Programming Embedded Systems

Lecture 1

Introduction to the course

Monday Jan 16, 2012

Philipp Rümmer
Uppsala University
Philipp.Ruemmer@it.uu.se

Lecture outline

- Organisation
    - Teachers
    - Lectures, exercises, labs, project
- Topics + focus of the course

- Recap of the C language

About myself (Philipp Rümmer)

- At UU since 2011,
  research assistant in
  embedded systems group
- Main background:
  formal methods, verification
- **In this course: lectures**

http://www.philipp.ruemmer.org
Philipp.Ruemmer@it.uu.se

About Kai Lampka

- At UU since 2012, lecturer in embedded systems group
- **In this course: lectures + exercises + labs**

http://www.it.uu.se/katalog/kaila126
kai.lampka@it.uu.se

About Othmane Rezine

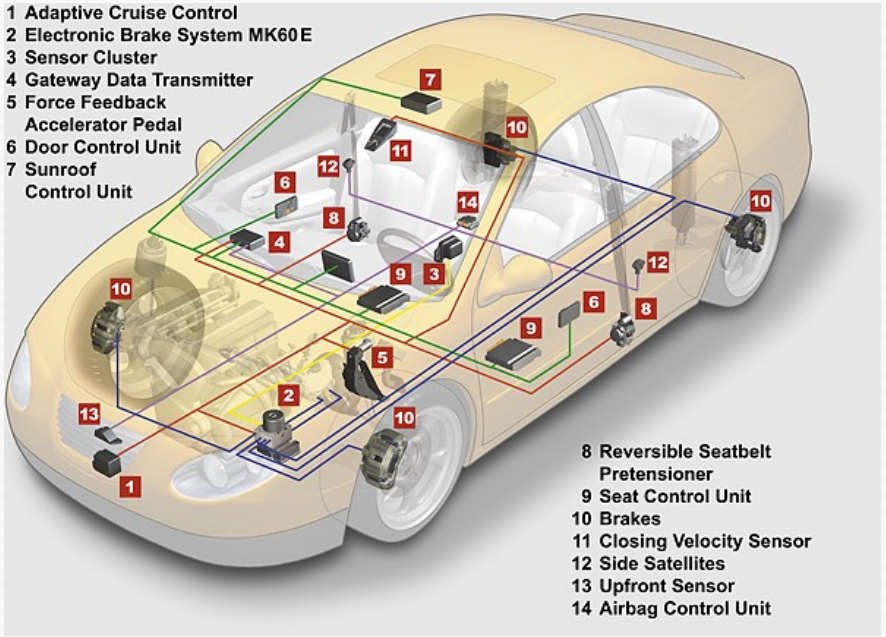
- PhD student in verification group
- **Will take care of exercises + labs**

http://www.it.uu.se/katalog/othre279
othmane.rezine@it.uu.se

Course topics

Recap: Embedded Systems

- Computer systems **integrated** into a larger device
- Hardware + **software** tailored to a particular purpose
- About **99%** of all computers are embedded



Pervasive:
Cell phones, cameras, trains, airplanes, traffic lights, home appliances, robots, industrial machines, etc.

Embedded systems (2)

