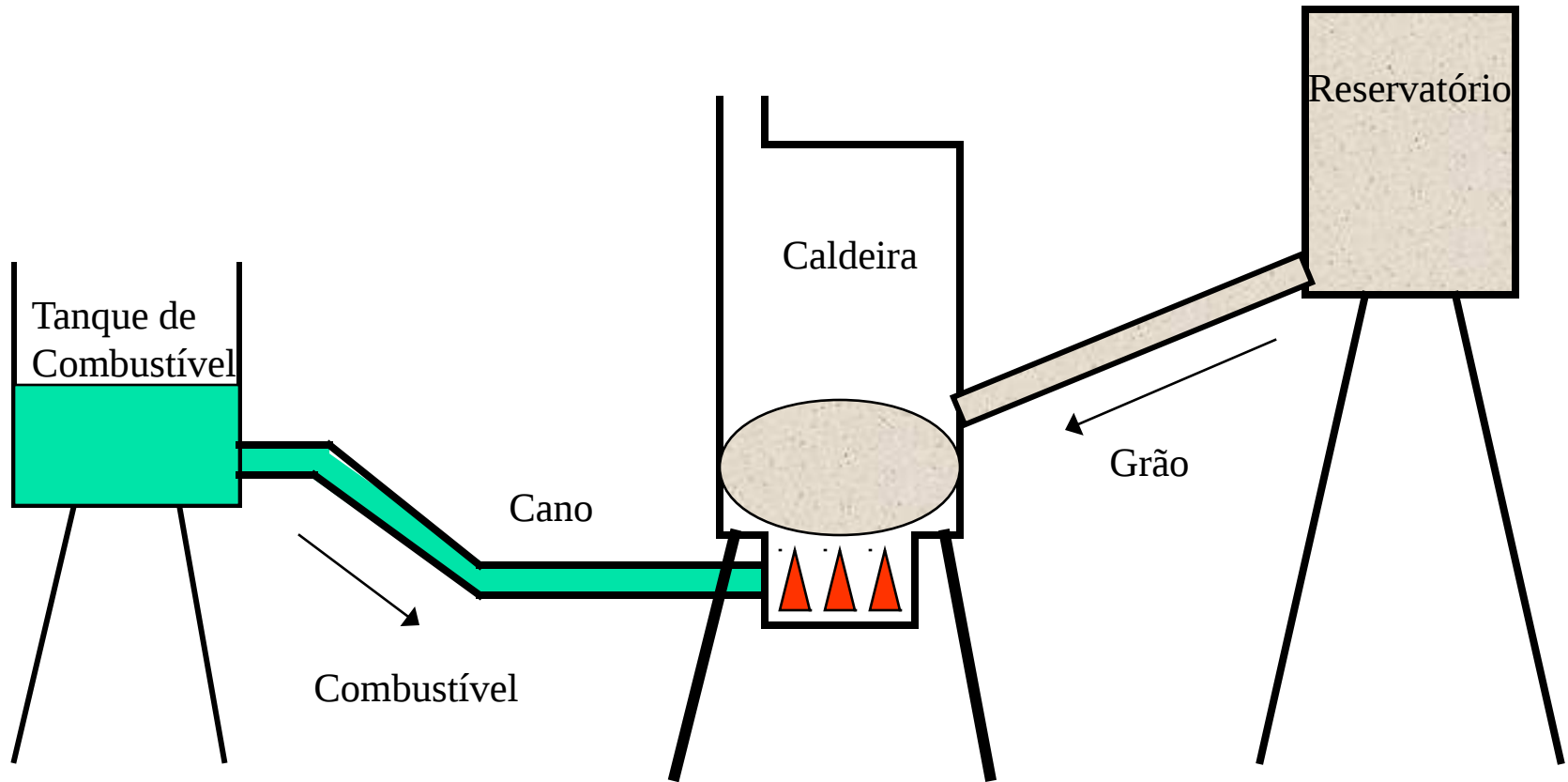
- **Time-triggered** — computação é disparada no momento de tempo definido
  - Lance atividade às 9:00
  - Lance a atividade a cada 25ms – uma atividade **periódica**
- **Event-trigger** — computação é disparada por um evento externo ou interno
  - A atividade lançada é dita **sporadic** se há um limite no intervalo de chegada do evento
  - A atividade lançada é dita **aperiodic** se não há tal limite



