Echtzeitsysteme
Lehrstuhl Informatik VI - Robotics and Embedded Systems

Echtzeitsysteme
Wintersemester 2012/2013

Prof. Dr. Alois Knoll

TU München

Lehrstuhl VI Robotics and Embedded Systems

Echtzeitsysteme: Organisation

Team

Prof. Dr. Alois Knoll

Philipp Heise

Steffen Wittmeier

Michael Jäntsch

Homepage der Vorlesung mit Folien, Übungsaufgaben und weiterem Material:
http://www6.informatik.tu-muenchen.de/Main/TeachingWs2012Echtzeitsysteme

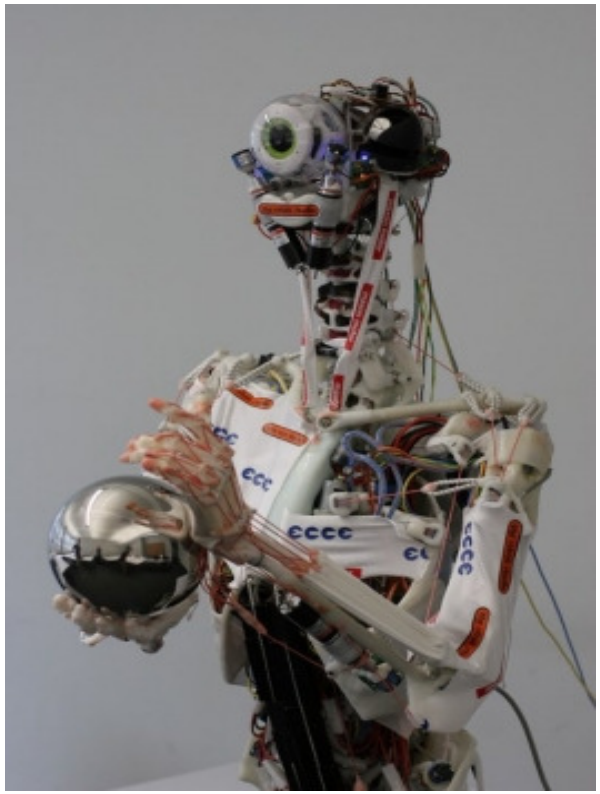
Bestandteile der Vorlesung

- Vorlesung:
  - Dienstag 10:15-11:45 Uhr MI HS 2
  - Donnerstag 12:00-12:45 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:
  - Hausaufgaben (6 x)
  - Praktikumsaufgaben (2 x)
  - Besprechung und Musterlösung der Hausaufgaben am Donnerstags-Termin (Beginn: 8.11.2012)
  - Weitere Termine bei Bedarf nach Vereinbarung
  - Beginn: erstes Blatt voraussichtlich am 08.11.2011
- Prüfung:
  - Schriftliche Klausur am Ende des Wintersemesters, falls ein Schein benötigt wird.

Informationen zur Übung

- Ziel: Praktisches Einüben von Vorlesungsinhalten
- Übungsaufgaben werden auf der Vorlesungsseite veröffentlicht
- Es werden diverse Aufgaben aus dem Bereich der Echtzeitprogrammierung angeboten, wie z.B. Aufgaben zu Threads, Semaphore, Kommunikation
- Programmiersprache ist überwiegend C/C++, zu Beginn der Übung wird eine kurze Einführung angeboten
- Falls Bedarf an weiteren Terminen besteht, senden Sie bitte eine Mail an Michael Jäntsch (michael.jaentsch@in.tum.de), Steffen Wittmeier (steffen.wittmeier@in.tum.de) oder Philipp Heise (heise@in.tum.de).
- Die Übungsinhalte sind nicht direkt prüfungsrelevant, **tragen aber stark zum Verständnis bei.**

Informationen zu Praktikumsaufgaben



- 2 Realzeitaufgaben anhand des EU-Projektes Eccerobot (www.eccerobot.eu)
- Bearbeitung der Aufgaben in 5er-Teams
- Ausarbeitung (Programmierung) muss in Heimarbeit erfolgen
- Testen/Debuggen der Lösungen im Rahmen von Praktikumsterminen (à 90 min)
- Terminvergabe durch ausgehängte Listen am 30.11.2012
- Ort: Bekanntgabe über Vorlesungshomepage

