Real-Time Systems I

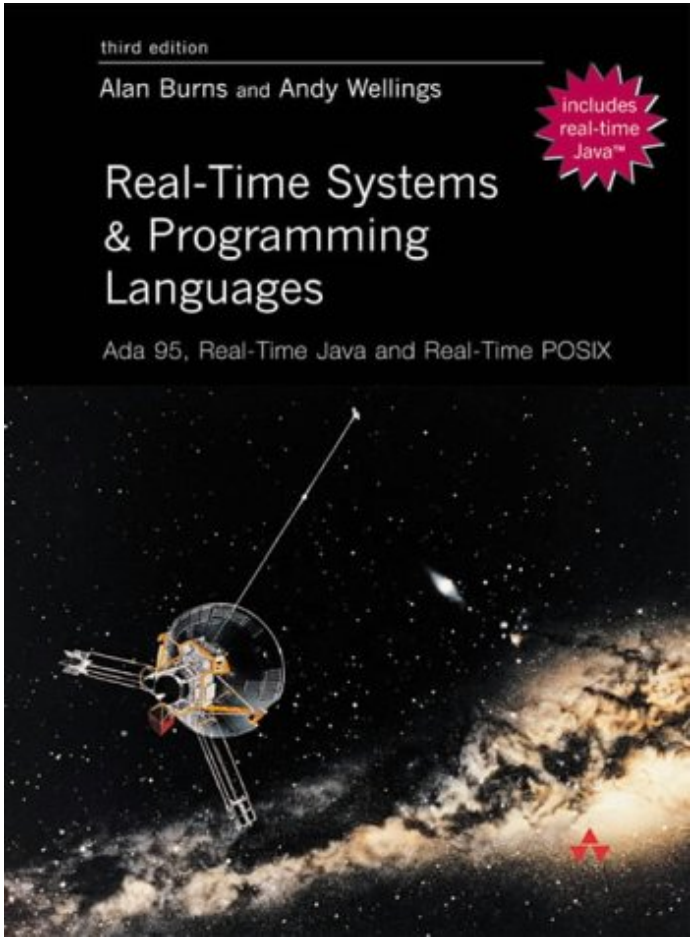
Alan Burns

University of York, UK

Three Talks

- **Introduction to real-time systems**
  - What they are, programming and languages, and some key notions
- **Fixed priority scheduling**
  - Three important analysis techniques and protocols
- **Scheduling the CAN bus**
  - Industrial application of the approach, but
  - Approach was flawed

Books


third edition
Alan Burns and Andy Wellings
includes real-time Java™
Real-Time Systems & Programming Languages
Ada 95, Real-Time Java and Real-Time POSIX

What is a real-time system?