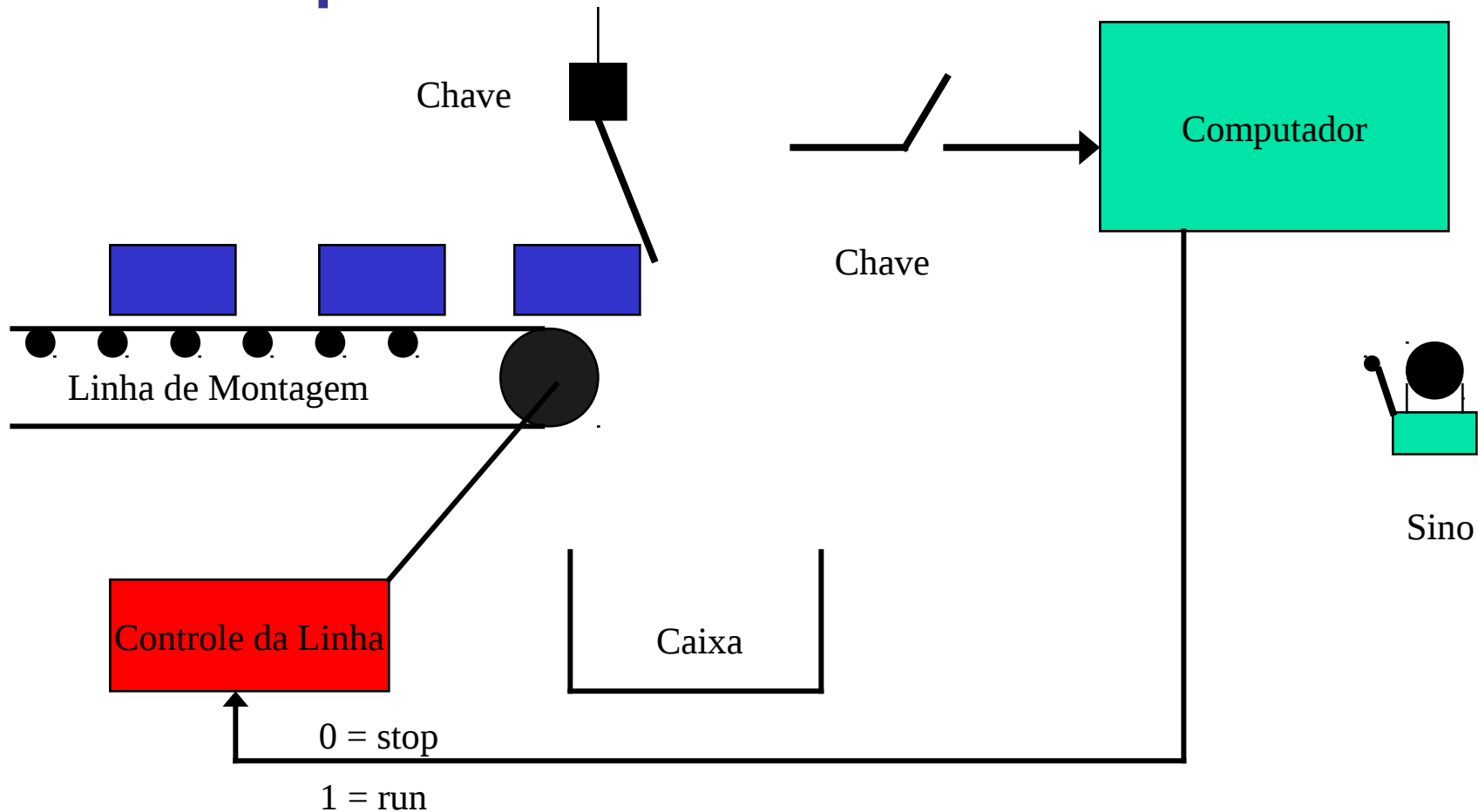
# Exemplo: Um simples sistemas de controle de fluidos.



