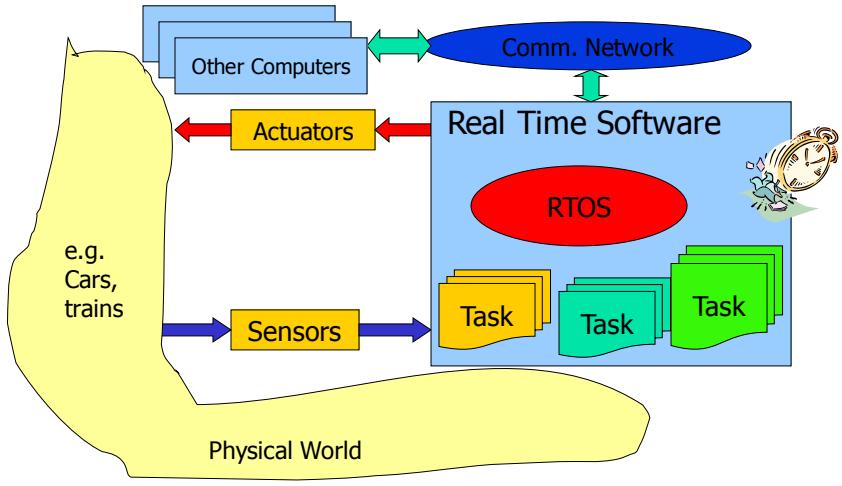
Course Outline (lectures)

- Introduction
  - Characteristics of RTS
- Real Time Operating Systems (RTOS)
  - OS support: tasking, scheduling, resource handling, OSEK
- Real Time Programming Languages
  - Language support, e.g. Ada tasking
- Timing Analysis of RT Software & Scheduling
  - Worst-case execitiom and response time analysis
- Distributed real time systems
  - Real Time Communication: CAN Bus
- Multiprocessor real-time systems (advanced topics)
  - Architectures and real-time scheduling
- Design and Validation (advanced topics)
  - Workload models, Modeling, Verification

Overall Structure of RT Systems

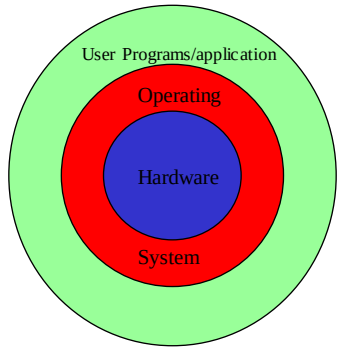
- Hardware (CPU, I/O device etc)
  - a clock!
- A real time OS (function as standard OS, with predictable behavior and well-defined functionality)
- A collection of RT tasks/processes (share resourses, communicate/synchronize with each other and the environment)

Components of RT Systems

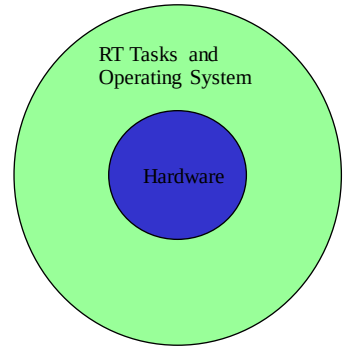


3

General-Purpose vs. Embedded RT Computer Systems



General-purpose computer systems

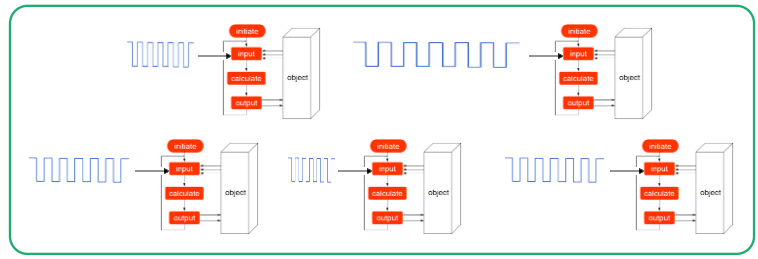


Typical Embedded Configuration
-- Real-Time Systems

4

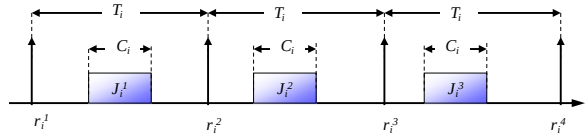
Embedded/Real-Time Software

- Multi-rate real-time tasks



- Each task

Utilization: C_i/T_i



Example: a Car Controller

Activities of a car control system. Let

1. C= worst case execution time
2. T= (sampling) period
3. D= deadline

- Speed measurment: C=4ms, T=20ms (50Hz), D=5ms
- ABS control: C=10ms,T=40ms (25Hz), D=40ms
- Fuel injection: C=40ms, T=80ms (12.5Hz),D=80ms
- Other software with soft deadlines e.g audio, air condition etc

Construct a controller meeting all the deadlines!