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

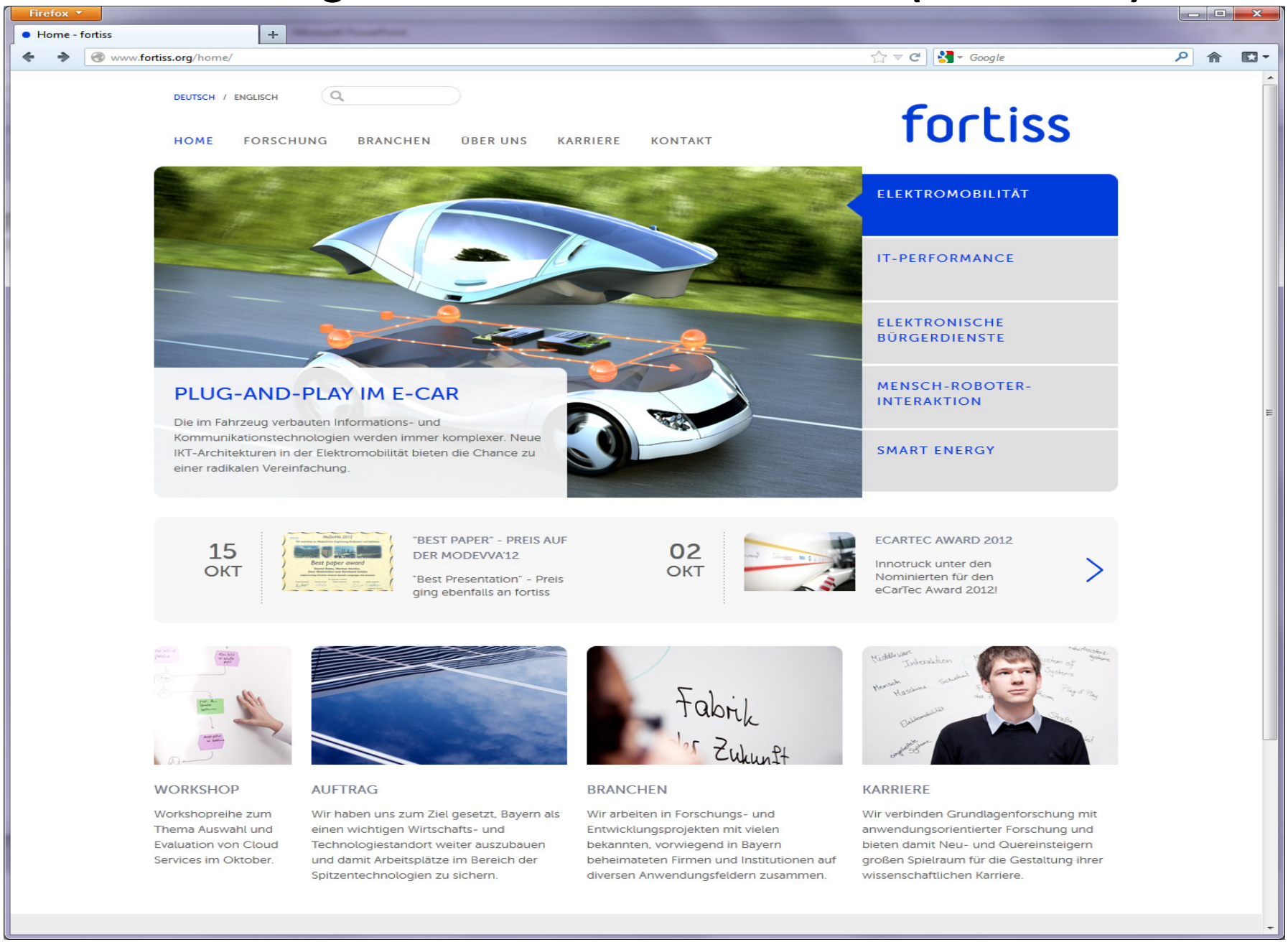
Grundsätzliches Konzept

- 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.
- In jedem Kapitel werden die relevanten Literaturhinweise referenziert
- Zur Erfolgskontrolle werden Klausuraufgaben im Rahmen der Übung diskutiert
- **Wir freuen uns jederzeit über Fragen, Verbesserungsvorschläge und konstruktive Kommentare!**

Weitere Angebote des Lehrstuhls

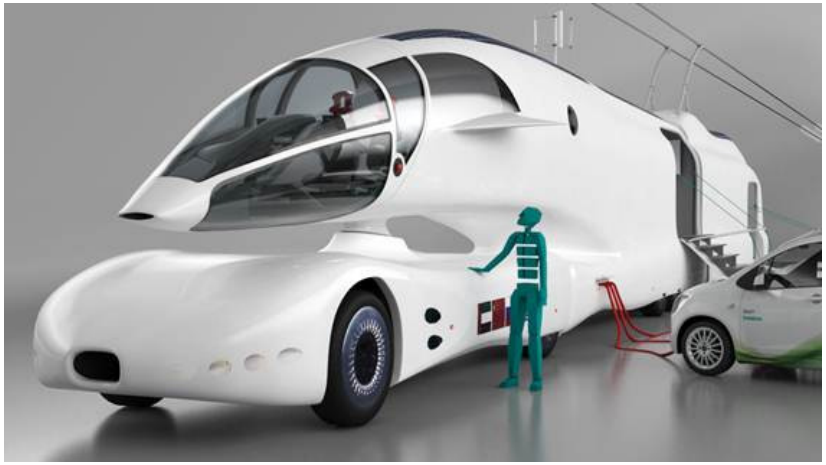
- Weitere Vorlesungen: Robotik, Digitale Signalverarbeitung, Maschinelles Lernen, Industrial Embedded Systems, Virtuelle Physik, Grundlagen der Künstlichen Intelligenz ...
- Praktika: Echtzeitsysteme, Roboterfußball, Industrieroboter, Neuronale Netze und Maschinelles Lernen, Computer Vision für Robotik, Signalverarbeitung ...
- Seminare: Robotik und kognitive Systeme, Human-Robot Interaction, Computer Vision for Robotics, Vision, Control and Human Machine Interaction in Robotic Surgery uvm. …
- Diplomarbeiten / Masterarbeiten
- Systementwicklungsprojekte / Bachelorarbeiten
- Guided Research, Stud. Hilfskräfte
- Unser gesamtes Angebot finden Sie unter **http://wwwknoll.in.tum.de**

Forschungs- und Transferinstitut fortiss (An-Institut)



InnoTruck – Diesel Reloaded

- Technologieträger für Elektromobilität z.B. in den Bereichen Antriebsstrang und Energiemanagement, Systemarchitektur und Mensch-Maschine-Interface
- Verbindung von Elektrofahrtzeugen mit „smart Homes“ und dem „smart Grid“ zur effizienten Energienutzung



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

Electric Car Software Architecture

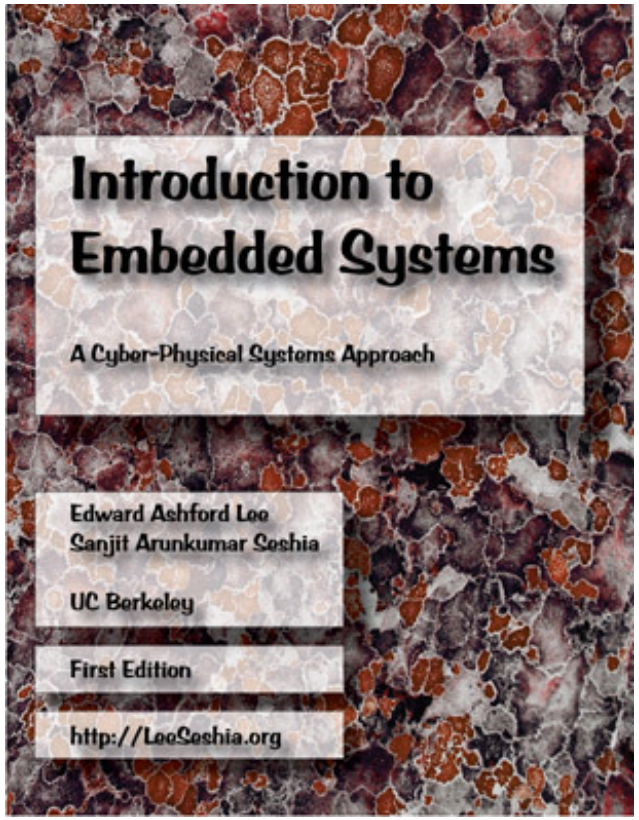
- The central concept of the architecture is **data flowing** from the sensors of the vehicle to its actuators, based on events and/or time predicates → *data-centric architecture with clear data structuring*
- **Avoid replication** of data and its acquisition → centralised logical data base or "*data 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*
- See also: **www.fortiss.org/ikt2030**

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*

Literatur

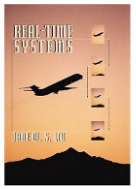


- Lee, Seshia: Introduction to Embedded Systems
  - Buch deckt einige Kapitel der Vorlesung ab
  - Kostenlos online verfügbar unter http://leeseshia.org/
  - Vorlesung wird in Zukunft an dieses Buch angepasst

Weitere Literatur

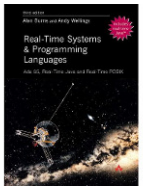
Weitere Literaturangaben befinden sich in den jeweiligen Abschnitten.