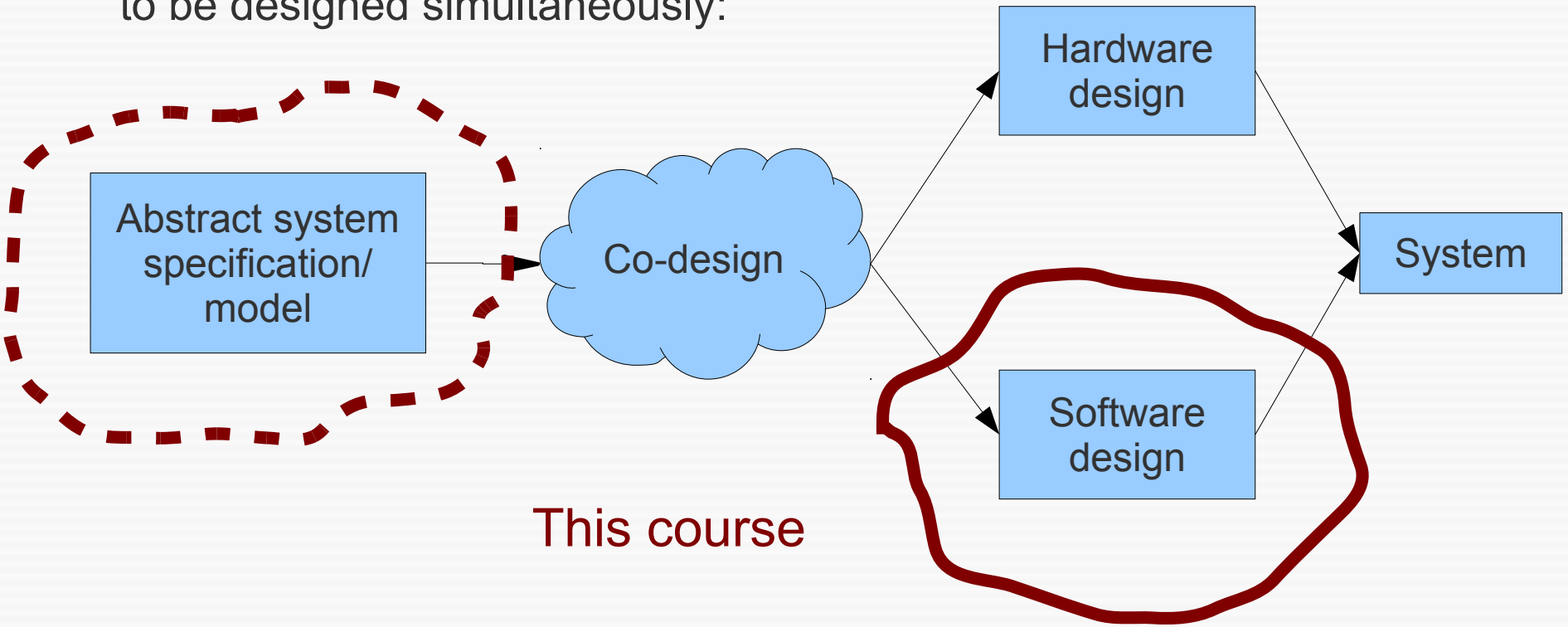
- System: **hardware + software**
- **Often constrained** in various ways:
  - Timing (real-time requirements)
  - Severely limited resources:
    weight, power, memory,
    computation power
  - Have to be cost-effective

Reliability

- Embedded systems are often **complex** and **safety-critical**
  - → Millions LOC
  - → Failures might be fatal
- **How to ensure reliability?** (Recurring topic in this course)
- Connected to various research areas: e.g., verification, testing

Course location: hardware/software co-design

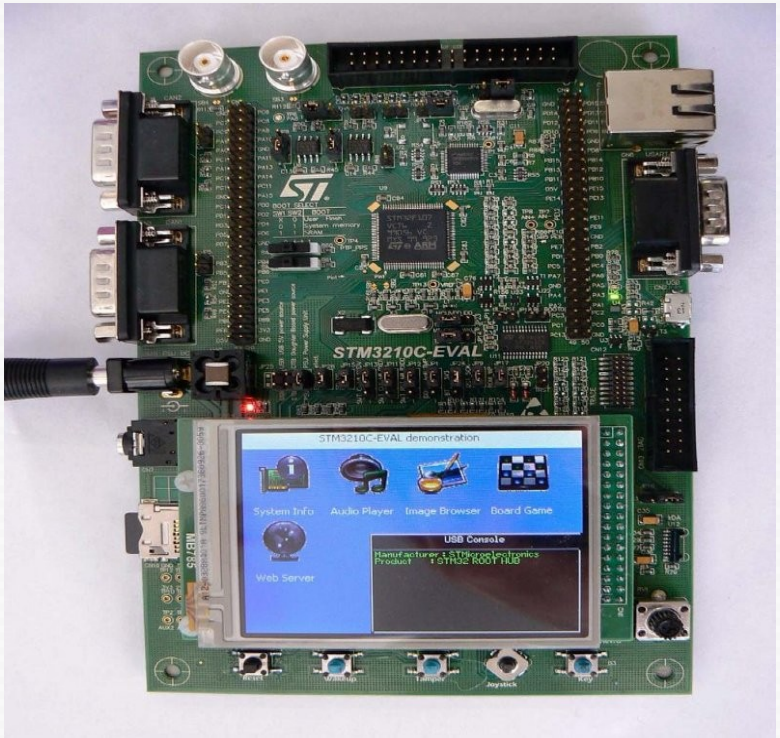
Embedded systems require
hardware and software
to be designed simultaneously:



Course covering (more) co-design:
Microcontroller Programming, Uwe Zimmermann

Topic 1: Practical stuff

- Development for embedded systems: **hardware features, programming, testing, debugging, simulation**
- Mainly considered: ARM CORTEX M3
- IDE + compiler: Keil/ARM µVision



