Module Description

Embedded Real-time Software

General Information

Number of ECTS Credits

3

Abbreviation

TSM_EmbReal

Version

30. August 2009

Responsible of module

Hans Buchmann, FHNW

Language

	Lausanne	Bern	Zürich
Instruction	x E x F	☐ D ☐ E ☐ F	x D x E
Documentation	x E ☐ F	☐ D ☐ E ☐ F	☐ D x E
Examination	x E x F	☐ D ☐ E ☐ F	x D x E

Module category

☐ Fundamental theoretical principles

x Technical/scientific specialization module

☐ Context module

Lessons

x 2 lecture periods and 1 tutorial period per week

☐ 2 lecture periods per week

Brief course description of module objectives and content

The module Embedded Real-Time Software conveys the most important topics, concepts and issues related to the development of embedded applications. Target embedded systems include simple environment without operating system as well as more complex environment with real-time kernel and additional features. This module will address the relevant development steps involved in this kind of development, from the concepts, design and implementation to the deployment, debugging and maintenance.

Aims, content, methods

Learning objectives and acquired competencies

At the end of this module, the student will be able:

- to describe the different architectures of embedded systems on which software may be deployed;
- to recognize the various kinds of synchronous/asynchronous interactions between hardware and software;
- to deduce from software requirements domain-oriented software architectures and interface specifications;
- to design an embedded application taking into account limited hardware-related and time-related constraints;
- to understand how behavioral models are verified and implemented;
- to explain the basics of operating systems;
- to discuss the properties and consequences of different real-time scheduling policies;
- to discuss the pros and cons of operating systems on the market and to select an adequate operating system for a specific application;
- to explain how applications are developed with programming formalisms and modeling frameworks intended to be deployed in different execution environments (with/without OS, monitors, etc.);
- to write embedded applications which deal with hardware (drivers, bootstrap code, etc.);
- to debug applications by means of specific debugging and monitoring tools;
- to port applications from an architecture to another.
- to avoid typical pitfalls of embedded real-time software

Contents of module with emphasis on teaching content

Application Development

- Problem Analysis:
    - Domain-orientation, processes and devices, functionality and response times, domain oriented separation

of concerns, generic software architecture and development process.

- Modelling of Reactive Components:
  - model-based abstraction, formalisms and methods, interacting state machines, architecture and behaviour, synchronous and asynchronous composition, behavioural correspondence, model verification and code generation.
- Input/Output Connection of Reactive Components:
  - interaction of external processes and reactive components, specification of interaction layers, interaction functions as interlayer services, cycle-based execution of interaction functions.
- Real-Time Scheduling Concepts:
  - static and dynamic scheduling algorithms, priorities and deadlines, cycle-based execution architecture, real-time task servers, scheduling hierarchies, schedulability verification.

Real-Time System Execution Environment

- Concurrency
  - Thread vs. EventDriven
- Real-Time Operating System

  OS Architecture and subsystems, kernel/user space and syscalls, memory management, task scheduling, inter-task communication and synchronization, OS on the market
- Multicore and Virtualization
  - Processor architecture and multicore, SMP, Hyperthreading, virtual machine monitor, full/para-virtualization, address space and driver isolation
- Failure handling

Hardware / Software Related Interactions

- System Architecture
  - Generic architecture, finite state machine, monoprocessors and multiprocessors, coprocessors, symmetric multiprocessing, non-uniform memory access.
- Hardware Design with Real-Time Constraint
  - Bus interactions, memory architecture, memory hierarchy, cache, memory management unit, interrupt controller
- Software Access for Hardware Resources
  - Board Support Package, bootloader, system startup, hardware debugging, JTAG, embedded file systems (flash, RAM, etc.)
- Drivers
  - Driver structure, Driver types (char, block, network, etc.), processor mode levels, communication between mode levels, driver models, major/minor, concurrency in the kernel, interactions between user and kernel space, DMA, interrupt service routines and deferred processing

Teaching and learning methods

- Ex-cathedra teaching
- Exercises
- Self-study (study of papers, case studies)

Prerequisites, previous knowledge, entrance competencies

- Programming , language C
- Computer architectures
- Fundamentals of Operating Systems

Literature

- Burns, A. Wellings. Real-Time Systems and Programming Languages. Addison-Wesley, third edition, 2001.
- F. Cottet, J. Delacroix, C. Kaiser, Z. Mammeri. /Scheduling in Real-Time Systems/. Wiley, 2002.
- G. C. Buttazzo. Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Applications. Second Edition, Kluwer Academic Publishers, 2005.
- L. Zaffalon. Programmation synchrone de systèmes réactifs avec Esterel et les SyncCharts. PPUR, 2005. (English Version planned for late 2008)
- Colin Walls. Embedded Software: The Works. Newnes, 2006

Jack Ganssle. The Firmware Handbook. Newnes, 2004

Assessment

Certification requirements for final examinations (conditions for attestation)

none

Written module examination

Duration of exam :	120 minutes
Permissible aids:	Personal summary