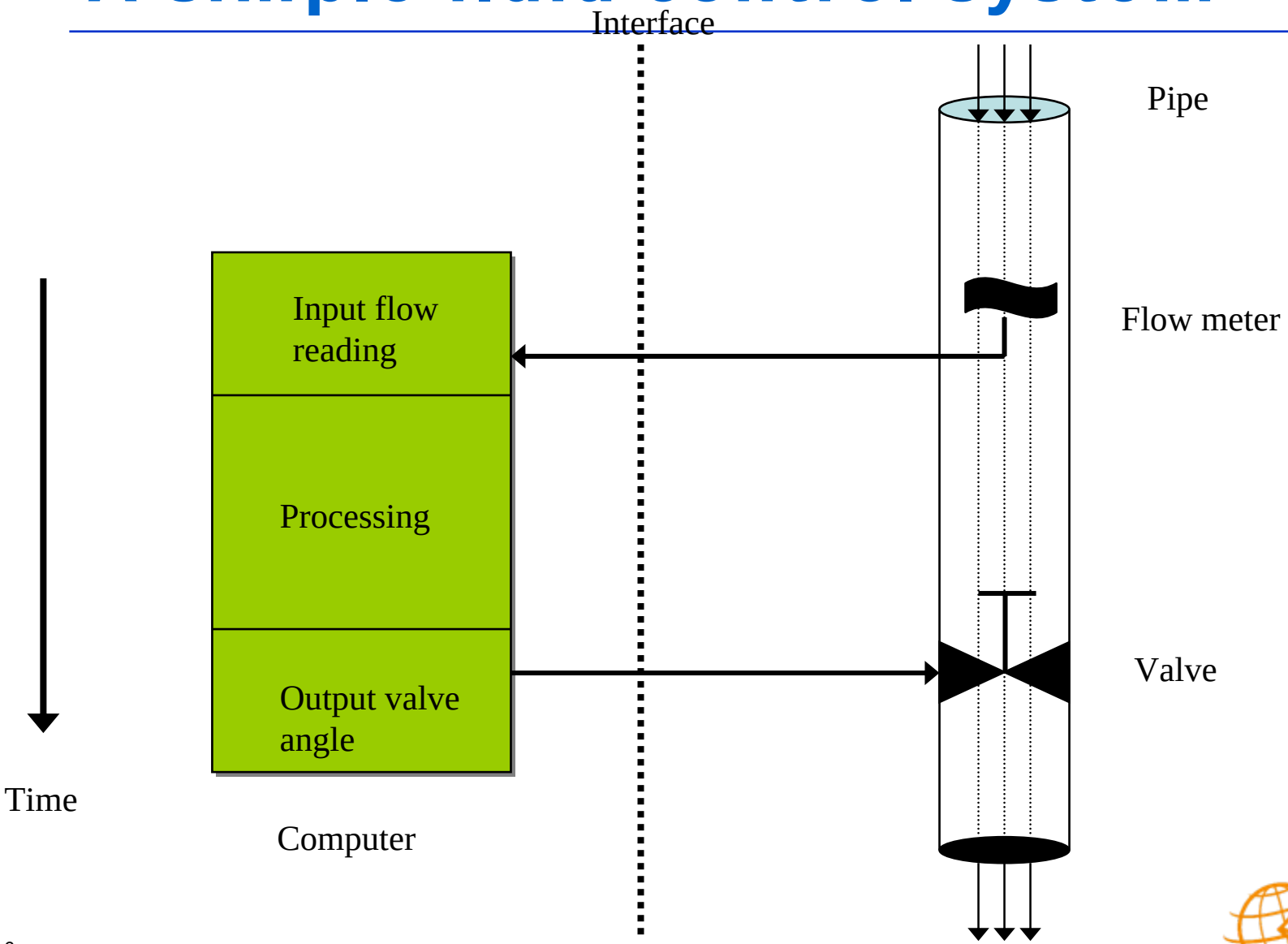
- **A real-time system is any information processing system which has to respond to externally generated input stimuli within a finite and specified period**
  - the correctness depends not only on the logical result but also the time it was delivered
  - failure to respond is as bad as the wrong response!
- **The computer is a component in a larger engineering system => EMBEDDED COMPUTER SYSTEM**
- **99% of all processors are for the embedded systems market**

Terminology

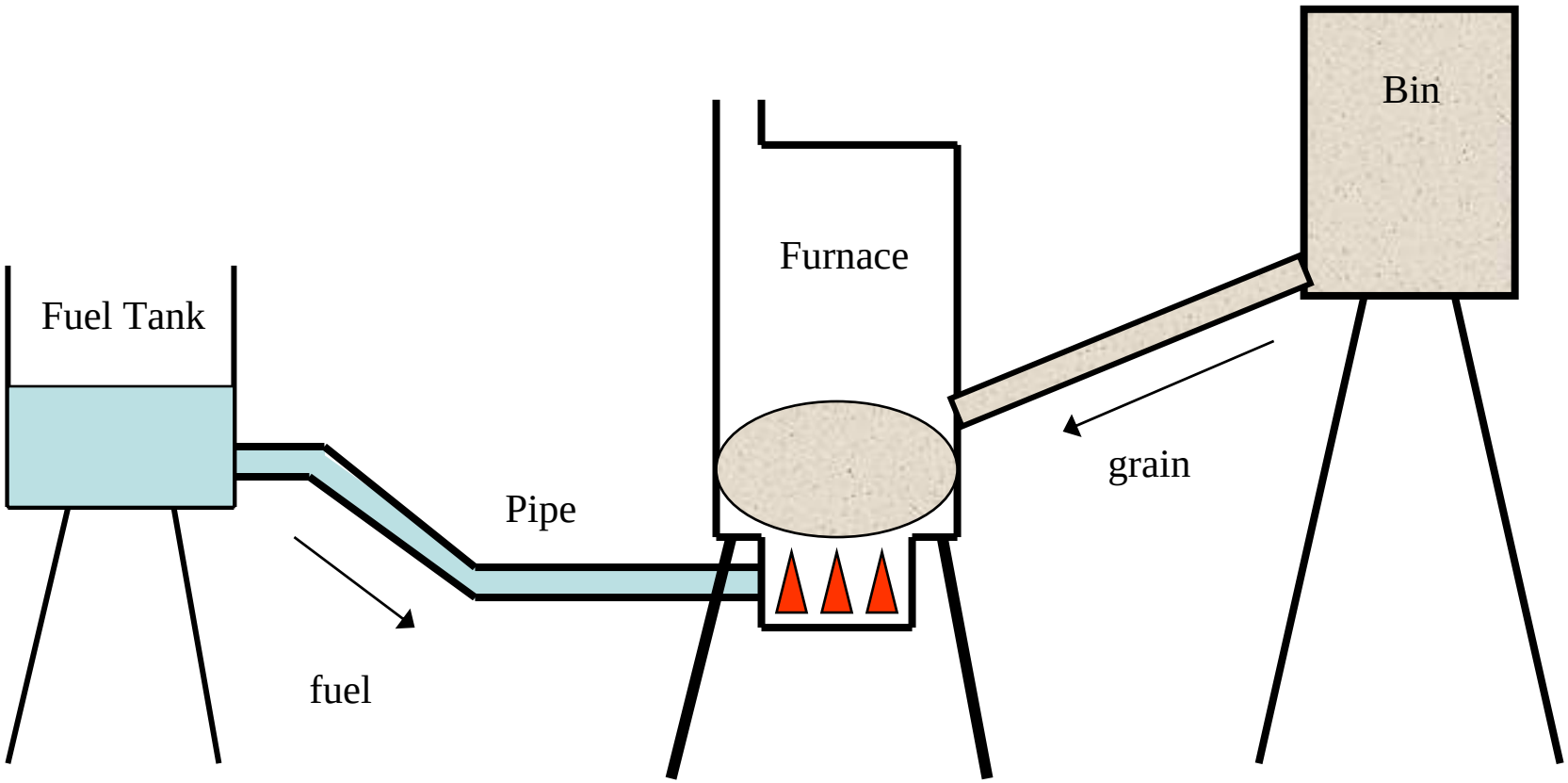
- **Hard real-time — systems where it is absolutely imperative that responses occur within the required deadline. E.g. Flight control systems.**
- **Soft real-time — systems where deadlines are important but which will still function correctly if deadlines are occasionally missed. E.g. Data acquisition system.**
- **Real real-time — systems which are hard real-time and which the response times are very short. E.g. Missile guidance system.**
- **Firm real-time — systems which are soft real-time but in which there is no benefit from late delivery of service.**

