Echtzeitsysteme
Lehrstuhl Informatik VI - Robotics and Embedded Systems

Echtzeitsysteme
Wintersemester 2011/2012

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: Nadine Keddis, Dominik Sojer, Stephan Sommer

Homepage der Vorlesung mit Folien, Übungsaufgaben und weiterem Material:
http://www6.informatik.tu-muenchen.de/Main/TeachingWs2011Echtzeitsysteme

Bestandteile der Vorlesung

- Vorlesung:
  - Dienstag 10:15-11:45 Uhr MI HS 2
  - Donnerstag 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 14:30 – 15:30 Uhr
  - Weitere Termine bei Bedarf nach Vereinbarung
  - Beginn: voraussichtlich ab 07.11.2011, Anmeldung ab sofort über TUMonline (http://www.tumonline.de, siehe Tutorübung zu Echtzeitsysteme)
- 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 Stephan Sommer (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

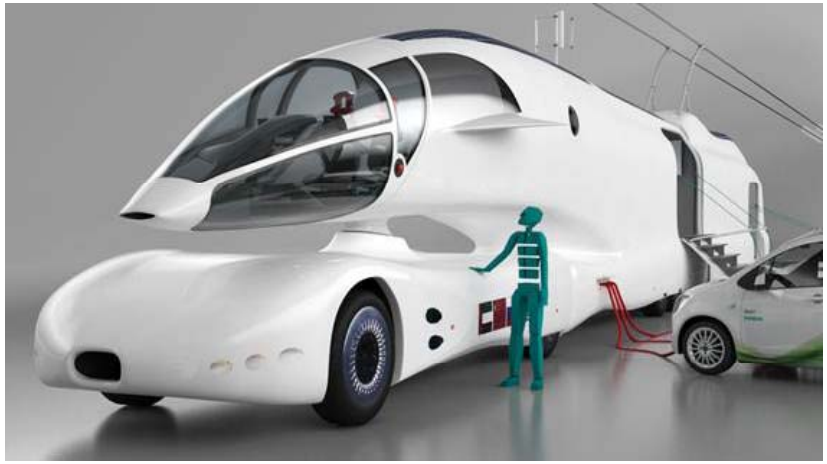
- Am Lehrstuhl werden eine große Zahl von Projekten im Bereich embedded systems bzw. cyber-physical systems durchgeführt
- Für die Hörer dieser Vorlesung bieten sich zum Einstieg zwei Projekte besonders an: die Leitprojekte "**E-Fahrzeug 2.0**" und "**InnoTruck**"
- "E-Fahrzeug 2.0" ist ein Projekt des **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

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



Electric Car Software Architecture

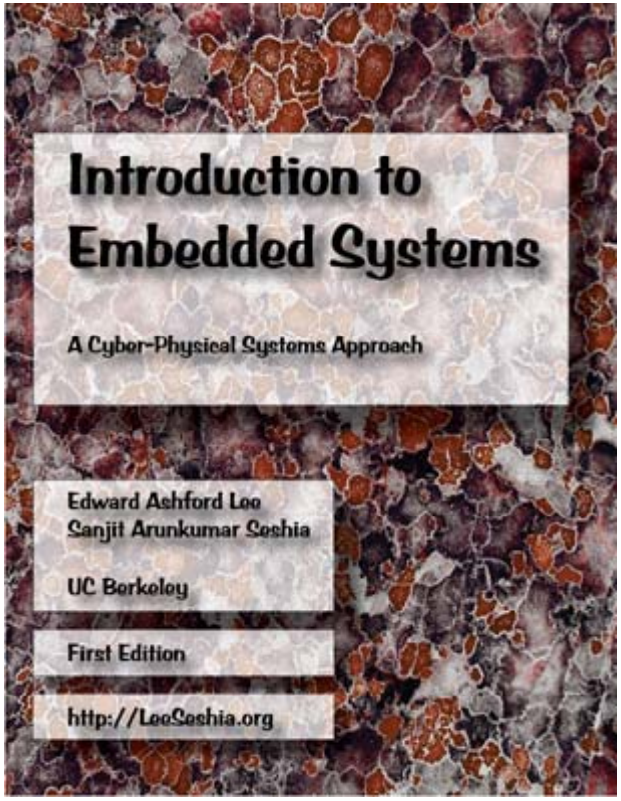
- The central concept of the architecture is **data flowing** from the sensors of the vehicle to its actuators, based on events and/or time predicates → *data-centric architecture with clear data structuring*
- **Avoid replication** of data and its acquisition → centralised logical data base or "*data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

