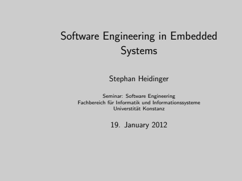
2012-03-13

Embedded Systems



Software Engineering in Embedded Systems

Stephan Heidinger

Seminar: Software Engineering
Fachbereich für Informatik und Informationssysteme
Universtität Konstanz

19. January 2012

Software Engineering in Embedded Systems

Stephan Heidinger

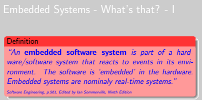
Seminar: Software Engineering
Fachbereich für Informatik und Informationssysteme
Universtität Konstanz

19. January 2012

2012-03-13

Embedded Systems

└─Embedded Systems - What's that? - I


Embedded Systems - What's that? - I

Definition

*"An **embedded software system** is part of a hardware/software system that reacts to events in its environment. The software is 'embedded' in the hardware. Embedded systems are nominaly real-time systems."*

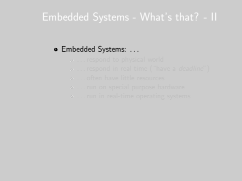
Software Engineering, p.561, Edited by Ian Sommerville, Ninth Edition

Embedded Systems - What's that? - I

Definition

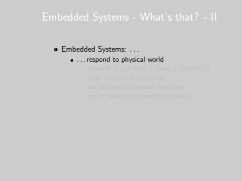
*"An **embedded software system** is part of a hardware/software system that reacts to events in its environment. The software is 'embedded' in the hardware. Embedded systems are nominaly real-time systems."*

Software Engineering, p.561, Edited by Ian Sommerville, Ninth Edition



Embedded Systems - What's that? - II

- Embedded Systems: ...
  - ...respond to physical world
  - ...respond in real time ("have a *deadline*")
  - ...often have little resources
  - ...run on special purpose hardware
  - ...run in real-time operating systems



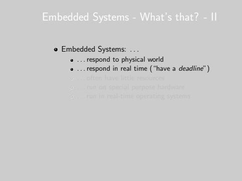
Embedded Systems - What's that? - II

- Embedded Systems: ...
  - ... respond to physical world
  - ... respond in real time ("have a deadline")
  - ... often have little resources
  - ... run on special purpose hardware
  - ... run in real-time operating systems

respond to physical world

Embedded Systems - What's that? - II

- Embedded Systems: ...
  - ... respond to physical world
  - ... respond in real time ("have a *deadline*")
  - ... often have little resources
  - ... run on special purpose hardware
  - ... run in real-time operating systems



Embedded Systems - What's that? - II

- Embedded Systems: ...
  - ...respond to physical world
  - ...respond in real time ("have a *deadline*")
  - ...often have little resources
  - ...run on special purpose hardware
  - ...run in real-time operating systems

resüpm om real time
(“have a deadline”) → time in which the result is produced

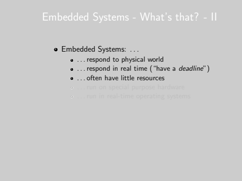
Embedded Systems - What's that? - II

- Embedded Systems: ...
  - ...respond to physical world
  - ...respond in real time (“have a *deadline*”)
  - ...often have little resources
  - ...run on special purpose hardware
  - ...run in real-time operating systems

2012-03-13

Embedded Systems

└─Embedded Systems - What's that? - II


Embedded Systems - What's that? - II

- Embedded Systems: ...
  - ... respond to physical world
  - ... respond in real time ("have a *deadline*")
  - ... often have little resources
  - ... run on special purpose hardware
  - ... run in real-time operating systems

often have litte resources

i.e. not 'computers'

Embedded Systems - What's that? - II

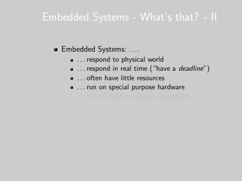
Embedded Systems Design

Architectural patterns

Timing analysis

Real-time operating systems

- Embedded Systems: ...
  - ... respond to physical world
  - ... respond in real time ("have a *deadline*")
  - ... often have little resources
  - ... run on special purpose hardware
  - ... run in real-time operating systems



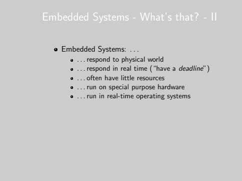
Embedded Systems - What's that? - II

- Embedded Systems: ...
  - ...respond to physical world
  - ...respond in real time ("have a *deadline*")
  - ...often have little resources
  - ...run on special purpose hardware
  - ...run in real-time operating systems

run on special purpose hardware

Embedded Systems - What's that? - II

- Embedded Systems: ...
  - ...respond to physical world
  - ...respond in real time ("have a *deadline*")
  - ...often have little resources
  - ...run on special purpose hardware
  - ...run in real-time operating systems


Embedded Systems - What's that? - II

- Embedded Systems: ...
  - ...respond to physical world
  - ...respond in real time ("have a *deadline*")
  - ...often have little resources
  - ...run on special purpose hardware
  - ...run in real-time operating systems

run in real time operating systems

Embedded Systems - What's that? - II

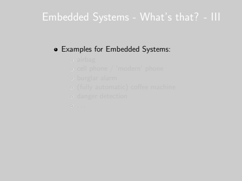
Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- Embedded Systems: ...
  - ...respond to physical world
  - ...respond in real time ("have a *deadline*")
  - ...often have little resources
  - ...run on special purpose hardware
  - ...run in real-time operating systems

2012-03-13

Embedded Systems

└Embedded Systems - What's that? - III



Embedded Systems - What's that? - III

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - ...

Embedded Systems - What's that? - III

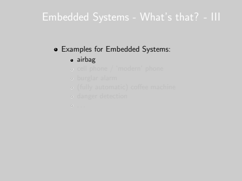
Embedded Systems Design

Architectural patterns

Timing analysis

Real-time operating systems

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - ...



Embedded Systems - What's that? - III

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - ...

airbag:

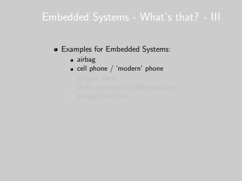
strict deadline

catastrophic result on failure

Embedded Systems - What's that? - III

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - ...

└Embedded Systems - What's that? - III

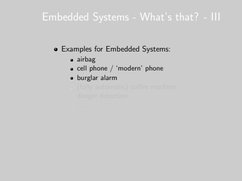


cell phone:

phone must be answered before call quit vom other side

Embedded Systems - What's that? - III

- Examples for Embedded Systems:
  - airbag
  - cell phone / ‘modern’ phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - . . .



Embedded Systems - What's that? - III

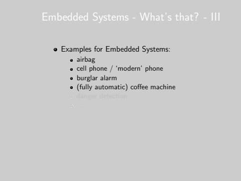
- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - ...

burglar alarm:

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

Embedded Systems - What's that? - III

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - ...



Embedded Systems - What's that? - III

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection

coffee machine:

dont want to have coffee, when its cold...

Embedded Systems - What's that? - III

Embedded Systems Design

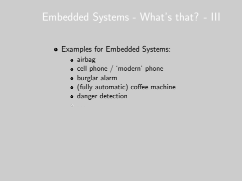
Architectural patterns

Timing analysis

Real-time operating systems

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - ...

└Embedded Systems - What's that? - III


Embedded Systems - What's that? - III

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection

danger detection:

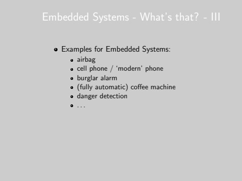
earthquake, toxins, . . .

depending on the kind of danger, absolutely no time to spare.

Embedded Systems - What's that? - III

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - . . .



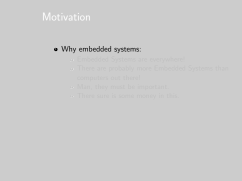
Embedded Systems - What's that? - III

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - ...

One can produce more examples on a whim especially in cars

Embedded Systems - What's that? - III

- Examples for Embedded Systems:
  - airbag
  - cell phone / 'modern' phone
  - burglar alarm
  - (fully automatic) coffee machine
  - danger detection
  - ...



Why embedded sytems

Motivation

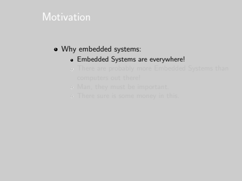
Embedded Systems Design

Architectural patterns

Timing analysis

Real-time operating systems

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.

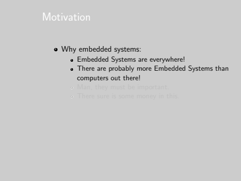

Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.

Embedded Systems are everywhere!

Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.



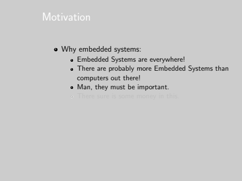
Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.

There are probably more Embedded Systems than computers out there!

Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.



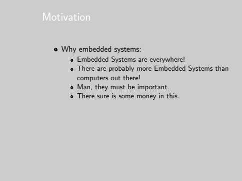
Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.

they must be important.

Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.



Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.

there sure is some money in this

some money:

C-programing

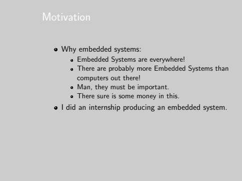
special skillsand i did an internship

producing an embedded system at PSI

monitoring device for the detector of an particle accelerator

Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.



Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.
- I did an internship producing an embedded system.

and i did an internship
producing an embedded system at PSI
monitoring device for the detector of an particle accelerator

Motivation

- Why embedded systems:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
  - Man, they must be important.
  - There sure is some money in this.
- I did an internship producing an embedded system.

Outline

1. Embedded Systems Design
2. Architectural patterns
3. Timing analysis
4. Real-time operating systems

this is the structure of my talk:
embedded systems DESIGN
architectural patterns
timing analysis
real-time operating systems

Outline

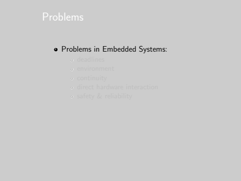
1. Embedded Systems Design
2. Architectural patterns
3. Timing analysis
4. Real-time operating systems

Outline

1 Embedded Systems Design
2 Architectural patterns
3 Timing analysis
4 Real-time operating systems

Outline

1. Embedded Systems Design
2. Architectural patterns
3. Timing analysis
4. Real-time operating systems



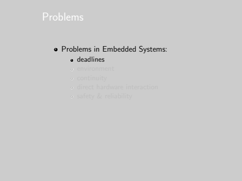
Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability

several problems in emb-systems that are not in “normal” systems

Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability



Problems

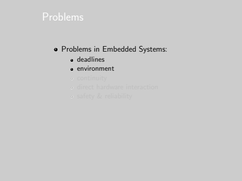
- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability

deadlines: every process has deadline until result must exist
hard systems: deadline not met, failure
soft system: deadline not met, bad results

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability



Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability

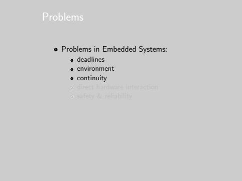
environment:

is unpredictable

embedded Software ⇒ must be concurrent

Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability



Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability

continuity:

embedded Software ⇒ does not normally terminate

software has to be reliable

may need update while operating

Problems

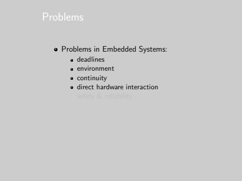
Embedded Systems Design

Architectural patterns

Timing analysis

Real-time operating systems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability



Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability

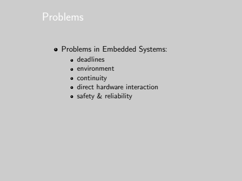
direct hardware interaction:

uncommon hardware (i.e. detonator in airbag)

speed issues (hardware is faster than software)

Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability


Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability

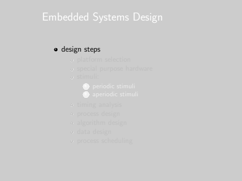
safety & reliability:

cost of failure high

either economical or in human life

Problems

- Problems in Embedded Systems:
  - deadlines
  - environment
  - continuity
  - direct hardware interaction
  - safety & reliability


Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

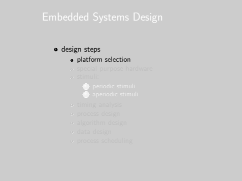
design steps

not all are necessary, but most will be.

no definite order

Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling


Embedded Systems Design
- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

Platform selection:

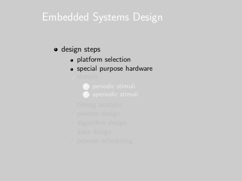
what hardware?

choice of Real-time operating system (later)

power consumption (mobile device, backup)

Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling


Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

special purpose hardware:

What is to be implemented in software, what in hardware

do we need uncommon hardware?

design special hardware?

replace software by hardware?

Embedded Systems Design

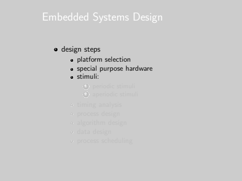
Embedded Systems Design

Architectural patterns

Timing analysis

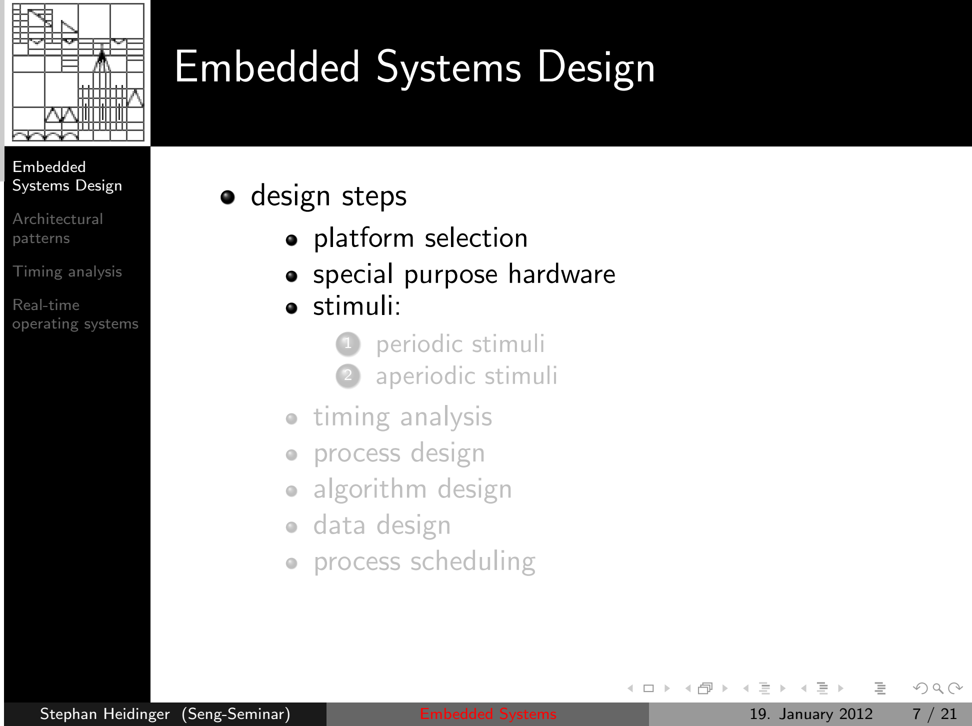
Real-time operating systems

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling



think about **stimuli:**
describe behavior of system by listing received stimuli and reactions
stimuli = signals
often: stimulus *Rightarrow* defined response

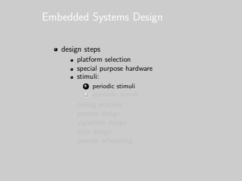
example AFTER THIS SLIDE



Embedded Systems Design

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling



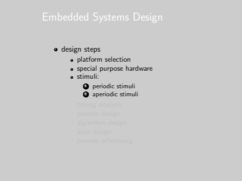
Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

periodic stimuli:
occur at predictable intervals
predefined reaction per stimulus
i.e. polling

Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling



Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

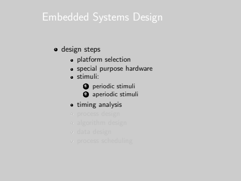
aperiodic stimuli:
occurr irregularly and unpredictably
often interrupts
i.e. alarms, failures, IO operation finished, etc**stimuli list:**
best practice: stimuli list with **all** stimuli.

example AFTER THIS SLIDE

Embedded Systems Design

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

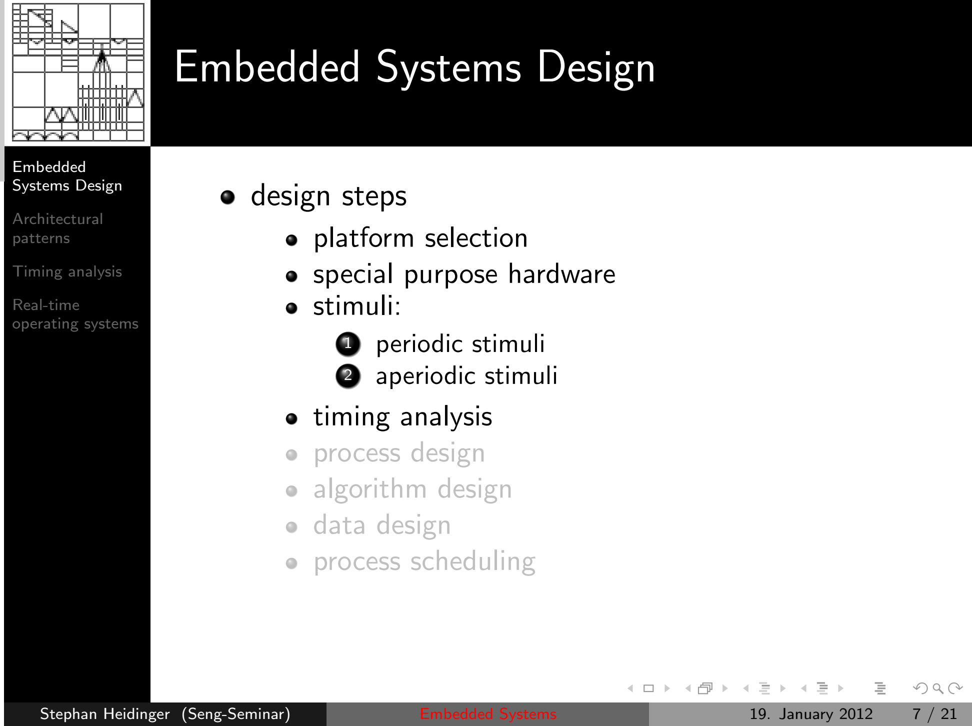
- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling



