Echtzeitsysteme

Wintersemester 2010/2011

Prof. Dr. Alois Knoll, Dr. Christian Buckl

TU München

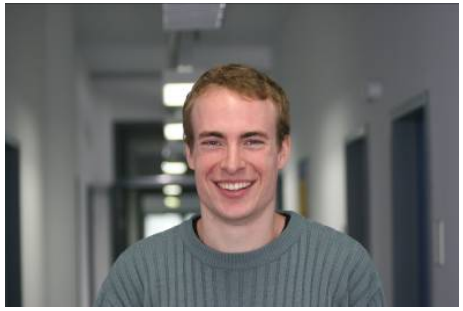
Lehrstuhl VI Robotics and Embedded Systems

Echtzeitsysteme: Organisation

Team

Prof. Dr. Alois Knoll

Dr. Christian Buckl



Übungen: Simon Barner, Dominik Sojer, Stephan Sommer

Homepage der Vorlesung mit Folien, Übungsaufgaben und weiterem Material:
http://www6.informatik.tu-muenchen.de/Main/TeachingWs2010Echtzeitsysteme

Bestandteile der Vorlesung

- Vorlesung:
  - Dienstag 10:15-11:45 Uhr MI HS 2
  - Mittwoch 12:15-13:00 Uhr MI HS 2
  - 6 ECTS Punkte
  - Wahlpflichtvorlesung im Gebiet Echtzeitsysteme (Technische Informatik)
  - Wahlpflichtvorlesung für Studenten der Elektro- und Informationstechnik
  - Pflichtvorlesung für Studenten des Maschinenbau Richtung Mechatronik
- Übung:
  - zweistündige Tutorübung, im Raum 03.05.012
  - Montags 14:00 – 15:30 Uhr
  - Dienstags 8:30 – 10:00 Uhr
  - Mittwochs 10:15 – 11:45 Uhr und 15:30 – 17:00 Uhr
  - Weitere Termine bei Bedarf nach Vereinbarung
  - Beginn: voraussichtlich ab 08.11.2010, Anmeldung ab sofort über TUMonline
- Prüfung:
  - Schriftliche Klausur am Ende des Wintersemesters, falls ein Schein benötigt wird.

Informationen zur Übung

- Ziel: Praktisches Einüben von Vorlesungsinhalten
- Übungsaufgaben werden in Gruppen zu zweit am Computer gelöst
- Platz ist begrenzt (8 Computer ⇔ 16 Studenten) ⇒ **Anmeldung erforderlich**
- Übungsaufgaben sind auch auf der Vorlesungsseite verfügbar
- Es werden diverse Aufgaben aus dem Bereich der Echtzeitprogrammierung angeboten, wie z.B. Aufgaben zu Threads, Semaphore, Kommunikation
- Programmiersprache ist überwiegend C, zu Beginn der Übung wird eine kurze Einführung in C angeboten
- Die Anmeldung erfolgt über **TUMonline** (Tutorübungen zu Echtzeitsysteme (IN2060))
- Falls Bedarf an weiteren Terminen besteht, senden Sie bitte eine Mail an sommerst@in.tum.de.
- Die Übungsinhalte sind nicht direkt prüfungsrelevant, **tragen aber stark zum Verständnis bei.**

Mögliche Übungsinhalte

- Modellierungssprachen/-werkzeuge (Esterel Studio, SCADE, Ptolemy, EasyLab, FTOS)
- Programmierung von Echtzeitsystemen (Programmiersprache C, Betriebssysteme VxWorks, PikeOS)
  - Nebenläufigkeit (Threads, Prozesse)
  - Interprozesskommunikation / Synchronisation: Semaphore, Signale, Nachrichtenwarteschlangen
  - Unterbrechungsbehandlung
- Kommunikation (Ethernet, CAN)
- Programmierung von Adhoc-Sensornetzwerken (TinyOS, ZigBee)
- Umsetzung von diversen Demonstratoren (Aufzug, Kugelfall, Murmelbahn)

- **Ihre Rückmeldung ist wichtig, denn sie bestimmt über die Inhalte!**

Klausur

- Für Studenten, die einen Schein benötigen, wird am Ende der Vorlesung eine schriftliche Klausur angeboten.
- Stoff der Klausur sind die Inhalte der Vorlesung.
- Die Inhalte der Übung sind nicht direkt prüfungsrelevant, tragen allerdings zum Verständnis des Prüfungsstoffes bei.
- Voraussichtlicher Termin: letzte Vorlesungswoche (Rückmeldung mit Prüfungsamt steht noch aus)
- Voraussichtlich erlaubte Hilfsmittel: keine