Hermann Kopetz: Real-Time Systems (Überblick)

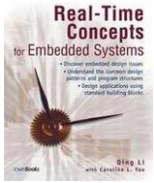


Jane W. S. Liu: Real-Time Systems
(Überblick, Schwerpunkt Scheduling)

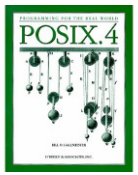
Stuart Bennet: Real-Time Computer Control:
An Introduction (Überblick, Hardware)



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)

Vorlesungsinhalte

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

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

Inhalt I

- Kapitel Einführung (ca. 1 Vorlesungswochen)
  - Definition Echtzeitsysteme
  - Klassifikation
  - Modellierung, Design und Analyse von Echtzeitsystemen
  - Echtzeitsysteme im täglichen Einsatz
- Zeit und Uhren (ca. 1 Vorlesungswoche)
  - Zeit und Echtzeit
  - Zeitmessung
  - Synchronisierung

Inhalt II

- Kapitel Hardware (ca. 2 Vorlesungswochen)
  - Prozessoren (e.g. x86, ARM, AVR)
  - Speicher-Architekturen
  - Ein- und Ausgabe
- Kapitel Kommunikation (ca. 2 Vorlesungswochen)
  - I2C, SPI
  - CAN-Bus
  - FlexRay
  - Real-time Ethernet

Inhalt III

- Kapitel Echtzeitbetriebssysteme (ca. 1 Vorlesungswoche)
  - QNX, VxWorks
  - Echtzeit-Linux
- Nebenläufigkeit/Prozesse (ca. 2 Vorlesungswochen)
  - Prozesse/Threads
  - Interprozesskommunikation
- Scheduling (ca. 2 Vorlesungswochen)
  - EDF, Least Slack Time
  - Scheduling mit Prioritäten (FIFO, Round Robin)
  - Scheduling periodischer Prozesse
  - Scheduling Probleme

Inhalt IV

- Kapitel Programmiersprachen (ca. 1 Vorlesungswoche)
  - Ada/Erlang
  - C/C++ mit POSIX.4
  - Real-time Java
- Kapitel Fehlertoleranz (ca. 1 Vorlesungswochen)
  - Bekannte Softwarefehler
  - Definitionen
  - Fehlerarten
  - Fehlerhypothesen
  - Fehlervermeidung
  - Fehlertoleranzmechanismen

Inhalt V

- Modellierung und Werkzeuge (ca. 1 Vorlesungswoche)
  - Allgemeine Einführung
  - Ptolemy
  - Simulink
- Regelungstechnik (ca. 1 Vorlesungswoche, optional)
  - Definitionen
  - PID-Regler

Kapitel 1

Einführung Echtzeitsysteme

Inhalt

- Definition Echtzeitsysteme
- Klassifikation von Echtzeitsystemen
- Modellierung, Design und Analyse von Echtzeitsystemen
- Modelle zur Berechnung
- Echtzeitsysteme im täglichen Leben

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.

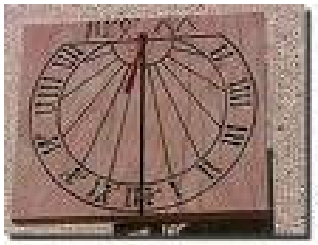
Hermann Kopetz, TU Wien

Was ist besonders an Echtzeitsystemen?

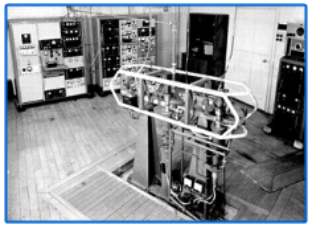
- Integration von Berechnungen und physikalischen Prozessen
- Erfassung und Veränderung von physischen Prozessen über die Zeit
- Es geht nicht um die Vereinigung von virtueller und realer Welt, sondern um deren Schnittstelle
- Rechen- bzw. Maschinenzeit nicht mehr losgelöst von der realen Zeit physischer Prozesse
- Korrektheit ist nicht nur durch das logische Ergebnis , sondern auch durch den **physikalischen Zeitpunkt** bestimmt

Was ist Zeit?

- Historisch:
  - Die Zeitspanne zwischen zwei aufeinanderfolgenden Höchstständen der Sonne heißt Tag (genauer gesagt: ein Sonnentag).
  - Eine Sonnensekunde ist 1/86400 dieser Spanne.
- Aktuell:
  - Atomuhr: Messung von Zeit auf Basis der charakteristischen Frequenz von Strahlungsübergängen der Elektronen freier Cäsium-Atome
- Rechnersystem:
  - Messung von Zeit auf Basis der Schwingung eine Quarz-Kristalls
  - Genauigkeit gegenüber Atomuhr deutlich geringer



Sonnenuhr
Deutsches Museum



Erste Cäsiumatomuhr

Quarz

Zeitproblematik bei Echtzeitsystemen

- Probleme bei der Messsung und Verarbeitung der Zeit
  - Diskretisierung
  - Drift
  - Synchronisation von Zeitmessung
- Probleme bei der Ausführung über die Zeit
  - Determinismus
  - Parallelität
  - Synchronisation von Prozessen
- Die Zeit ist die einzige allgemein gültige Bezugs-Größe, die jedem (System) **überall gleichrangig** zur Verfügung steht

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 häufig fehlertolerant sein, da sie die Umgebung beeinflussen und ein Hardware-/Softwareausfall fatale (!) Folgen haben kann
  - 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) Uhrensynchronisation notwendig
  - Für die einzelnen Berechnungen ist jeweils ein Zeitslot reserviert) 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 ausgelöst
  - Wichtig sind bei ereignisgesteuerten Anwendungen garantierte Antwortzeiten
  - Das Scheduling erfolgt dynamisch, da zur Übersetzungszeit keine Aussage über den zeitlichen Ablauf getroffen werden kann.