Timing analysis:

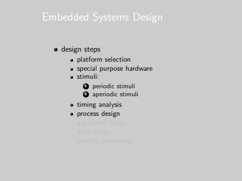
For each stimulus and response ⇒ find timing constraints

timing constraints ⇒ deadlines

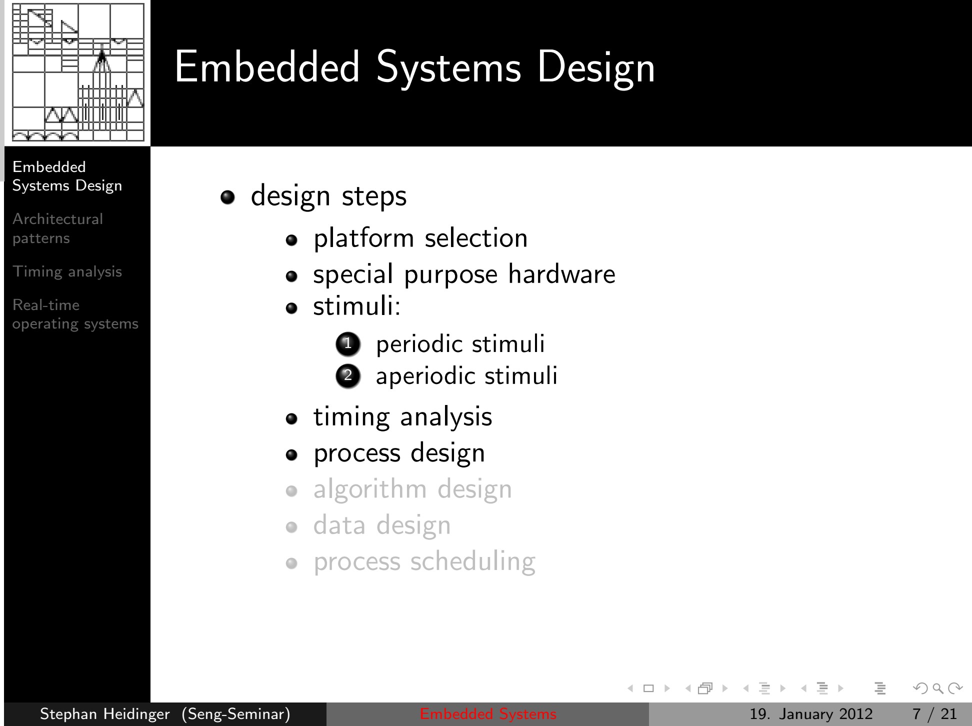


Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling



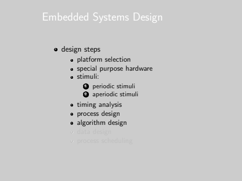
Process design:
aggregate the stimuli & responses into concurrent processes
SEE Architectural patterns



Embedded Systems Design

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling



Embedded Systems Design

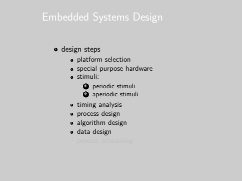
- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

Algorithm design:
For each stimulus & response ⇒ design algorithm
especially important for computationally intensive tasks (signal processing)
Do we need to implement these in hardware? (i.e. control systems)

Embedded Systems Design

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

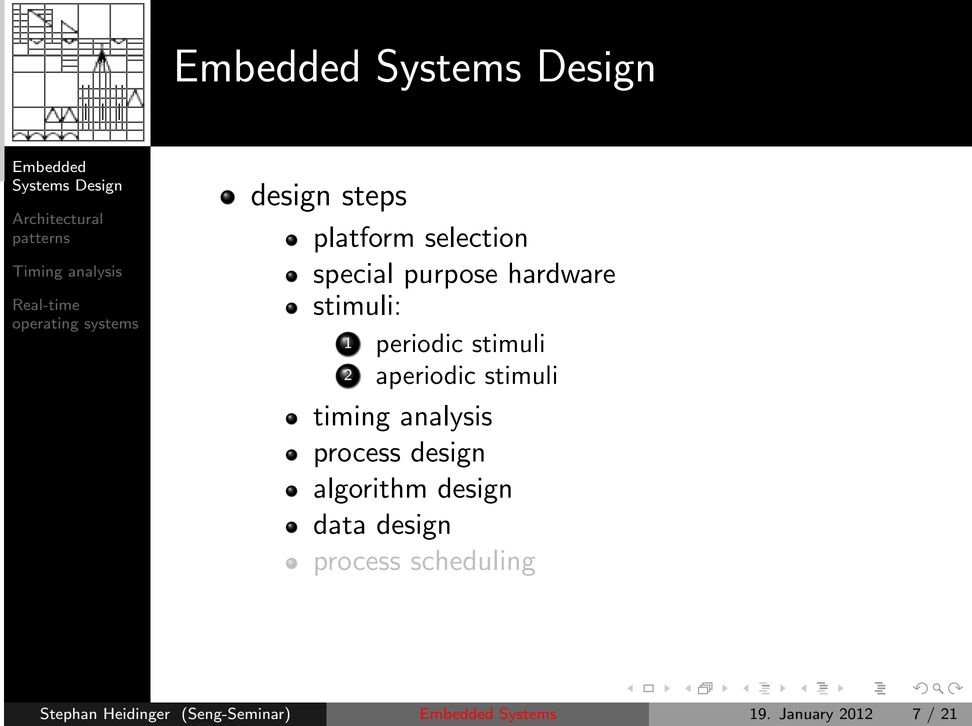


Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

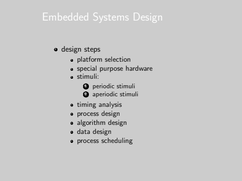
Data design:
How to store data, that will be exchanged
semaphore & critical regions & monitors & . . .

circular buffer: producer & consumer may run at different speeds



Embedded Systems Design

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling


Embedded Systems Design
- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

Process scheduling:
ensure, that processes meet their deadline

all shown:
not all need to be done, but most probably will
which & order depends on what we design

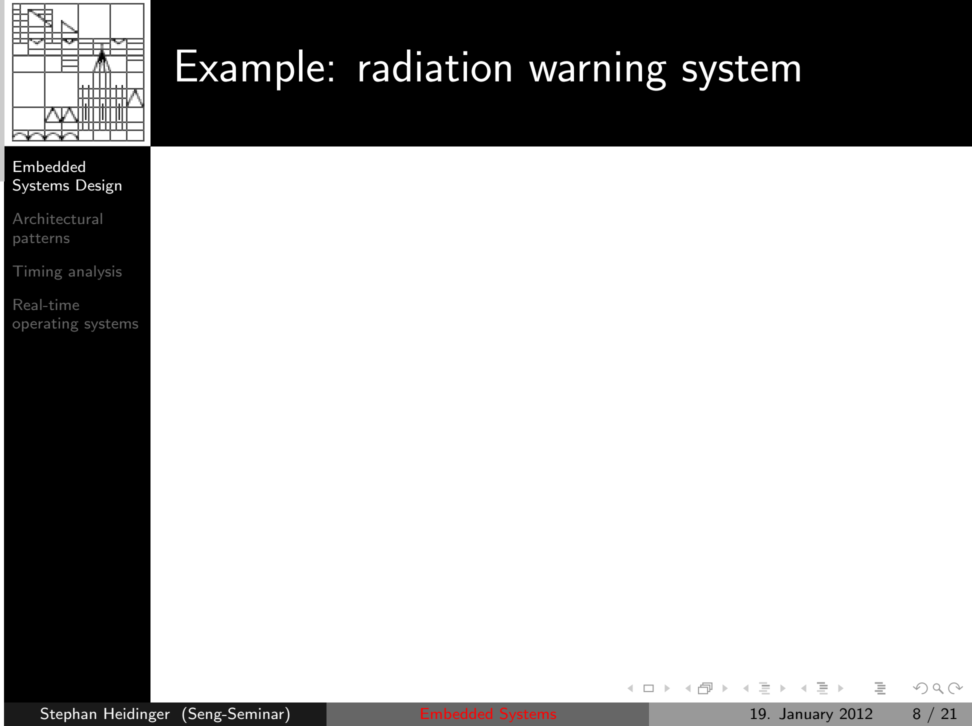
after this design steps:
make sure system can meet deadlines
static analysis
simulation