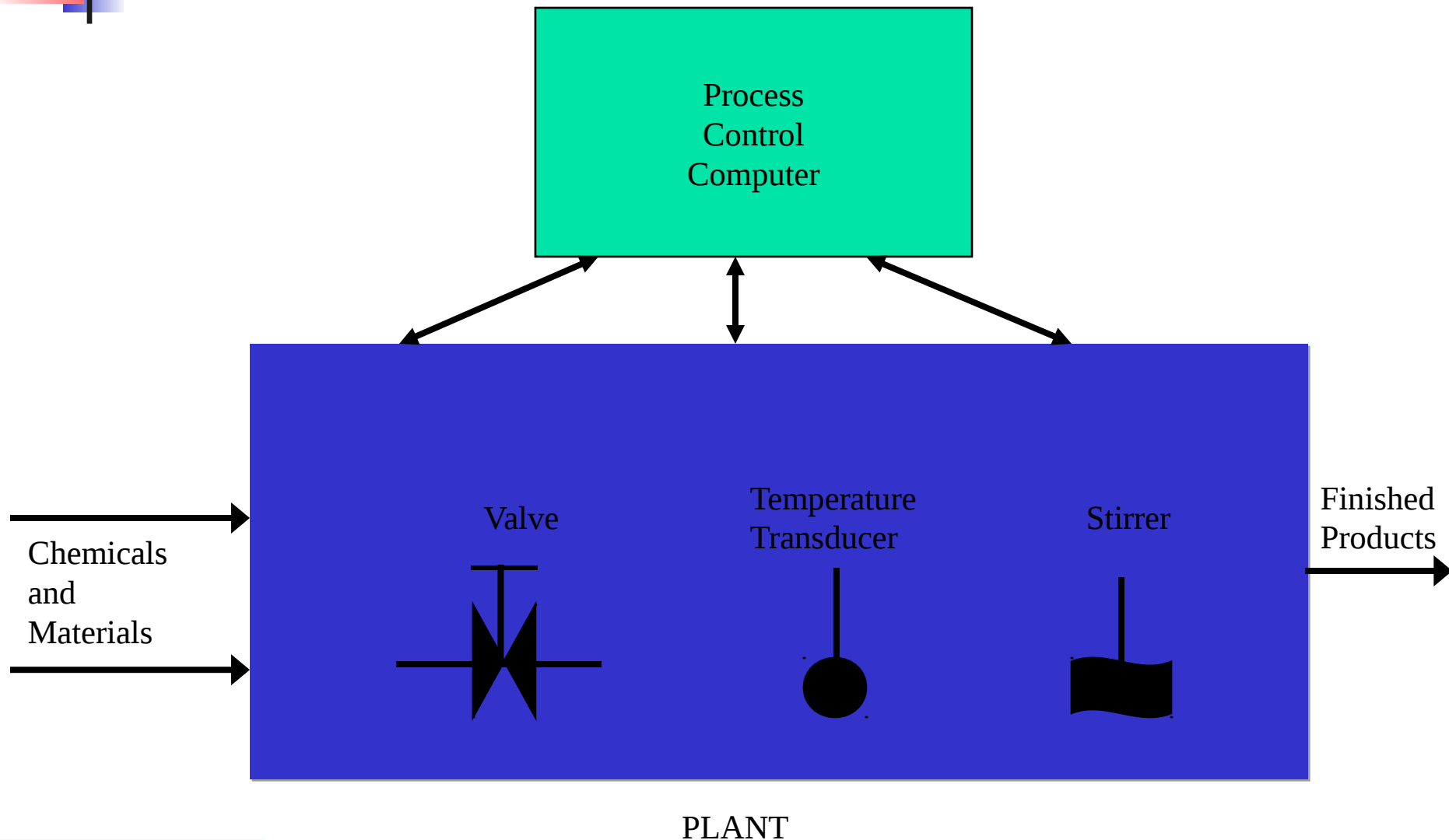
# Planta de um torrar grãos



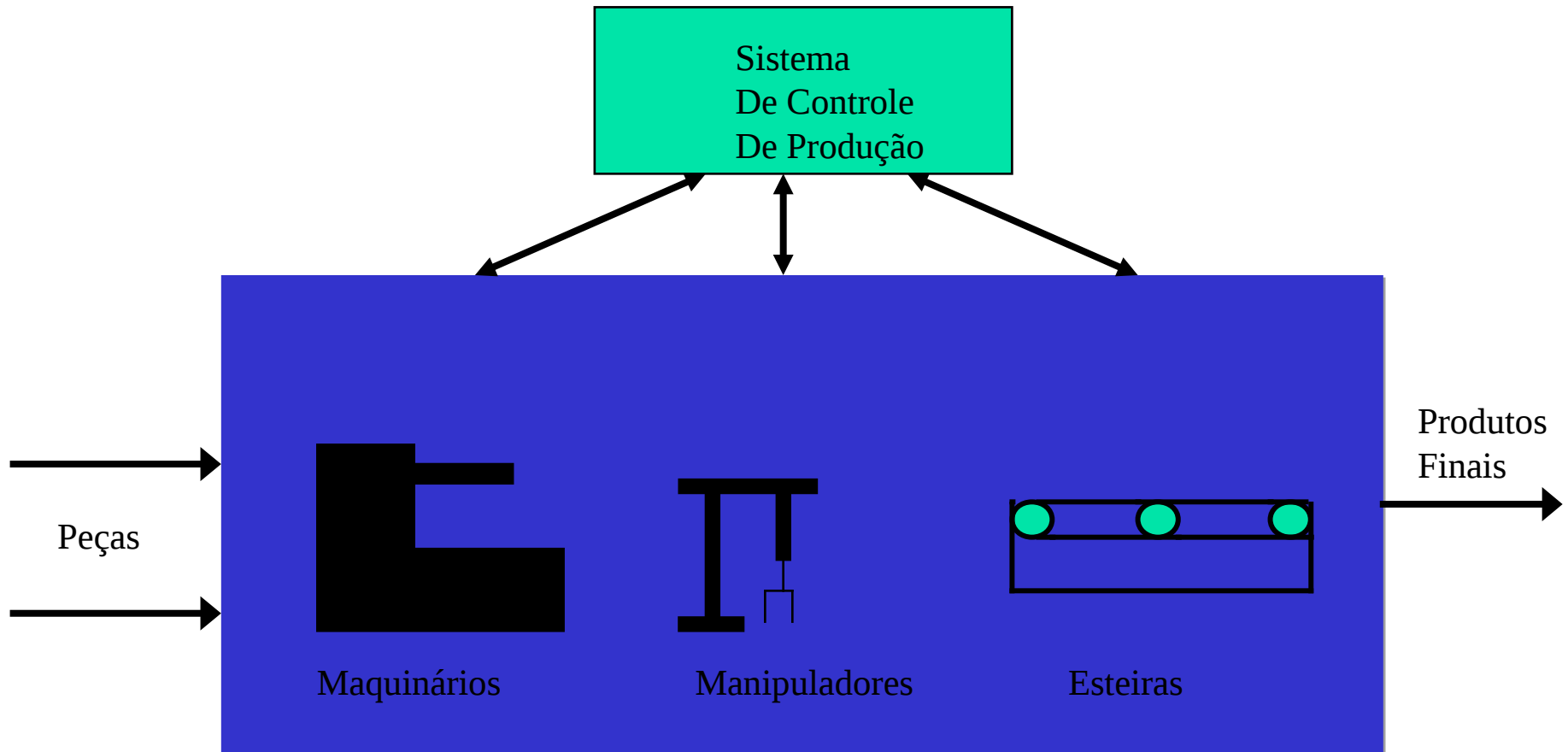
# Estação de empacotamento



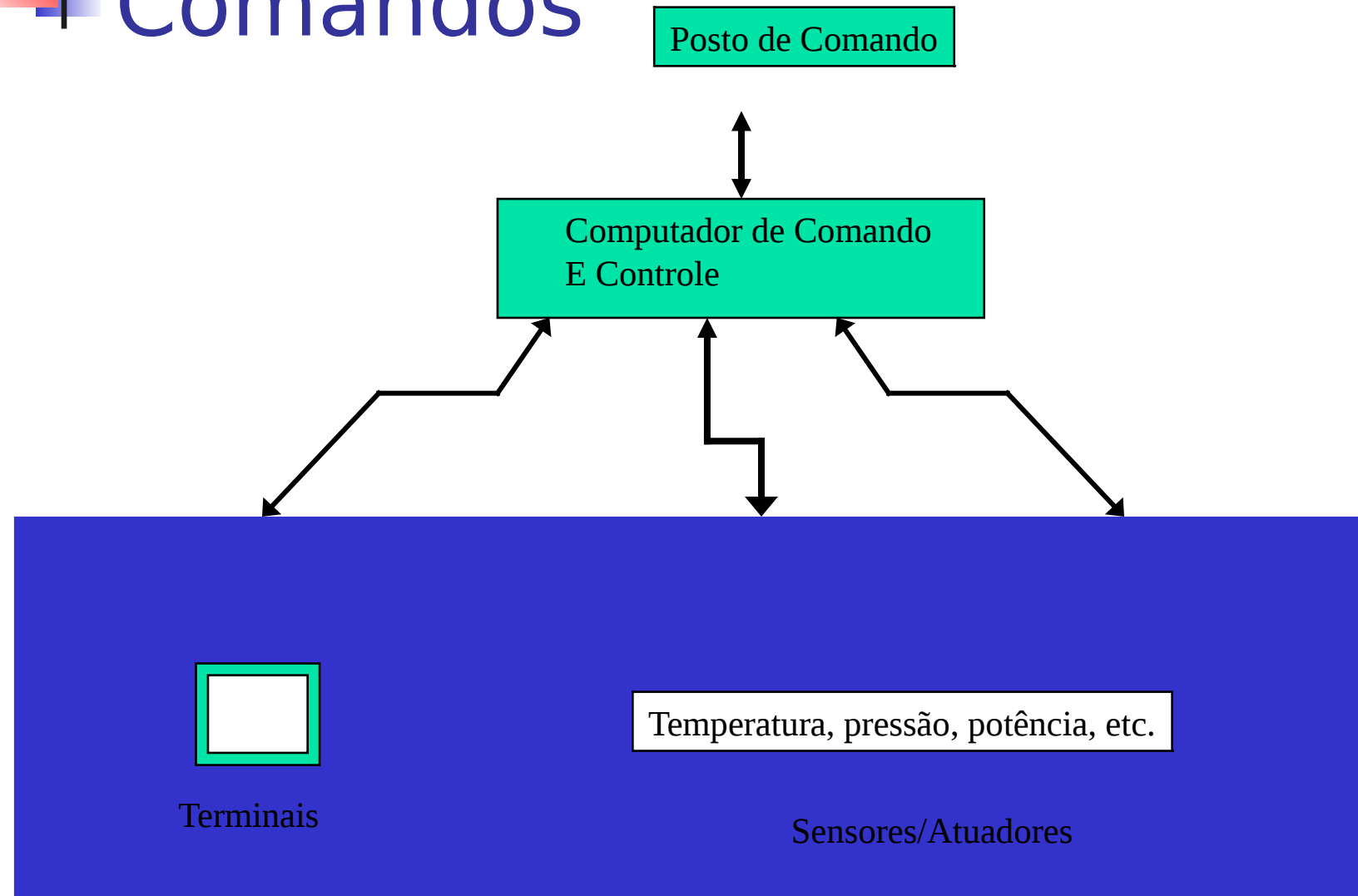
# A Process Control System



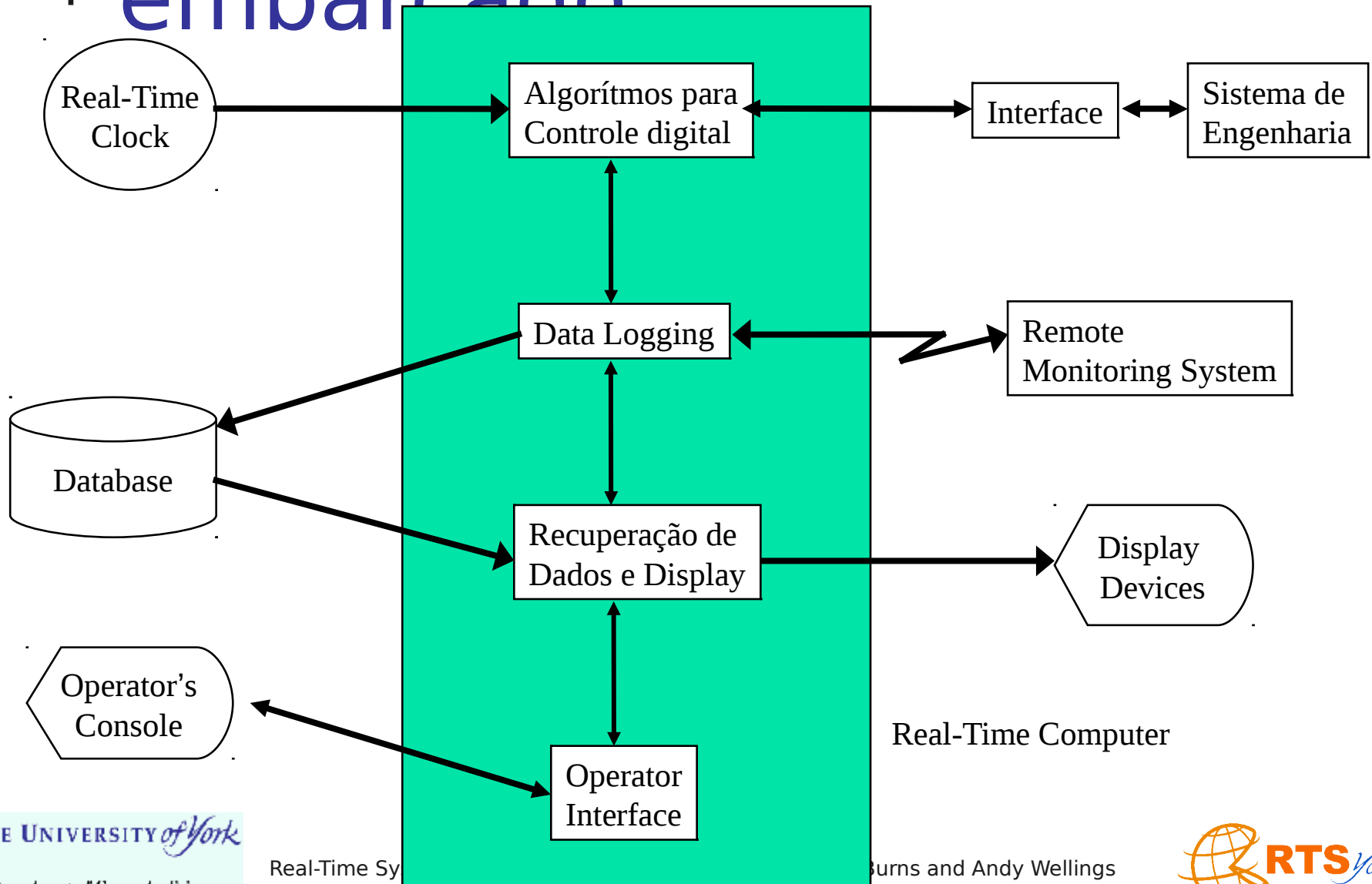
# Sistema de Controle de Produção



# Sistema de Controle e Comandos



# Um típico sistema embarcado





# Outros Sistemas de Tempo

## Real

- Sistemas Multimídia
