

Echtzeitsystemlabor

Analyse, Entwurf, Implementierung

Fabian Scheler
Peter Ulbrich
Wolfgang Schröder-Preikschat

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

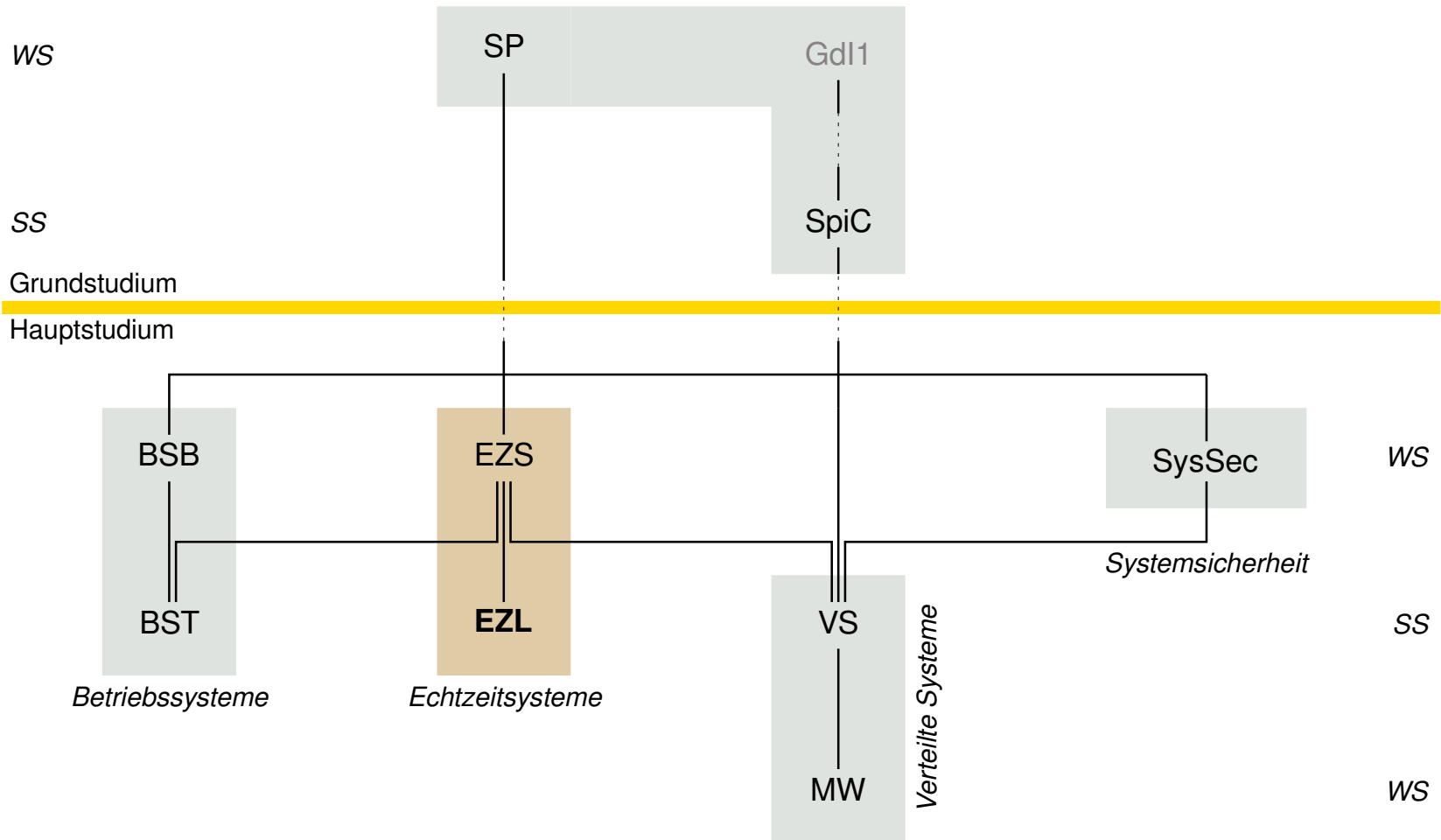
<http://www4.cs.fau.de/~{scheler,ulbrich,wosch}>
{scheler,ulbrich,wosch}@cs.fau.de



Übersicht

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





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/SS09/V_EZL
- **kein Skript** – Folien zu Vorlesung/Übung