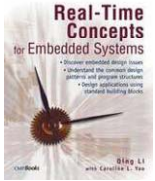
Weitere Literaturangaben befinden sich in den jeweiligen Abschnitten.

Hermann Kopetz: Real-Time Systems (Überblick)

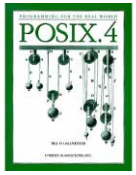
Jane W. S. Liu: Real-Time Systems
(Überblick, Schwerpunkt Scheduling)

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. Uhren
3. Modellierung und Werkzeuge
4. Nebenläufigkeit
5. Scheduling
6. Kommunikation
6. Echtzeitbetriebssysteme
7. Programmiersprachen
8. Fehlertolerante Systeme
9. Spezielle Hardware
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 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 nc werden, insgesamt kommt es durch die Fristverletzung evtl. zu einer Diens
  Beispiel für ein weiches Echtzeitsystem: Video
  Konsequenz von Fristverletzungen: einzelne Videoframes gehen verloren,

- **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 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

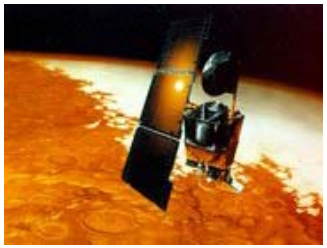
Echtzeitsysteme sind allgegenwärtig!











Beispiel: Kuka Robocoaster




http://www.robocoaster.com

Einleitung Echtzeitsysteme

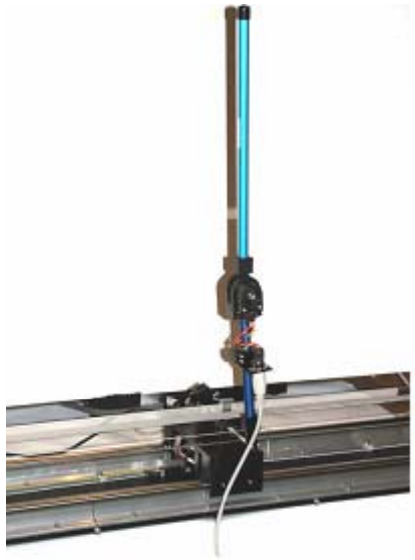
Anwendungen am Lehrstuhl / fortiss

Steuerungsaufgaben (Praktika+Studienarbeiten)

Regelungsaufgaben (Praktika+Studienarbeiten)