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.

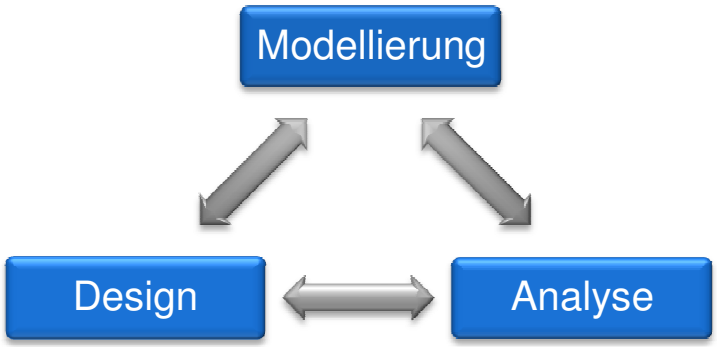
Definition Cyber-Physical System

- A cyber-physical system (CPS) is an integration of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa. As an intellectual challenge, CPS is about the intersection, not the union, of the physical and the cyber. It is not sufficient to separately understand the physical components and the computational components. We must instead understand their interaction. (Lee)
- CPS = embedded Systems + (Internet) communication

Modellierung, Design und Analyse von Echtzeitsystemen

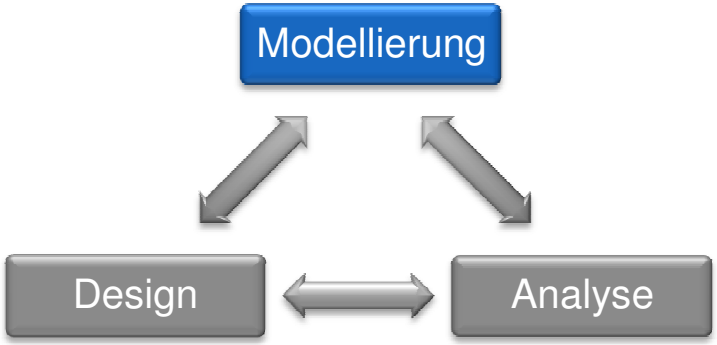
- "Naives Codieren" von Echtzeitsystemen nicht sinnvoll
  - Zusicherbare/Garantierbare Leistungen notwendig
  - Systeme müssen fehlertolerant sein
  - Verfizierbarkeit gegenüber einem Modell
  - Potenziell Menschenleben in Gefahr
- Deshalb ist die sorgfältige Modellierung des Verhaltens des Prozesses und des Systemverhaltens unabdingbar
- Erst das Modell eröffnet die Möglichkeit zur formalen Überprüfung!

Modellierung, Design und Analyse von Echtzeitsystemen



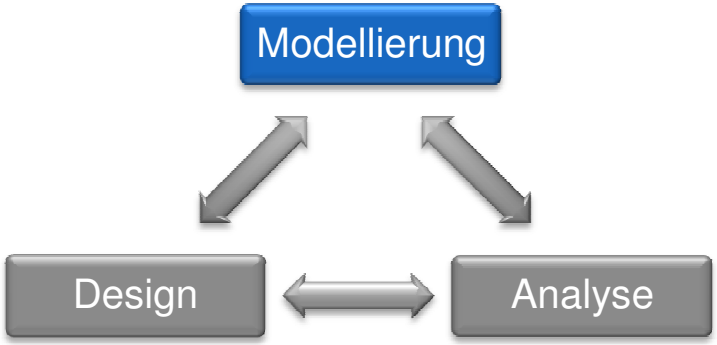
- Modellierung:
  - Modell ist ein **Abbild der Realität** (im Regelfall bezogen nur auf interessierende Eigenschaften)
  - Widerspiegelung von Systemeigenschaften
  - Bildet ab, *was* das System macht
- Design:
  - Umsetzung/Durchführung (“Implementierung”)
  - Hardware/Software
  - Gibt vor, *wie* das System eine Leistung erbringt
- Analyse:
  - Verständnis für das System
  - Erklärt *warum* das System macht, was es macht

Modellierungsaspekte



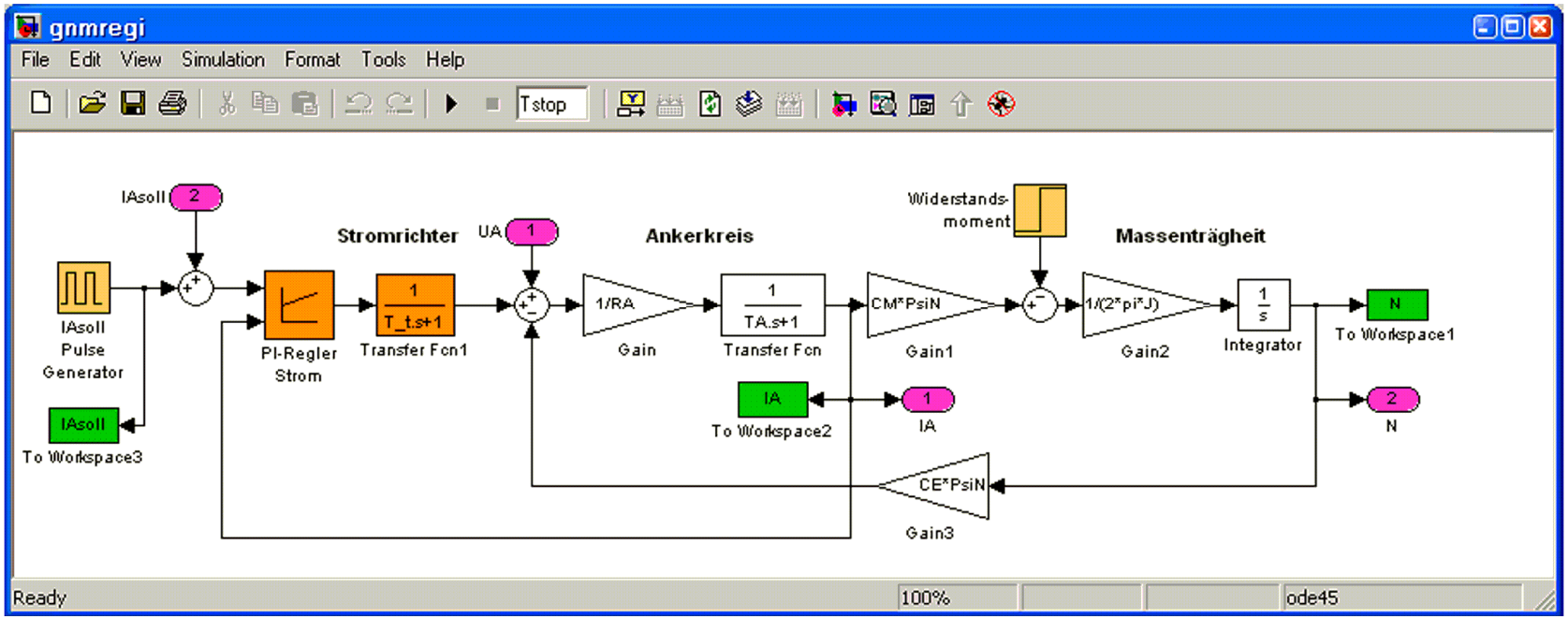
- Kontinuierliche Dynamik
- Diskrete Dynamik
- Hybride Systeme
- State Machines
- Concurrent Models of Computation
- Werkzeuge zur Modellierung

