Real-time Software
Objectives

- To explain the concept of a real-time system and why these systems are usually implemented as concurrent processes
- To describe some design issues
- To introduce generic process architectures for monitoring and control and data acquisition systems
Topics covered

- Real-time systems
- Design issues
- Monitoring and control systems
- Data acquisition systems
Real-time systems

- Systems which **monitor and control** their environment.
- Inevitably associated with hardware devices
  - **Sensors**: Collect data from the system environment;
  - **Actuators**: Change (in some way) the system's environment;
- **Time is critical**. Real-time systems **MUST** respond within specified times.
Definition: real-time system

A real-time system is a software system where the correct functioning of the system depends on:

- the results produced by the system
- and the time at which these results are produced.
Definition: **Soft & Hard real-time systems**

- A **soft real-time system** is a system whose operation is *degraded* if results are not produced according to the specified timing requirements.
- A **hard real-time system** is a system whose operation is *incorrect* if results are not produced according to the timing specification.
Stimulus/Response Systems

- Given a stimulus, the system must produce a response within a specified time.

- Two classes of stimuli:
  - Periodic stimuli
  - Aperiodic stimuli
Periodic stimuli

- Occur at **predictable time intervals**
- Provide info on the system’s **state and environment**
- Example: a temperature sensor may be **polled 10 times per second** - The system takes action according to the sensor value (stimulus).
Aperiodic stimuli

- Occur at unpredictable times (irregular)
- Often indicate some exceptional conditions
- May be generated either by sensors or actuators.
- Example: a system power failure may trigger an interrupt which must be processed by the system.
Architectural considerations

- Because of the need to respond to timing demands made by different stimuli/responses, the system architecture must allow for fast switching between stimulus handlers.
- Timing demands of different stimuli are different so a simple sequential loop is not usually adequate.
- Real-time systems are therefore usually designed as cooperating processes with a real-time executive controlling these processes.
A real-time system model
Sensor/actuator processes

Sensor

Stimulus

Sensor control

Data processor

Actuator control

Actuator

Response
System elements

- Sensor control processes
  - Collect information from sensors. May buffer information collected in response to a sensor stimulus.

- Data processor
  - Carries out processing of collected information and computes the system response.

- Actuator control processes
  - Generates control signals for the actuators.
Real-time programming

- **Hard-real time** systems may have to be programmed in **assembly** language to ensure that deadlines are met.

- Languages such as C allow efficient programs to be written but do not have constructs to support concurrency or shared resource management.
Java as a real-time language

- Java supports lightweight concurrency (threads and synchronized methods) and can be used for some soft real-time systems.

- Java 2.0 is not suitable for hard RT programming
• **Partition functions** to either hardware or software.

• **Hardware** delivers better performance but potentially **longer development** and less scope for change.

• Design decisions should be made on the basis on **non-functional** system requirements.
Timing constraints

- May require extensive simulation and experiment to ensure that these are met by the system.

- May mean that certain design strategies such as object-oriented design cannot be used because of the additional overhead involved (see comments).

- May mean that low-level programming language features have to be used for performance reasons.

- HW components have better performance than equivalent SW

- Use **HW components** for sys-processing bottlenecks
Comments on OO design & RT systems

- **OO design is not suitable** for RT systems.
- **OO involves**:
  - hiding data
  - Accessing attributes via operations defined with the object
- **Performance loss** because extra code is required to access attributes and handle calls to operations.
- May be impossible to meet RTS deadlines.
Real-time system modelling

- UML Finite state machines can be used for modelling real-time systems.
State Machine model of a Microwave

**Full power**
- **Full power**
  - do: operate oven
- **Half power**
  - do: set power = 300
  - do: display 'Waiting'
- **Half power**
  - do: set power = 600
  - Timer
  - do: display time

**Operation**
- do: operate oven
- Cancel
- Door open
- do: display 'Ready'
- Disabled
- do: display 'Waiting'
- do: operate oven

**Set time**
- do: get number
  - exit: set time
- Full power

**Waiting**
- do: display time
- Half power
- do: set power = 600
- do: display 'Waiting'
- Timer

**Number**
- do: display time
State Machine model of a Petrol (gas) pump
Process priority

- **Interrupt level priority.** Highest priority which is allocated to processes requiring a very fast response.