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 12:00 – 14:00 (0.031)
- 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

■ Rechnerübung

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

→ Kontinuität und aktive Mitarbeit als 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 / Abschlusspräsentation
- Poster

■ benoteter Schein

- unbenoteter Schein
- Bewertung der erbrachten Leistungen



Kontakt

- Fabian Scheler
`www4.informatik.uni-erlangen.de/~scheler`
`scheler@informatik.uni-erlangen.de`
- 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/WS07/V_EZS



Lernziele

- Echtzeitprogrammierung in Teams

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

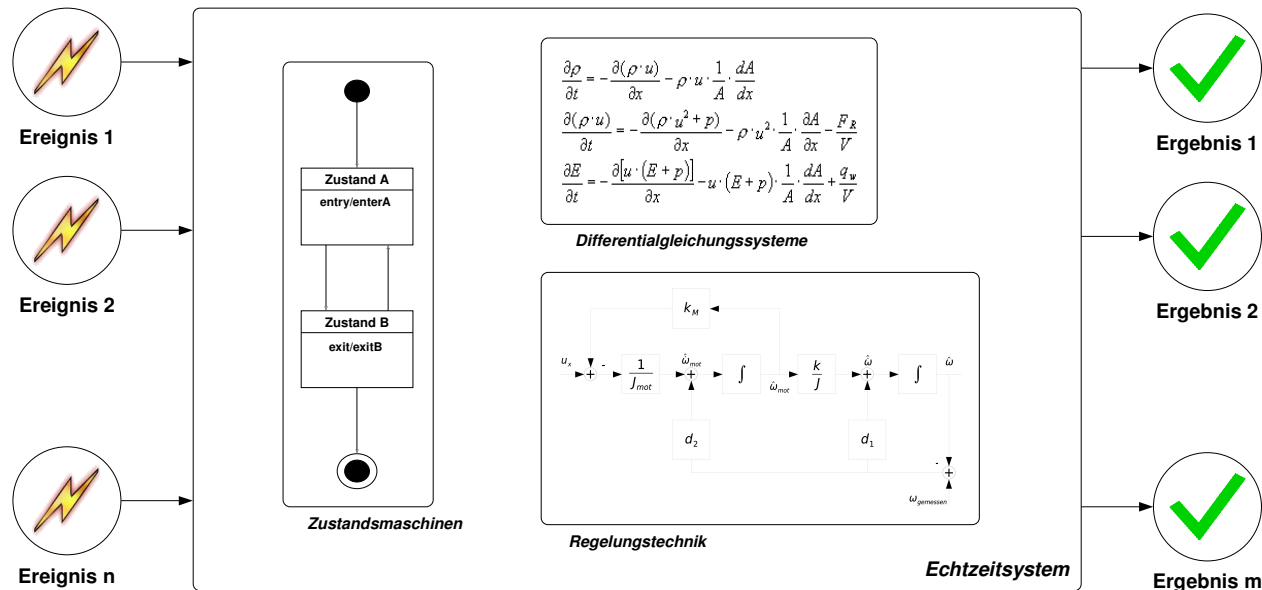
- Vertiefung von Grundlagenwissen
 - durch experimentelles Arbeiten