Embedded Systems Design

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli
  - timing analysis
  - process design
  - algorithm design
  - data design
  - process scheduling

Now for the examples for stimuli:



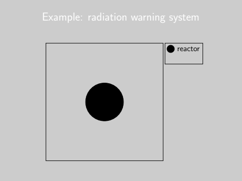
Example: radiation warning system

Embedded Systems Design

Architectural patterns

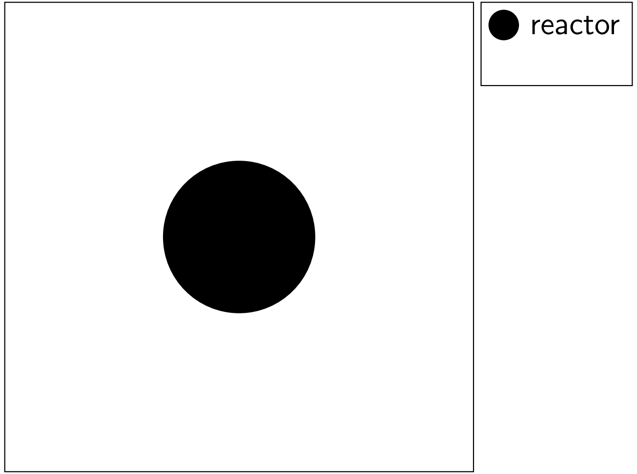
Timing analysis

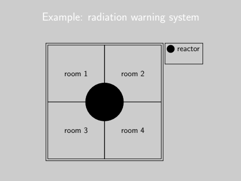
Real-time operating systems



we have this room with an reactor inside

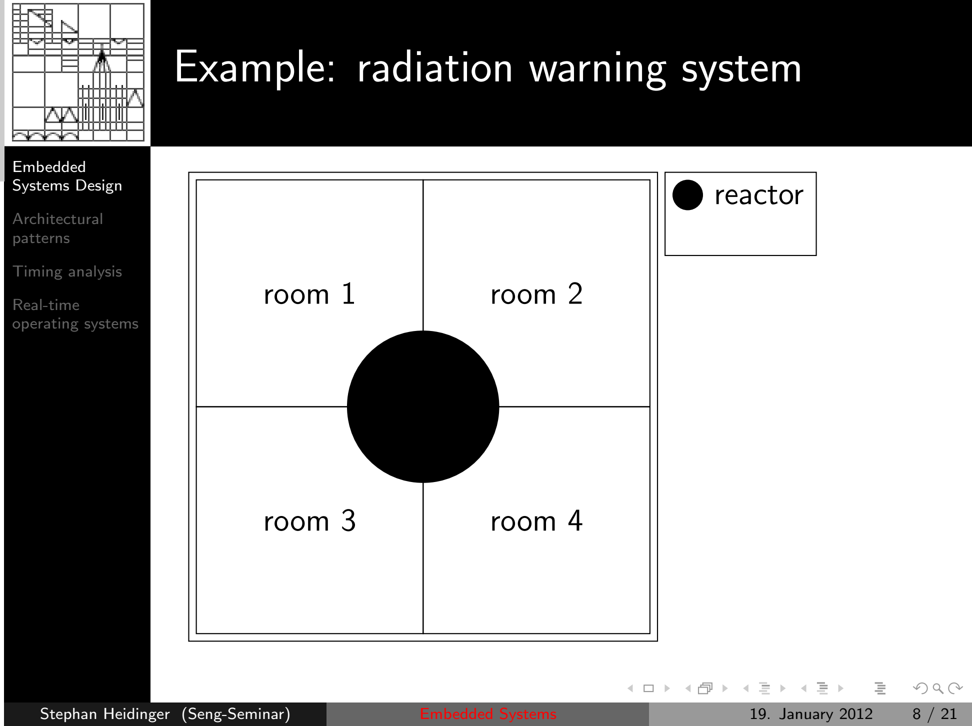
Example: radiation warning system





because several people work here, we put several rooms around the reactor

walls are shielded



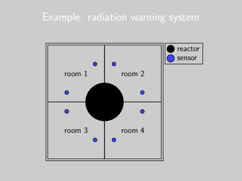
Example: radiation warning system

Embedded Systems Design

Architectural patterns

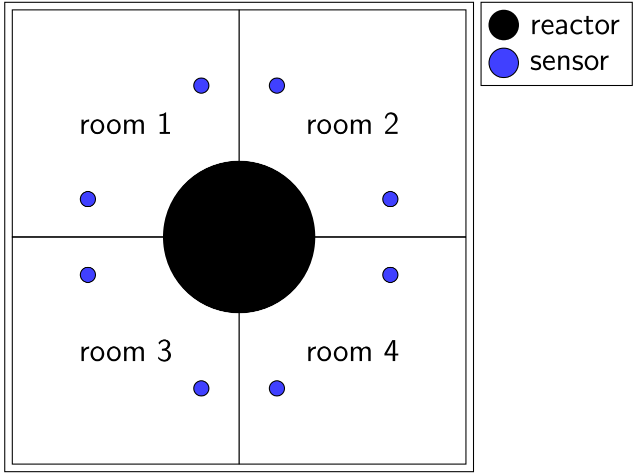
Timing analysis

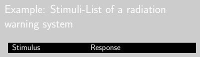
Real-time operating systems



because people work here, we need some sensors to detect radiation leaks

Example: radiation warning system




Example: Stimuli-List of a radiation warning system

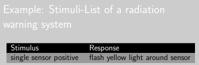
Stimulus Response

Now we built a list of stimuli and responses

Example: Stimuli-List of a radiation warning system

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

Stimulus	Response



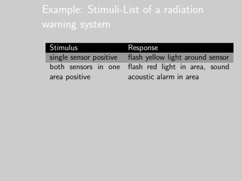
Example: Stimuli-List of a radiation warning system

Stimulus	Response
single sensor positive	flash yellow light around sensor

single sensor positive
want to warn people, that there is something
flash a yellow light around the sensor

Example: Stimuli-List of a radiation warning system

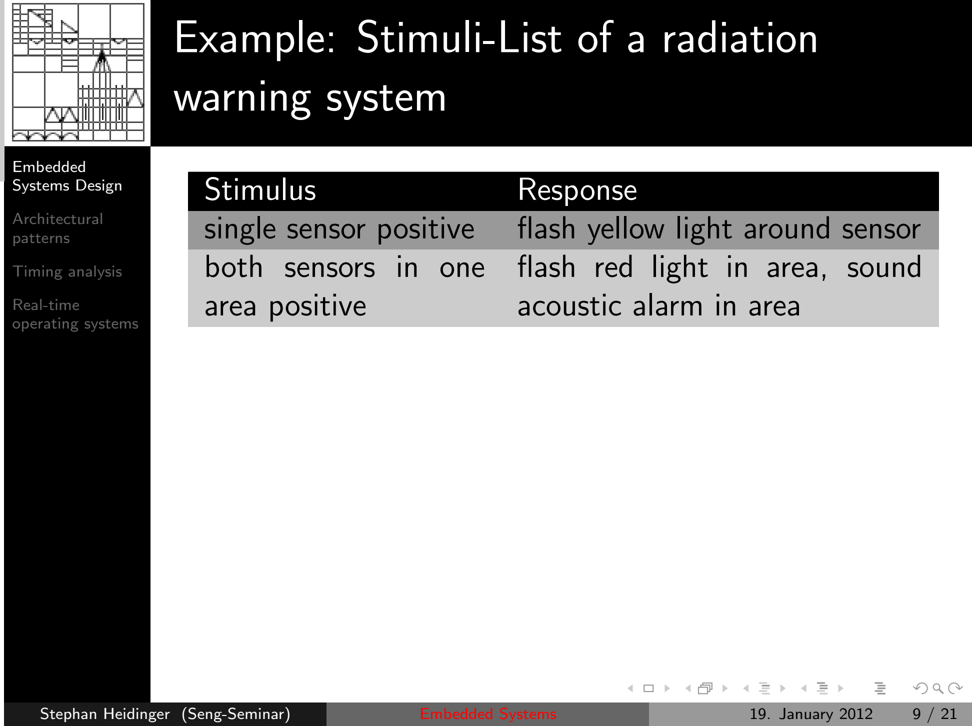
Stimulus	Response
single sensor positive	flash yellow light around sensor



Example: Stimuli-List of a radiation warning system

Stimulus	Response
single sensor positive	flash yellow light around sensor
both sensors in one area positive	flash red light in area, sound acoustic alarm in area

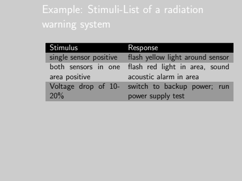
two sensors in one are positive
something is really wrong
flash red light in are
sound alarm



Example: Stimuli-List of a radiation warning system

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

Stimulus	Response
single sensor positive	flash yellow light around sensor
both sensors in one area positive	flash red light in area, sound acoustic alarm in area