Modellierung - Werkzeuge



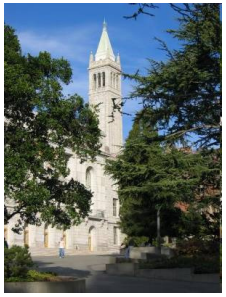
- Bleistift und Papier
- Matlab/Simulink
- Scade
- Esterel
- Ptolemy
- ……

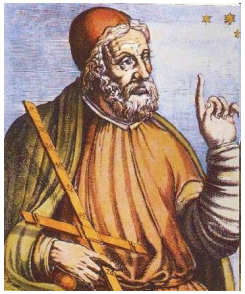
Modellierung – Werkzeuge – Simulink

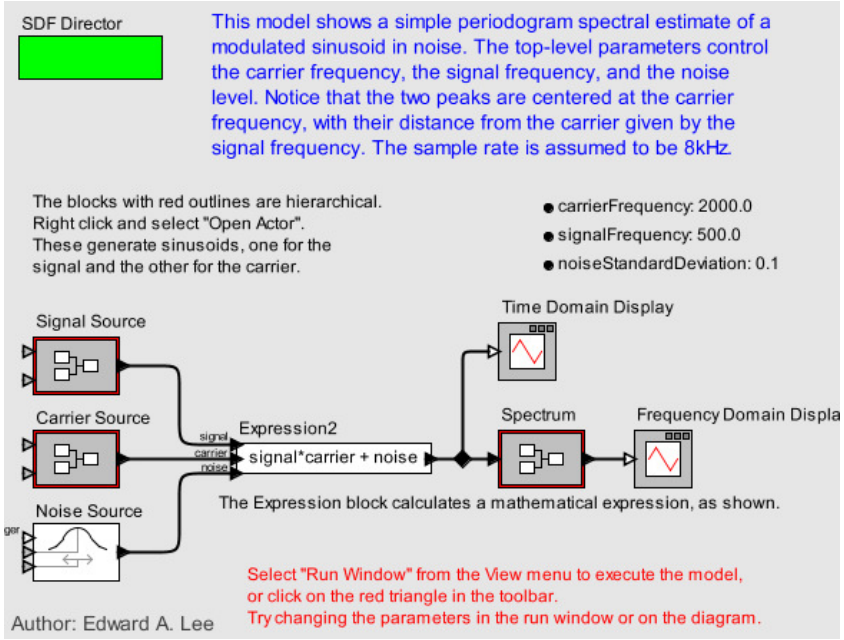


- Modellierung kontinuierlicher und diskreter dynamischer Systeme

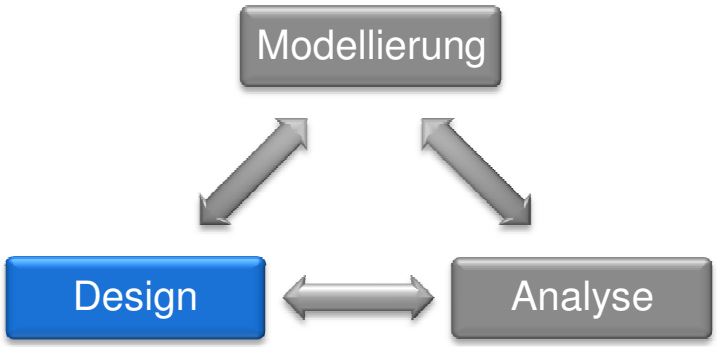
Modellierung – Werkzeuge – Ptolemy







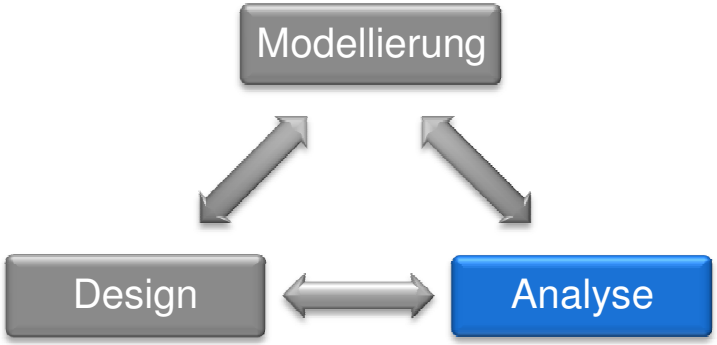
- Ptolemy-Projekt der UC Berkeley
- Benannt nach **Claudius Ptolemaeus**, (* um 100, vermutlich in Ptolemais Hermii, Ägypten; † um 175)
- Unterstützung verschiedener Ausführungs-Modelle (Models of Computation)
- Ptolemy unterstützt
  - Modellierung
  - Simulation
  - Codegenerierung
  - Formale Verifikation (teilweise)
- Weitere Informationen und kostenloser Dowload unter: http://ptolemy.eecs.berkeley.edu/



Design

- Hardware:
  - Prozessor
  - Speicherarchitektur
  - Ein-/Ausgabe
- Software:
  - Betriebssystem
  - Nebenläufigkeit
  - Scheduling
  - Implementierungstechnik (z.B. Codegenerierung)
  - ...

Analyse



- Invarianz
- Erreichbarkeits-Analyse
- Model-Checking
- Ausführungs-Zeit
- Fehlertoleranz
- ....