Keil/ARM µVision

- Installed on Windows lab computers (in 1313)
- If you want to use your own computer: evaluation licence from http://www.keil.com/uvision/ (sufficient for this course)

Topic 2: Operating Systems

- OS simplifies development of systems:
  - Multi-tasking, scheduling, task pre-emption, deadlines
  - Synchronisation, shared resources
  - Drivers for communication, periphery
  - Interrupt handling
- Large variety of OSs common for embedded systems
  - e.g, LynxOS, VxWorks, Windows CE, RT-Linux, FreeRTOS, ECOS, OSE, QNX, Integrity, …

Main OS used here: FreeRTOS

- Small industrial OS, open-source (GPL)
- C API
- Satisfies hard real-time requirements
- Pre-emptive/cooperative multi-tasking, co-routines
- Fixed-priority scheduler
- Platforms: ARM, x86, Freescale, ...

http://www.freertos.org/

FreeRTOS (2)

- Will be introduced in lectures,
  used for assignments + labs + project

- Supporting book:
  Richard Barry, *“Using the FreeRTOS Real Time Kernel - a Practical Guide”*

Real-time Linux

- Larger, more powerful OS
- Introduced towards end of period 3

Related course topics

- Interrupt handling
- Accessing ports,
  devices like sensors, actuators, buses
- Memory management
- Synchronisation,
  inter-task communication

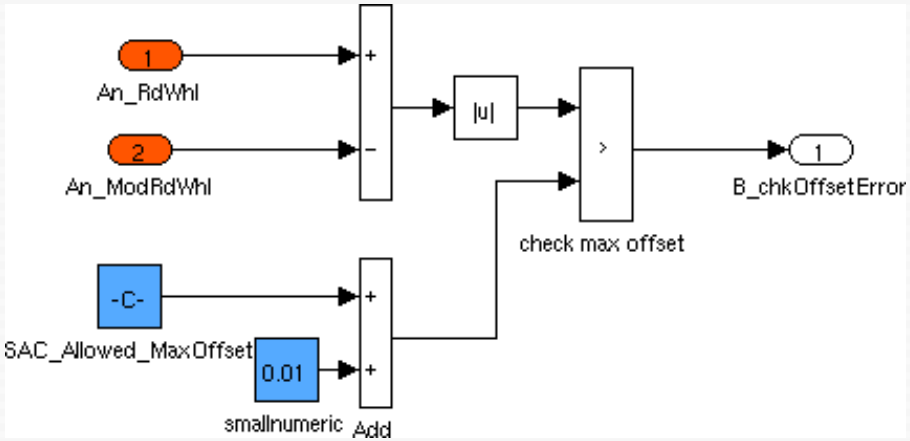
Topic 3: programming lang.

- **Which language to write embedded software in?**
- Traditional: low-level languages,C
- Trends: **high-level, declarative, model-based, component-based** languages

C

```
void setupActuatorModule() {
  TIM_TimeBaseInitTypeDef timInit;

  /* Setup timer TIM3 for pulse-width modulation:
     100kHz, periodically counting from 0 to 9999
  RCC_APB1PeriphClockCmd( RCC_APB1Periph_TIM3, ENA

  TIM_DeInit( TIM3 );
  TIM_TimeBaseStructInit( &timInit );

  timInit.TIM_Period = (unsigned portSHORT)0x270F;
  timInit.TIM_Prescaler = 720;

  timInit.TIM_ClockDivision = TIM_CKD_DIV1;
  timInit.TIM_CounterMode = TIM_CounterMode_Up;
```

Simulink

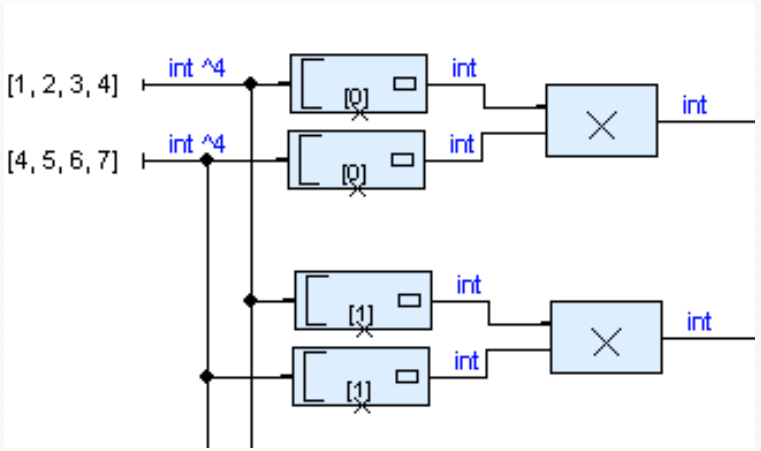


Low-level programming

- Most of the course will be **based on C**
- Knowledge of C programming is needed for the course
- We will give some recap and exercises in the beginning of the course

Lustre, synchronous prog.