Schwebender Stab

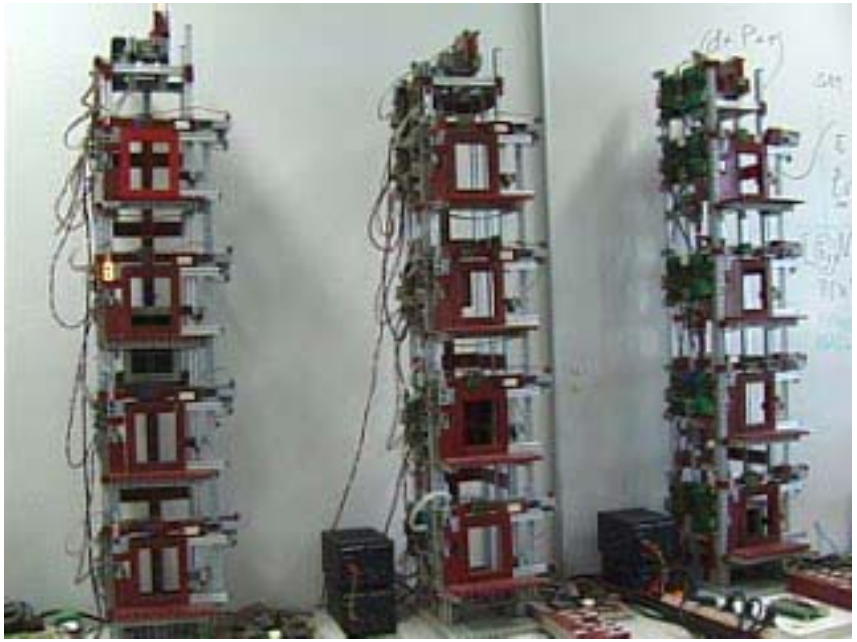
Produktionstechnik



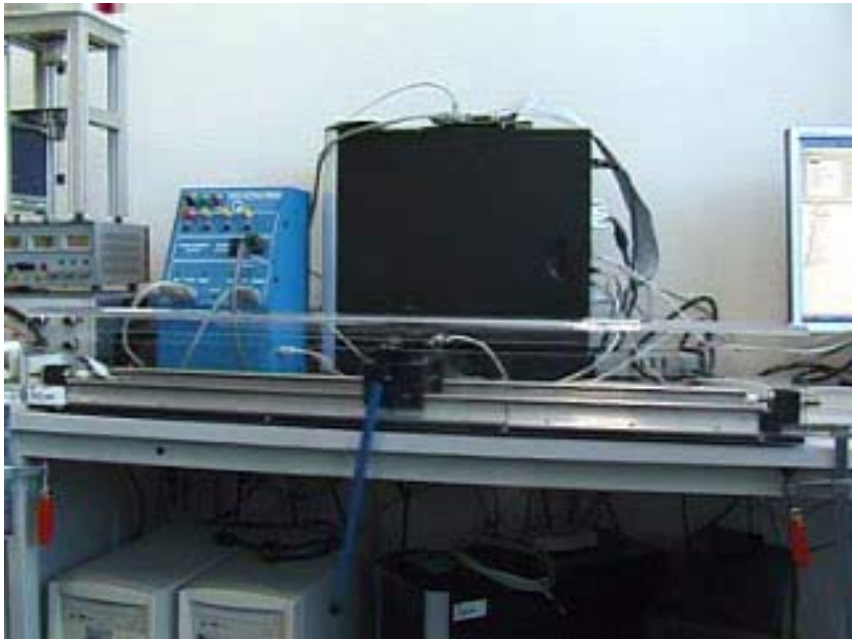
Invertiertes Pendel

Elektroauto

Videos



Software entwickelt mit FTOS

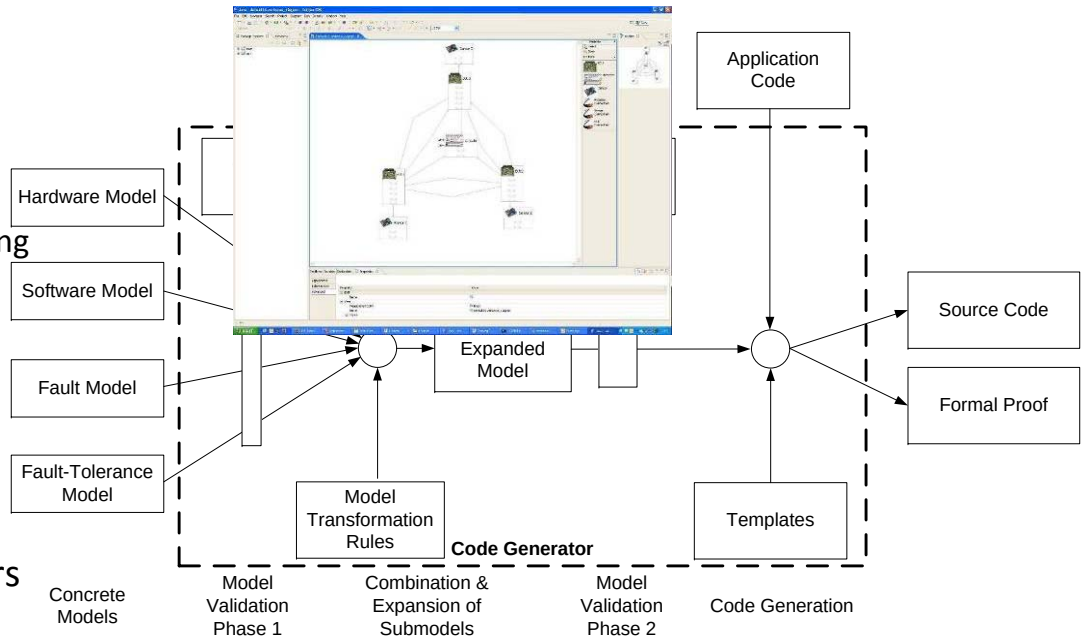


Software entwickelt mit EasyLab

FTOS: Modellbasierte Entwicklung fehlertoleranter Echtzeitsysteme

Ziele:

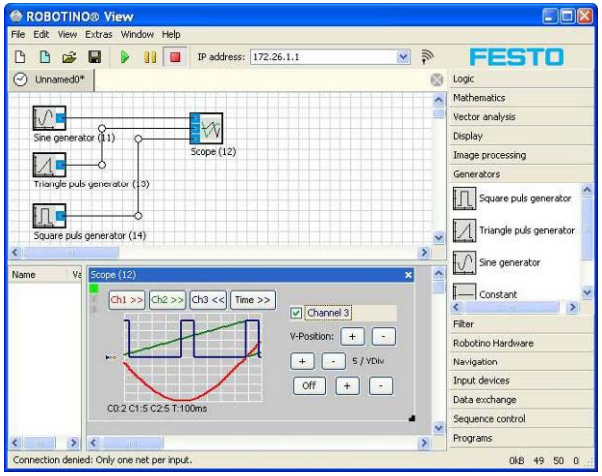
- Umfangreiche Generierung von Code auf Systemebene:
  - Fehlertoleranzmechanismen
  - Prozessmanagement, Scheduling
  - Kommunikation
- Erweiterbarkeit der Codegenerierung durch Verwendung eines vorlagen-basierten Codegenerators