Grundsätzliches Konzept

- Themen werden aus verschiedenen Blickrichtungen beleuchtet:
  - Stand der Technik in der Industrie
  - Stand der Technik in der Wissenschaft
  - Existierende Werkzeuge
  - Wichtig: nicht die detaillierte Umsetzung, sondern die Konzepte sollen verstanden werden
- Zur Verdeutlichung theoretischer Inhalte wird versucht, Analogien zum Alltag herzustellen. Wichtig: Praktische Aufgaben in der Vorlesung und der Übung
- In jedem Kapitel werden die relevanten Literaturhinweise referenziert
- Zur Erfolgskontrolle werden Klausuraufgaben der letzten Jahre am Ende eines Kapitels diskutiert
- **Wir freuen uns jederzeit über Fragen, Verbesserungsvorschläge und konstruktive Kommentare!**

Weitere Angebote des Lehrstuhls

- Weitere Vorlesungen: Robotik, Digitale Signalverarbeitung, Maschinelles Lernen und bioinspirierte Optimierung I&II, Sensor- und kamerageführte Roboter
- Praktika: Echtzeitsysteme, Roboterfußball, Industrieroboter, Neuronale Netze und Maschinelles Lernen, Bildverarbeitung, Signalverarbeitung
- Seminare: Sensornetzwerke, Modellierungswerkzeuge, Busprotokolle, Objekterkennung und Lernen, Neurocomputing,
- Diplomarbeiten / Masterarbeiten
- Systementwicklungsprojekte / Bachelorarbeiten
- Guided Research, Stud. Hilfskräfte
- Unser gesamtes Angebot finden Sie unter http://wwwknoll.in.tum.de

Leitprojekte des Lehrstuhls/fortiss im Bereich ES

- Am Lehrstuhl werden eine große Zahl von Projekten im Bereich embedded systems bzw. cyber-physical systems durchgeführt
- Für die Hörer dieser Vorlesung bieten sich zum Einstieg zwei Projekte besonders an: die Leitprojekte "**E-Fahrzeug 2.0**" und "**InnoTruck**"
- "E-Fahrzeug 2.0" ist ein Projekt des **Transferinstituts fortiss** eGmbH (www.fortiss.org)
- Der "InnoTruck" entsteht im Projekt "Diesel Reloaded" des Rudolf-Diesel-Fellows Prof. G. Spiegelberg **des TUM Institute for Advanced Studies** (http://www.ias.tum.de/)

E-Fahrzeug 2.0

- Entwicklung eines Versuchsträgers für innovative Konzepte zur Steuerung eines rein elektrischen Fahrzeugs
- Basisdaten:
  - Vier Radnabenmotoren in “e-corners”, Lenkwinkel individuell voll steuerbar
  - Keine Bremsen, rein elektrische Verzögerung
  - Sidestick-Steuerung
  - Kommunikation basierend auf einer Echtzeit-Variante von Ethernet

Petronius Electric Car Software Architecture

- The central concept of Petronius is **data flowing** from the sensors of the vehicle to its actuators, based on events and/or time predicates → *data-centric architecture with clear data structuring*
- **Avoid replication** of data and its acquisition → centralised logical data base or "*data cloud*" for decoupling modules
- **Modular design** for easy extensibility, testability, independent yet safe development → provision for specifying *abstract data dependencies*
- Simple **network structure** → logical or *virtual networking*
- **Scalability and portability** across vehicle classes → *flexible mapping* from logical architecture to target hardware through suitable tools
- **Fault tolerance** is mandatory → *fault recognition and redundancy management*

Experimental Platform

Key Facts (per wheel):

- *Drive motor: 2…8 kW*
- *RPM: 520 → 48 km/h*
- *Maximum torque: 160 Nm*
- *No Brakes*
- *X-by-Wire*
- *Sidestick control*

Studentenarbeiten im Rahmen des eCar-Projektes (1)

- **Verbesserung eines internen Kommunikationsnetzes (HIWI)**
  Im Rahmen dieser Arbeit soll ein auf Echtzeit-Ethernet basierendes Kommunikationsnetz weiterentwickelt werden.
- **Integration eines Winkelsensors zur Messung des aktuellen Radwinkels (SEP, Bachelor, HIWI)**
  Es soll mit Hilfe eines Winkelsensors die absolute Radposition ermittelt werden.
- **Erweiterung der Mensch-Maschine Schnittstelle (HIWI)**
  Ziel ist es die MMI zu erweitern und besser auf die Bedürfnisse des Fahrers abzustimmen.
- **Implementierung einer Fahrdynamikregelung (SEP, Bachelor, HIWI)**
  Die Aufgabe besteht in der Portierung einer Matlab/Simulink Regelung auf das eCar.

