

Echtzeitsystemlabor

Analyse, Entwurf, Implementierung

Peter Ulbrich
Fabian Scheler
Wolfgang Schröder-Preikschat

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme
Friedrich-Alexander Universität Erlangen-Nürnberg

<http://www4.cs.fau.de/~{scheler,ulbrich,wosch}>
{scheler,ulbrich,wosch}@cs.fau.de



1

Übersicht

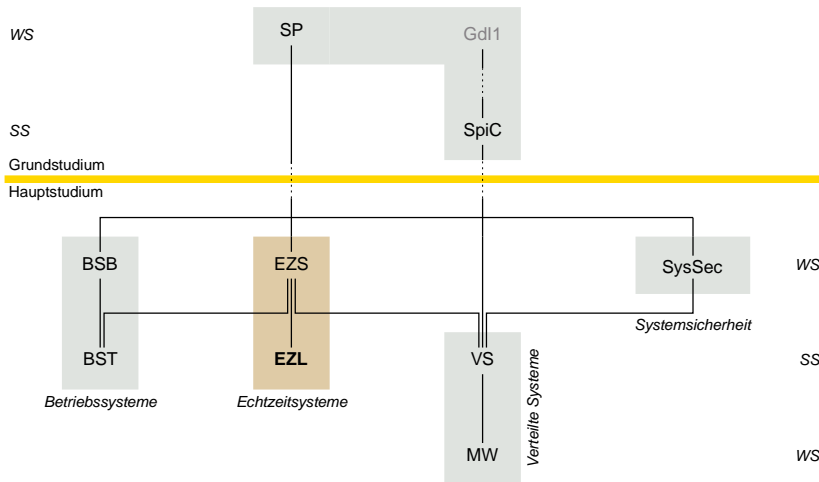
- Einordnung
- Aufbau/Lehrform
- Voraussetzungen
- Leistungsnachweis
- Literatur
- Lernziele & Lehrinhalt
- Experimente



© {scheler,ulbrich,wosch}@cs.fau.de - EZL (SS 2010)

2

Lehre@I4



© {scheler,ulbrich,wosch}@cs.fau.de - EZL (SS 2010)

3

Integrierte Lehrveranstaltung

- Termine
 - Vorlesung (2 SWS) + Übung (2 SWS) = 4 SWS (3 Stunden/Woche)
 - 5 ECTS-Punkte
- Vor-/Nacharbeit
 - N Stunden/Woche: $0 \leq N \leq (165 - X)$
 - $X < 165$: Zeitäquivalent anderer Pflichten
- Folien/Handout
 - www4.informatik.uni-erlangen.de/Lehre/SS10/V_EZL
 - **kein Skript** – Folien zu Vorlesung/Übung



© {scheler,ulbrich,wosch}@cs.fau.de - EZL (SS 2010)

4

Ablauf

■ Modus

- im 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 (0.035)
- Möglichkeit 2: Vorschlag ...
- Möglichkeit 3: Vorschlag ...



Übungsbetrieb

■ Anmeldung

- über das Waffel
- <http://waffel.informatik.uni-erlangen.de/signup/?univisid=20669384>
- Begin: ab sofort ...

■ Projektarbeit

- ≤ 3 Mitarbeiter je Gruppe
 - *Einzelkämpfer* zu sein, ist nicht zielführend
- Statusberichte, Abschlusspräsentation, Poster

■ Rechnerübung

- Manlobbi (0.058), West-Side Labs (0.055), Bello-Labs (0.033)
- freies Arbeiten, Betreuung „on demand“

→ **Kontinuität** und **aktive Mitarbeit** → Schlüssel zum Erfolg!



Voraussetzungen

- Grundlagen von **Echtzeitbetriebssystemen**: EZS
 - 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 und Abschlusspräsentation
- Poster

■ benoteter Schein

- Voraussetzung: unbenoteter Schein
- Bewertung der erbrachten Leistungen



Kontakt

- Peter Ulbrich
www4.informatik.uni-erlangen.de/~ulbrich
ulbrich@informatik.uni-erlangen.de
- Fabian Scheler
www4.informatik.uni-erlangen.de/~scheler
scheler@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 Analysis.
Wiley, third edition, 2004.
- W. Schröder-Preikschat.
Echtzeitsysteme.
www4.informatik.uni-erlangen.de/Lehre/WS09/V_EZS



