Echtzeitsysteme 2

Analyse, Entwurf, Implementierung

Fabian Scheler
Michael Stilkerich
Wolfgang Schröder-Preikschat

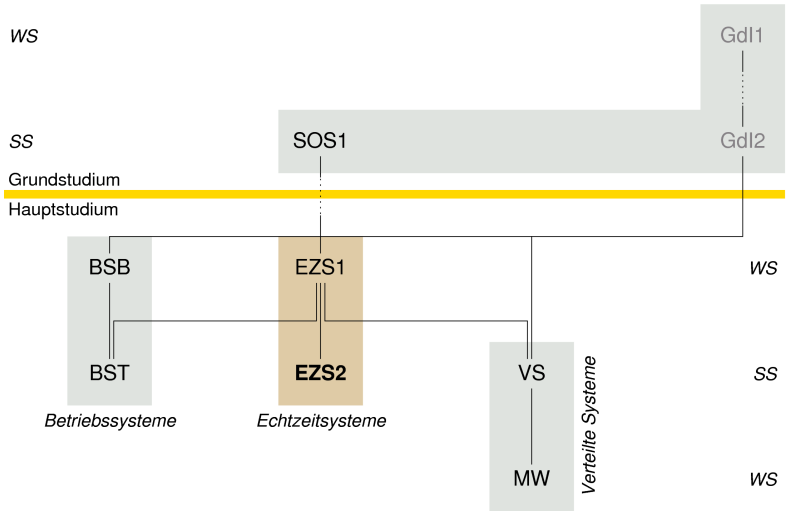
Lehrstuhl für Informatik IV
Verteilte Systeme und Betriebssysteme
Friedrich-Alexander Universität Erlangen-Nürnberg

http://www4.cs.fau.de/~{scheler,mike,wosch}
{scheler,mike,wosch}@cs.fau.de

Übersicht

- Einordnung
- Aufbau/Lehrform
- Voraussetzungen
- Leistungsnachweis
- Literatur
- Lernziele & Lehrinhalt
- Experimente

Lehrprofil von I4



Integrierte Lehrveranstaltung

- Termine
  - Vorlesung (2 SWS) + Übung (2 SWS)
    = 4 SWS (3 Stunden/Woche)
- Vor-/Nacharbeit
  - N Stunden/Woche: $0 \leq N \leq (165 - X)$
  - $X < 165$: Zeitäquivalent anderer Pflichten
- Folien/Handout
  - `www4.informatik.uni-erlangen.de/Lehre/SS07/V_EZS2`
  - kein Skript – Folien zu Vorlesung/Übung

Ablauf/Material

- Modus
  - wöchentlicher Wechsel (2 SWS):
    - Vorlesung = inhaltliche Einführung zu den Projektphasen
    - Übung = Statusbericht der Gruppen zu den Projekten
  - „restlichen“ 2 SWS:
    - Bearbeitung der Projekte
- Ort und Termin???

→ Abstimmung

▪ Möglichkeit 1:	Dienstag	10:00 – 12:00	(0.031)
▪ Möglichkeit 2:	Mittwoch	14:00 – 16:00	(0.031)
▪ Möglichkeit 3:	Donnerstag	14:00 – 16:00	(00.153)

Übungsbetrieb

- Anmeldung
  - über W.A.S. (Web-Anmeldesystem)
  - `www.was.dienste.uni-erlangen.de/content`
  - Begin: ab sofort ...
- Projektarbeit
  - ≤ 3 Mitarbeiter je Gruppe
    - Einzelkämpferdasein ist nicht zielführend
  - 14-tägige Statusberichte
- Rechnerübung: manlobbi – Raum 0.058
  - freies Arbeiten, Betreuung „on demand“

→ Kontinuität und aktive Mitarbeit als Schlüssel zum Erfolg!

Voraussetzungen

- Grundlagen von Echtzeitbetriebssystem: EZS 1
  - zeit- und ereignisgesteuerte Systeme
  - periodische, aperiodische und sporadische Aufgaben
  - Einplanung und Koordination
- Systemprogrammierung in C/C++ und Assembler
  - maschinen- d.h. hardwarenahe Programmierung

→ Betriebssystemkenntnisse sind fördernd,
daher erwünscht und hilfreich

Leistungsnachweis

- unbenoteter Schein
  - erfolgreiche Bearbeitung des Projekts
  - Statusberichte / Abschlusspräsentation
  - Poster
- benoteter Schein
  - unbenoteter Schein
  - Bewertung der erbrachten Leistungen

Kontakt

- Fabian Scheler
  www4.informatik.uni-erlangen.de/~scheler
  scheler@informatik.uni-erlangen.de
- Michael Stilkerich
  www4.informatik.uni-erlangen.de/~mike
  stilkerich@informatik.uni-erlangen.de
- auf folgenden Wegen
  - persönlich
  - e-Mail (**Mailingliste!**)
  - ICQ, Jabber, ...