Projektphasen 1: Anwendungsanalyse

1. Analyse des physikalischen Objekts

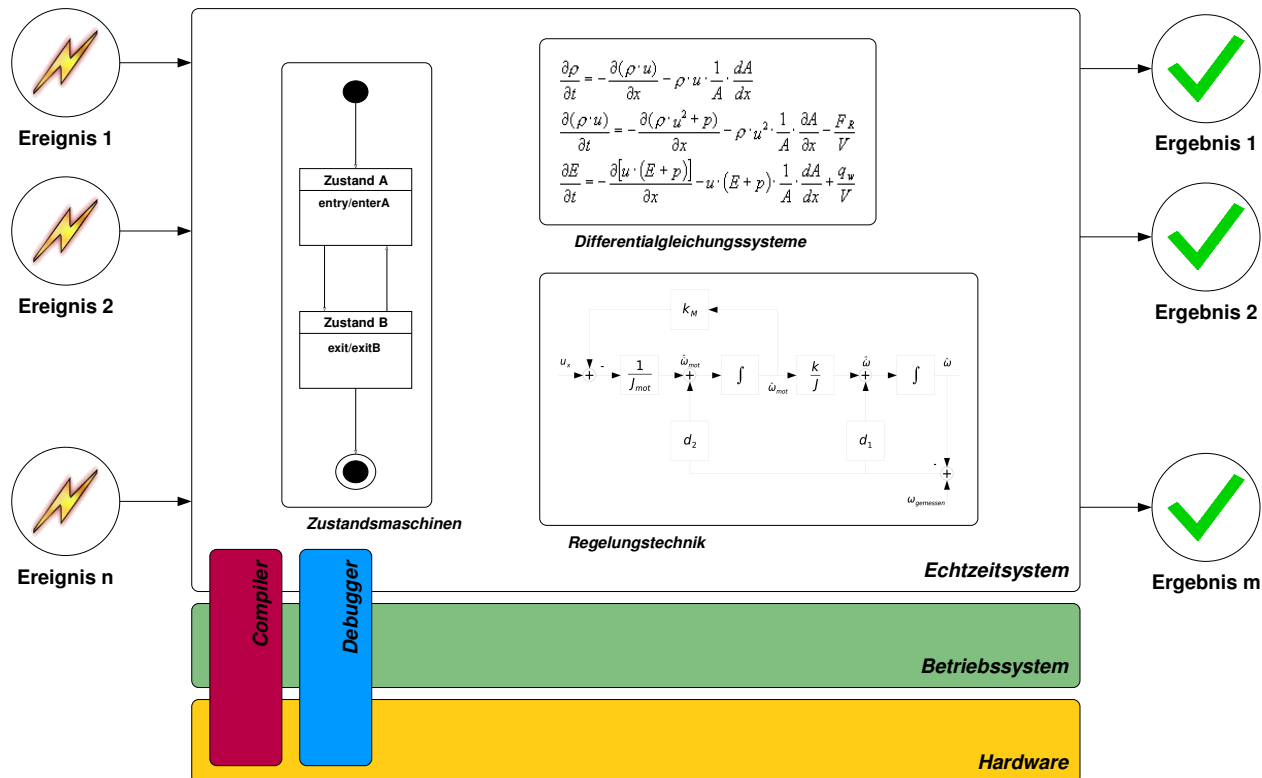
- Erfassen **physikalischer Zusammenhänge**
- Wie sieht das physikalische Modell aus, wo abstrahiert man?
- Existieren **Termine**? Gibt es **periodisch/aperiodisch Ereignisse**?



Exkurs

2. Entwicklungsumgebung

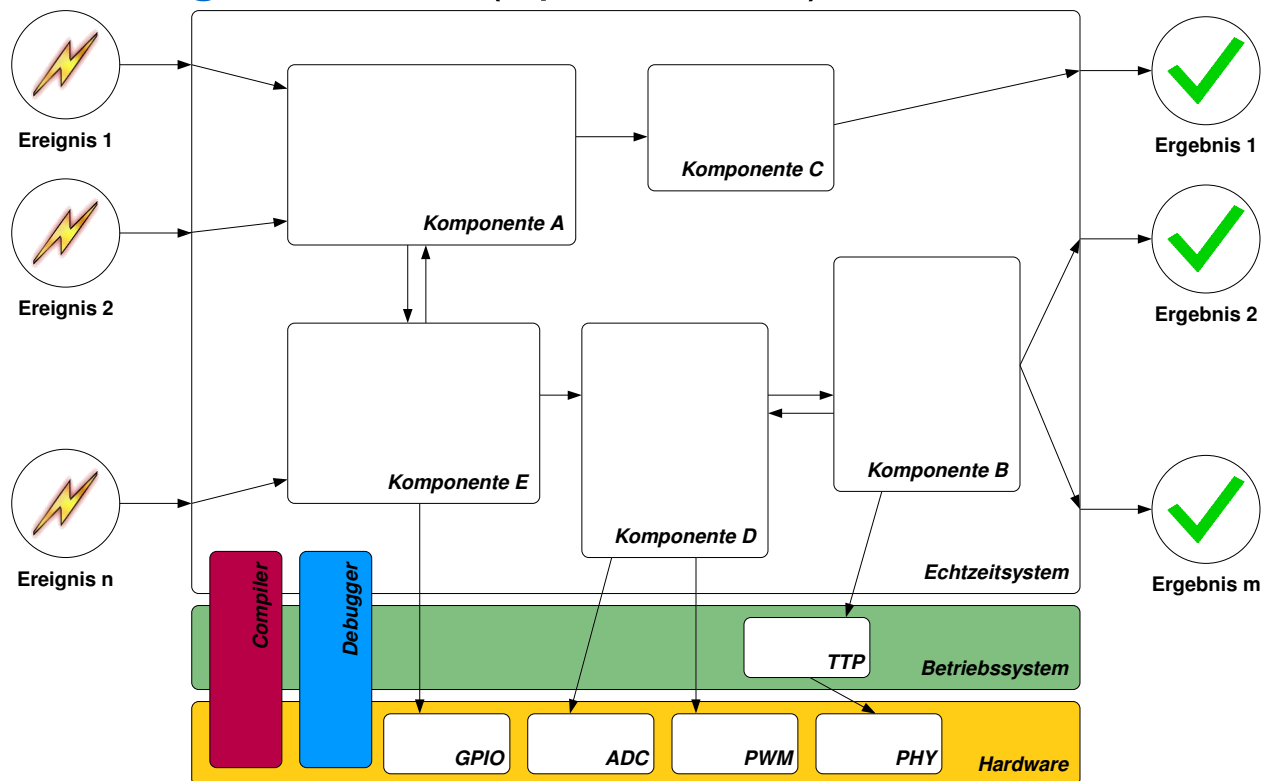
- Welcher/s Prozessor, Compiler, Betriebssystem wird verwendet?
- Wie funktioniert das Debuggen?
- Gibt es irgendwelche Besonderheiten?