- **Lustre**, Esterel, Signal
- Execution governed by a global clock, static scheduling
- Determinism is guaranteed (despite concurrency)
- Sometimes also used for modelling/prototyping

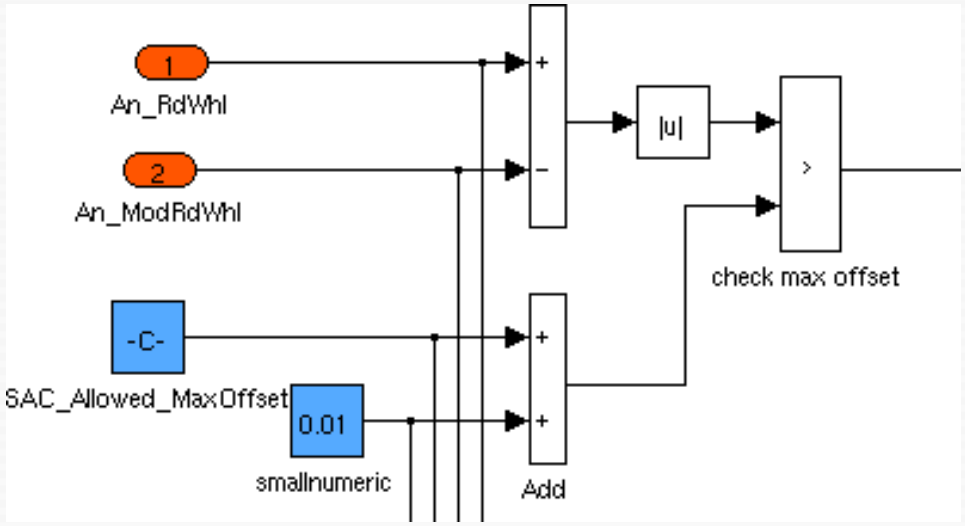


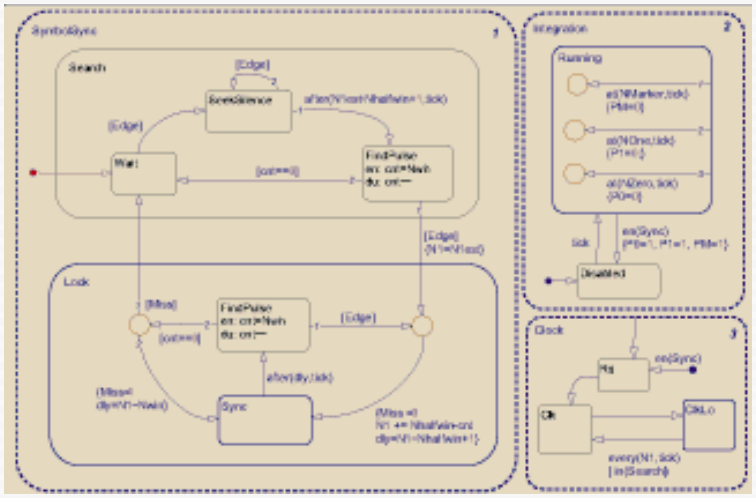
High-level imperative lang.

- **Real-time Java**, Ada 95
- High-level heap model
- Scoped memory
  (garbage collectors are difficult in real-time systems)
- Built-in real-time primitives

Graphical languages

- Matlab/Simulink, SCADE/Lustre
- Mostly done in course *“Model-based design of embedded software,” Bengt Jonsson*