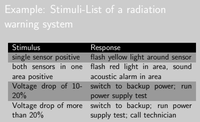

Example: Stimuli-List of a radiation warning system

Stimulus	Response
single sensor positive	flash yellow light around sensor
both sensors in one area positive	flash red light in area, sound acoustic alarm in area
Voltage drop of 10-20%	switch to backup power; run power supply test

small voltage drop
probably nothing bad
switch to backup power
run power supply test

Example: Stimuli-List of a radiation warning system

Stimulus	Response
single sensor positive	flash yellow light around sensor
both sensors in one area positive	flash red light in area, sound acoustic alarm in area
Voltage drop of 10-20%	switch to backup power; run power supply test

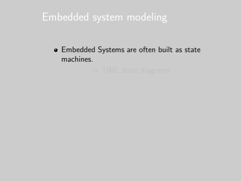

Example: Stimuli-List of a radiation warning system

Stimulus	Response
single sensor positive	flash yellow light around sensor
both sensors in one area positive	flash red light in area, sound acoustic alarm in area
Voltage drop of 10-20%	switch to backup power; run power supply test
Voltage drop of more than 20%	switch to backup; run power supply test; call technician

big voltage drop
do the same as on small drop
call technician
LAST CELL:

Example: Stimuli-List of a radiation warning system

Stimulus	Response
single sensor positive	flash yellow light around sensor
both sensors in one area positive	flash red light in area, sound acoustic alarm in area
Voltage drop of 10-20%	switch to backup power; run power supply test
Voltage drop of more than 20%	switch to backup; run power supply test; call technician



Embedded system modeling

- Embedded Systems are often built as state machines.

⇒ UML state diagrams

Embedded Systems ⇒ often build as state machines

Embedded system modeling

Embedded Systems Design

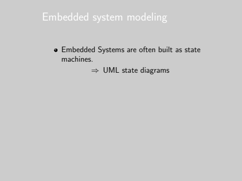
Architectural patterns

Timing analysis

Real-time operating systems

- Embedded Systems are often built as state machines.

⇒ UML state diagrams



Embedded system modeling

- Embedded Systems are often built as state machines.

⇒ UML state diagrams

of course ⇒ UML state diagrams
very good for understanding the workings of the system
something like thismodelled stimuli+responses into states
here i modelled two sensor as a result of one sensor ⇒ may be done differently

Embedded system modeling

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- Embedded Systems are often built as state machines.

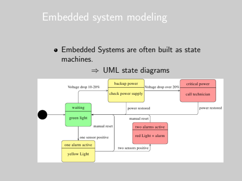
⇒ UML state diagrams

2012-03-13

Embedded Systems

└Embedded Systems Design

└Embedded system modeling


Embedded system modeling

- Embedded Systems are often built as state machines.

⇒ UML state diagrams

Embedded system modeling

Embedded Systems Design

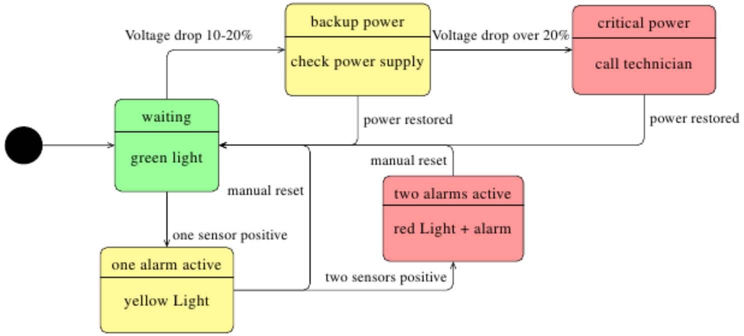
Architectural patterns

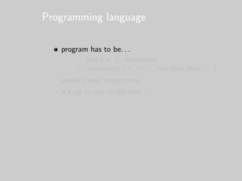
Timing analysis

Real-time operating systems

- Embedded Systems are often built as state machines.

  ⇒ UML state diagrams





Programming language

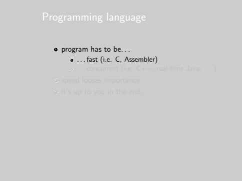
- program has to be...
  - ...fast (i.e. C, Assembler)
  - ...concurrent (i.e. C++, real time Java, ...)
- speed looses importance
- it's up to you in the end ...

the programming language
several things need to be taken into account

program has to be

Programming language

- program has to be...
  - ...fast (i.e. C, Assembler)
  - ...concurrent (i.e. C++, real time Java, ...)
- speed looses importance
- it's up to you in the end ...


Programming language

- program has to be. . .
  - . . . fast (i.e. C, Assembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance
- it's up to you in the end . . .

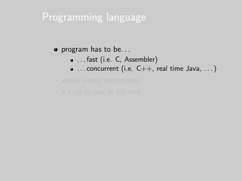
fast: C, Assembler:

No concurrency

no built-in system for shared resources

Programming language

- program has to be. . .
  - . . . fast (i.e. C, Assembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance
- it's up to you in the end . . .



Programming language

- program has to be. . .
  - . . . fast (i.e. C, Assembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance
- it's up to you in the end . . .

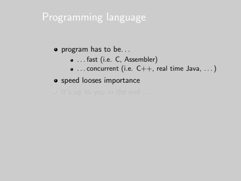
concurrent:
and manage shared resources

concurrent or speed??:
depends on what is more important
simulate concurrency with frequent polling
do something yourself about shared resources

Programming language

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- program has to be. . .
  - . . . fast (i.e. C, Assembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance
- it's up to you in the end . . .


Programming language

- program has to be. . .
  - . . . fast (i.e. C, Assembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance
- it's up to you in the end . . .

speed:
due to faster hardware
ie monitoring device written in C++
ie cell phones in java, objective C, . . .
still there are some areas, where you need C & assembler. . .

Programming language

- program has to be. . .
  - . . . fast (i.e. C, Assembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance
- it's up to you in the end . . .

Programming language

- program has to be. . .
  - . . . fast (i.e. C, Assembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance
- it's up to you in the end . . .

it's up to you in the end. . .
evaluate the needs and decide. . .

Programming language

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- program has to be. . .
  - . . . fast (i.e. C, Assembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance
- it's up to you in the end . . .

2012-03-13

Embedded Systems
└Architectural patterns
└Outline

Outline

1 Embedded Systems Design

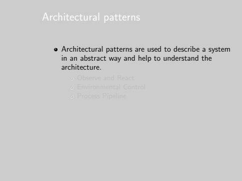
2 Architectural patterns

3 Timing analysis

4 Real-time operating systems

Outline

1. Embedded Systems Design
2. Architectural patterns
3. Timing analysis
4. Real-time operating systems



Architectural patterns

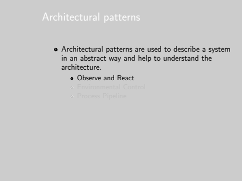
- Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.
  - Observe and React
  - Environmental Control
  - Process Pipeline

Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.

note on the 3 patterns:
The summerville book describes three rough design pattern
there are finer patterns that will lead to more exact design

Architectural patterns

- Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.
  - Observe and React
  - Environmental Control
  - Process Pipeline



Architectural patterns

- Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.
  - Observe and React
  - Environmental Control
  - Process Pipeline

Observe and React:

set of monitored sensors

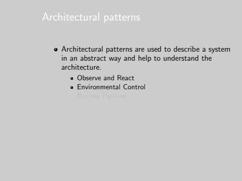
Something exeptional happens ⇒ we do something

i.e. monitoring, incoming phone call

Architectural patterns

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.
  - Observe and React
  - Environmental Control
  - Process Pipeline



Architectural patterns

- Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.
  - Observe and React
  - Environmental Control
  - Process Pipeline

Environmental Control:

set of sensors and actuators

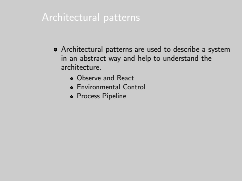
can change environment

i.e. flash light, when sensor fires

i.e. control water level in a tank

Architectural patterns

- Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.
  - Observe and React
  - Environmental Control
  - Process Pipeline


Architectural patterns

- Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.
  - Observe and React
  - Environmental Control
  - Process Pipeline

Process Pipeline:
data transformation
series of processing steps
preferably concurrent

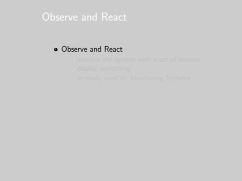
all of those: can be combined
often more than one pattern in the system
ie monitor the actuators

design patterns: will lead to **inefficient** system ⇒ only for understanding system

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

Architectural patterns

- Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.
  - Observe and React
  - Environmental Control
  - Process Pipeline

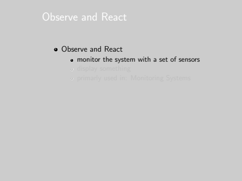

Observe and React

- Observe and React
  - monitor the system with a set of sensors
  - display something
  - primarly used in: Monitoring Systems

Observer & React

Observe and React

- Observe and React
  - monitor the system with a set of sensors
  - display something
  - primarly used in: Monitoring Systems