Programming the car controller (1)

Process Speed: Loop read sensor,compute,display... sleep (0.02) /*period*/ End loop	Process ABS Loop Read sensor, compute, react sleep(0.04) End loop
Process Fuel Loop read data, compute, inject ... sleep(0.08) End loop	Soft RT Processes Loop read temperature el hiss, stereo .... End loop

7

Any problem?

- We forgot the execution times …

e.g. Process speed:

20ms = execution time + sleep(X)

8

Programming the car controller (2)

Process Speed: Loop next := get-time + 0.02 read sensor,compute,display... sleep until next End loop	Process ABS Loop next:=get-time + 0.04 Read sensor, compute, react sleep until next End loop
Process Fuel Loop next:=get-time + 0.08 read data, compute, inject ... sleep until next End loop	Soft RT Processes Loop read temperature elevator, stereo .... End loop

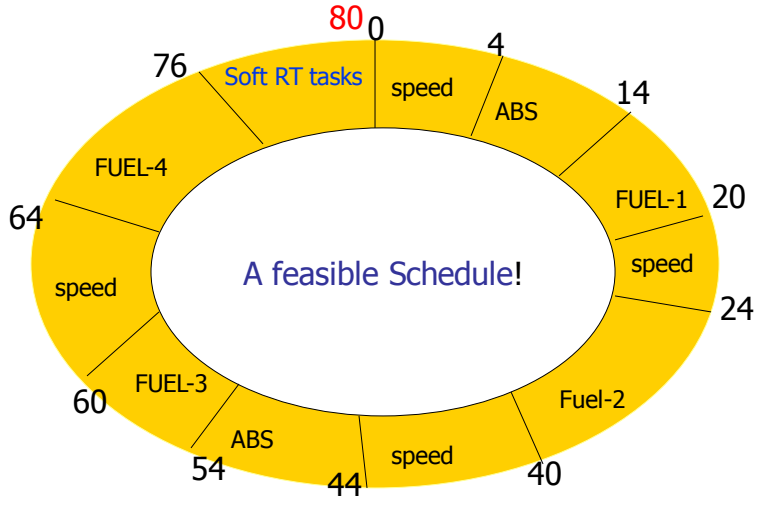
9

To guarantee the deadlines are not violated:

- We need to know the execution times
- We need to do schedulability analysis
- We need to construct a schedule

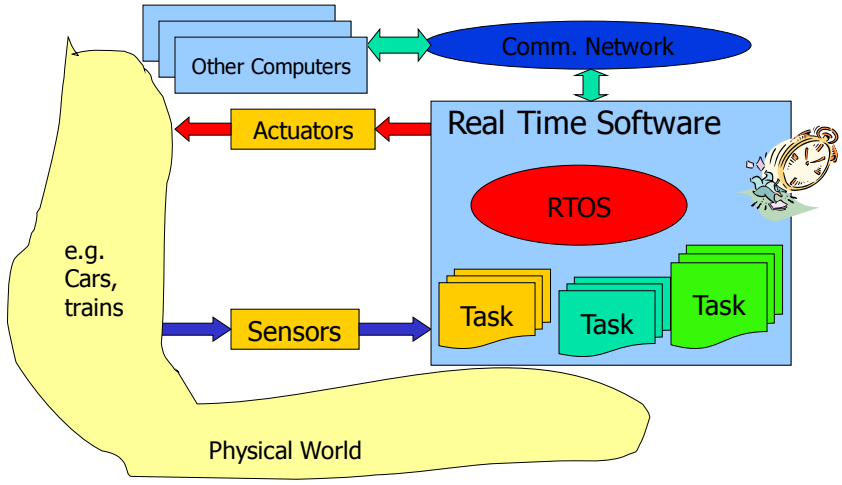
10

Programming the car controller (3)



