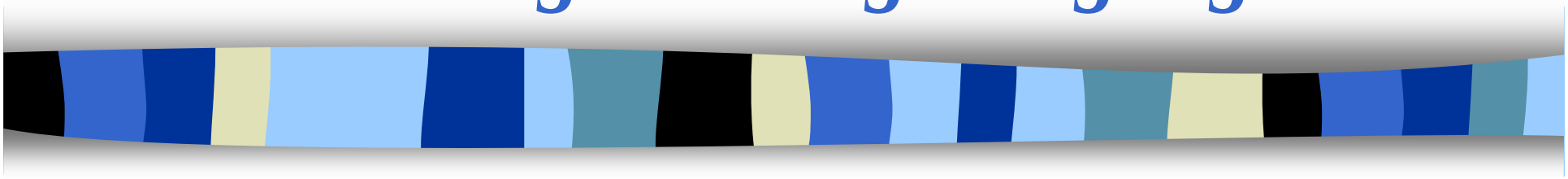
Real-Time Systems and Programming Languages



- Buy Real-Time Systems: Ada 95, Real-Time Java and Real-Time POSIX by Burns and Wellings
- See www.cs.york.ac.uk/rts/RTSBookThirdEdition.html
- Foils: /usr/course/rts/foils
- Practicals: email me your preference

Prerequisites

- Basic understanding of Ada and C
- Basic understanding of Computer Architectures.
- Basic understanding of Operating Systems

Course Aims

- Understanding of the broad concept
- Practical understanding for industry
- To stimulate research interest

Overall Technical Aims of the Course

- To understand the basic requirements of real-time systems and how these requirements have influenced the design of real-time programming languages and real-time operating systems.
- To understand the implementation and analysis techniques which enable the requirements to be realized.