Observe and React

- Observe and React
  - monitor the system with a set of sensors
  - display something
  - primarly used in: Monitoring Systems

monitoring:

monitor the system with a set of sensors

Observe and React

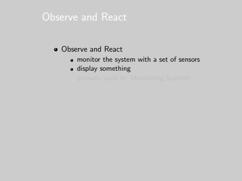
Embedded Systems Design

Architectural patterns

Timing analysis

Real-time operating systems

- Observe and React
  - monitor the system with a set of sensors
  - display something
  - primarly used in: Monitoring Systems



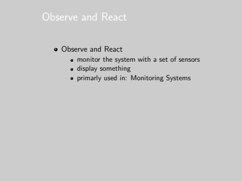
Observe and React

- Observe and React
  - monitor the system with a set of sensors
  - display something
  - primarly used in: Monitoring Systems

display:
monitoring screen
on exceptional behaviour: alarms, shutdown

Observe and React

- Observe and React
  - monitor the system with a set of sensors
  - display something
  - primarly used in: Monitoring Systems



Observe and React

- Observe and React
  - monitor the system with a set of sensors
  - display something
  - primarly used in: Monitoring Systems

primarly in **monitoring systems:**
often consist of more than one O&R pattern
one for each sensor
optimisation: combine something, ie display on one monitor

Observe and React

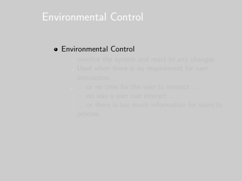
- Observe and React
  - monitor the system with a set of sensors
  - display something
  - primarly used in: Monitoring Systems

2012-03-13

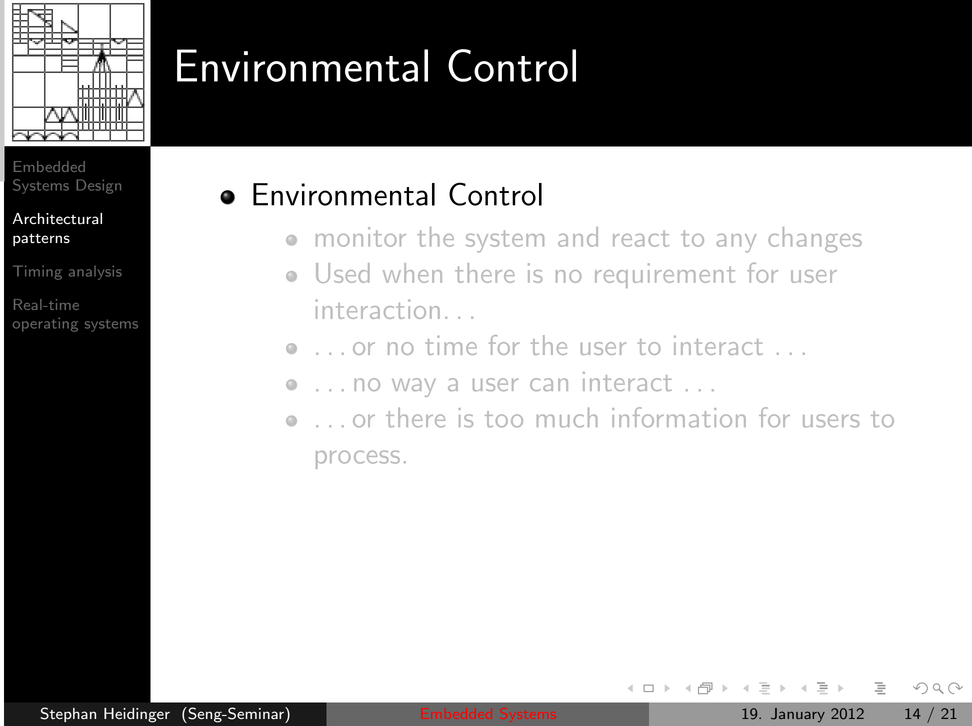
Embedded Systems

└Architectural patterns

└Environmental Control



Environmental Control



Environmental Control

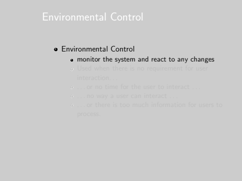
- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction…
  - …or no time for the user to interact …
  - …no way a user can interact …
  - …or there is too much information for users to process.

Embedded Systems Design

Architectural patterns

Timing analysis

Real-time operating systems

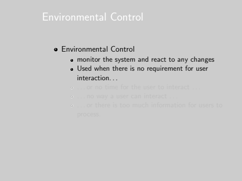

Environmental Control

- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction…
  - …or no time for the user to interact …
  - …no way a user can interact …
  - …or there is too much information for users to process.

monitor sytem and react to any changes

Environmental Control

- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction…
  - …or no time for the user to interact …
  - …no way a user can interact …
  - …or there is too much information for users to process.



Environmental Control

- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction. . .
  - . . . or no time for the user to interact . . .
  - . . . no way a user can interact . . .
  - . . . or there is too much information for users to process.

no required user interaction

examples:

cruise control

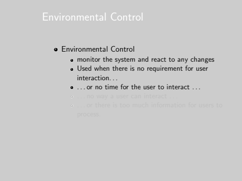
water level

pressure control

. . .

Environmental Control

- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction. . .
  - . . . or no time for the user to interact . . .
  - . . . no way a user can interact . . .
  - . . . or there is too much information for users to process.



Environmental Control

- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction. . .
  - . . . or no time for the user to interact . . .
  - . . . no way a user can interact . . .
  - . . . or there is too much information for users to process.

no time for user interaction

examples:

break assist

airbag

Environmental Control

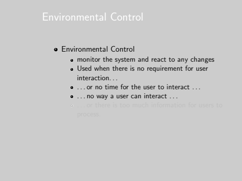
Embedded Systems Design

Architectural patterns

Timing analysis

Real-time operating systems

- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction. . .
  - . . . or no time for the user to interact . . .
  - . . . no way a user can interact . . .
  - . . . or there is too much information for users to process.



Environmental Control

- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction. . .
  - . . . or no time for the user to interact . . .
  - . . . no way a user can interact . . .
  - . . . or there is too much information for users to process.

no way for user interaction

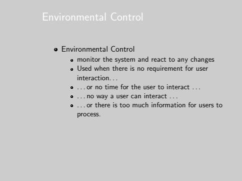
example:

CYPRES (parachute, Möllemann did not activate his in 2003)

self desctruct of military/sensitive equipment

Environmental Control

- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction. . .
  - . . . or no time for the user to interact . . .
  - . . . no way a user can interact . . .
  - . . . or there is too much information for users to process.



Environmental Control

- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction. . .
  - . . . or no time for the user to interact . . .
  - . . . no way a user can interact . . .
  - . . . or there is too much information for users to process.

too much information for users

example:

Nuclear Power Plant

Airplane

Car

virtually any big system with many subsystems

Environmental Control

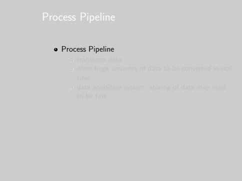
- Environmental Control
  - monitor the system and react to any changes
  - Used when there is no requirement for user interaction. . .
  - . . . or no time for the user to interact . . .
  - . . . no way a user can interact . . .
  - . . . or there is too much information for users to process.

2012-03-13

Embedded Systems

└Architectural patterns

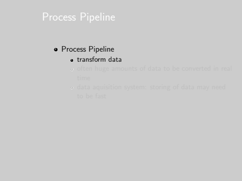
└Process Pipeline


Process Pipeline

- Process Pipeline
  - transform data
  - often huge amounts of data to be converted in real time
  - data aquisition system: storing of data may need to be fast

Process Pipeline

- Process Pipeline
  - transform data
  - often huge amounts of data to be converted in real time
  - data aquisition system: storing of data may need to be fast



Process Pipeline

- Process Pipeline
  - transform data
  - often huge amounts of data to be converted in real time
  - data aquisition system: storing of data may need to be fast

transform data

examples:

signal processing from sensors in other systems

optical sensor

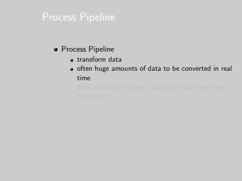
convert digital data to audio

Process Pipeline

- Process Pipeline
  - transform data
  - often huge amounts of data to be converted in real time
  - data aquisition system: storing of data may need to be fast

huge amount in real time:

concurrency + multicore is the key



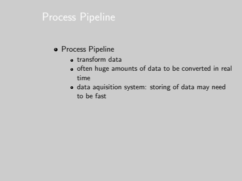
Process Pipeline

- Process Pipeline
  - transform data
  - often huge amounts of data to be converted in real time
  - data aquisition system: storing of data may need to be fast

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

Process Pipeline

- Process Pipeline
  - transform data
  - often huge amounts of data to be converted in real time
  - data aquisition system: storing of data may need to be fast


Process Pipeline

- Process Pipeline
  - transform data
  - often huge amounts of data to be converted in real time
  - data aquisition system: storing of data may need to be fast

data aquisition system**example:**
particle accelerator
chemical reactions
. . .
if storing not fast, data will be lost