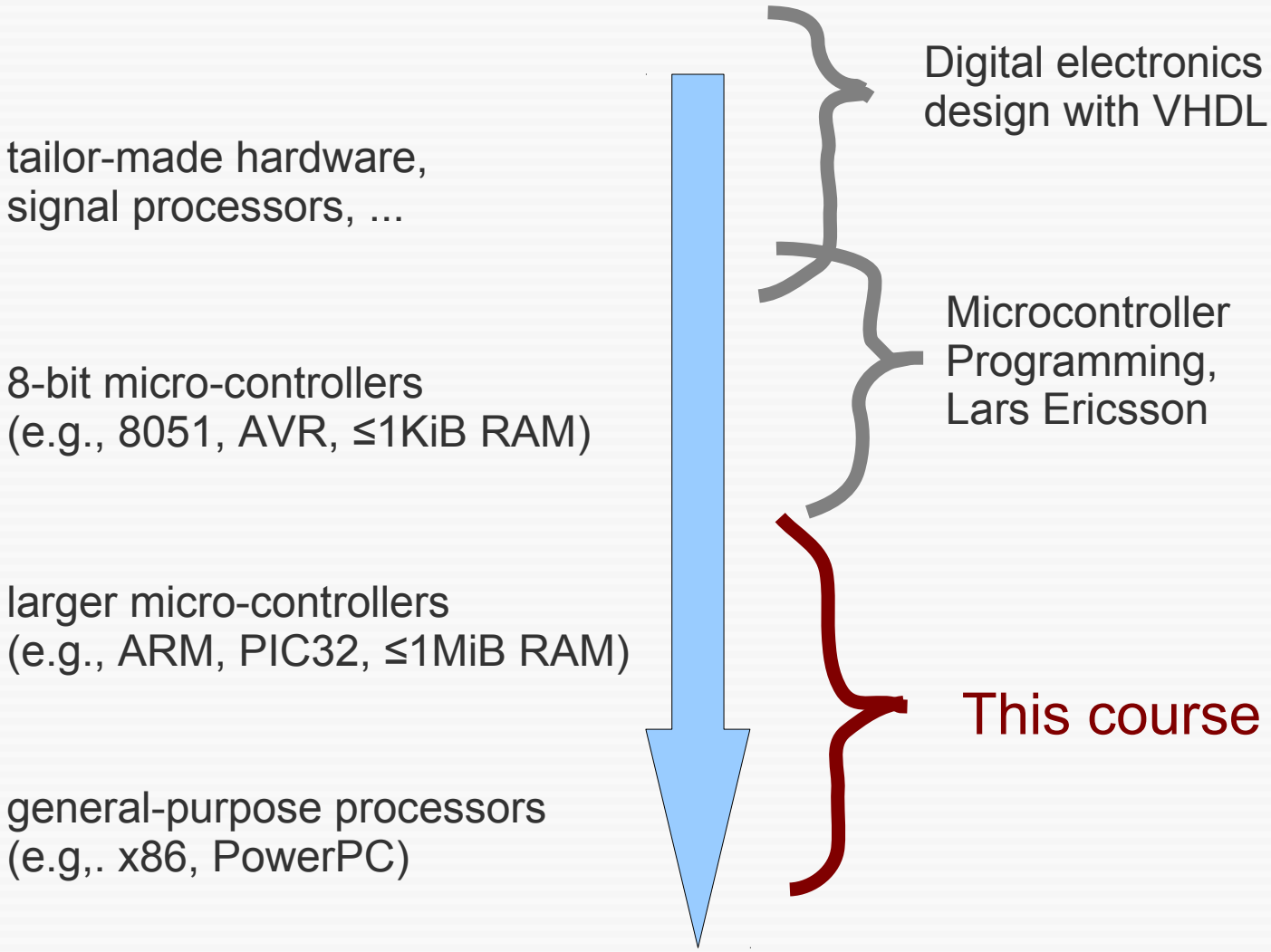




Topic 4: correctness + reliability

- Requirements, safety properties
- Correctness:
  simulation, testing, debugging,
  verification
- Fault tolerance, redundancy
- Determinism, predictability
- Pitfalls with arithmetic datatypes
  (floating-point, fixed-point)