Studentenarbeiten im Rahmen des eCar-Projektes (2)

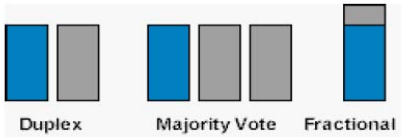
- **Entwicklung eines modellgetriebenen Tools für die Funktionsentwicklung des eCar (SEP, Bachelor, Master, Diplom, HIWI)**
  Um die Funktionsentwicklung für das eCar zu erleichtern, soll ein modellgetriebenes Tool entwickelt werden, das das Hinzufügen von weiteren Funktionen unterstützt.
- **Ermittlung des aktuellen Verbrauchs (SEP, Bachelor, HIWI)**
  Die Aufgabe besteht in bedarfsgerechter Ermittlung der Energiemenge. Diese Information soll in einem ersten Schritt dem Fahrer zur Verfügung gestellt werden.
- **Diverse weitere Themen**
  Außerdem gibt es noch unzählige weitere Themen die im Rahmen des eCar-Projektes bearbeitet werden können.

Kontakt:

ecar@fortiss.org

Nähere Informationen:

http://www.fortiss.org/en/research/cyber-physical-systems/projects/ecar.html

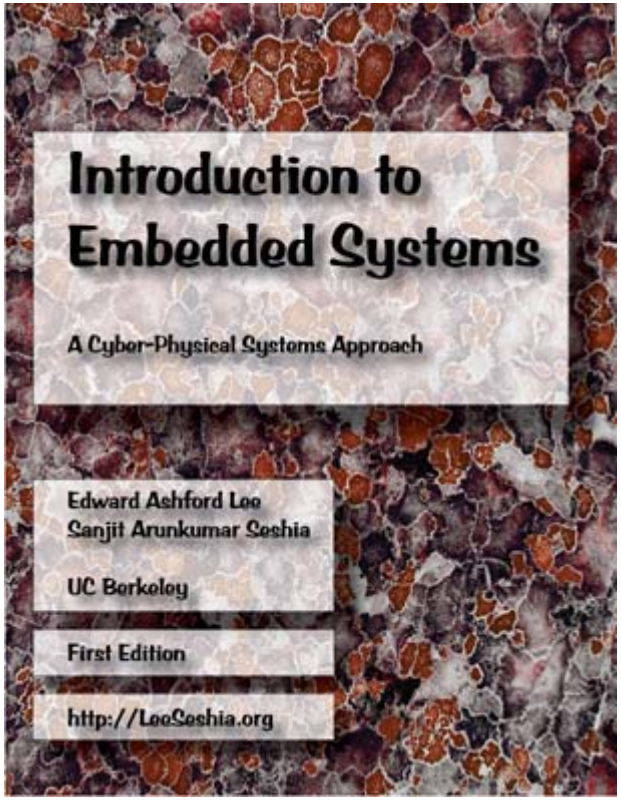


PUMA: HiWi Wanted

- The PUMA doctoral school is looking for long-term (> 4 months) student assistants to work with us on the implementation (extension) of **Gecko**, a novel framework combining game solving and fault-tolerant embedded systems.
- Requirements (applicants should fulfill one of them):
  - 1. Knowledge in verification or automata theory
    - You can work on extensions in model translation or engines
  - 2. Knowledge in embedded/real-time systems
    - You can work on extensions which establish links between concrete examples (production line system based on Festo MPS) and our tool
- For further information (job packages), please contact
- Chih-Hong Cheng (chengch@in.tum.de).