Ergänzende Literatur

- *Hermann Kopetz.*
  Real-Time Systems: Design Principles for Distributed Embedded Applications.
  Kluwer Academic Publishers, 1997.
- *Jane W. S. Liu.*
  Real-Time Systems.
  Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.
- *Jim Cooling.*
  Software Engineering for Real-Time Systems.
  Addison-Wesley, 2003.
- *Phillip A. Laplante.*
  Real-Time Systems Design and Analysys.
  Wiley, third edition, 2004.
- *W. Schröder-Preikschat.*
  Echtzeitsysteme.
  www4.informatik.uni-erlangen.de/Lehre/WS06/V_EZS

Lernziele

- Echtzeitprogrammierung in Teams
- Echtzeitanwendung entwickeln
  - Anforderungen/Gegebenheiten analysieren und spezifizieren
  - Echtzeitsystem problemorientiert implementieren
  - Experimente aufbauen und durchführen
- → Vertiefung von Grundlagenwissen durch experimentelles Arbeiten

Lehrinhalte - Projektphasen

1. Analyse des physikalischen Objekts
   - Erfassen physikalischer Zusammenhänge
   - Wo existieren Deadlines?
   - Wie sieht das physikalische Modell aus, wo abstrahiert man?
2. Entwicklungsumgebung
   - Was für ein
     - Prozessor
     - Compiler
     - Betriebssystem

     wird verwendet?
   - Wie funktioniert das Debuggen?
   - Gibt es irgendwelche Besonderheiten?

Lehrinhalte - Projektphasen

3. Komponenten
   - Welche Komponenten existieren in meinem System?
   - Was tun diese Komponenten? (Spezifikation)
   - Wie interagieren diese Komponenten? (Spezifikation)
   - Implementierung der Komponenten?

4. Testen: Komponenten
   - einfache Testumgebung:
     - übersetzt und bindet Testfall
     - führt Testfall aus
     - überprüft das Ergebnis
   - Spezifikation der Testfälle
   - Getestet wird: Funktionalität, WCET und Speicherplatz

Lehrinhalte - Projektphasen

5. Komposition
   - Abbildung: Komponenten → Ereignisbehandlungen
   - Planung:
     - Prioritätenvergabe
     - statische Ablaufpläne
   - Planbarkeitsanalyse

6. Integration
   - Implementierung des kompletten Systems
   - Abbildung auf Betriebssystemstrukturen

Lehrinhalte - Projektphasen

7. Testen: komplettes System → Akzeptanztest
   - Ausprobieren
   - Debuggen

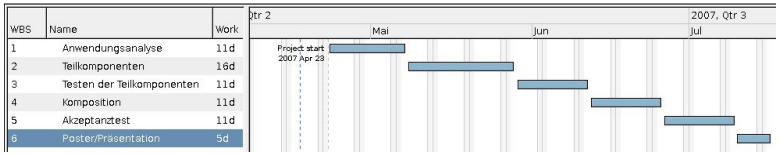
8. Präsentation
   - Zusammenfassung des Projekts

Lehrinhalt - Struktur

- zu jeder Phase gibt es einen Zwischenbericht
  - ca. 10-minütige Präsentation
  - Zu welchen Ergebnissen ist man gekommen?
  - Welche Designentscheidungen wurden getroffen? Warum?
  - Welche Konsequenzen haben diese Entscheidungen?

- Abschlussbericht / Abschlusspräsentation
  - ca. 45-minütige Präsentation / Demonstration
  - Überblick über das Projekt
  - Poster
  - setzt sich aus Zwischenberichten zusammen

Zeitplanung



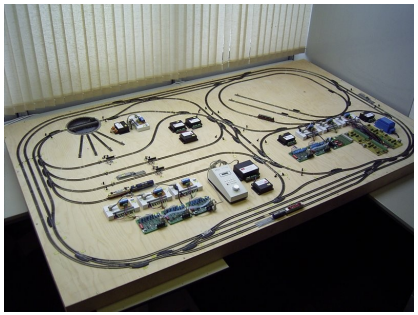
	Phase	Start	Ende
1	Anwendungsanalyse	23.04.2007	07.05.2007
2	Teilkomponenten (Entwurf & Implementierung)	08.05.2007	28.05.2007
3	Teilkomponenten (Testen & WCET)	29.05.2007	11.06.2007
4	Komposition	12.06.2007	25.06.2007
5	Akzeptanztest	26.06.2007	09.07.2007
6	Poster/Präsentation	10.07.2007	16.07.2007

Experiment 1: Hau den Lukas

- Aufgabe: den Eisenkern ...
  - anheben und abbremsen
  - schrittweise anheben/fallen lassen
  - gebremst fallen lassen
  - pendeln lassen
- Entwicklungsumgebung
  - ProOSEKtime
  - GCC
  - TriCore
  - Digital I/O
  - A/D-Wandler
  - TTCAN

Experiment 2 & 3: Eisenbahn

- Aufgabe: Zug-/Weichensteuerung
  - (un)veränderliches Streckennetz
  - Weichenstellung
  - Signalanlagen
- Entwicklungsumgebung
  - KESO/ProOSEKtime
  - TriCore
  - Digital I/O
  - serielle Schnittstelle
  - TTCAN



Experiment 4: Robertino

- Aufgabe: Steuerung
  - Bewegung in der Ebene
  - Vermeidung von Kollisionen
- Entwicklungsumgebung
  - KESO
  - TriCore
  - Digital I/O
  - D/A-Wandler
  - CAN