A single system may have all hard, soft and firm real-time subsystems

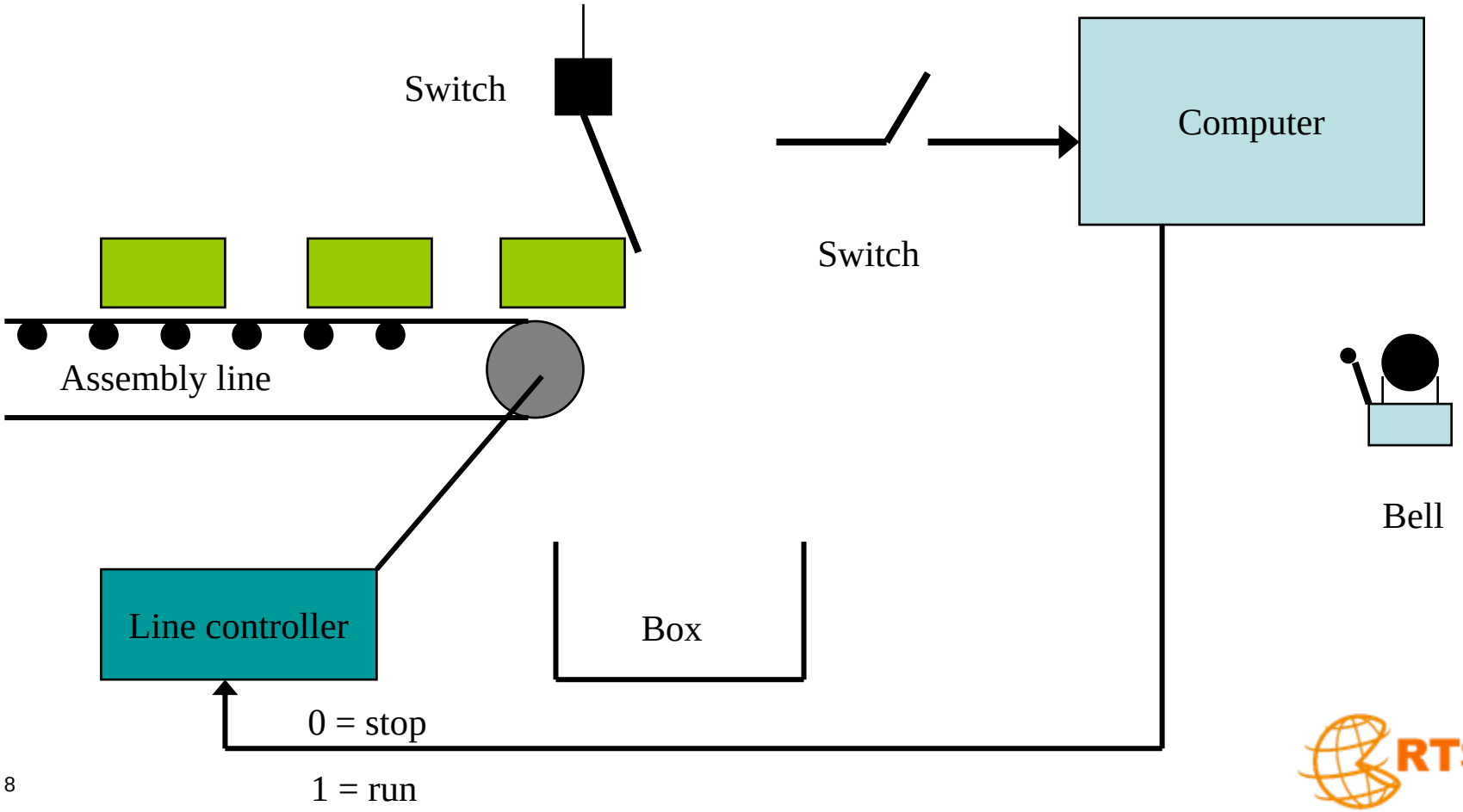
A simple fluid control system



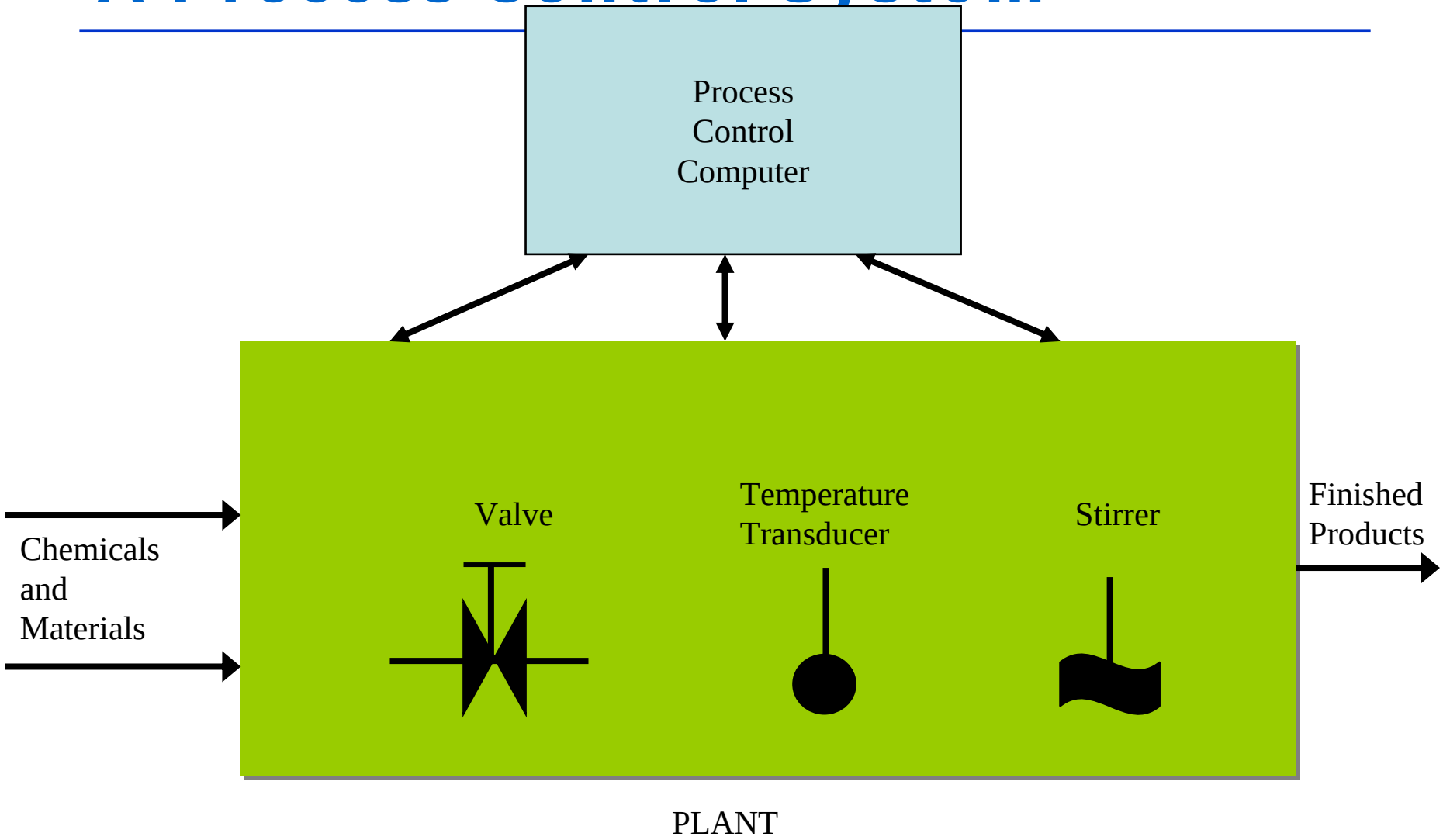
A Grain-Roasting Plant



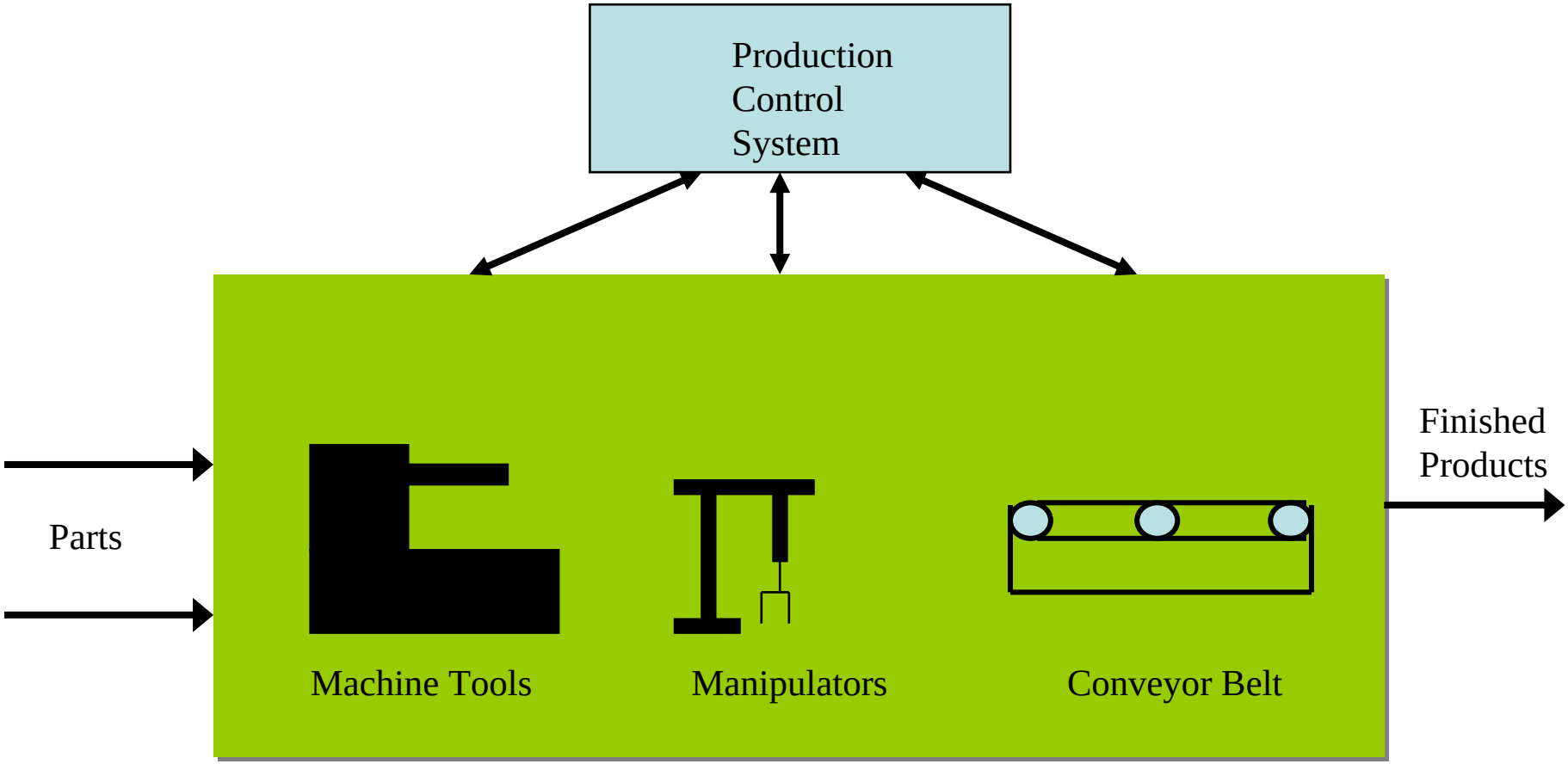
A Widget-Packing Station