  - Incluindo dispositivos móveis (smartphones)
- Sistemas Cyber-Físicos
  - Conectando sistemas baseados em web com o mundo real





# Características de um STR (RTS)

---

- **Tempos de resposta garantidos** — nós precisamos estar aptos a predizer com precisão o pior caso de tempo de resposta para os sistemas; eficiência é importante, mas predição é essencial
- **Controle concorrente de componentes do sistema** — dispositivos operam em paralelo no mundo real; melhor então é modelar este paralelismo por entidades concorrentes no programa
- **Facilidade de interagir com hardware de propósito geral**



# Características de um STR

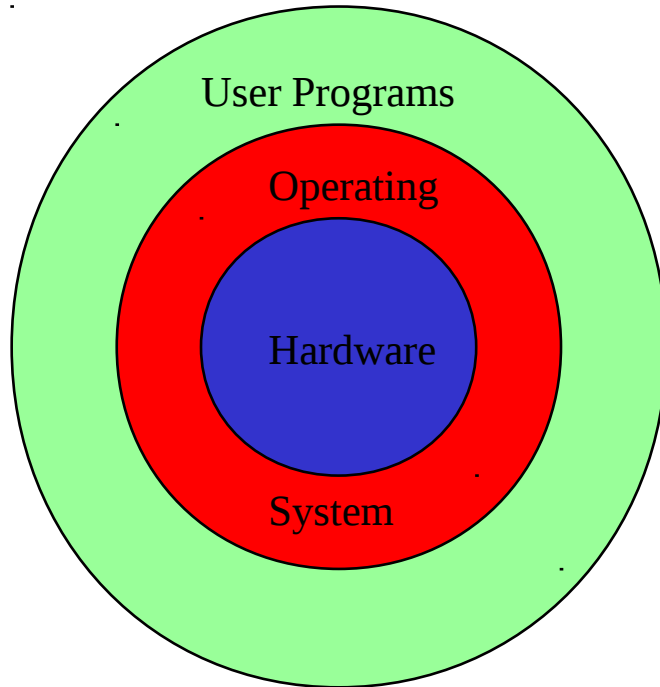
- **Suporte para computação numérica** – estar apto a suportar computação discreta/contínua necessária para os algoritmos retro-alimentados.
- **Tamanho e Complexidade** — variam de algumas centenas de linhas de códigos em assembler ou C até 20 milhões de linhas de ADA. Questão da variedade quanto o tamanho.
- **Confiabilidade e Segurança Extremas** — sistemas embarcados tipicamente controlam o ambiente em que eles operam, falha deste controle pode resultar em perda de vida, agressão ao meio-ambiente ou perdas econômicas