Projektphase 2: Komponenten

3. Komponenten und ihre Implementierung

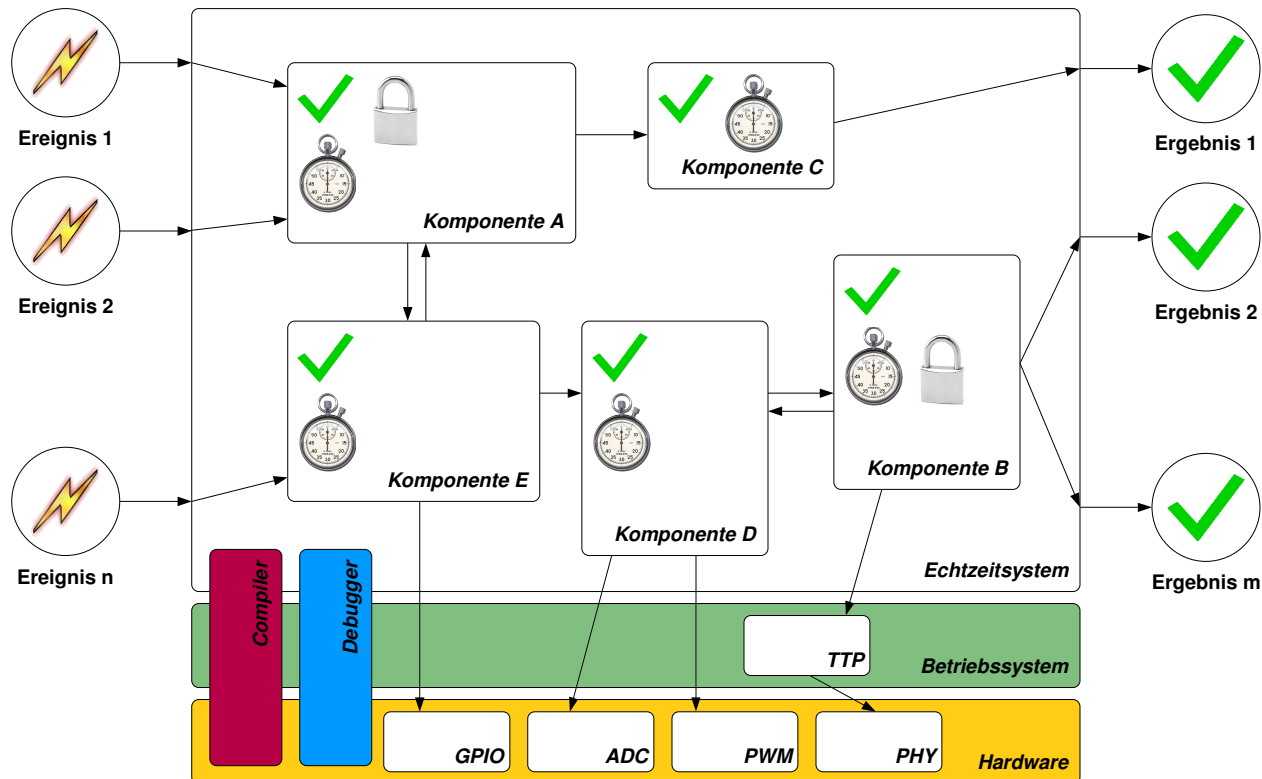
- **Welche Komponenten** existieren in meinem System?
- **Was tun sie?** (Spezifikation)
- **Wie interagieren sie?** (Spezifikation)