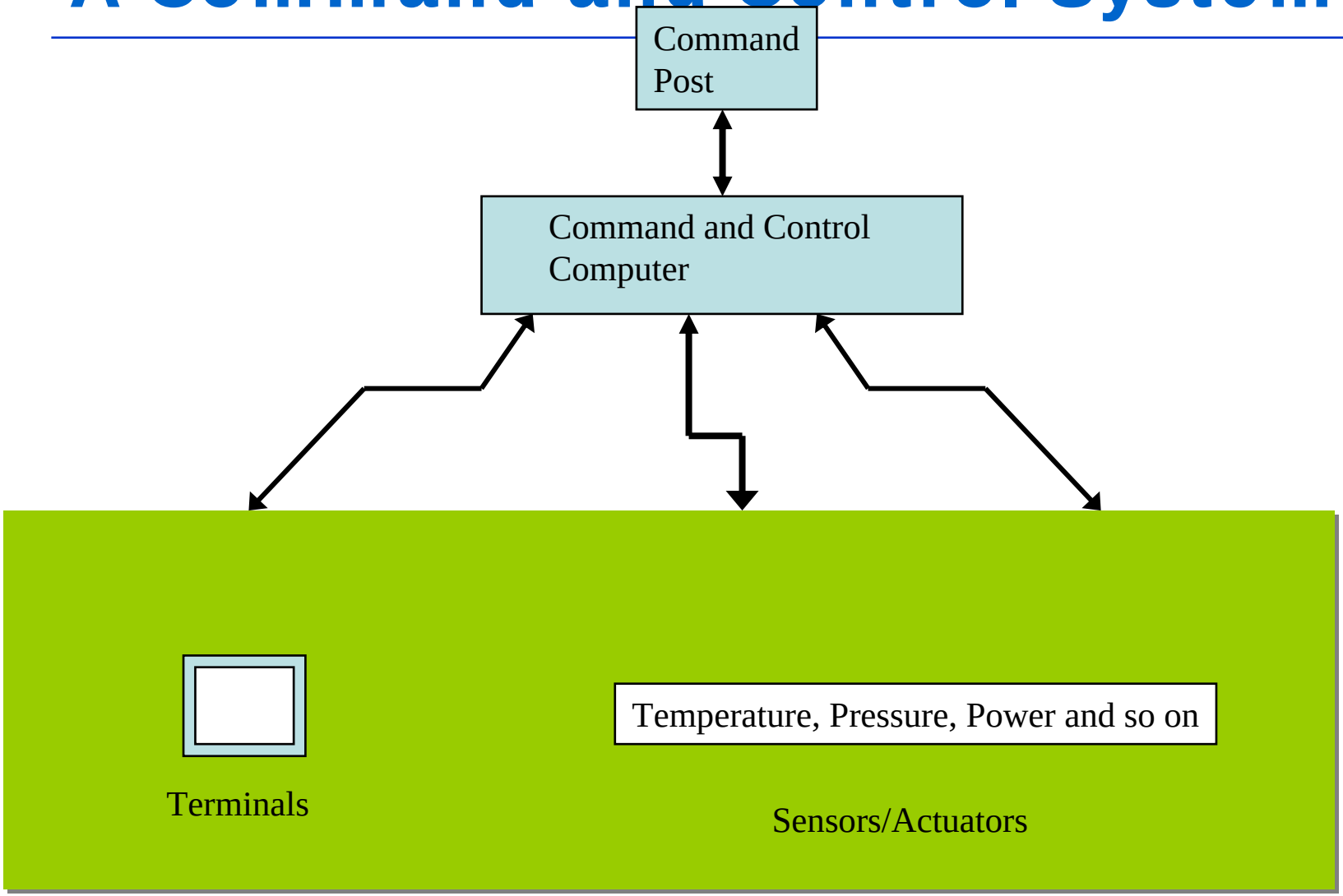
A Process Control System



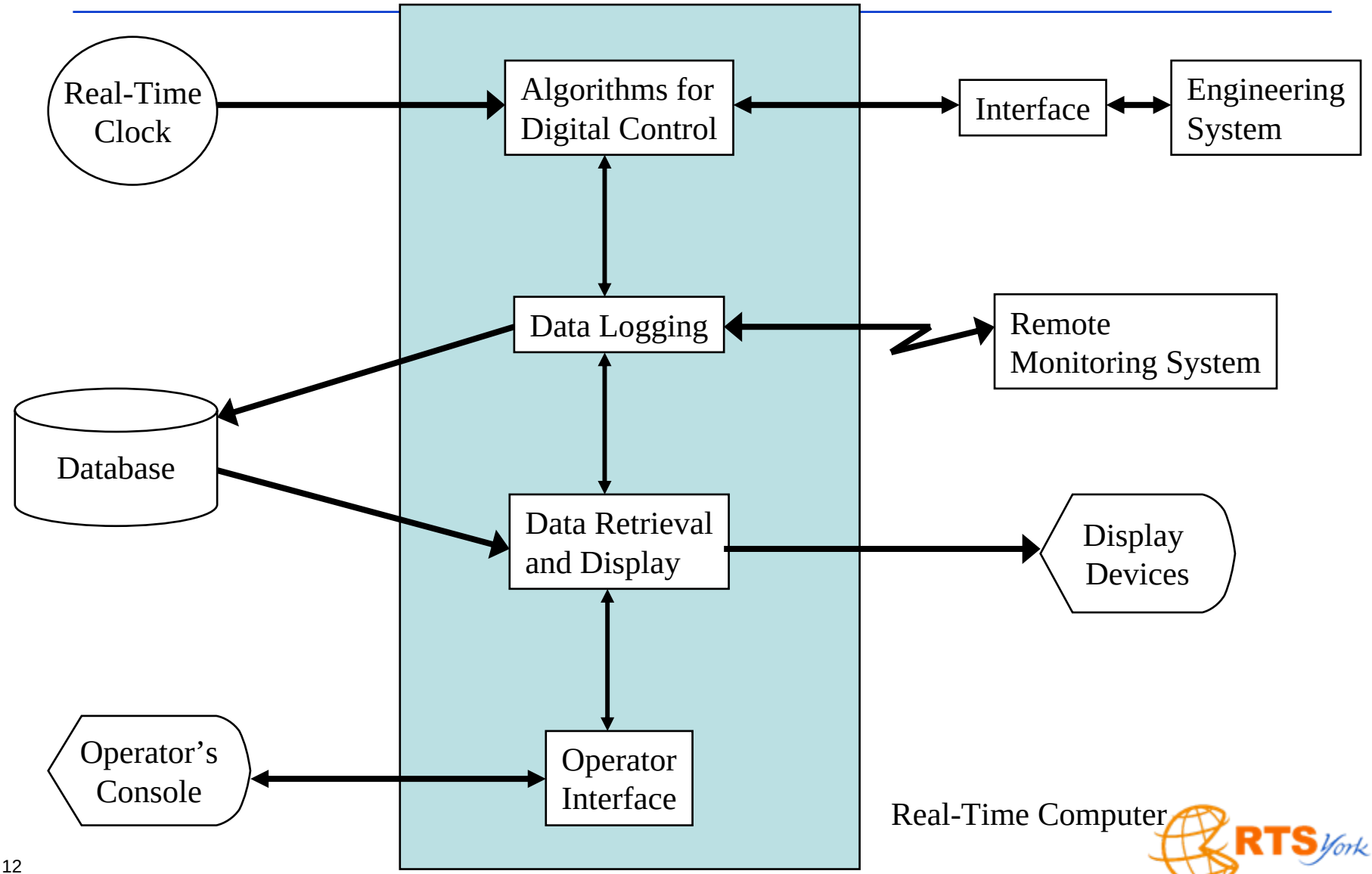
A Production Control System



A Command and Control System



A Typical Embedded System



Other Real-Time Systems

- **Multi-media systems**
  - Including mobile devices
- **Cyber-physical systems**
  - Linking web-based information and the sensed physical world

Characteristics of RTS software

- Large and complex — vary from a few hundred lines of assembler or C to 20 million lines of Ada estimated for the Space Station Freedom
- Concurrent control of separate system components — devices operate in parallel in the real-world; better to model this parallelism by concurrent entities in the program
- Facilities to interact with special purpose hardware — need to be able to program devices in a reliable and abstract way