Course location: considered hardware



Course location: software architectures

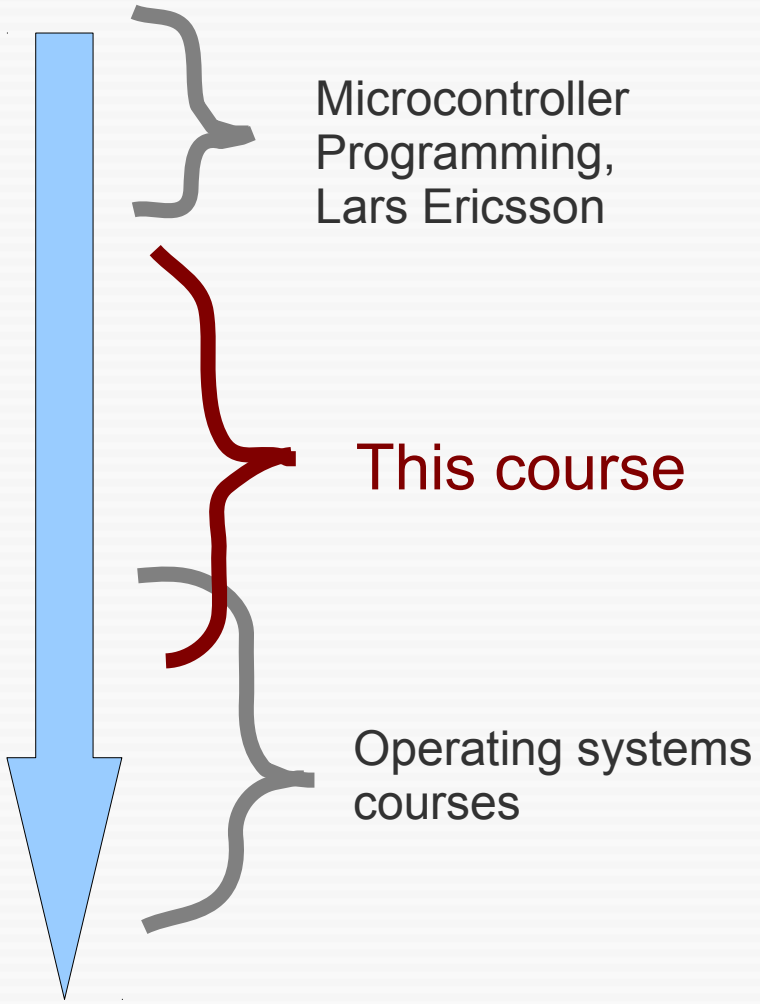
no operating system,
simple control loop

dedicated RTOS
(e.g., LynxOS, VxWorks,
Windows CE)

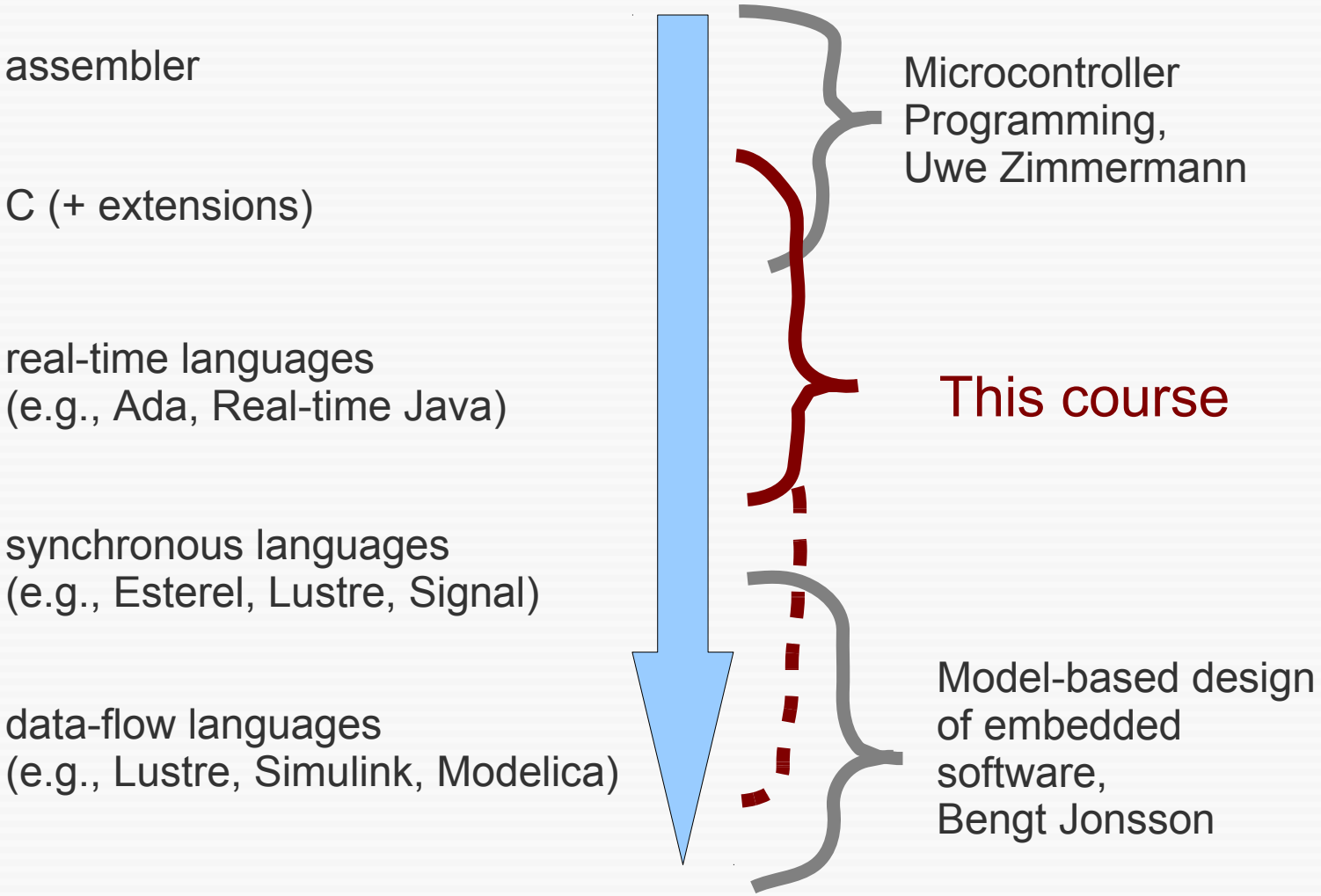
POSIX 1003.1b
(standard for real-time OSs)

