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

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 system

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 system

2012-01-09

Embedded Systems

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

Embedded Systems - What's that? - II

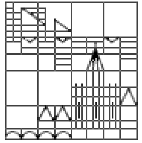
- Embedded Systems: ...
  - ...respond to physical world
  - ...respond in real time ("have a *deadline*")
  - ...often have little resources
  - ...run on special purpose hardware
  - ...run in real-time operating system

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 system

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 system



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 system

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

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 system

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 system

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 system

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
  - ...

2012-01-09

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

- 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
  - (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
  - ...

2012-01-09

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

- 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

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

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
  - ...

Embedded Systems - What's that? - III

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

2012-01-09

Embedded Systems

└ Motivation

Motivation

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

Motivation

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

Motivation

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

Motivation

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

Motivation

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

Motivation

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

Motivation

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

Motivation

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

Motivation
- We see:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
- We realize:
  - Man, they must be important.
  - There sure is some money in this.

Motivation

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

Motivation

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

some money:

C-programing

special skills

Motivation

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

Motivation

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

Motivation

- We see:
  - Embedded Systems are everywhere!
  - There are probably more Embedded Systems than computers out there!
- We realize:
  - 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

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

2012-01-09

Embedded Systems
└ Embedded Systems Design
  └ Outline

Outline

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

Embedded Systems Design

Architectural patterns

Timing analysis

Real-time operating systems

Outline

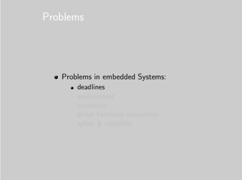
1. Embedded Systems Design
2. Architectural patterns
3. Timing analysis
4. Real-time operating systems

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

Problems

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

environment:

is unpredictable

embedded Software $\Rightarrow$ 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 $\Rightarrow$ does not normally terminate

software has to be reliable

may need update while operating

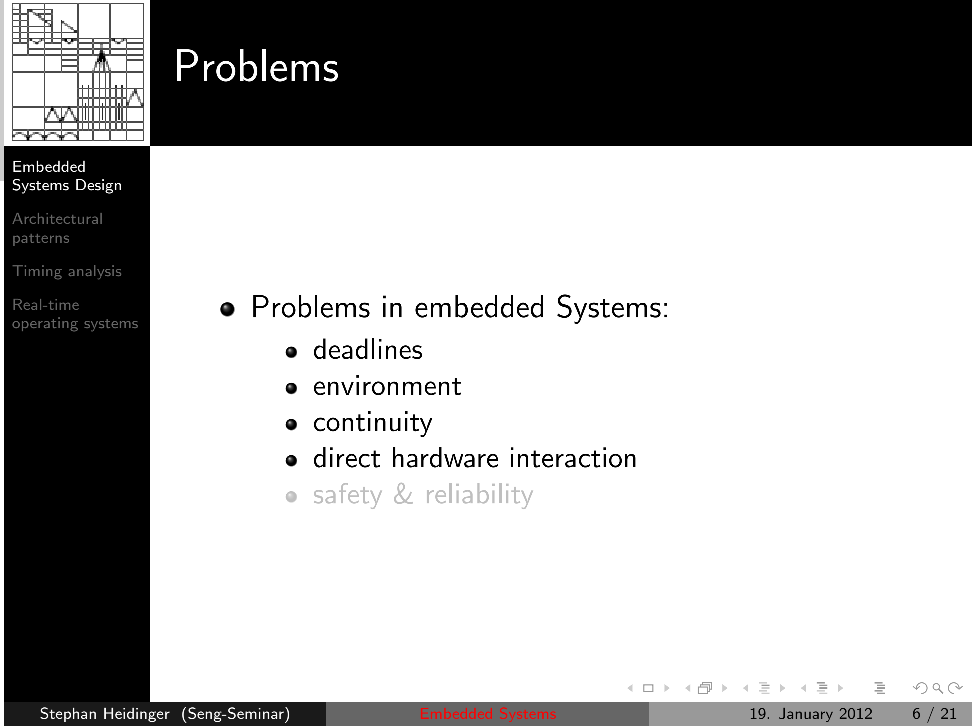
Problems

- 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)



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

next thing:

not all are necessary, but most will be.