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

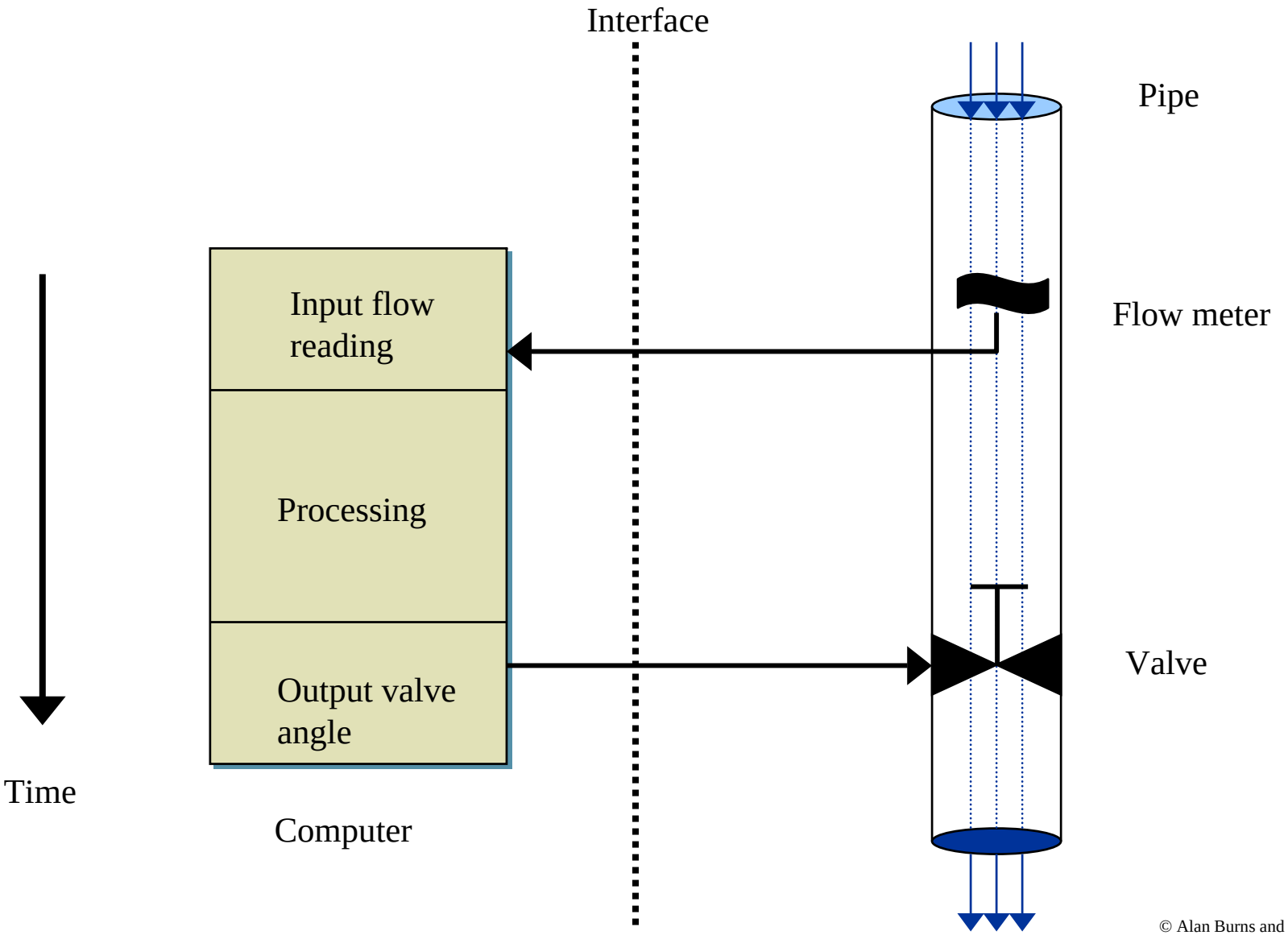
Terminology

- 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