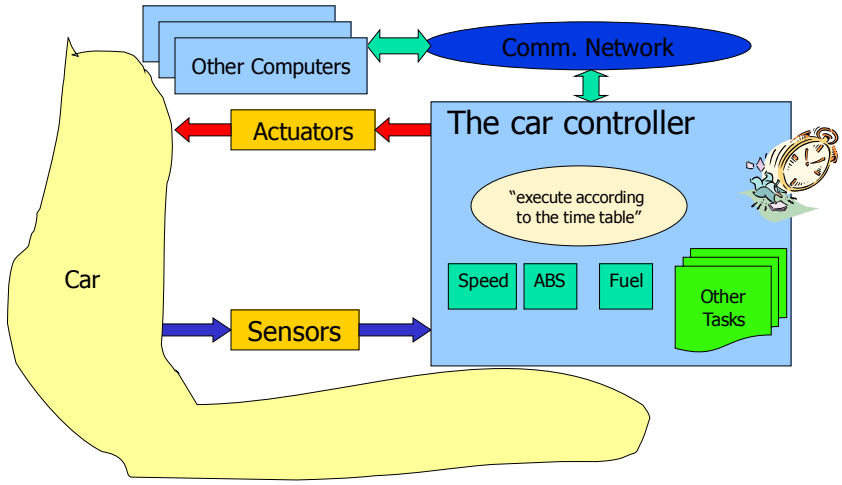
11

Components of RT Systems



12

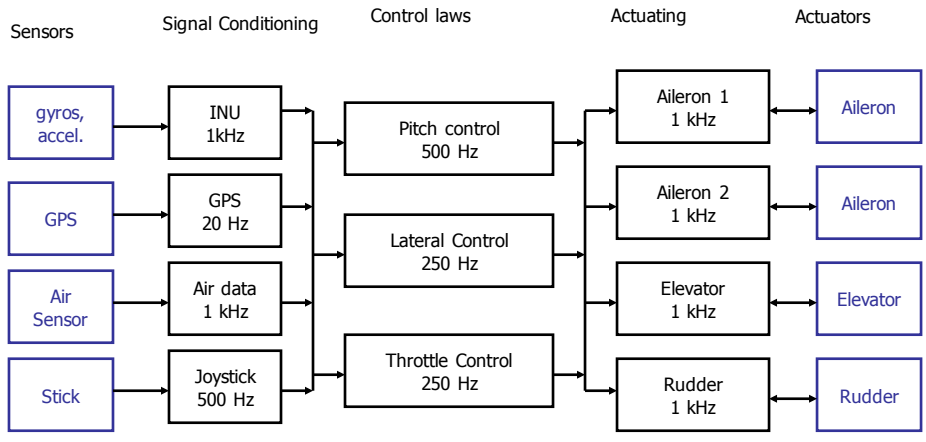
Components of RT Systems



13

Example: Fly-by-wire Avionics:

Hard real-time system with multi-rate tasks (Hz = #cycles per sec)



Challenges in RT Systems Design

- Predictability: the system behaviour is known before it is put into operation!
  e.g. Response times, deadlock freedom etc
- Testability: easy to test e.g. any deadline miss?
- Cost optimality: e.g. Energy consumption, memory blocks etc

15

Main desirable properties of RT Systems (2)

- Maintainability: modular structure to ease system modification
- Robustness: must not collapse when subject to peak load, exception, manage all possible scenarios
- Fault tolerance: hardware and software failures should not cause the system to crash - function down-grading

16

Certisfiability: provide clear evidance to show your system works reliably and safe according to ISO standards

--- predictability is the key challenge

17

Difficult to achieve predictability: Hardware and OS

- Cache sharing, processor pipelines, multicores, DMA …
- Interrupt handling may introduce unbounded delays
- Priority inversion (low-prority tasks blocking high-prior taskts)
- Memory management (static allocation may not be enough, dynamic data structures e.g. Queue), no virtual memory
- Communication delays in a distributed environment
- New hardware platforms … Multicores
  - system updates?

18

Difficult to achieve predictability: Software

- Difficult to calculate the worst case execution time for tasks (theoretically impossible, halting problem)
  - Avoid dynamic data structures
  - Avoid recursion
  - Bounded loops e.g. For-loops only
- Complex synchronization patterns between tasks: potential deadlocks (formal verification)
- Multi-tasking, tasks that share resources

- New software components
  - Software updates?

19