Embedded Systems Design - I

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli

Platform selection:

what hardware?

Real-time operating system (later)

What is to be implemented in software, what in hardware

need to design special hardware?

power consumption (mobile device, backup)

Embedded Systems Design - I

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli

Embedded Systems Design - I

- design steps
  - platform selection
  - special purpose hardware
  - periodic stimuli
  - aperiodic stimuli

special purpose hardware:
do we need special hardware?
design special hardware?
replace software by hardware?

Embedded Systems Design - I

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli

stimuli:

describe behavior of system by listing received stimuli and reactions

stimuli = signals

Embedded Systems Design - I

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli

Embedded Systems Design - I
- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli

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

Embedded Systems Design - I

- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli

Embedded Systems Design - I
- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli

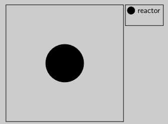
aperiodic stimuli:
occurr irregularly and unpredictably
often interrupts
i.e. alarms, failures, IO operation finished, etc**stimuli list:**
best practice: stimuli list with **all** stimuli.

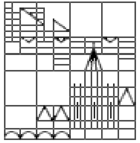
example next slide

Embedded Systems Design - I

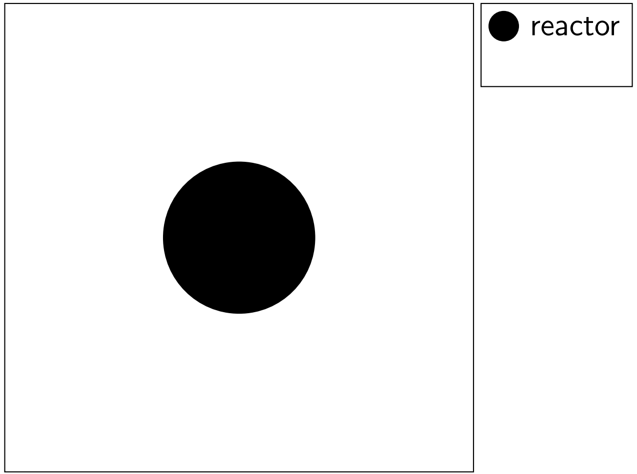
- design steps
  - platform selection
  - special purpose hardware
  - stimuli:
    1. periodic stimuli
    2. aperiodic stimuli