Projektphase 3: Testen

4. Testen der einzelnen Komponenten

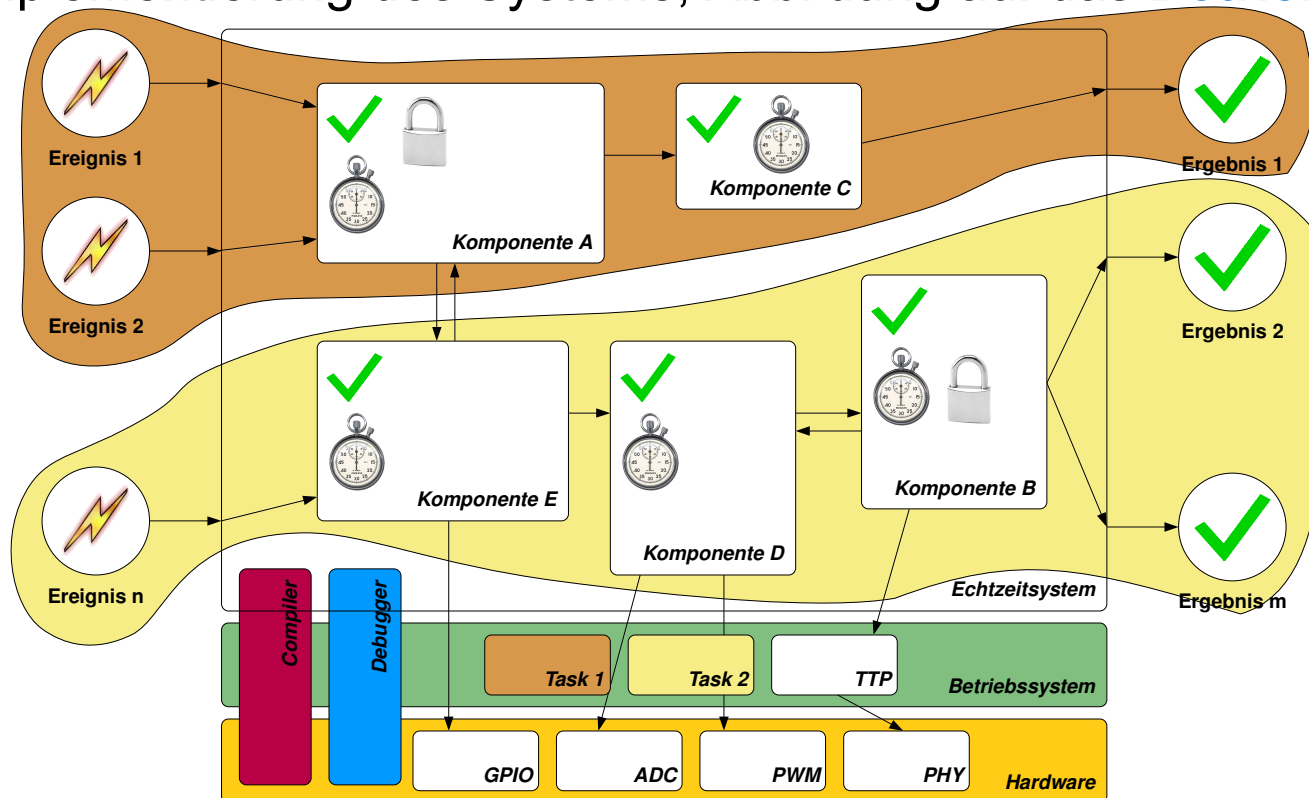
- einfache Testumgebung: Testfall übersetzen, binden, ausführen
- Spezifikation der Testfälle
- Getestet wird: **Funktionalität**, **WCET** (und Speicherplatz)