# Linguagens de Programação

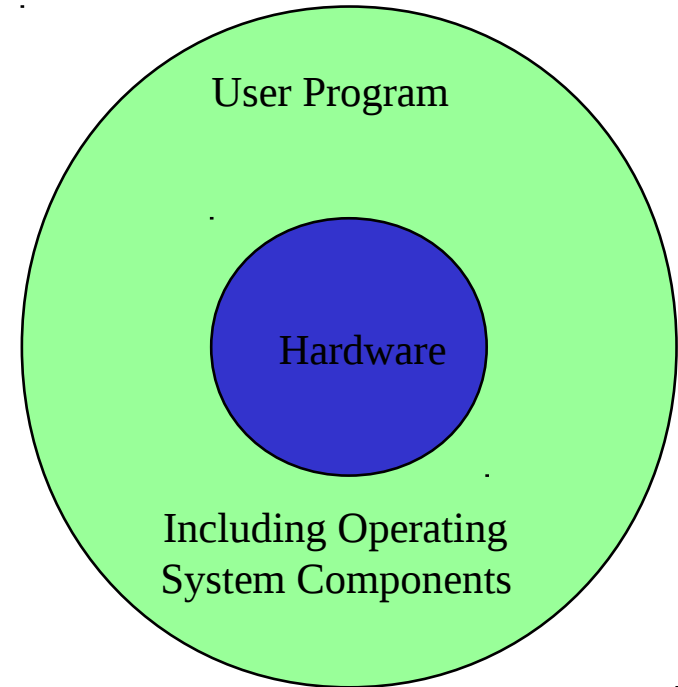
## RT

- Linguagens Assembly
- Linguagens de implementação de sistemas Sequenciais — ex.: RTL/2, Coral 66, Jovial, C.
- Ambas normalmente precisam de um S.O.
- Linguagens concorrentes de alto-nível. ex.: Ada, Chill, Modula-2, Mesa, Java.
- Sem suporte de S.O. (não obrigatório)!
- Vamos considerar aqui:
  - Java/Real-Time Java
  - C e Real-Time POSIX (não em detalhes)
  - Ada 2005