- Zertifizierung des Codegenerators



GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Modellbasierte Software-Entwicklung für mechatronische Systeme

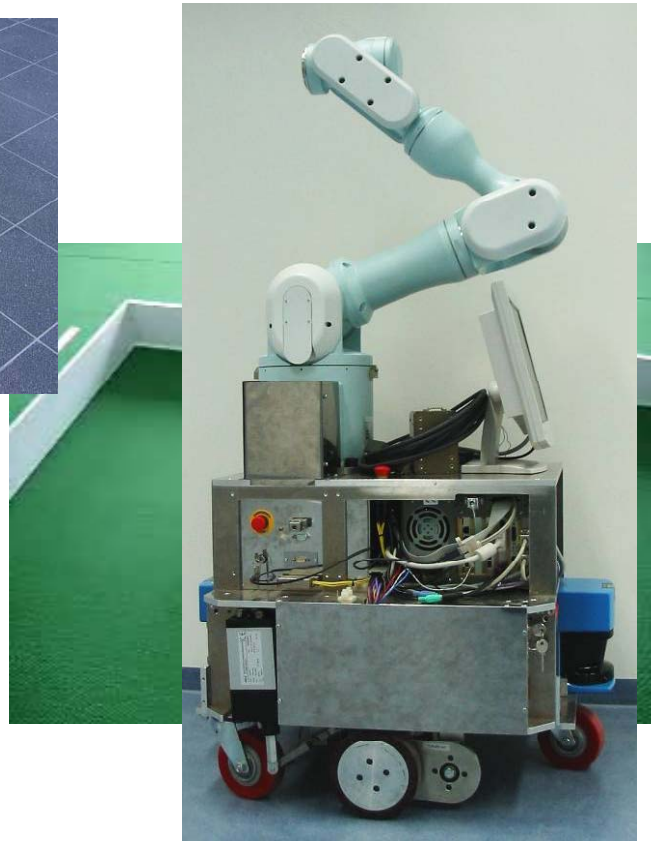
- Entwicklung von komponentenbasierten Architekturen für mechatronische Systeme (Mechanik, Elektronik, Software)
- Ziele:
  - Reduzierung der Entwicklungszeiten
  - Vereinfachung des Entwicklungsprozesses
- Komponenten:
  - Hardwaremodule
  - Softwaremodule
  - Werkzeugkette:
    - Codegenerierung
 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



GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

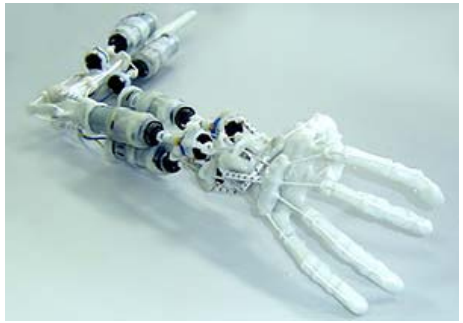
Robotersteuerung

Robotino



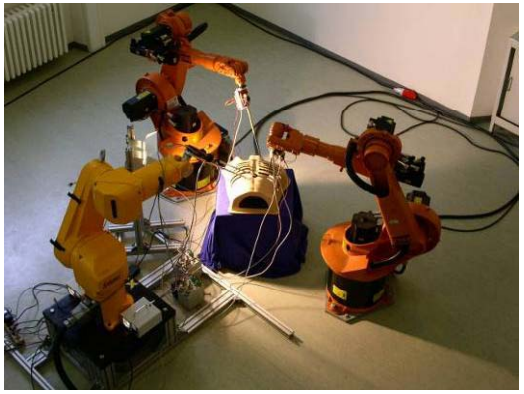
Leonardo

Stäubli



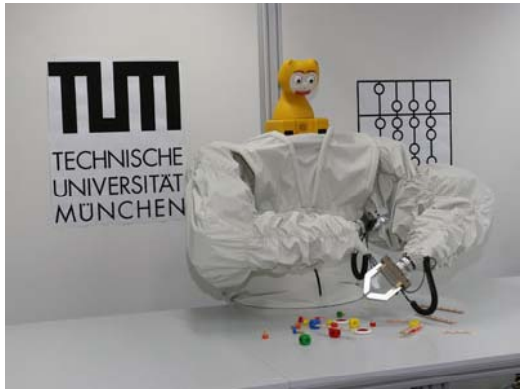
Tumanoid

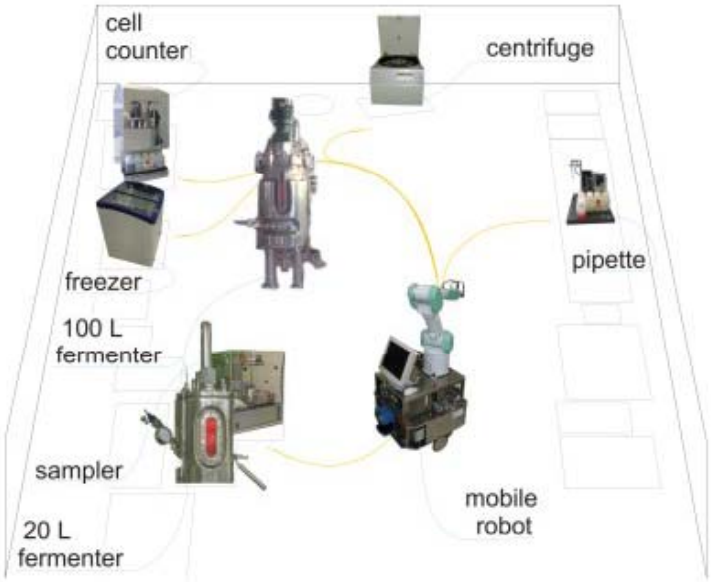
Anwendungen der Robotik



Telemedizin

Jast





Automatisiertes biotechnisches Labor

Automatisiertes Labor

Kapitel 2

Uhren & Synchronisation

Inhalt

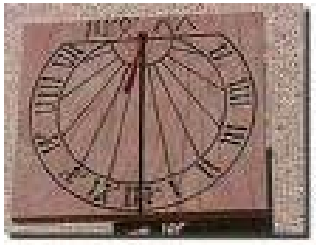
- Motivation
  - Definition Zeit
- Uhren
- Synchronisation
  - Algorithmus von Cristian
  - Algorithmus aus Berkeley
  - NTP-Protokoll
  - Synchronisation bei fehlerbehafteten Uhren

Literatur

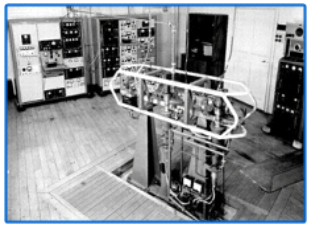
- Links zum Thema Zeit:
  - http://www.ptb.de/de/zeit/uhrzeit.html
  - http://www.maa.mhn.de/Scholar/dt_times.html
- Uhrensynchronisation:
  - Leslie Lamport: Synchronizing clocks in the presence of faults, 1985