.

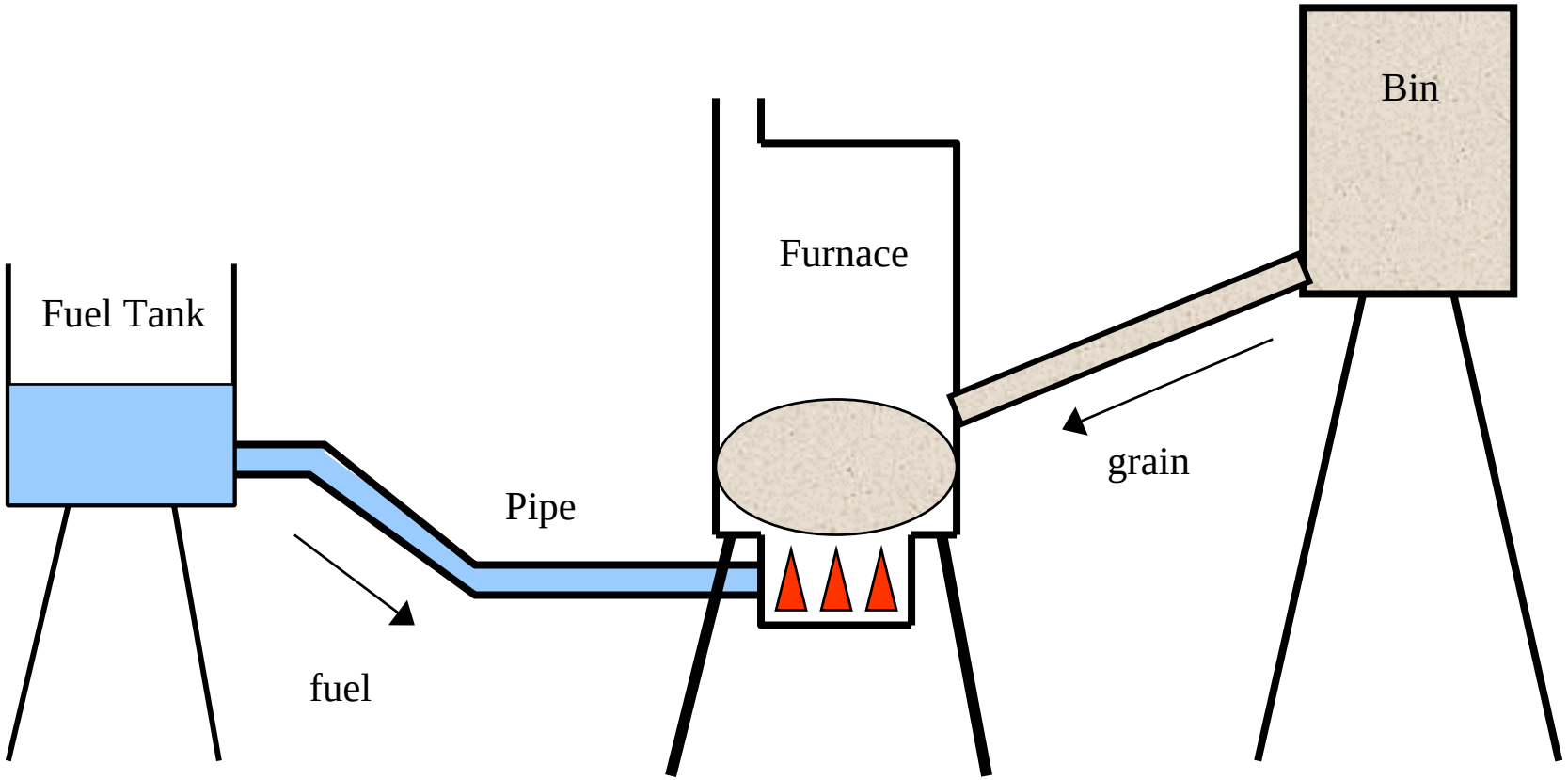
A single system may have all hard, soft and real real-time subsystems