# Linguagens RT e os SOs



Typical OS Configuration



Typical Embedded Configuration

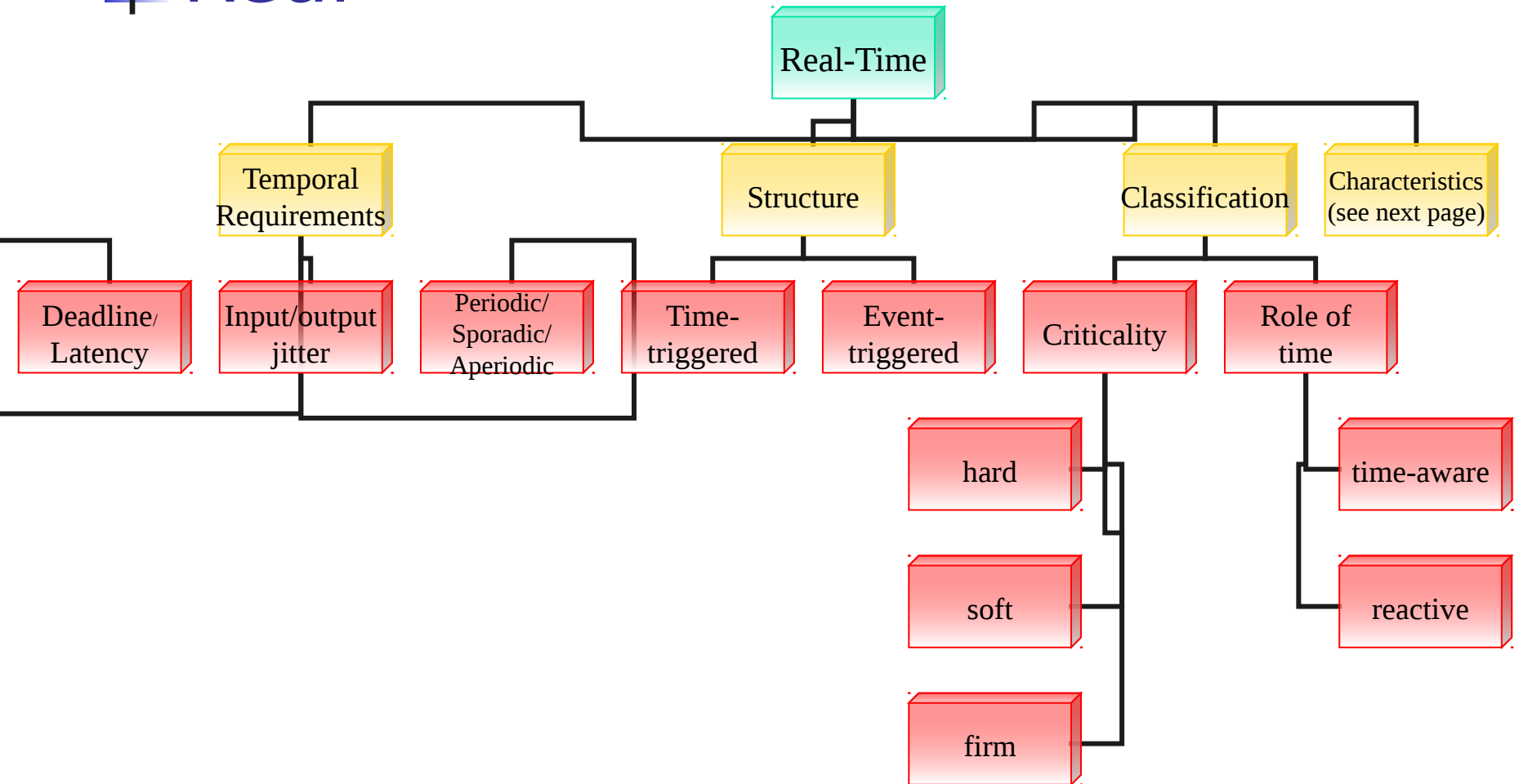


# Resumo

---

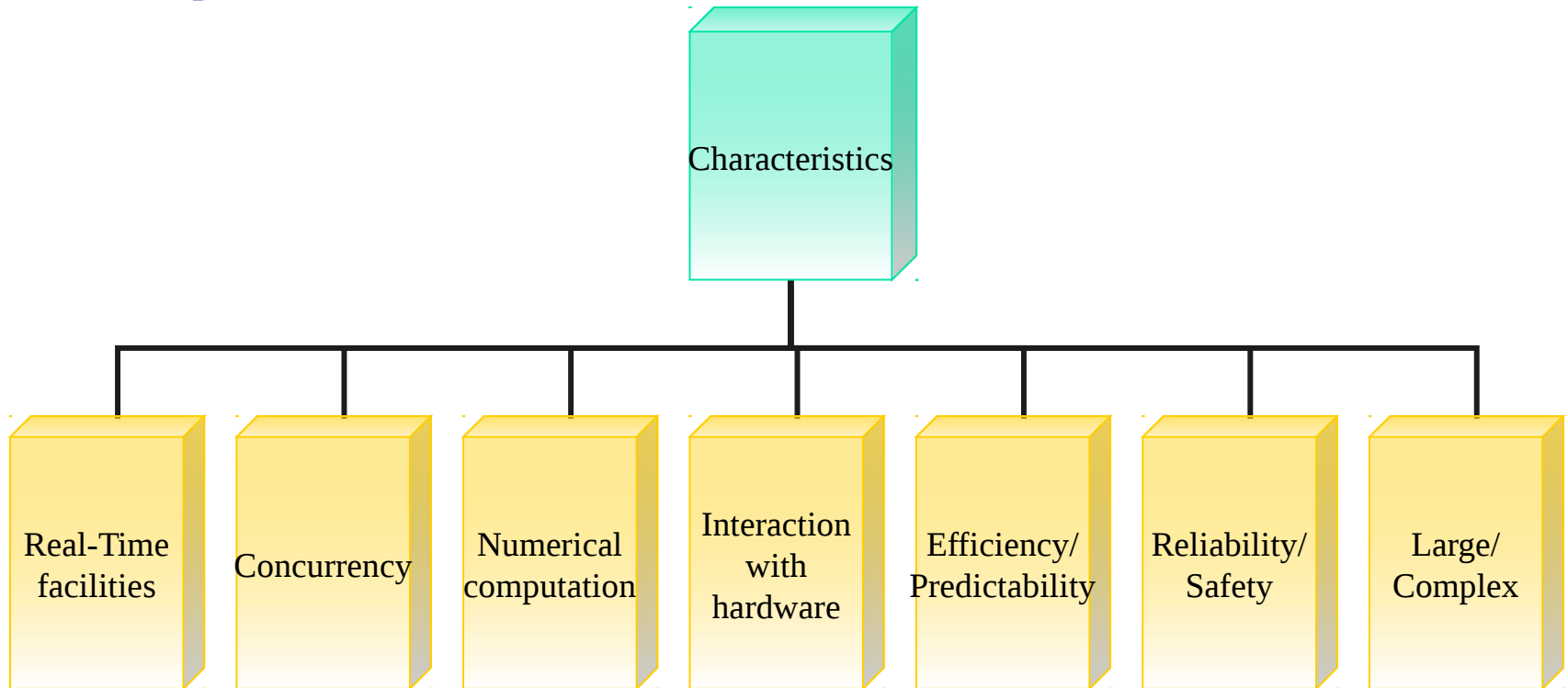
- Esta aula introduz definições chaves e exemplos de sistemas de tempo real
- Os aspectos básicos de um sistemas de tempo real são representados no seguinte diagrama

# Aspectos de Sistemas de Tempo Real





# Aspects of Real-Time Systems





# Exercícios

## ■ Responda

- O que é um sistema de tempo real?
- Diferencie os STR *hard*, *soft* e *firm*.
- Defina a linguagem ADA e que versão ela está?
- A partir do site <http://ideone.com>, compile e execute

```
--Programa Alo Mundo  
  
With Ada.Text_IO; Use Ada.Text_IO;  
  
Procedure AloMundo is  
  
begin  
    Put("Programação em Ada!");  
    New_Line;  
    Put_Line("Exemplo do Olá Mundo!!");  
end AloMundo;
```