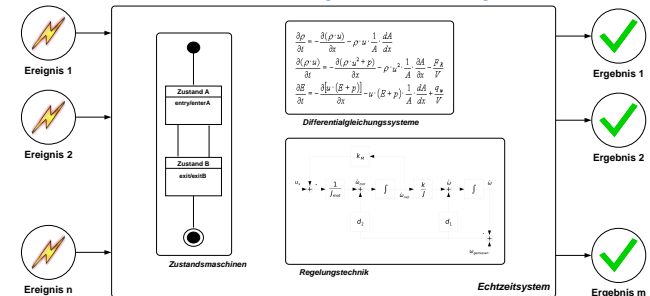
Lernziele

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



Projektphasen 1: Anwendungsanalyse

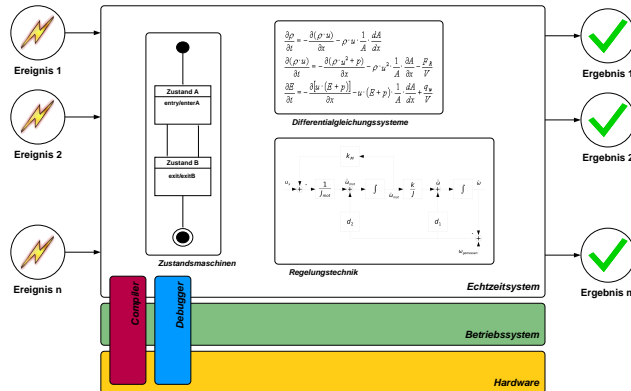
1. Analyse des physikalischen Objekts
 - Erfassen **physikalischer Zusammenhänge**
 - Wie sieht das physikalische Modell aus, wo abstrahiert man?
 - Existieren **Termine**? Gibt es **periodisch/apperiodisch Ereignisse**?



Exkurs

2. Entwicklungsumgebung

- Welcher/s Prozessor, Compiler, Betriebssystem wird verwendet?
- Wie funktioniert das Debuggen?



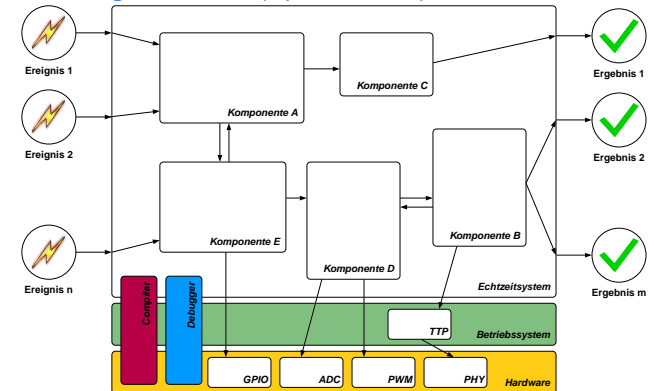
© [scheler,ulbrich,wosch]@cs.fau.de - EZL (SS 2010)

13

Projektphase 2: Komponenten

3. Komponenten und ihre Implementierung

- Welche **Komponenten** existieren in meinem System?
- Was **tun sie?** (Spezifikation)
- Wie **interagieren sie?** (Spezifikation)



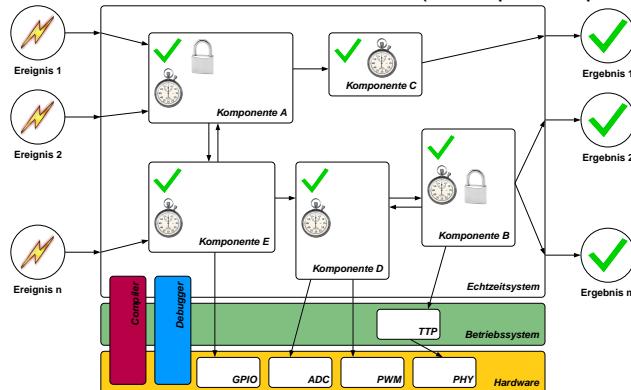
© [scheler,ulbrich,wosch]@cs.fau.de - EZL (SS 2010)

14

Projektphase 3: Testen

4. Testen der einzelnen Komponenten

- einfache Testumgebung: Testfall übersetzen, binden, ausführen
- Spezifikation der Testfälle
- Getestet wird: **Funktionalität**, **WCET** (und Speicherplatz)



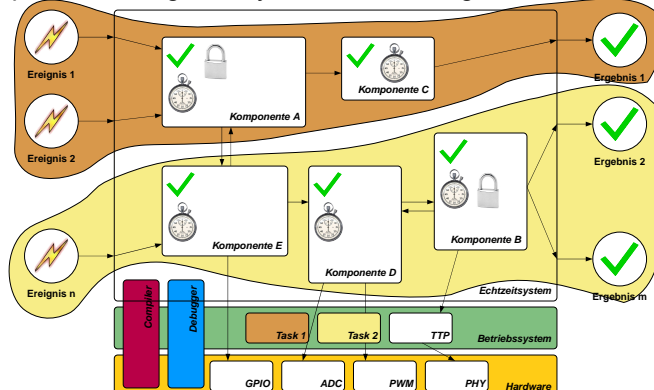
© [scheler,ulbrich,wosch]@cs.fau.de - EZL (SS 2010)

15

Projektphase 4: Komposition

5. Komposition & Integration

- Abbildung:** Komponenten → **Ereignisbehandlungen**
- Planung:** Prioritätenvergabe & statische Ablaufpläne, Zulässigkeit
- Implementierung des Systems, Abbildung auf das **Betriebssystem**