In reality many systems will have a cost function associated with missing each deadline

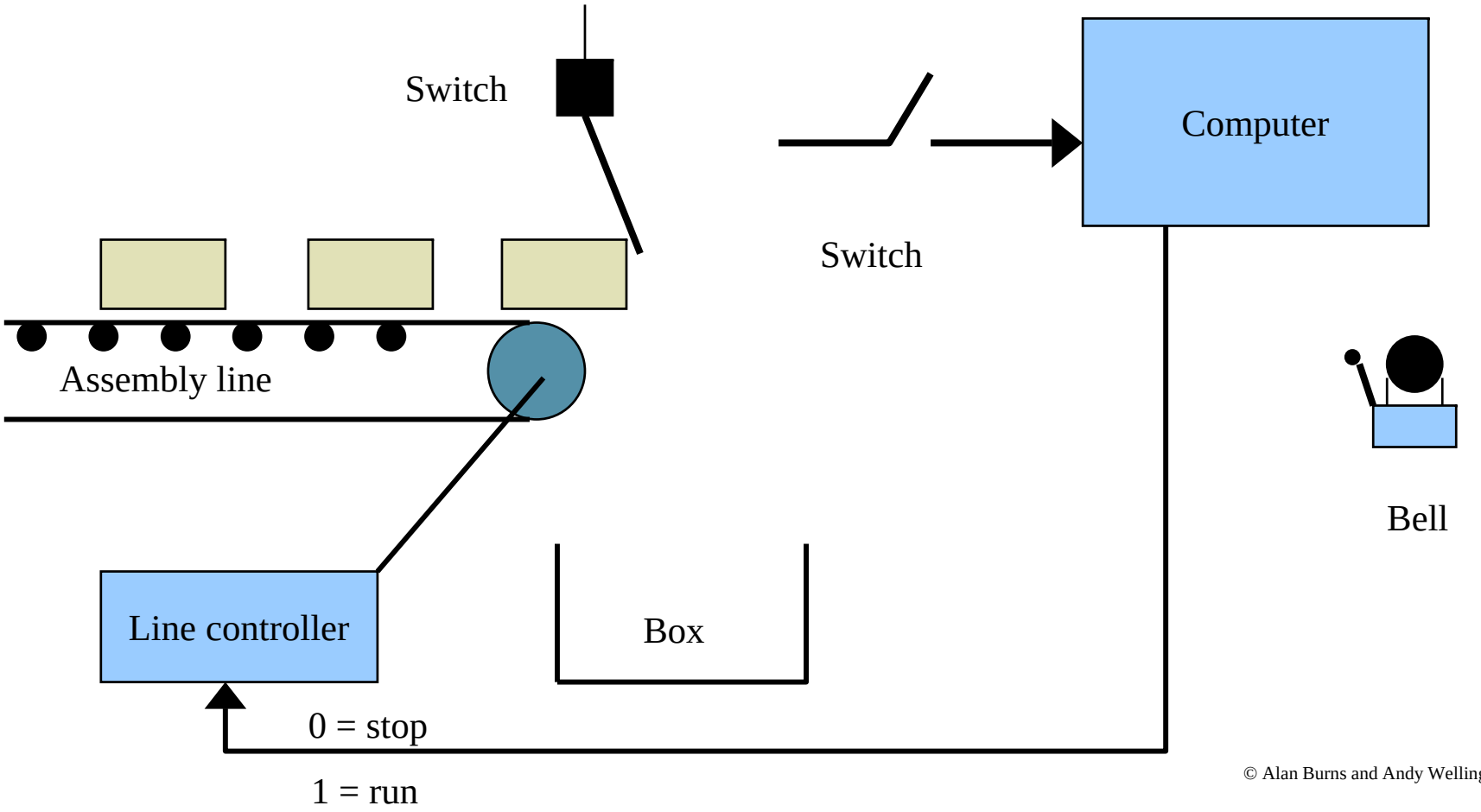
A simple fluid control system



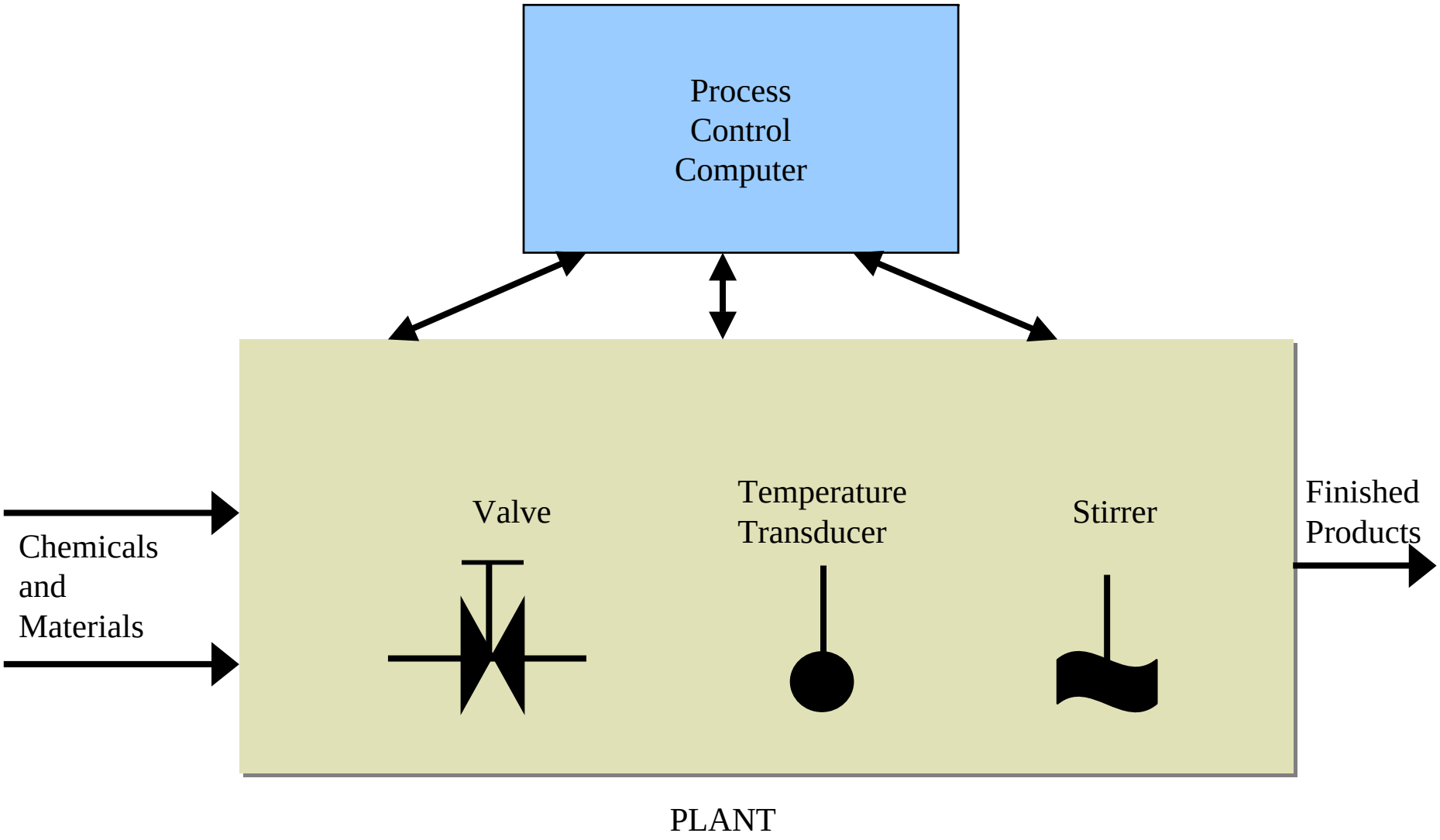
A Grain-Roasting Plant



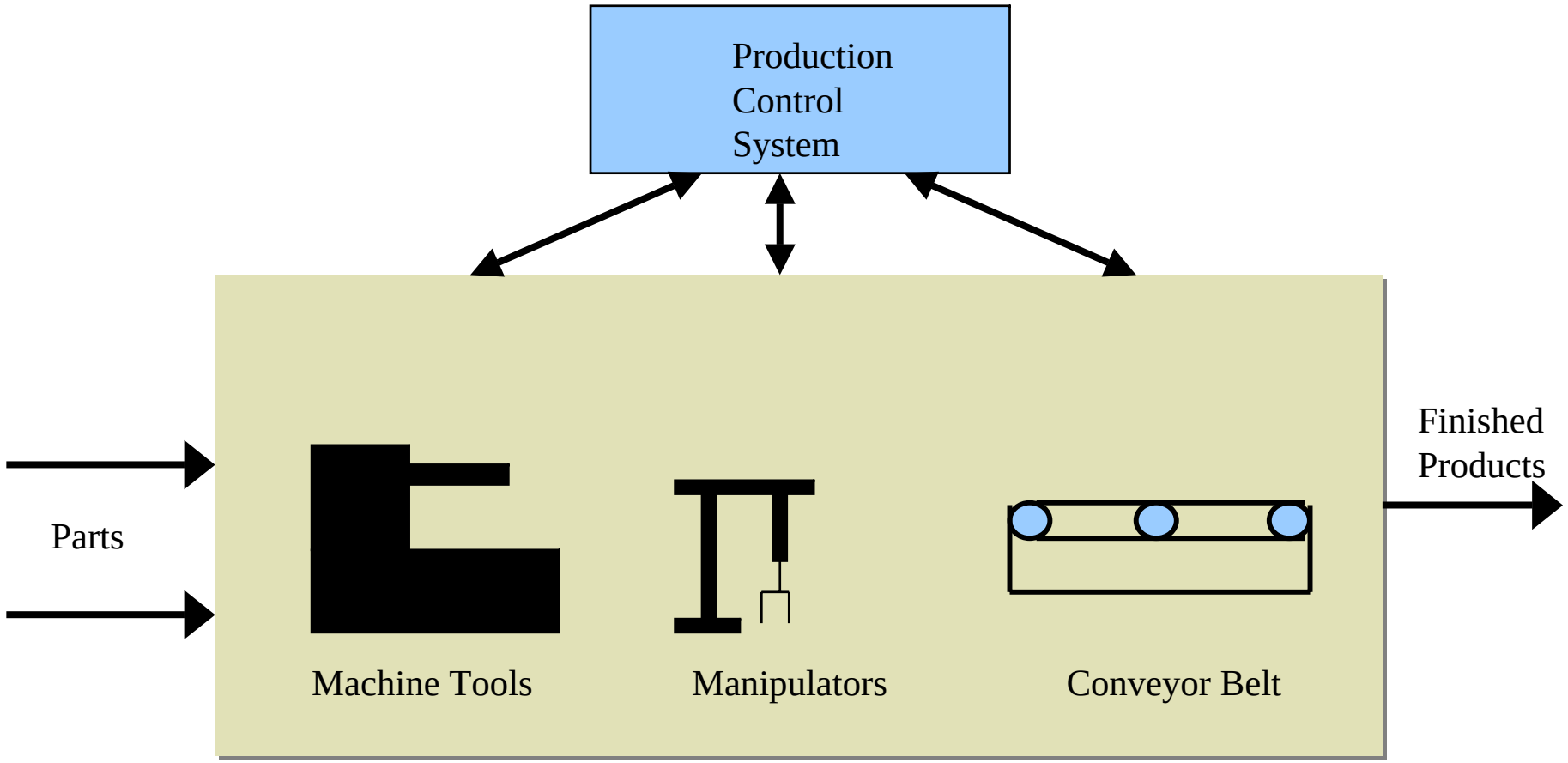
A Widget-Packing Station



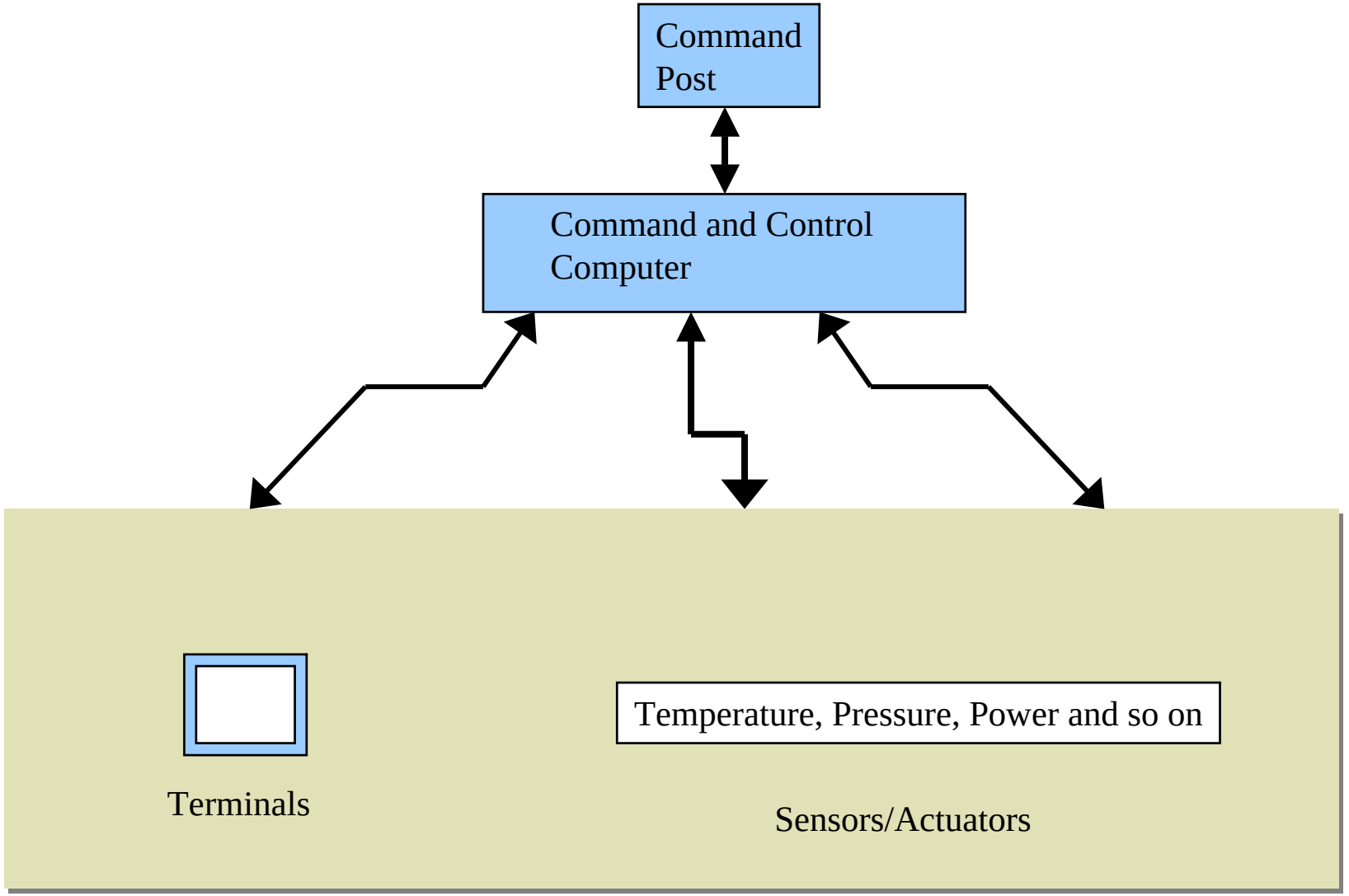
