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

- **Scheduling and Timing Analysis**
  - Worst-case execution and response time analysis

- **Distributed real time systems**
  - Real Time Communication: CAN Bus

- **Workload Models (advanced topic)**
  - Graph-based task models

- **Multiprocessor real-time systems (advanced topics)**
  - Architectures and real-time scheduling

- **Design and Validation (advanced topics)**
  - Modeling and 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 resources, communicate/synchronize with each other and the environment)
Components of RT Systems

- Actuators
- Sensors
- Physical World: e.g., Cars, trains
- Other Computers
- Communication Network
- Real Time Software
  - RTOS
  - Tasks

Physical World
General-Purpose vs. Embedded RT Computer Systems

General-purpose computer systems

Typical Embedded Configuration
-- Real-Time Systems
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 = resource budget (worst case execution time)
2. T = period (rate, 1/period)
3. D = deadline

- Speed control: C=4ms, T=20ms (50Hz), D=5ms
- Brake control: C=10ms, T=40ms (25Hz), D=40ms
- Engine control: 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)

<table>
<thead>
<tr>
<th>Process Speed:</th>
<th>Process Brake</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop</td>
<td>Loop</td>
</tr>
<tr>
<td>read sensor, compute, display...</td>
<td>Read sensor, compute, react</td>
</tr>
<tr>
<td>sleep (0.02) /<em>period</em>/</td>
<td>sleep(0.04)</td>
</tr>
<tr>
<td>End loop</td>
<td>End loop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process Engine</th>
<th>Soft RT Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop</td>
<td>Loop</td>
</tr>
<tr>
<td>read data, compute, inject ...</td>
<td>read temperature</td>
</tr>
<tr>
<td>sleep(0.08)</td>
<td>el hiss, stereo</td>
</tr>
<tr>
<td>End loop</td>
<td>....</td>
</tr>
<tr>
<td></td>
<td>End loop</td>
</tr>
</tbody>
</table>
Any problem?

- We forgot the execution times ...

  e.g. Process speed:

  \[ 20 \text{ms} = \text{execution time} + \text{sleep}(X) \]
### Programming the car controller (2)

<table>
<thead>
<tr>
<th>Process Speed: Loop</th>
<th>Process Brake Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>next := get-time + 0.02</td>
<td>next := get-time + 0.04</td>
</tr>
<tr>
<td>read sensor, compute, display...</td>
<td>Read sensor, compute, react</td>
</tr>
<tr>
<td>sleep until next</td>
<td>sleep until next</td>
</tr>
<tr>
<td>End loop</td>
<td>End loop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process Engine Loop</th>
<th>Soft RT Processes Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>next := get-time + 0.08</td>
<td>read temperature</td>
</tr>
<tr>
<td>read data, compute, inject ...</td>
<td>elevator, stereo</td>
</tr>
<tr>
<td>sleep until next</td>
<td>....</td>
</tr>
<tr>
<td>End loop</td>
<td>End loop</td>
</tr>
</tbody>
</table>
To ensure that the deadlines are not violated:

- We need to know the execution times
- We need to do schedulability analysis
- We need to construct a schedule
Programming the car controller (3)

A feasible Schedule!
Components of RT Systems

Actuators

Sensors

Car

Other Computers

Comm. Network

The car controller

“execute according to the time table”

Speed
Brake
Engine

Other Tasks
Example: Fly-by-wire Avionics:

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

Sensors | Signal Conditioning | Control laws | Actuating | Actuators
---|---|---|---|---
gyros, accel. | INU 1kHz | Pitch control 500 Hz | Aileron 1 1 kHz | Aileron
GPS | GPS 20 Hz | Lateral Control 250 Hz | Aileron 2 1 kHz | Aileron
Air Sensor | Air data 1 kHz | Throttle Control 250 Hz | Elevator 1 kHz | Elevator
Stick | Joystick 500 Hz | | Rudder 1 kHz | Rudder
Challenges in RT Systems Design

- **Predictability**: the system behaviour is known before it is put into operation!
  
  e.g. Response times, deadlock freedom etc

- **Certisfiability**: provide clear evidence to show your system works reliably and safe according to ISO standards

  --- predictability is the key challenge
Challenges in RT Systems Design

- **Cost optimality**: e.g. Energy consumption, memory blocks etc
- **Testability**: easy to test e.g. any deadline miss?
- **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 - down-grading
- **Maintainability**: modular structure for local changes
Difficult to achieve predictability: Hardware and OS

- Cache sharing, processor pipelines, multicores, DMA ...
- Interrupt handling may introduce unbounded delays
- Priority inversion (low-priority tasks blocking high-priority tasks)
- 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
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
- Software components provided by a third party