PUMA Graduiertenkolleg

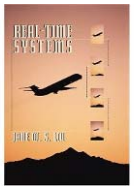
Literatur



- Lee, Seshia: Introduction to Embedded Systems
  - Buch deckt die meisten Kapitel der Vorlesung (bis auf Kommunikation) ab
  - Kostenlos online verfügbar unter http://leeseshia.org/
  - Vorlesung wird in Zukunft an dieses Buch angepasst (Buch ist gerade erst erschienen)

Weitere Literatur

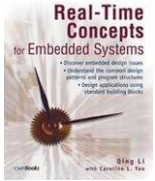
Hermann Kopetz: Real-Time Systems (Überblick)



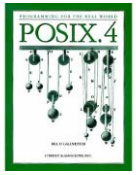
Jane W. S. Liu: Real-Time Systems
(Überblick, Schwerpunkt Scheduling)

Stuart Bennet: Real-Time Computer Control:
An Introduction (Überblick, Hardware)

Alan Burns, Andy Wellings: Real-Time Systems and Programming
Languages (Schwerpunkt: Programmiersprachen)



Qing Li, Caroline Yao: Real-Time Concepts for
Embedded Systems (Schwerpunkt: Programmierung)



Bill O. Gallmeister: Programming for the Real-World: POSIX.4
(Schwerpunkt: Posix)

Weitere Literaturangaben befinden sich in den jeweiligen Abschnitten.

Vorlesungsinhalte

1. Einführung Echtzeitsysteme
2. Modellierung und Werkzeuge
3. Nebenläufigkeit
4. Scheduling
5. Echtzeitbetriebssysteme
6. Programmiersprachen
7. Uhren
8. Kommunikation
9. Fehlertolerante Systeme
10. Spezielle Hardware
11. Regelungstechnik

Weitere Themen können bei Interesse aufgenommen werden. Melden Sie sich einfach nach der Vorlesung oder per Email.

Inhalt I

- Kapitel Einführung (ca. 1 Vorlesungswoche)
  - Definition Echtzeitsysteme
  - Klassifikation
  - Echtzeitsysteme im täglichen Einsatz
  - Beispielanwendungen am Lehrstuhl
- Kapitel Modellierung/Werkzeuge (ca. 3 Vorlesungswochen)
  - Allgemeine Einführung
  - Grundsätzlicher Aufbau, Models of Computation, Ptolemy
  - Synchrone Sprachen (Esterel, Lustre), SCADE, EasyLab
  - Zeitgesteuerte Systeme: Giotto, FTOS, TTA
  - Exkurs: Formale Methoden

Inhalt II

- Kapitel Nebenläufigkeit (2 Vorlesungswochen)
  - Prozesse, Threads
  - Interprozesskommunikation
- Kapitel Scheduling (2 Vorlesungswochen)
  - Kriterien
  - Planung Einrechner-System, Mehrrechnersysteme
  - EDF, Least Slack Time
  - Scheduling mit Prioritäten (FIFO, Round Robin)
  - Scheduling periodischer Prozesse
  - Scheduling Probleme

Inhalt III

- Kapitel Echtzeitbetriebssysteme (1 Vorlesungswoche)
  - QNX, VxWorks, PikeOS
  - RTLinux, RTAI, Linux Kernel 2.6
  - TinyOS, eCos
  - OSEK
- Kapitel Programmiersprachen (1 Vorlesungswoche)
  - Ada
  - Erlang
  - C mit POSIX.4
  - Real-time Java
- Kapitel Uhren (1 Vorlesungswoche)
  - Uhren
  - Synchronisation von verteilten Uhren

Inhalt IV

- Kapitel Echtzeitfähige Kommunikation (1 Vorlesungswoche)
  - Token-Ring
  - CAN-Bus
  - TTP, FlexRay
  - Real-Time Ethernet
- Kapitel Fehlertoleranz (ca. 2 Vorlesungswochen)
  - Bekannte Softwarefehler
  - Definitionen
  - Fehlerarten
  - Fehlerhypothesen
  - Fehlervermeidung
  - Fehlertoleranzmechanismen

Potentielle zusätzliche Inhalte

- Kapitel: Spezielle Hardware (1 Vorlesungswoche)
  - Digital-Analog-Converter (DAC)
  - Analog-Digital-Converter (ADC)
  - Speicherprogrammierbare Steuerung (SPS)
- Kapitel: Regelungstechnik (ca. 2 Vorlesungswochen)
  - Definitionen
  - P-Regler
  - PI-Regler
  - PID-Regler
  - Fuzzy-Logic

Kapitel 1

Einführung Echtzeitsysteme

Inhalt

- Definition Echtzeitsysteme
- Klassifikation von Echtzeitsystemen
- Echtzeitsysteme im täglichen Leben
- Beispielanwendungen am Lehrstuhl

Definition Echtzeitsystem

Ein Echtzeit-Computersystem ist ein Computersystem, in dem die Korrektheit des Systems nicht nur vom logischen Ergebnis der Berechnung abhängt, sondern auch vom physikalischen Moment, in dem das Ergebnis produziert wird.

