Echtzeitsystemlabor

Analyse, Entwurf, Implementierung

Peter Ulbrich
Wolfgang Schröder-Preikschat

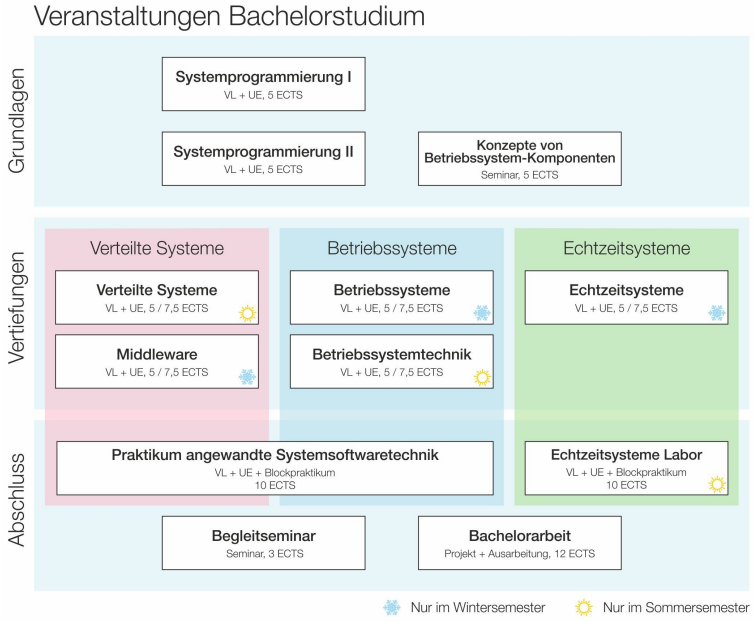
Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme
Friedrich-Alexander Universität Erlangen-Nürnberg

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

Übersicht

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

Lehre@I4



Integrierte Lehrveranstaltung

- Termine
  - Vorlesung + Übung + Praktikum
  - → 10 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/SS11/V_EZL`
  - **kein Skript** – Folien zu Vorlesung/Übung

Ablauf

- **Modus**
  - im Wechsel (Vorlesung):
    - Vorlesung = inhaltliche Einführung zu den Projektphasen
    - Übung = Statusbericht der Gruppen zu den Projekten
  - *Restliche Zeit*:
    - Bearbeitung der Projekte
- **Ort** und **Termin**???
  - → Abstimmung
  - Möglichkeit 1: Mittwoch 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 / Praktikumsnachweis**
  - Voraussetzung: unbenoteter Schein
  - Bewertung der erbrachten Leistungen

Kontakt

- Peter Ulbrich
  `www4.informatik.uni-erlangen.de/~ulbrich`
  `ulbrich@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

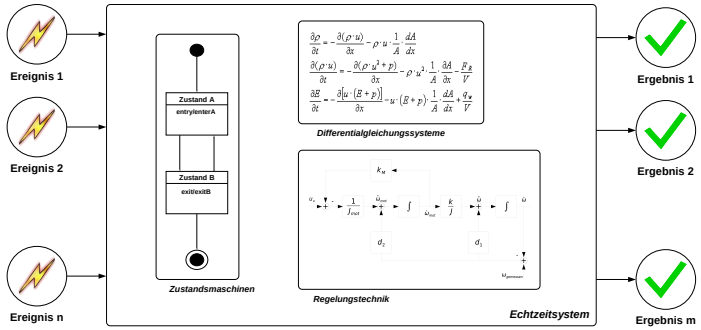
- Echtzeitprogrammierung in Teams

- Echtzeitanwendung entwickeln
  - Anforderungen/Gegebenheiten analysieren und spezifizieren
  - Echtzeitsysteme problemorientiert implementieren
  - Experimente aufbauen und durchführen

→ Vertiefung von Grundlagenwissen
  - durch experimentelles Arbeiten

Projektphasen 1: Anwendungsanalyse

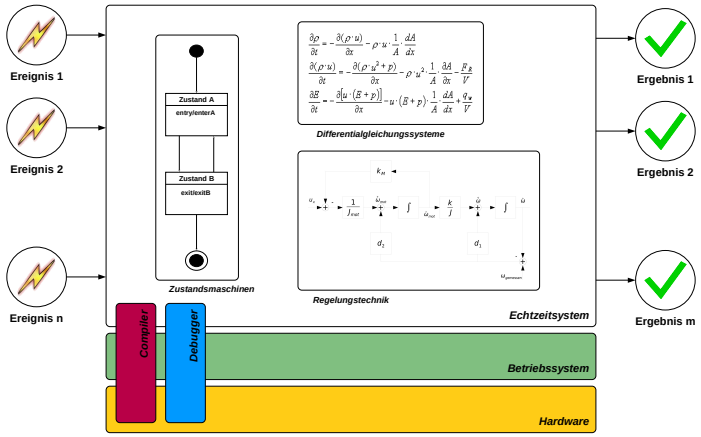
1. Analyse des physikalischen Objekts
   - Erfassen **physikalischer Zusammenhänge**
   - Wie sieht das physikalische Modell aus, wo abstrahiert man?
   - Existieren **Termine**? Gibt es **periodisch/aperiodisch Ereignisse**?