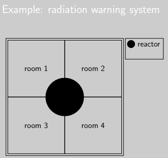
Example: radiation warning system



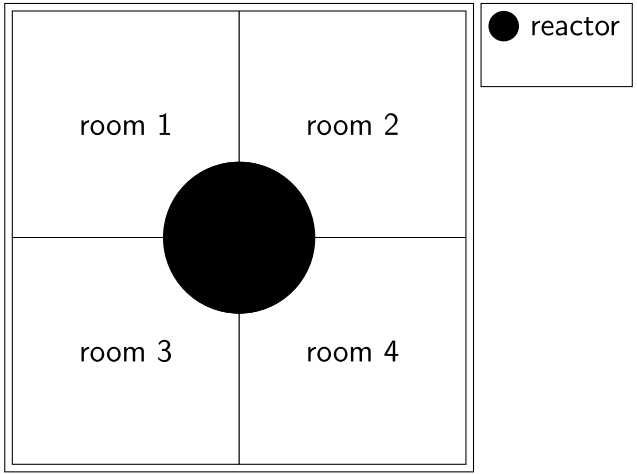


Example: radiation warning system



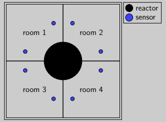


Example: radiation warning system

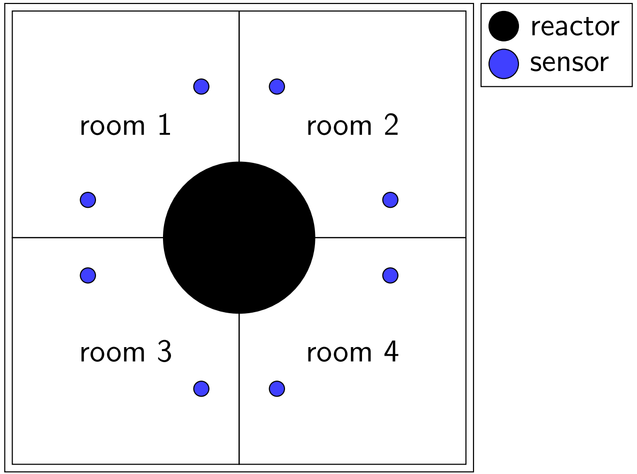


2012-01-09

Example: radiation warning system



Example: radiation warning system



Example: Stimuli-List of a radiation warning system
Stimulus Response

Example: Stimuli-List of a radiation warning system

Stimulus	Response

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

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 around sensor, sound acoustic alarm 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 around sensor, sound acoustic alarm 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 around sensor, sound acoustic alarm around sensor
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 around sensor, sound acoustic alarm around sensor
Voltage drop of 10-20%	switch to backup power; run power supply test

2012-01-09

Embedded Systems

└ Embedded Systems Design

└ Example: Stimuli-List of a radiation warning system

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 around sensor, sound acoustic alarm around sensor
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 maintainer

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 around sensor, sound acoustic alarm around sensor
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 maintainer

2012-01-09

Embedded Systems
└ Embedded Systems Design
  └ Embedded Systems Design - II

Embedded Systems Design - II

- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

Embedded Systems Design - II

- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

Embedded Systems Design - II
design steps - continued
Timing analysis
Process design
Algorithm design
Data design
Process scheduling

Timing analysis:

For each stimulus and response $\Rightarrow$ find timing constraints

timing constraints $\Rightarrow$ deadlines

Embedded Systems Design - II

- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

Embedded Systems Design - II
- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

Process design:

aggregate the stimuli & responses into concurrent processes

See Architectural design

Embedded Systems Design - II

- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

Embedded Systems Design - II
- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

Algorithm design:

For each stimulus & response $\Rightarrow$ design algorithm

especially important for computationally intensive tasks (signal processing)

Do we need to implement these in hardware?

Embedded Systems Design - II

- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

Embedded Systems Design - II

- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design

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 - II

- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

Embedded Systems Design - II
- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

Process scheduling:
ensure, that processes meet their deadline
probably among the hardest (own opionion)
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 - II

- design steps - continued
  - Timing analysis
  - Process design
  - Algorithm design
  - Data design
  - Process scheduling

2012-01-09

Embedded Systems

└Embedded Systems Design

└Embedded system modeling

Embedded system modeling

- Embedded Systems are often built as state machines.

$\Rightarrow$ UML state diagrams

Embedded system modeling

- Embedded Systems are often built as state machines.

  $\Rightarrow$ UML state diagrams

2012-01-09

Embedded Systems
└Embedded Systems Design
└Embedded system modeling

Embedded system modeling

- Embedded Systems are often built as state machines.

$\Rightarrow$ UML state diagrams

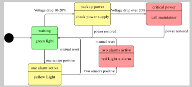
Embedded system modeling

- Embedded Systems are often built as state machines.

$\Rightarrow$ UML state diagrams

2012-01-09

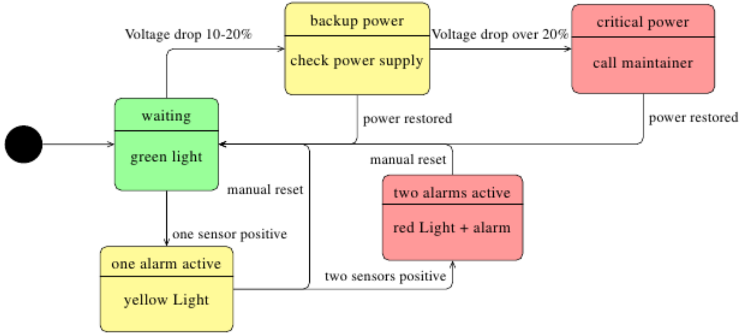
Embedded Systems
└Embedded Systems Design
└Embedded system modeling



Embedded system modeling

- Embedded Systems are often built as state machines.