Einführung Echtzeitsysteme

Echtzeitsysteme im Alltag

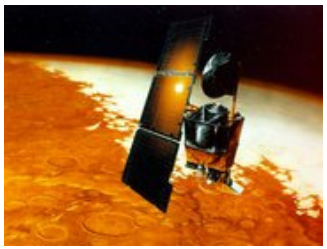
Echtzeitsysteme sind allgegenwärtig!











Beispiele

- Auto
  - Motormanagement:
    - Benzinmenge und Zündzeitpunkt anhängig von Motordrehzahl
    - Hohe Drehzahl erfordert Reaktionszeiten unter 1 ms
  - Airbag:
    - Permanente Überwachung von Sensorik, um festzustellen, ob der Airbag ausgelöst werden soll
    - Meist Redundante Sensorik um Airbag nur im Ernstfall zu zünden
    - Reaktionszeit im Bereich von 1 ms

Weitere Beispiele

- Aufzugsteuerung
- Ampelsteuerung
- Thermostat/Klimaanlage
- Mobiltelefone
- ABS-System
- …

Beispiel: Kuka Robocoaster

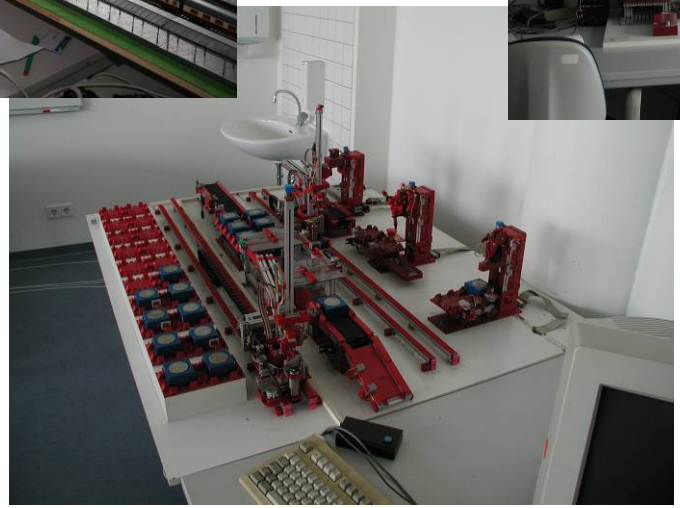



http://www.robocoaster.com

Einleitung Echtzeitsysteme

Anwendungen am Lehrstuhl / fortiss

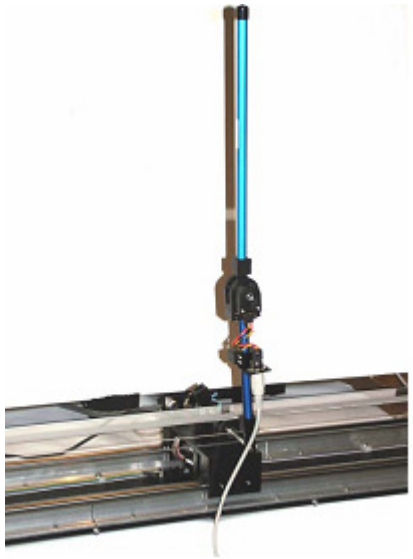
Steuerungsaufgaben (Praktika+Studienarbeiten)



Regelungsaufgaben (Praktika+Studienarbeiten)