A Process Control System



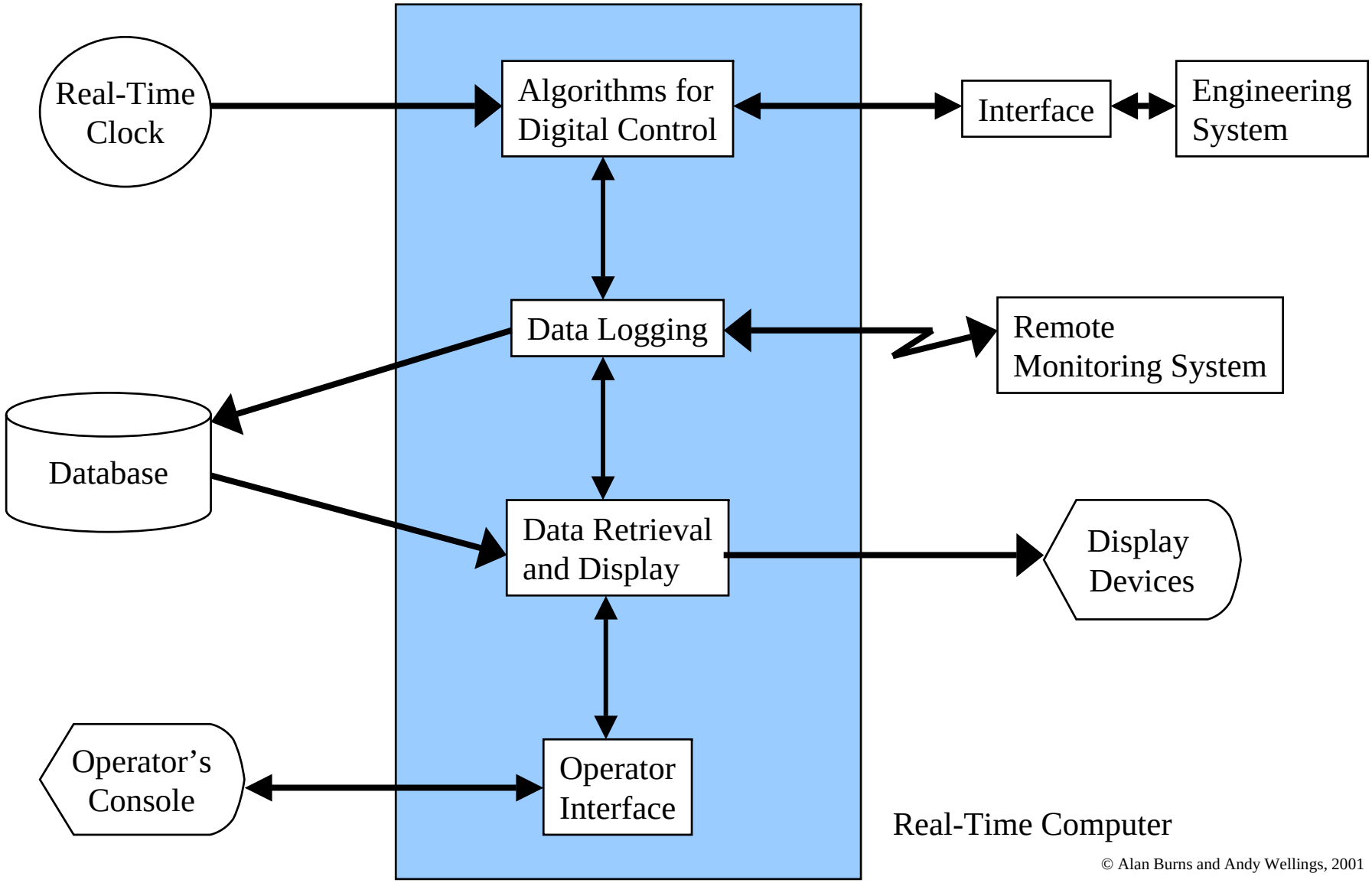
A Production Control System



A Command and Control System



A Typical Embedded System



Characteristics of a RTS

- 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

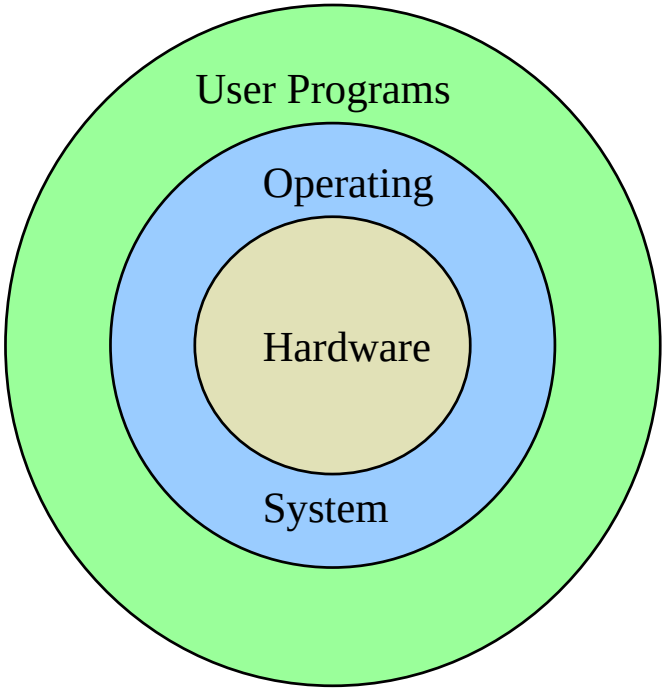
Characteristics of a RTS

- 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