Projektphase 4: Komposition

5. Komposition & Integration

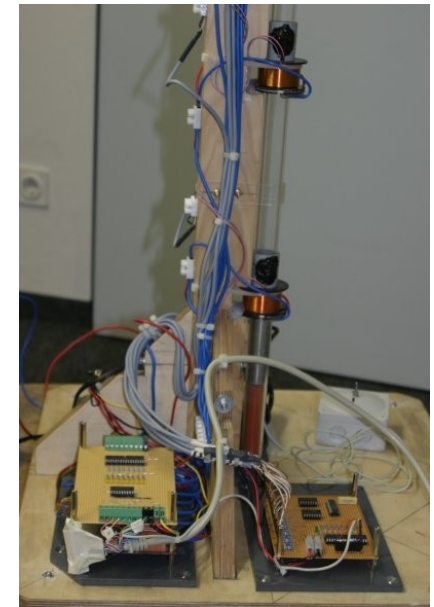
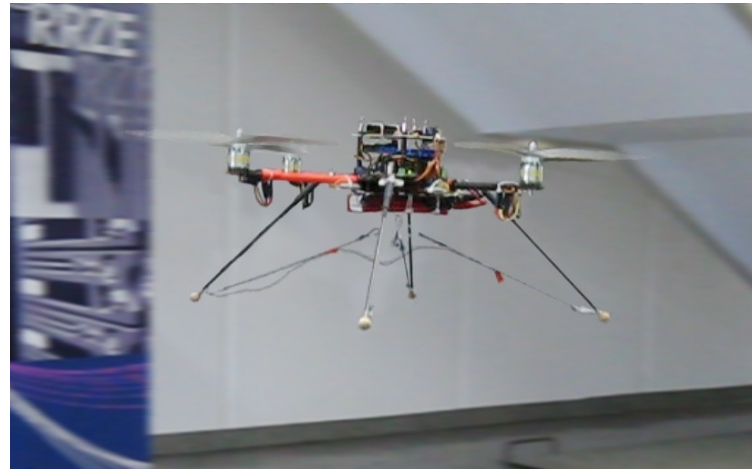
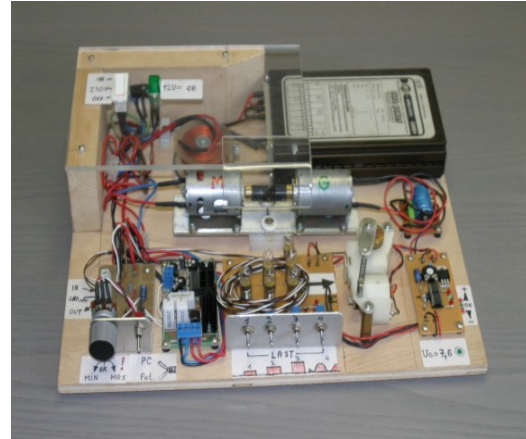
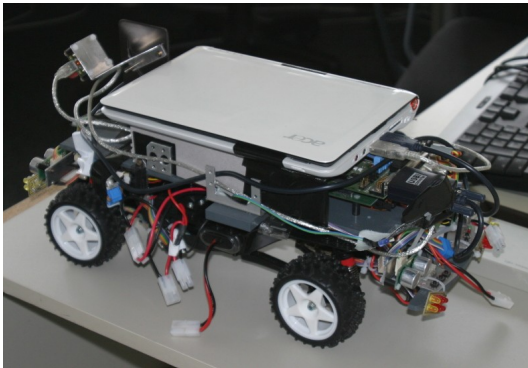
- Abbildung: Komponenten → **Ereignisbehandlungen**
- **Planung**: Prioritätenvergabe & statische Ablaufpläne, Zulässigkeit
- Implementierung des Systems, Abbildung auf das **Betriebssystem**