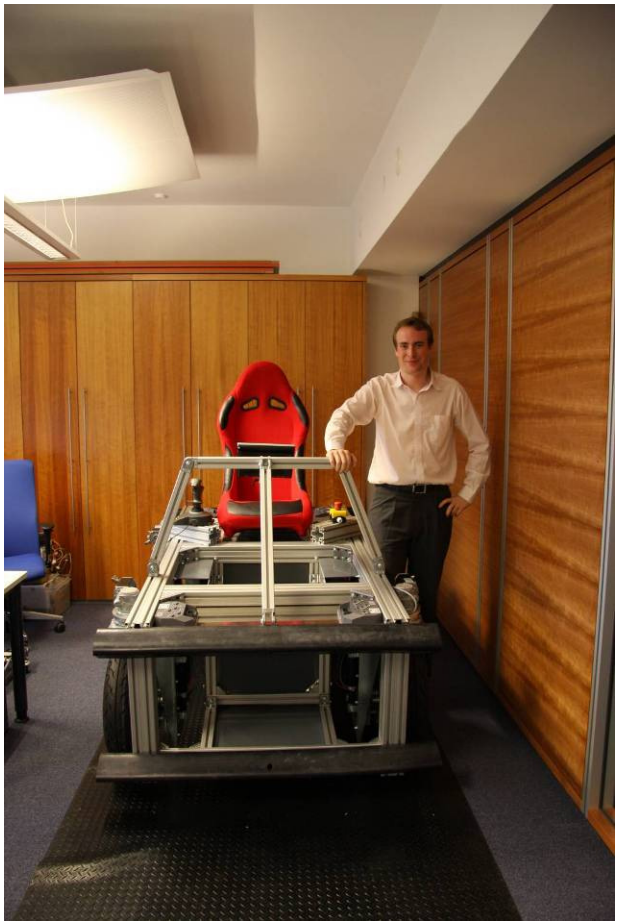
Schwebender Stab

Produktionstechnik

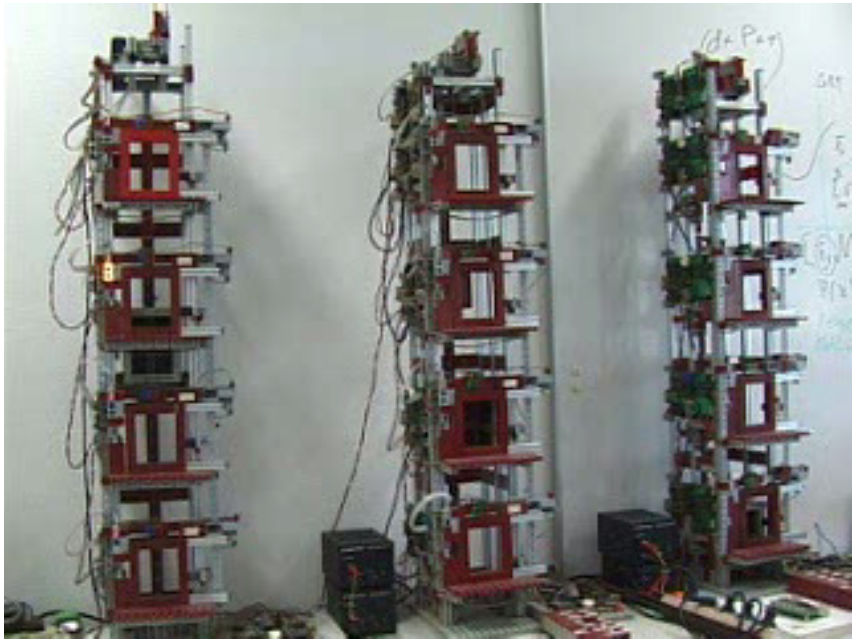


Invertiertes Pendel

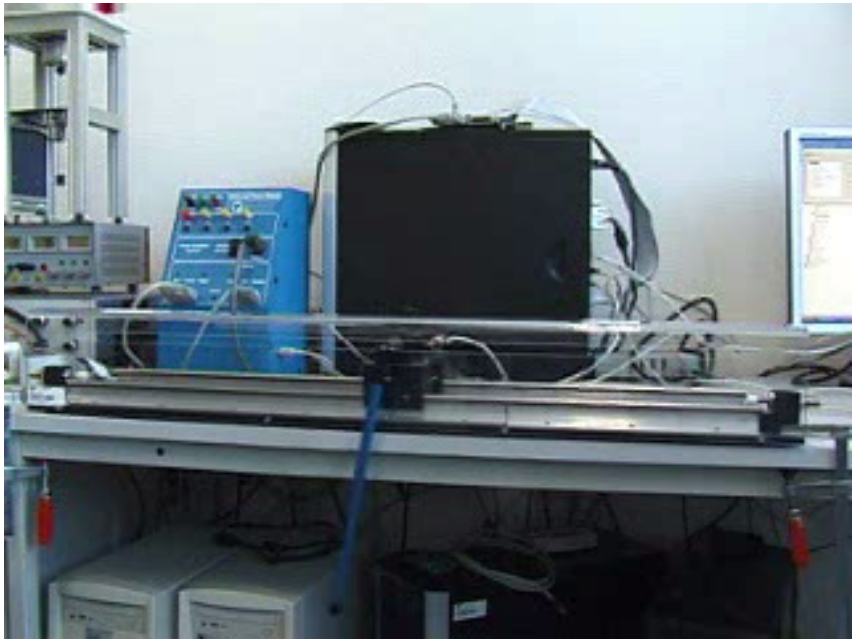
Elektroauto



Videos



Software entwickelt mit FTOS

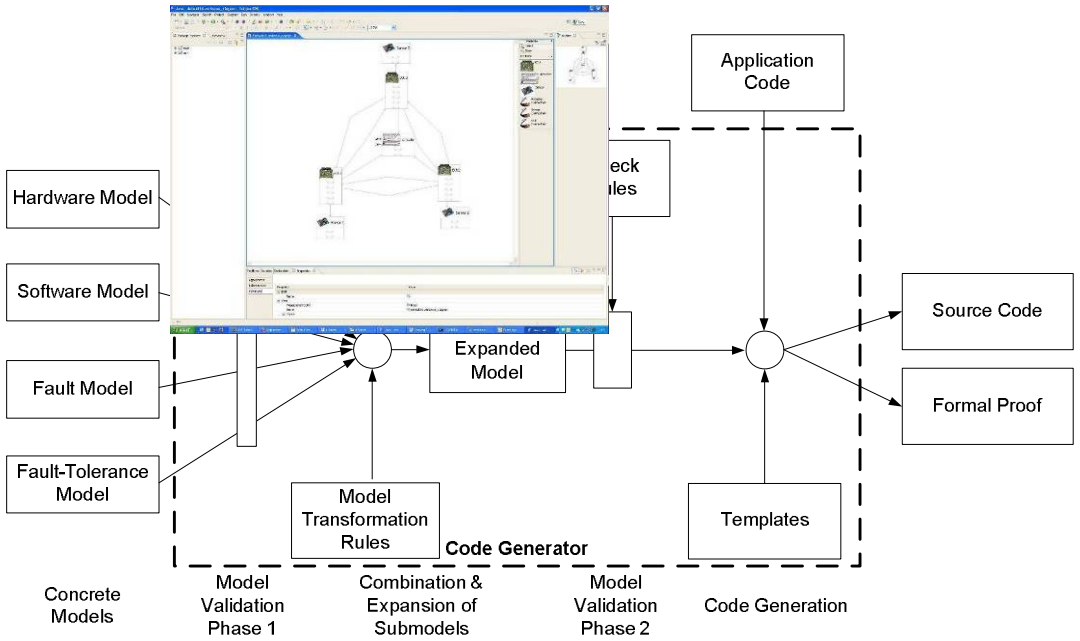


Software entwickelt mit EasyLab

FTOS: Modellbasierte Entwicklung fehlertoleranter Echtzeitsysteme

Ziele:

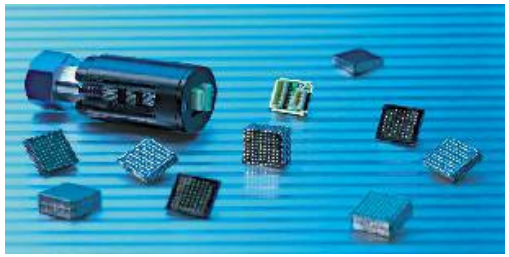
- Umfangreiche Generierung von Code auf Systemebene:
  - Fehlertoleranzmechanismen
  - Prozessmanagement, Scheduling
  - Kommunikation
- Erweiterbarkeit der Codegenerierung durch Verwendung eines vorlagen-basierten Codegenerators
- Zertifizierung des Codegenerators

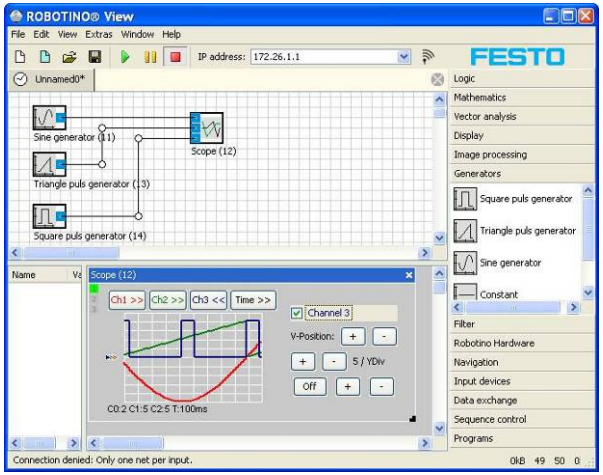


GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Modellbasierte Software-Entwicklung für mechatronische Systeme