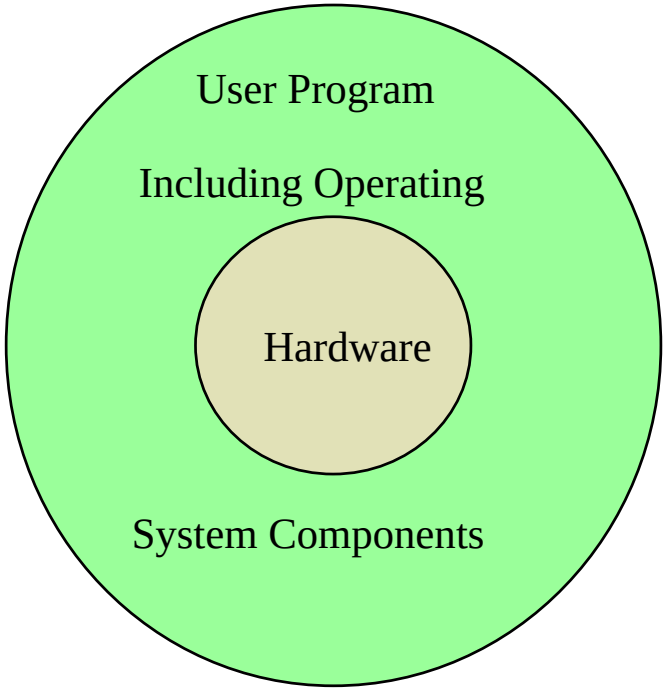
Real-time Programming Languages

- Assembly languages
- Sequential systems implementation languages — e.g. RTL/2, Coral 66, Jovial, C.
- Both normally require operating system support.
- High-level concurrent languages. Impetus from the software crisis. e.g. Ada, Chill, Modula-2, Mesa, Java.
- No operating system support!
- We will consider:
  - Java/Real-Time Java
  - C and Real-Time POSIX
  - Ada 95
  - Also Modula-1 for device driving

Real-Time Languages and OSs



Typical OS Configuration



Typical Embedded Configuration

Summary

- Two main classes of such systems have been identified:
  - hard real-time systems
  - soft real-time systems
- The basic characteristics of a real-time or embedded computer system are:
  - largeness and complexity,
  - manipulation of real numbers,
  - extreme reliability and safety,
  - concurrent control of separate system components,
  - real-time control,
  - interaction with hardware interfaces,
  - efficient implementation.