  - http://www.ntp.org/

Definition Zeit

- Historisch:
  - Jeden Tag gegen Mittag erreicht die Sonne ihren höchsten Punkt am Himmel.
  - Die Zeitspanne zwischen zwei aufeinander folgenden Ereignissen dieses Typs heißt Tag (genauer gesagt: ein Sonnentag).
  - Eine Sonnensekunde ist 1/86400 dieser Spanne.
- Zeitmessung heute:
  - Verwendung von Atomuhren: eine Sekunde ist die 9.192.631.770-fache Periodendauer, der dem Übergang zwischen den beiden Hyperfeinstrukturniveaus des Grundzustands von ^{133}Cäsium-Atomen entsprechenden Strahlung.
  - Am 01.01.1958 entsprach die Atomsekunde genau einer Sonnensekunde.
  - Aufgrund von unregelmäßigen Schwankungen, sowie einer langfristigen Verlangsamung der Erdrotation unterscheiden sich die Atomsekunde und die Sonnensekunde.



Sonnenuhr
Deutsches Museum



Erste Cäsiumatomuhr

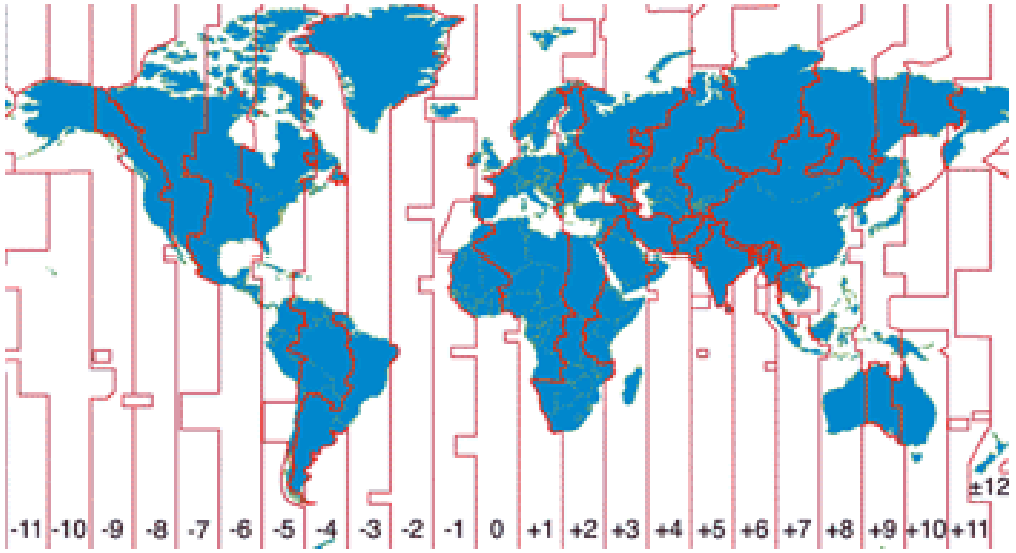
TAI (Temps Atomique International)