Exkurs

2. Entwicklungsumgebung

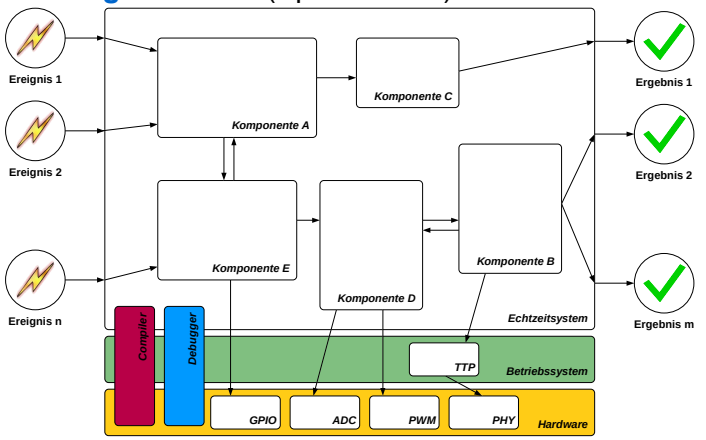
- Welcher/s Prozessor, Compiler, Betriebssystem wird verwendet?
- Wie funktioniert das Debuggen?



Projektphase 2: Komponenten

3. Komponenten und ihre Implementierung

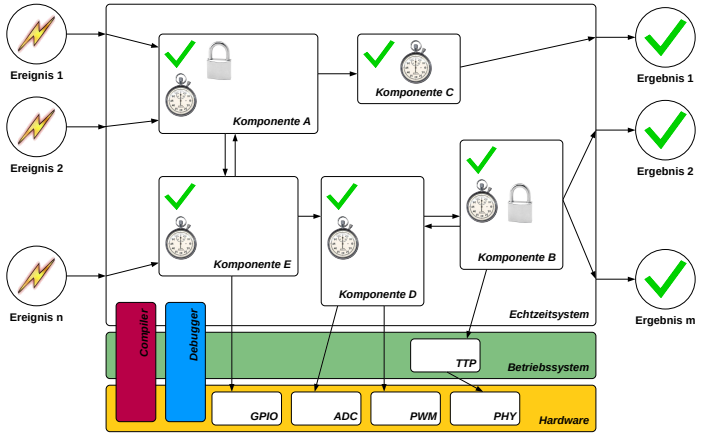
- **Welche Komponenten** existieren in meinem System?
- **Was tun sie?** (Spezifikation)
- **Wie interagieren sie?** (Spezifikation)



Projektphase 3: Testen

4. Testen der einzelnen Komponenten

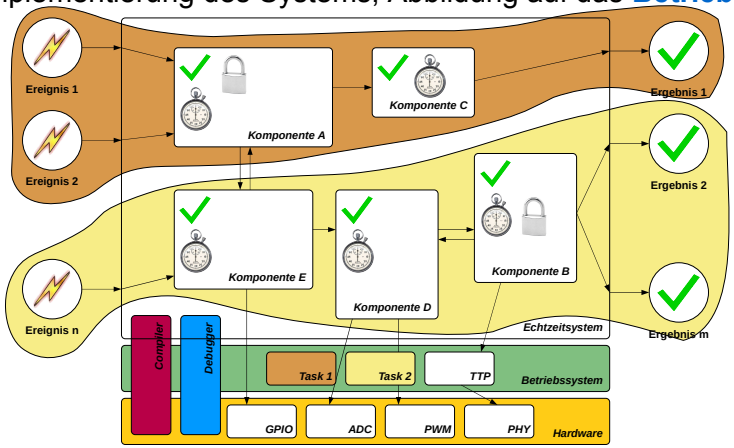
- einfache Testumgebung: Testfall übersetzen, binden, ausführen
- Spezifikation der Testfälle
- Getestet wird: **Funktionalität**, **WCET** (und Speicherplatz)



Projektphase 4: Komposition

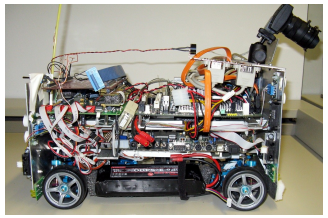
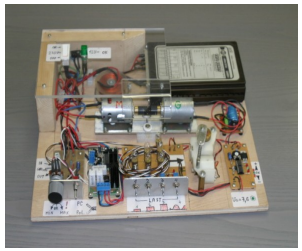
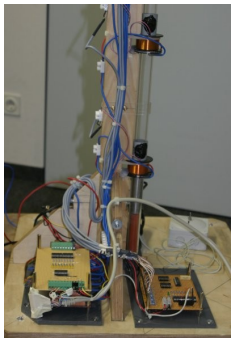
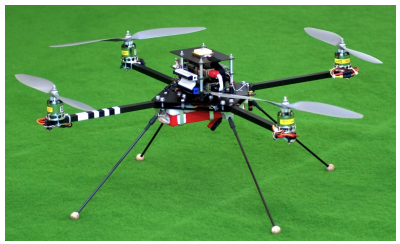
5. Komposition & Integration

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



Projektphase 5: Akzeptanztest

6. Testen: komplettes System → Akzeptanztest
   - Ausprobieren ;-)
   - Debuggen


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**

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

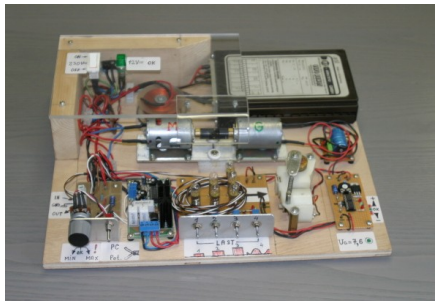
- besondere Termine
  - 25.05.2010: **Berg-Kerwa**
  - 22.06.2010: **frei**
  - 22.07.2010: **Semesterabschluss: Grillen** (unter Vorbehalt)

Experiment 1: Hau den Lukas

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

Experiment 2: Generator

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



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

Experiment 4: I4*Copter* (Optional)

- **„Proaktive“ Themenfindung**
- **Mögliche Aufgaben**
  - Höhenregelung (Druck- und Ultraschall-Sensoren)
  - Verhaltenssteuerung (Betriebszustandswechsel)