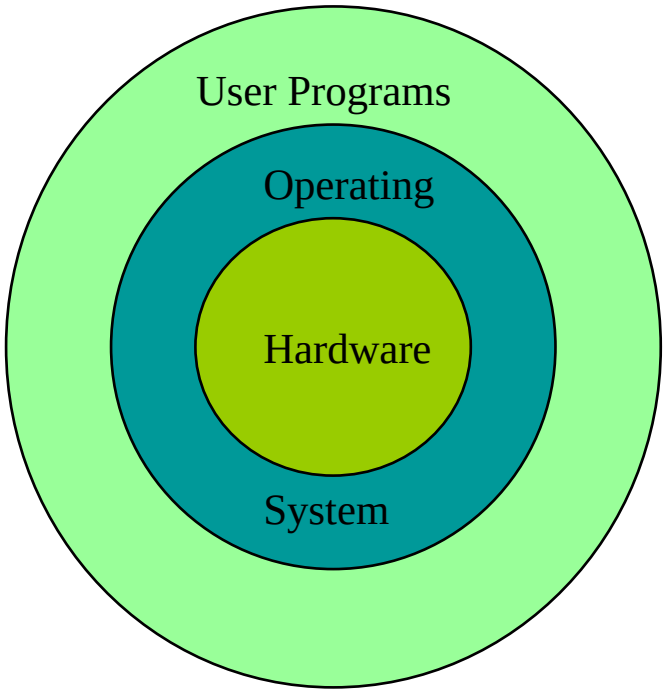
Characteristics of RTS software

- Extreme reliability and safe — embedded systems typically control the environment in which they operate; failure to control can result in loss of life, damage to environment or economic loss
- Guaranteed response times — we need to be able to predict with confidence the worst case response times for systems; efficiency is important but predictability is essential

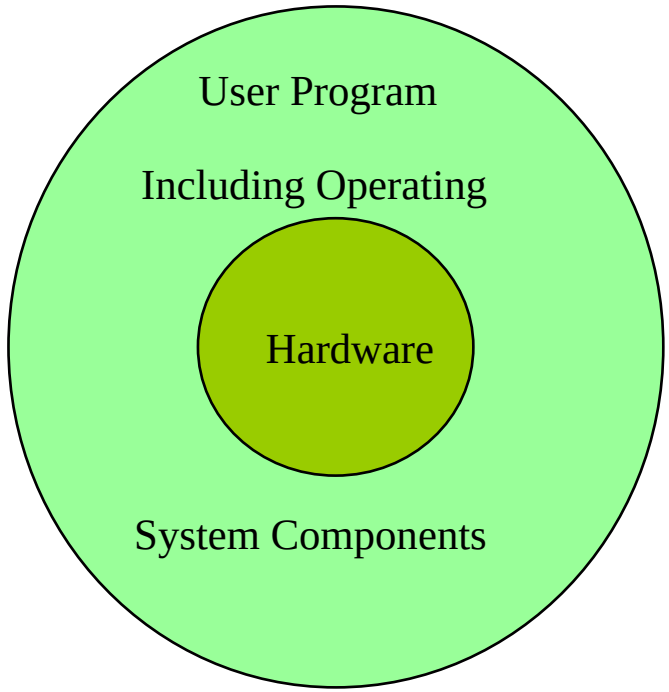
Programming Languages

- **Assembly languages**
- **Sequential systems implementation languages — e.g. RTL/2, Coral 66, Jovial, C, C++.**
  - Normally require operating system support.
- **High-level concurrent languages. e.g. Ada, Chill, Modula-2, Mesa, Java.**
  - No operating system support!

Real-Time Languages and OSs



Typical OS Configuration



Typical Embedded Configuration

Requirements Summary

- **The basic characteristics of a real-time or embedded computer system are:**
  - largeness and complexity
  - extreme reliability and safety
  - concurrent control of separate system components
  - real-time control
  - interaction with hardware interfaces
  - efficient implementation

Language Design Issues

- **Large and complex**
  - Modularity, ADTs, OO, interfaces etc
- **Concurrency**
  - Control over shared objects
- **Reliability**
  - Exception handling
- **Low-level Programming**
  - Device driving and interrupt handling
- **Real-time control**

Real-Time Control

- **Interaction with ‘time’**
- **Delays**
  - Wait for world to catch up
- **Deadlines**
  - Keep up with the world
- **Scheduling**
  - Make sufficient progress to meet deadlines
  - Use resources (CPUs, networks etc) effectively and predictably

Control Example

```
task body Temp_Controller is
  TR : Temp_Reading; HS : Heater_Setting;
  Next : Time;
  Interval : Time_Span := Milliseconds(30);
  Finish : Time_Span := Milliseconds(20);
begin
  Next := Clock;  -- start time
  loop
    Read(TR);
    Temp_Convert(TR,HS);
    Write(HS);
    deadline(Next + Finish);
    Next := Next + Interval;
    delay until Next;
  end loop;
end Temp_Controller;
```

Scheduling

- **In general, a scheduling scheme provides two features:**
  - An algorithm for ordering the use of system resources (in particular the CPUs)
  - A means of predicting the worst-case behaviour of the system when the scheduling algorithm is applied
- **The prediction can then be used to confirm the temporal requirements of the application**

Scheduling Techniques

- **Fixed priority scheduling**
  - Next lecture
- **Earliest Deadline First (EDF)**