generic OS extended for RT
(e.g., RT-Linux)

generic OS
(e.g., Unix, Windows)



Course location: programming languages



Organisation of the course

Main structure of the course

Part 1
period 3, week 3-11
15 lectures (±)
6 assignments, 1 lab (3hp)

Main topics:
operating systems, programming languages, development, debugging, testing, technology … for embedded systems

Part 2
period 4, week 12-21

Embedded systems project (4hp)

Exam: May 25th (3hp)

Lectures

- Normally 2 lectures per week,
  2 hours each
- Sometimes tutorial-style
  (black-board + computer),
  some more theoretic (slides)

- Lecture material (slides, examples)
  will be available on course page
  `http://www.it.uu.se/edu/course/homepage/pins/vt12`

Exercises

- Weekly, Thursdays or Fridays (check webpage for exact time)
- Mostly for discussing assignments + general discussions

- First exercise:
  Friday Jan 27th, 8:15 – 10:00, 1245

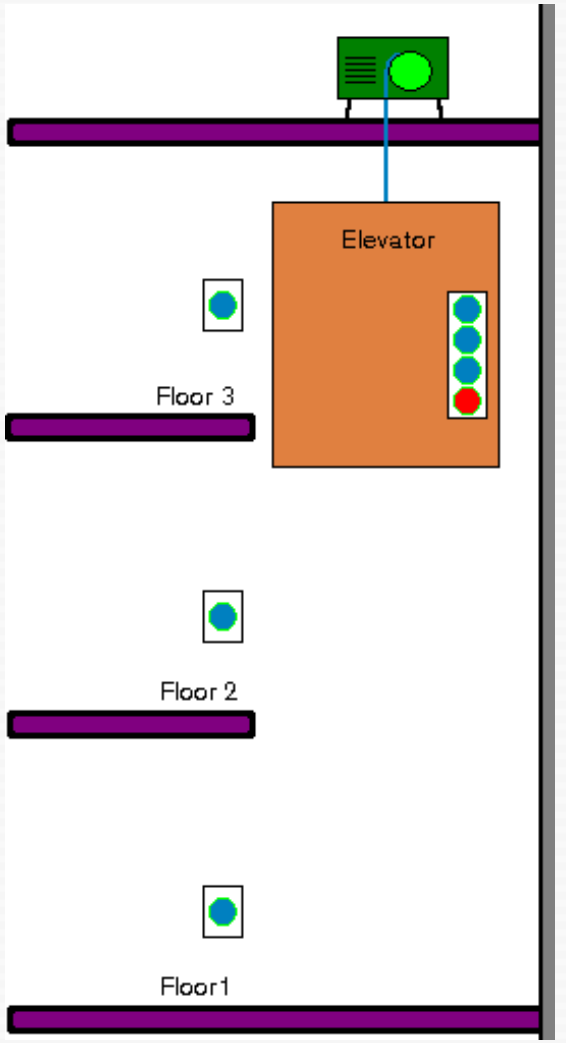
  (no exercise this week!)