- **Clock level priority.** Allocated to periodic processes.
Interrupt servicing

- Control is transferred automatically to a pre-determined memory location.
- This location contains an instruction to jump to an interrupt service routine.
Periodic process servicing

- Periodic processes are executed at specified time intervals (Examples: data acquisition & actuator control).

- The process manager selects a process which is ready for execution.
Monitoring and control systems

- Represents an important class of real-time systems.
- Continuously check sensors and take actions depending on sensor values.
- Monitoring systems examine sensors and report their results.
- Control systems take sensor values and control hardware actuators.
Monitoring and control systems: Generic architecture

- Sensor's monitoring processes
- Control processes
- Control panel processes
- Testing process
- Actuator's monitoring processes

S1 → P (S1) → P (A1) → A1
S2 → P (S2) → P (A2) → A2
S3 → P (S1) → P (A1) → A3

Burglar alarm system

- A system is required to monitor sensors on doors and windows to detect the presence of intruders in a building.
- When a sensor indicates a break-in, the system switches on lights around the area and calls police automatically.
- The system should include provision for operation without a mains power supply.
Burglar alarm system

- **Sensors**
  - Movement detectors, window sensors, door sensors;
  - 50 window sensors, 30 door sensors and 200 movement detectors;
  - Voltage drop sensor.

- **Actions**
  - When an intruder is detected, police are called automatically;
  - Lights are switched on in rooms with active sensors;
  - An audible alarm is switched on;
  - The system switches automatically to backup power when a voltage drop is detected.
The R-T system design process

- Identify stimuli and associated responses.
- Define the timing constraints associated with each stimulus and response.
- Allocate system functions to concurrent processes.
- Design algorithms for stimulus processing and response generation.
- Design a scheduling system which ensures that processes will always be scheduled to meet their deadlines.
Stimuli to be processed

- Power failure
  - Generated aperiodically by a circuit monitor. When received, the system must switch to backup power within 50 ms.

- Intruder alarm
  - Stimulus generated by system sensors. Response is to call the police, switch on building lights and the audible alarm.
## Timing requirements

<table>
<thead>
<tr>
<th>Stimulus/Response</th>
<th>Timing requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power fail interrupt</td>
<td>The switch to backup power must be completed within a deadline of 50 ms.</td>
</tr>
<tr>
<td>Door alarm</td>
<td>Each door alarm should be polled twice per second.</td>
</tr>
<tr>
<td>Window alarm</td>
<td>Each window alarm should be polled twice per second.</td>
</tr>
<tr>
<td>Movement detector</td>
<td>Each movement detector should be polled twice per second.</td>
</tr>
<tr>
<td>Audible alarm</td>
<td>The audible alarm should be switched on within 1/2 second of an alarm being raised by a sensor.</td>
</tr>
<tr>
<td>Lights switch</td>
<td>The lights should be switched on within 1/2 second of an alarm being raised by a sensor.</td>
</tr>
<tr>
<td>Communications</td>
<td>The call to the police should be started within 2 seconds of an alarm being raised by a sensor.</td>
</tr>
<tr>
<td>Voice synthesiser</td>
<td>A synthesised message should be available within 4 seconds of an alarm being raised by a sensor.</td>
</tr>
</tbody>
</table>
Burglar alarm system processes

- Movement detector process
- Door sensor process
- Window sensor process

- Detector status
- Sensor status
- Sensor status

- Building monitor process
- Communication process

- Power failure interrupt
- Building monitor
- Room number
- Alert message

- Power switch process
- Alarm system process
- Lighting control process
- Voice synthesiser process

- 400 Hz
- 60 Hz
- 100 Hz

- 560 Hz
A burglar alarm system is primarily a monitoring system. It collects data from sensors but no real-time actuator control.

Control systems are similar but, in response to sensor values, the system sends control signals to actuators.

An example of a monitoring and control system is a system that monitors temperature and switches heaters on and off.
A temperature control system

- Sensor process 500 Hz
- Sensor values
- Thermostat process
- Switch command Room number
- Heater control process 500 Hz
- Thermostat process
- Furnace control process
Data acquisition systems

- Collect data from sensors for subsequent processing and analysis.
- Data collection processes and processing processes may have different periods and deadlines.
- Data collection may be faster than processing e.g. collecting information about an explosion.
- Circular or ring buffers are a mechanism for smoothing speed differences.