- TAI: Atomzeitskala, die zur Koordination nationaler Atomzeiten ermittelt wird:
  - Beteiligung von 50 verschiedene Zeitinstitute mit ca. 250 Atomuhren
  - Zeit basiert auf der Atomsekunde
  - Referenzzeitpunkt ist der 1.Januar 1970
  - relative Genauigkeit von +/ - 10^{-15}, aber keine exakte Übereinstimmung mit der Sonnenzeit

Atomuhr der Physikalisch-Technischen Bundesanstalt in Braunschweig

UTC (Coordinated Universal Time)

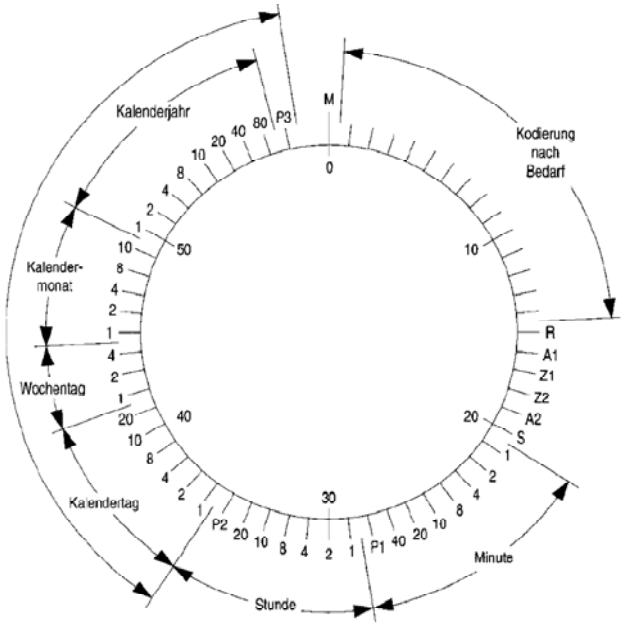
- eigentlicher Nachfolger der Greenwichzeit
- realisiert durch Atomuhren, die Zeiteinheit ist die SI-Sekunde
) hochkonstante Zeiteinheit
- zusätzlich Übereinstimmung mit dem Sonnenlauf
) einheitliche Grundlage zur Zeitbestimmung im täglichen Leben
- Durch Einfügen von Schalt-sekunden wird die UTC mit der universellen Sonnenzeit (UT1) synchronisiert
- Anpassung erfolgt zumeist zu Ende oder Mitte des Jahres (typischer Abstand: alle 18 Monate)



Zeitzonen

DCF77

- Das PTB überträgt die aktuelle Uhrzeit über den Langwellen-sender DCF77
- Die Zeitinformationen werden als digitales Signal (negative Modulation) Absenkung der Trägeramplitude) im Sekunden-takt übertragen.



- '0' und '1' werden durch eine Absenkung um 100ms bzw. 200 ms codiert. In der Sekunde 59 erfolgt keine Absenkung) Markierung der Beginn einer neuen Minute bei nächster Amplitudenabsenkung.
- Pro Minute stehen somit 59 Bit zur Verfügung (wobei Bit 0-14 für Betriebsinformationen verwendet werden)