Projektphase 5: Akzeptanztest

6. Testen: komplettes System → Akzeptanztest

- Ausprobieren ;-)
- Debuggen

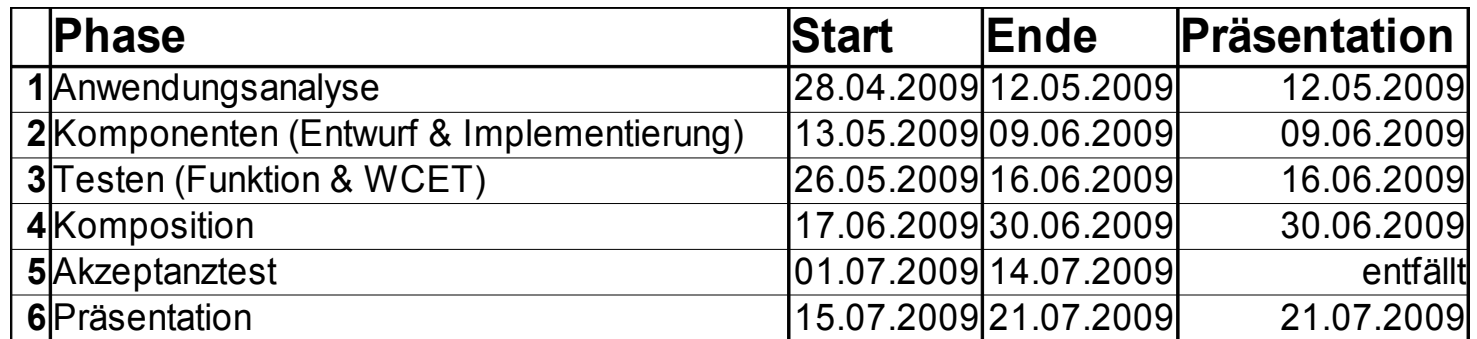


Lehrinhalt - Struktur

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

- **Abschlusspräsentation**
 - ca. **45-minütige Präsentation**
 - **Demonstration** des Experiments
 - **Überblick** über das Projekt
 - **Poster**
 - setzt sich im wesentlichen aus Zwischenberichten zusammen





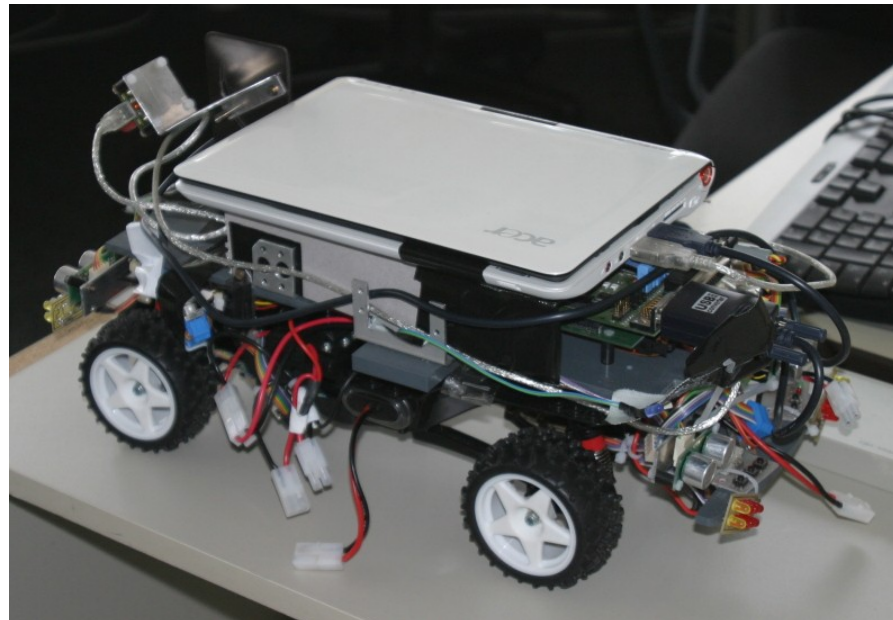
Experiment 1: I4Copter

- Aufgabenstellung
 - (Entwurf) und Implementierung einer
 - Komponenten-Überwachung und Verhaltensmodell
 - Am Beispiel der Gier- oder Höhen-Regelung
 - Ausnahmebehandlung
 - Implementierung als ereignis- / zeitgesteuertes System



Experiment 2: Carolo-Cup

- Aufgabenstellung
 - (Entwurf) und Implementierung einer
 - Geschwindigkeitsregelung
 - Orientierungssteuerung
 - Ausnahmebehandlung
 - Implementierung als zeitgesteuertes System



Experiment 3: Hau den Lukas

- Aufgabe 1: den Eisenkern ...
 - anheben und abbremsen
 - schrittweise anheben/fallen lassen
 - gebremst fallen lassen
 - pendeln lassen
 - „Not Aus“-Funktion



Experiment 4: Generator

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