Ein Echtzeit-Computer-System ist immer nur ein Teil eines größeren Systems, dieses größere System wird Echtzeit-System genannt.

Hermann Kopetz

TU Wien

Definition Eingebettetes System

Technisches System, das durch ein integriertes, von Software gesteuertes Rechensystem gesteuert wird. Das Rechensystem selbst ist meist nicht sichtbar und kann in der Regel nicht frei programmiert werden. Um die Steuerung zu ermöglichen ist zumeist eine Vielzahl von sehr speziellen Schnittstellen notwendig.

In der Regel werden leistungsärmere Mikroprozessoren mit starken Einschränkung in Bezug auf die Rechenleistung und Speicherfähigkeit eingesetzt.

Resultierende Eigenschaften

⇒ zeitliche Anforderungen

- Zeitliche Genauigkeit (nicht zu früh, nicht zu spät)
- Garantierte Antwortzeiten
- Synchronisation von Ereignissen / Daten
- **Aber nicht:** Allgemeine Geschwindigkeit

⇒ Eigenschaften aufgrund der Einbettung

- Echtzeitsysteme sind typischerweise sehr Eingabe/Ausgabe (E/A)-lastig
- Echtzeitsysteme müssen fehlertolerant sein, da sie die Umgebung beeinflussen
- Echtzeitsysteme sind häufig verteilt

Zeitlicher Determinismus vs. Leistung

- Konsequenz der Forderung nach deterministischer Ausführungszeit: Mechanismen, die die allgemeine Performance steigern, aber einen negativen, nicht exakt vorhersehbaren Effekt auf einzelne Prozesse haben können, werden in der Regel nicht verwendet:
  - Virtual Memory
  - Garbage Collection
  - Asynchrone IO-Zugriffe
  - rekursive Funktionsaufrufe

Klassifikation von Echtzeitsystemen

- Echtzeitsysteme können in verschiedene Klassen unterteilt werden:
  - Nach den Konsequenzen bei der Überschreitung von Fristen: harte vs. weiche Echtzeitsysteme
  - Nach dem Ausführungsmodell: zeitgesteuert (zyklisch, periodisch) vs. ereignisbasiert (aperiodisch)

Harte bzw. weiche Echtzeitsysteme

- **Weiche Echtzeitsysteme:**
  Die Berechnungen haben eine zeitliche Ausführungsfrist, eine Überschreitung dieser Fristen hat jedoch keine katastrophale Folgen. Eventuell können die Ergebnisse noch verwendet werden, insgesamt kommt es durch die Fristverletzung evtl. zu einer Dienstverschlechterung.
  Beispiel für ein weiches Echtzeitsystem: Video
  Konsequenz von Fristverletzungen: einzelne Videoframes gehen verloren, das Video hängt

- **Harte Echtzeitsysteme:**
  Eine Verletzung der Berechnungsfristen kann sofort zu fatalen Folgen (hohe Sachschäden oder sogar Gefährdung von Menschenleben) führen. Die Einhaltung der Fristen ist absolut notwendig.
  Beispiel für ein hartes Echtzeitsystem: Raketensteuerung
  Konsequenz von Fristverletzung: Absturz bzw. Selbstzerstörung der Rakete

Unterteilung nach Ausführungsmodell

- Zeitgesteuerte Applikationen:
  - Der gesamte zeitliche Systemablauf wird zur Übersetzungszeit festgelegt
  - Notwendigkeit einer präzisen, globalen Uhr $\Rightarrow$ Uhrensynchronisation notwendig
  - Für die einzelnen Berechnungen ist jeweils ein Zeitslot reserviert $\Rightarrow$ Abschätzung der maximalen Laufzeiten (**worst case execution times - WCET**) notwendig
  - **Vorteil**: Statisches Scheduling möglich und damit ein vorhersagbares (**deterministisches**) Verhalten
- Ereignisgesteuerte Applikationen:
  - Alle Ausführungen werden durch das Eintreten von Ereignissen angestoßen
  - Wichtig sind bei ereignisgesteuerten Anwendungen garantierte Antwortzeiten
  - Das Scheduling erfolgt dynamisch, da zur Übersetzungszeit keine Aussage über den zeitlichen Ablauf getroffen werden kann.

Einführung Echtzeitsysteme

Echtzeitsysteme im Alltag