- Entwicklung von komponentenbasierten Architekturen für mechatronische Systeme (Mechanik, Elektronik, Software)
- Ziele:
  - Reduzierung der Entwicklungszeiten
  - Vereinfachung des Entwicklungsprozesses
- Komponenten:
  - Hardwaremodule
  - Softwaremodule
  - Werkzeugkette:
    - Codegenerierung
    - Graphische Benutzerschnittstelle
    - Debugging-Werkzeug

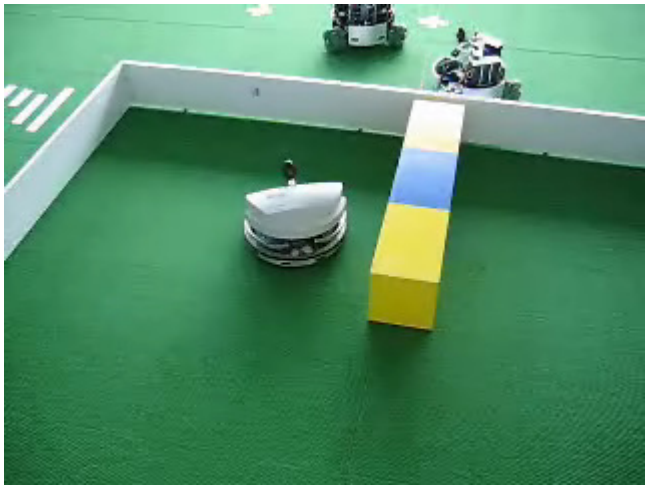


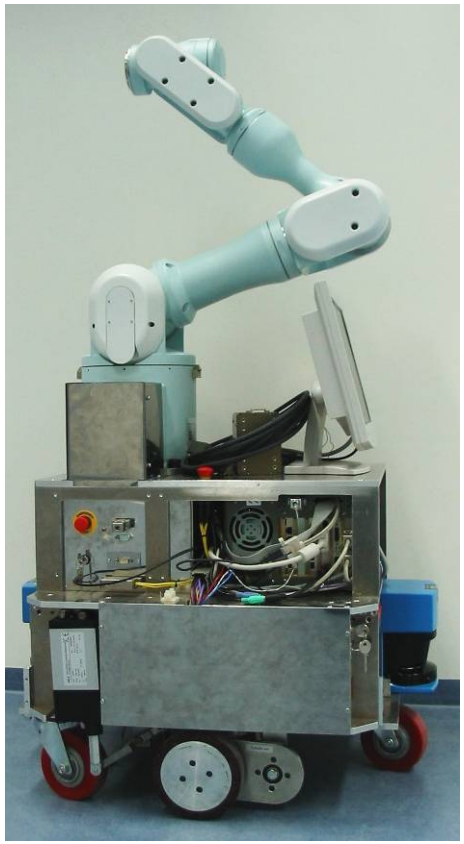


GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Robotersteuerung

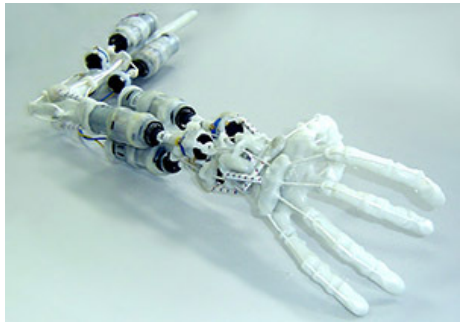
Robotino





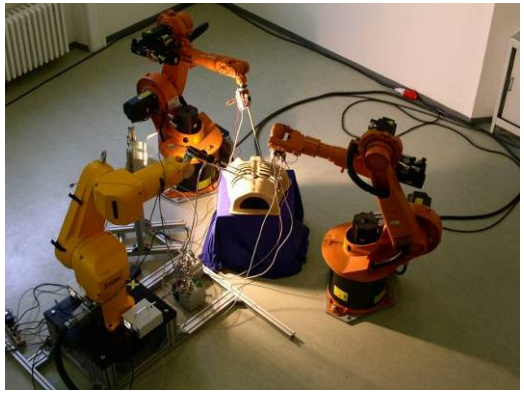
Leonardo

Stäubli



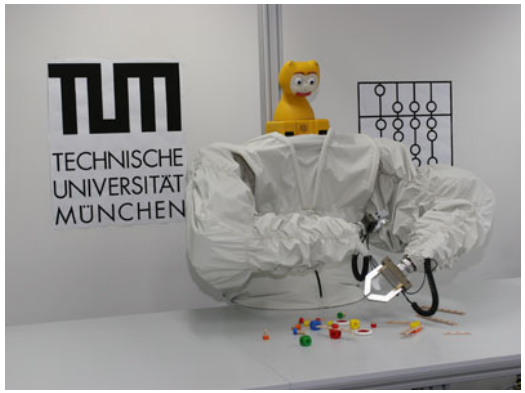
Tumanoid

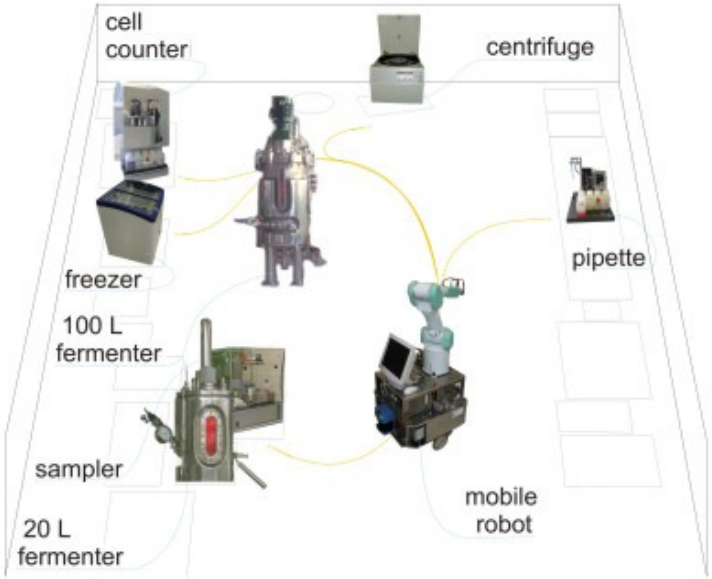
Anwendungen der Robotik



Telemedizin

Jast





Automatisiertes biotechnisches Labor

Automatisiertes Labor