  $\Rightarrow$ UML state diagrams



Embedded software programing

- programm has to be...

Embedded software programing

- programm has to be...
  - ...fast (i.e. C, Asssembler)
  - ...concurrent (i.e. C++, real time Java, ...)
- speed looses importance

C, Assembler:

No concurrency

no built-in system for shared resources

Embedded software programing

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

- programm has to be...
  - ...fast (i.e. C, Asssembler)
  - ...concurrent (i.e. C++, real time Java, ...)
- speed looses importance

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

Embedded software programing

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

- programm has to be. . .
  - . . . fast (i.e. C, Asssembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance

Embedded software programing

- programm has to be. . .
  - . . . fast (i.e. C, Asssembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance

speed:

due to faster hardware

ie monitoring device written in C++

ie cell phones in java, objective C, . . .

Embedded software programing

- programm has to be. . .
  - . . . fast (i.e. C, Asssembler)
  - . . . concurrent (i.e. C++, real time Java, . . .)
- speed looses importance

2012-01-09

Embedded Systems
└Architectural patterns
    └Outline

Outline

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

note on 3:

The source described three rough design pattern

there are finer patterns

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 happens *Rightarrow* we do something
ie incoming phone call

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

Environmental Control:

set of sensors and actuators

can change environment

ie flash light, when sensor fires

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 *Rightarrow* only for understanding system

Architectural patterns

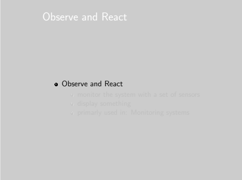
- Architectural patterns are used to describe a system in an abstract way and help to understand the architecture.
  - Observe and react
  - Environmental Control
  - Process Pipeline

2012-01-09

Embedded Systems

└─Architectural patterns

└─Observe and 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:

as stated before

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

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
  - monitor the system with a set of sensors
  - display something
  - primarly used in: Monitoring systems

monitoring systems:

often consist of more than one O&R patterns, 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

2012-01-09

Embedded Systems
└ Architectural patterns
└ 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.

Environmental Control

- Environmental Control
  - monitor the system 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.

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.

examples:

break assist

airbag

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 . . .

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.

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.

2012-01-09

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

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

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:

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

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

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

2012-01-09

Embedded Systems
└Timing analysis
└Outline

Outline

3 Timing analysis

Outline

2012-01-09

Embedded Systems
└Timing analysis
  └Timing Analysis - I

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 $\Rightarrow$ 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

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?

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

fast systems:

use only periodic stimuli

poll frequently for aperiodic stimuli

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 $\Rightarrow$ make assumptions

2012-01-09

Embedded Systems
└ Timing analysis
　└ Timing Analysis - II

Timing Analysis - II

- Consider:

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

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
voltage drop switch to backup: 50ms

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

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

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

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

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

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

LAST CELL:

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

2012-01-09

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

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

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 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

real-time clock:

provides information required to schedule processes

Real-time operating systems

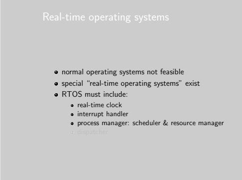
- normal operating systems not feasible
- special "real-time operating systems" exist
- RTOS must include:
  - real-time clock
  - interrupt handler
  - process manager: scheduler & resource manager
  - dispatcher

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



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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to 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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to timing analysis in (hard) Embedded Systems.

2012-01-09

Embedded Systems

Real-time operating systems

30 minutes in short

30 minutes in short

- What you should (at least) remember:
  - 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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to 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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to 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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to timing analysis in (hard) Embedded Systems.

2012-01-09

Embedded Systems
└Real-time operating systems
└30 minutes 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 reating to stimuli
  - State models 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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to 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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to 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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to 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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to 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 reating to stimuli
  - State models help understanding the System.
  - Architectural patterns can be used to help in designing the system.
  - Always to timing analysis in (hard) Embedded Systems.