Process Pipeline

- Process Pipeline
  - transform data
  - often huge amounts of data to be converted in real time
  - data aquisition system: storing of data may need to be fast

2012-03-13

Embedded Systems

Timing analysis

Outline

Outline

1 Embedded Systems Design

2 Architectural patterns

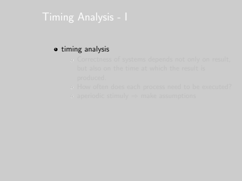
3 Timing analysis

4 Real-time operating systems

Outline

1. Embedded Systems Design
2. Architectural patterns
3. Timing analysis
4. Real-time operating systems

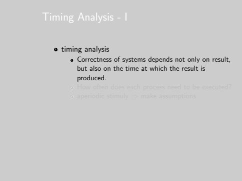
timing analysis


Timing Analysis - I

- timing analysis
  - Correctness of systems depends not only on result, but also on the time at which the result is produced.
  - How often does each process need to be executed?
  - aperiodic stimuly ⇒ make assumptions

Timing Analysis - I

- timing analysis
  - Correctness of systems depends not only on result, but also on the time at which the result is produced.
  - How often does each process need to be executed?
  - aperiodic stimuly ⇒ make assumptions



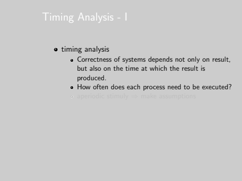
not only result, also time is important

soft and hard systems

Timing Analysis - I

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

- timing analysis
  - Correctness of systems depends not only on result, but also on the time at which the result is produced.
  - How often does each process need to be executed?
  - aperiodic stimuly ⇒ make assumptions



Timing Analysis - I

- timing analysis
  - Correctness of systems depends not only on result, but also on the time at which the result is produced.
  - How often does each process need to be executed?
  - aperiodic stimuly ⇒ make assumptions

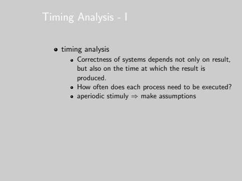
how often?:

then we check, if our system can deliver this

this can be quite hard, when *mixture of aperiodic and periodic stimuli* or *many aperiodic stimuli* are expected

Timing Analysis - I

- timing analysis
  - Correctness of systems depends not only on result, but also on the time at which the result is produced.
  - How often does each process need to be executed?
  - aperiodic stimuly ⇒ make assumptions



Timing Analysis - I

- timing analysis
  - Correctness of systems depends not only on result, but also on the time at which the result is produced.
  - How often does each process need to be executed?
  - aperiodic stimuly ⇒ make assumptions

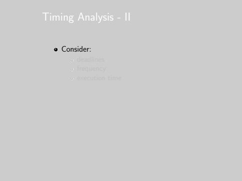
aperiodic stimuli:
make assumptions

fast systems:
use only periodic stimuli
poll frequently for aperiodic stimuli

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

Timing Analysis - I

- timing analysis
  - Correctness of systems depends not only on result, but also on the time at which the result is produced.
  - How often does each process need to be executed?
  - aperiodic stimuly ⇒ make assumptions

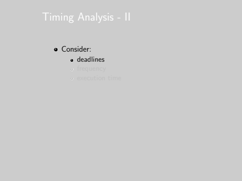

Timing Analysis - II

- Consider:
  - deadlines
  - frequency
  - execution time

Timing analysis must consider:

Timing Analysis - II

- Consider:
  - deadlines
  - frequency
  - execution time



Timing Analysis - II

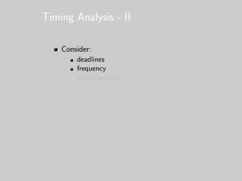
- Consider:
  - deadlines
  - frequency
  - execution time

deadlines:

By which time must the process have ended.

Timing Analysis - II

- Consider:
  - deadlines
  - frequency
  - execution time



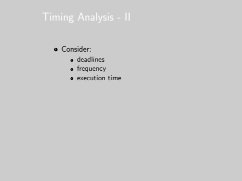
Timing Analysis - II

- Consider:
  - deadlines
  - frequency
  - execution time

frequency:
The number of times a process must be executed in a given span, so that the *system* meets all deadlines

Timing Analysis - II

- Consider:
  - deadlines
  - frequency
  - execution time



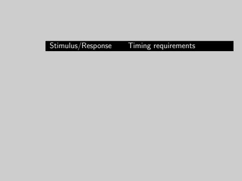
Timing Analysis - II

- Consider:
  - deadlines
  - frequency
  - execution time

execution time:
How long does each single process take (average & worst case)
hard: conditional execution, delays waiting, . . .
hard systems: always worst case

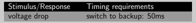
Timing Analysis - II

- Consider:
  - deadlines
  - frequency
  - execution time


Stimulus/Response Timing requirements

We list stimuli and response
then think about how fast this needs to work

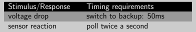
Stimulus/Response Timing requirements



Stimulus/Response	Timing requirements
voltage drop	switch to backup: 50ms

voltage drop ⇒ 50ms

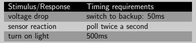
Stimulus/Response	Timing requirements
voltage drop	switch to backup: 50ms



Stimulus/Response	Timing requirements
voltage drop	switch to backup: 50ms
sensor reaction	poll twice a second

sensor reaction ⇒ poll twice a second

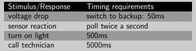
Stimulus/Response	Timing requirements
voltage drop	switch to backup: 50ms
sensor reaction	poll twice a second



Stimulus/Response	Timing requirements
voltage drop	switch to backup: 50ms
sensor reaction	poll twice a second
turn on light	500ms

turn on light ⇒ 500ms

Stimulus/Response	Timing requirements
voltage drop	switch to backup: 50ms
sensor reaction	poll twice a second
turn on light	500ms



Stimulus/Response	Timing requirements
voltage drop	switch to backup: 50ms
sensor reaction	poll twice a second
turn on light	500ms
call technician	5000ms

call technician ⇒ 5000ms
may take longer, as technician reaction time is low anyways**LAST CELL:**

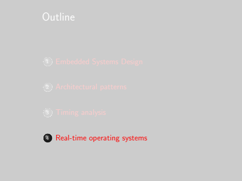
Stimulus/Response	Timing requirements
voltage drop	switch to backup: 50ms
sensor reaction	poll twice a second
turn on light	500ms
call technician	5000ms

2012-03-13

Embedded Systems

Real-time operating systems

Outline



Outline

1 Embedded Systems Design

2 Architectural patterns

3 Timing analysis

4 Real-time operating systems

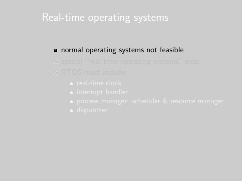
Outline

1 Embedded Systems Design

2 Architectural patterns

3 Timing analysis

4 Real-time operating systems

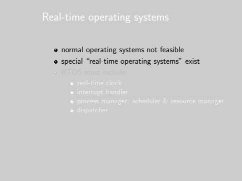

Real-time operating systems

- normal operating systems not feasible
- special “real-time operating systems” exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

normal operating systems:
too large, too bulky, too slow

Real-time operating systems

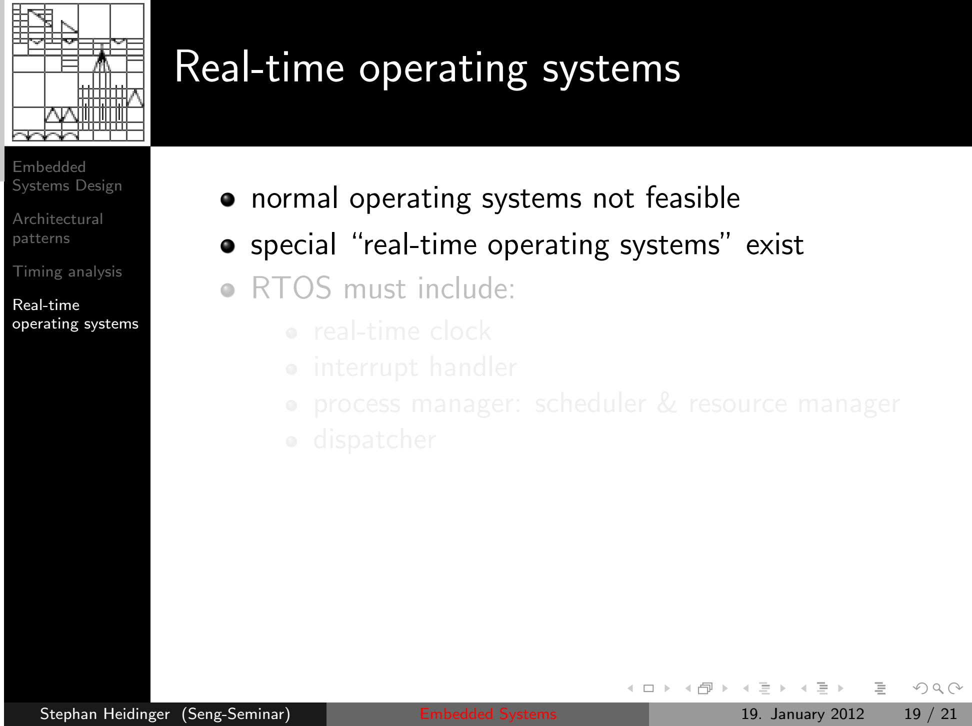
- normal operating systems not feasible
- special “real-time operating systems” exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher


Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

real-time operating systems:
Windows/CE
Vxworks
RTLinux
emdebian

they are small and damn fast



Embedded Systems Design

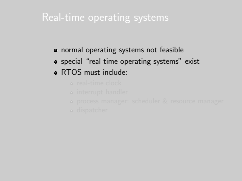
Architectural patterns

Timing analysis

Real-time operating systems

Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

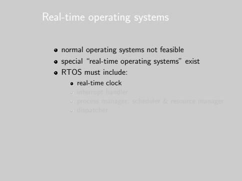

Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

RTOS must include

Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher



Real-time operating systems

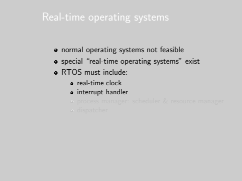
- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

real-time clock:

provides information required to schedule processes

Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher



Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

interrupt handler:

manages aperiodic requests for service

may be inside process manager

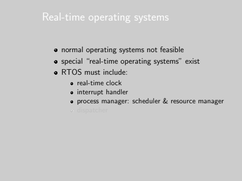
at least **2 levels:**

interrupt for processes with fast response time & *clock level* fore regular processes

often also background processes with low priority (self checks etc)

Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher


Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

scheduler

examines processes and chooses one for execution

processes need enough processor time to *finish before their deadline*

commonly used:

non-pre-emptive & pre-emtive (execution of processes may be stopped)

round robin

rate monolithic scheduling (SJF)

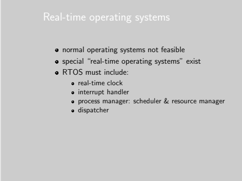
shortes deadline first (HPF)

resource manager:

allocates memory and processor resources scheduled for execution

Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

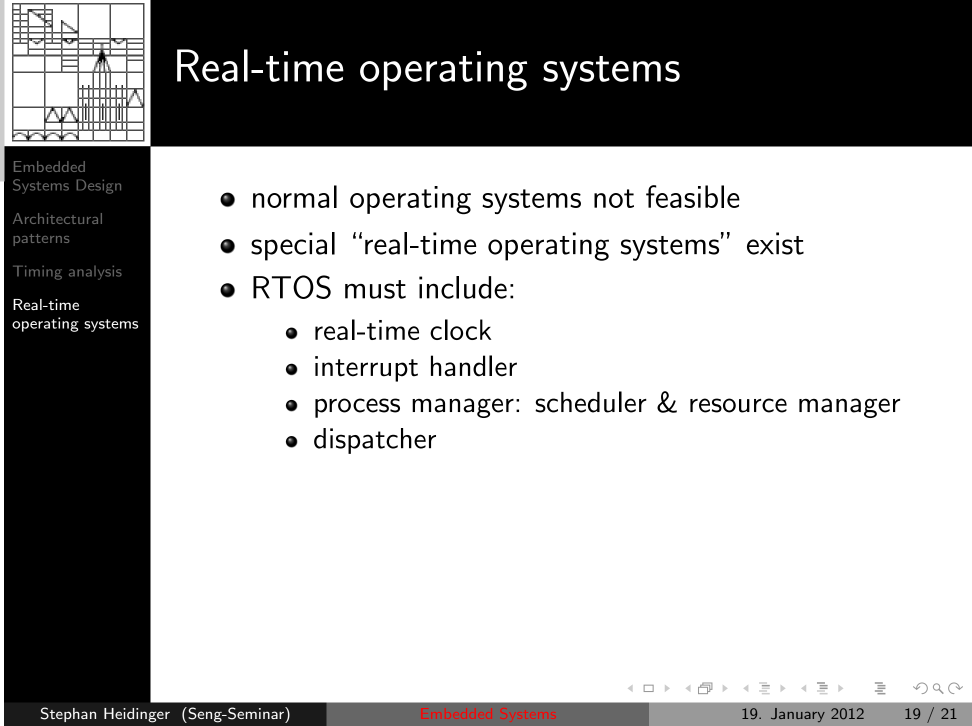


Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

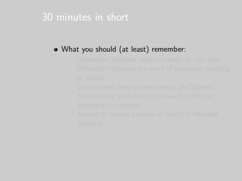
dispatcher:

starts execution of processes



Real-time operating systems

- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher


30 minutes in short

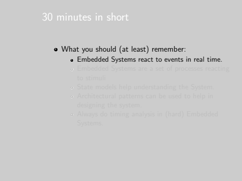
- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.

nearly done

important stuff in short

30 minutes in short

- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.



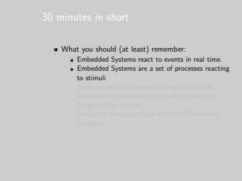
30 minutes in short

- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.

Embedded Systems
react to events in real time

30 minutes in short

- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.



30 minutes in short

- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.

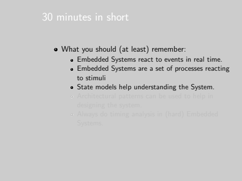
Embedded Systems

react to events in real time

are a set of processes reacting to stimuli

30 minutes in short

- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.



30 minutes in short

- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.

Embedded Systems

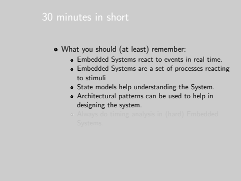
react to events in real time

are a set of processes reacting to stimuli

state model help understanding the system

30 minutes in short

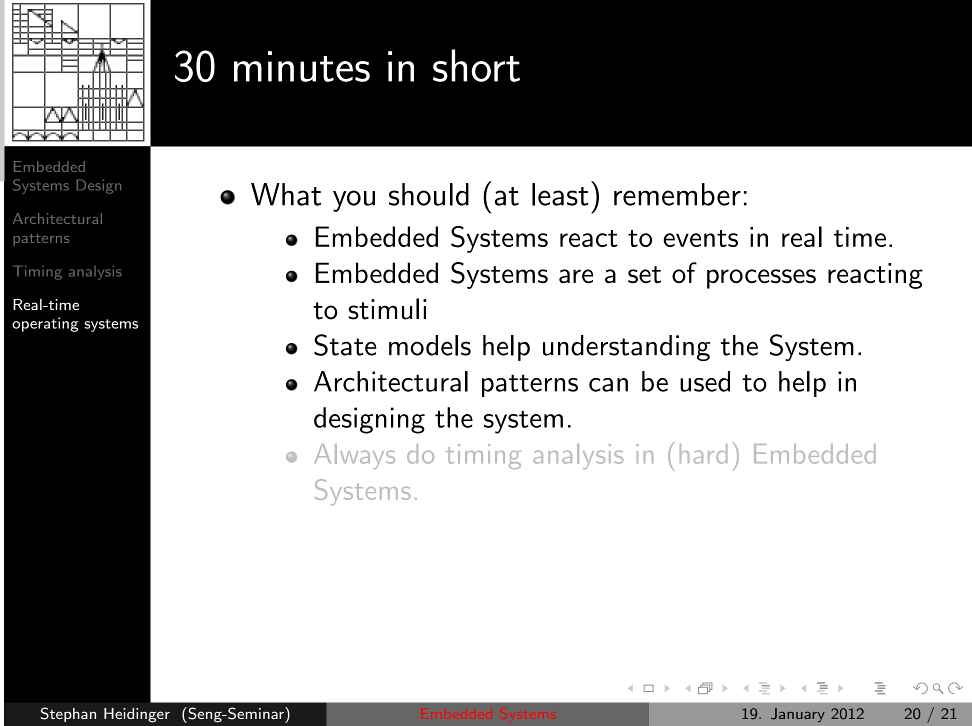
- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.



30 minutes in short

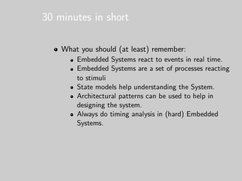
- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.

Embedded Systems
react to events in real time
are a set of processes reacting to stimuli
state model help understanding the system
Architectural patterns help designing the system especially first steps



30 minutes in short

- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.



30 minutes in short

- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.

Embedded Systems
react to events in real time
are a set of processes reacting to stimuli
state model help understanding the system
Architectural patterns help designing the system especially first steps
timing analysis must always be done in (hard) systems

Embedded Systems Design
Architectural patterns
Timing analysis
Real-time operating systems

30 minutes in short

- What you should (at least) remember:
  - Embedded Systems react to events in real time.
  - Embedded Systems are a set of processes reacting to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always do timing analysis in (hard) Embedded Systems.

2012-03-13

Embedded Systems

└Real-time operating systems

└Questions?

Questions?

Questions?

Questions?

Embedded Systems Design

Architectural patterns

Timing analysis

Real-time operating systems

Questions?