Assignments

- 6 weekly assignments, **solved by students individually**
- Graded with points: 0 - 20
- To pass an assignment, **≥ 12 points** have to be reached
- **≥ 4 assignments have to be handed in + passed**
- Assignment solutions are discussed in exercises

Lab

- **Done in groups (2 people)**
- Various aspects of developing an embedded system (elevator system): *specification, design, implementation, testing*
- **Running weeks 5 - 10**
- Done using simulator
  → no real embedded hardware



More infos later + on course page

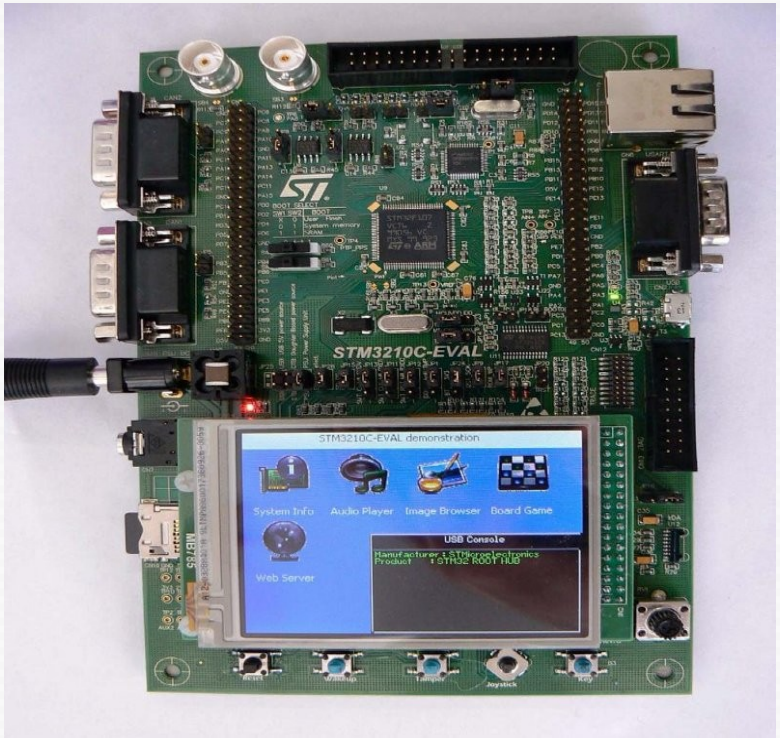
Lab (2)

- We will give lab support once a week (starting week 5)
- **What you should do already now:**
  - Choose your groups
  - Sign up for groups on `studentportalen.uu.se`

More infos later + on course page

Project (period 4)

- Larger groups (3-4 people)
- Use of actual “embedded” hardware
- **Project results will by graded U, 3, 4, 5** (→ part of overall course grade later)
- More details later



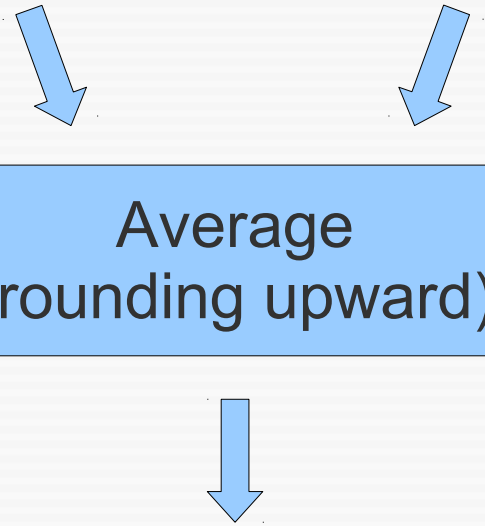
Exam

- May 25th
- Graded U, 3, 4, 5
- Will be short (probably 2 hours)
- Not all topics from the course will be relevant for exam (since some are tested in assignments + project)
- **Precise list of relevant topics will be made available on course page**

Course grade

Project grade
(groups, 3, 4, 5)

Exam grade
(individual, 3, 4, 5)



Individual
overall course grade
(3, 4, 5)

What remains

Further information

- Course page:
  http://www.it.uu.se/edu/course/homepage/pins/vt12
- **There is a forum for questions on studentportalen.se**

Always check the forum before sending us an email!

Further reading

- **"An embedded software primer"**
  David E. Simon, Addison-Wesley, 1999
- **"Hard Real Time Computing Systems – Predictable Scheduling Algorithms and Applications"**
  Giorgio Buttazzo, Springer, 2005
- **"Using the FreeRTOS Real Time Kernel - a Practical Guide"**
  Richard Barry, generic CORTEX M3 ed.

Next lecture

Pol_1212

- Wednesday, Jan 18, 10:15, Pol_1245
- Intro to fixed-priority scheduling
- Intro + tutorial to FreeRTOS

Rest of this lecture

- Questionnaire
- Recap of C programming