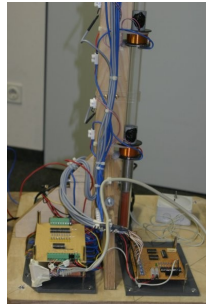
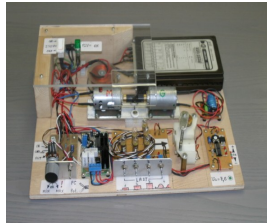
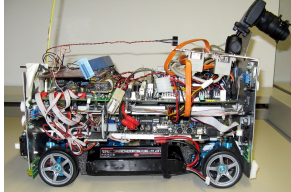
© [scheler,ulbrich,wosch]@cs.fau.de - EZL (SS 2010)

16

Projektphase 5: Akzeptanztest

6. Testen: komplettes System → Akzeptanztest

- Ausprobieren ;-)
- Debuggen



© {scheler,ulbrich,wosch}@cs.fau.de - EZL (SS 2010)

17

Lehrinhalt - Struktur

- zu jeder Phase gibt es einen **Statusbericht**
 - **ca. 10-minütige Präsentation**
 - Zu welchen **Ergebnissen** ist man gekommen?
 - Welche **Entwurfsentscheidungen** wurden getroffen? Warum?
 - Welche **Konsequenzen** haben diese Entscheidungen?
- **Abschlusspräsentation**
 - **ca. 45-minütige Präsentation**
 - setzt sich im wesentlichen aus Statusberichten zusammen
 - **Demonstration** des Experiments
 - **Überblick** über das Projekt
 - **Poster**

© {scheler,ulbrich,wosch}@cs.fau.de - EZL (SS 2010)

18

Zeitplanung

Phase	Start	Ende	Präsentation
Anwendungsanalyse	26.04.2010	11.05.2010	11.05.2010
Komponenten (Entwurf & Implementierung)	12.05.2010	08.06.2010	08.06.2010
Testen (Funktion & WCET)	01.06.2010	15.06.2010	15.06.2010
Komposition	15.06.2010	29.06.2010	29.06.2010
Akzeptanztest	29.06.2010	13.07.2010	13.07.2010
Präsentation	13.07.2010	20.07.2010	20.07.2010

▪ besondere Termine

- 25.05.2010: **Berg-Kerwa**
- 22.06.2010: **frei**
- 22.07.2010: **Semesterabschluss: Grillen** (unter Vorbehalt)

Experiment 1: Hau den Lukas

- **Aufgabe:** den Eisenkern ...
 - anheben und abbremsen
 - schrittweise anheben/fallen lassen
 - gebremst fallen lassen
 - pendeln lassen
 - „Not Aus“-Funktion



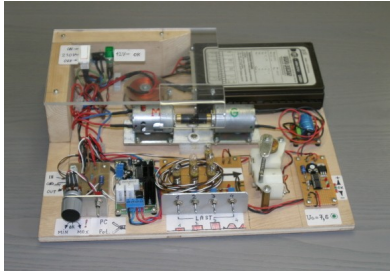
© {scheler,ulbrich,wosch}@cs.fau.de - EZL (SS 2010)

20

Experiment 2: Generator

Aufgabe

- Regelung einer Motor-Generator-Strecke
- Implementierung
 - des Regelsystems
 - einer *naiven* Regelung
- Vergleich der implementierten Regler



© {scheler,ulbrich,wosch}@cs.fau.de - EZL (SS 2010)

21

Experiment 3: NXT Standalone

Aufgabe

- stabiler Stand
- fahren: vorwärts/rückwärts
- Lenken: drehen nach rechts/links
- Fernsteuerung (Bluetooth)
- Hinderniserkennung (Sonar)

Alternativen

- Automatikgetriebe
- Sortieranlage
- NXT On A Ball



© {scheler,ulbrich,wosch}@cs.fau.de - EZL (SS 2010)

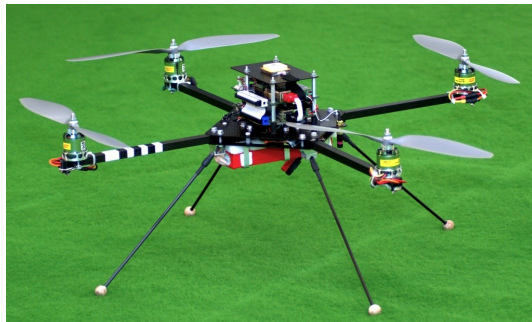
22

Experiment 4: I4Copter (Optional)

„Proaktive“ Themenfindung

Mögliche Aufgaben

- Höhenregelung (Druck- und Ultraschall-Sensoren)
- Verhaltenssteuerung (Betriebszustandswechsel)



© {scheler,ulbrich,wosch}@cs.fau.de - EZL (SS 2010)

23