

# Echtzeitverarbeitung

R. Kaiser, K. Beckmann, R. Kröger

(HTTP: <http://www.cs.hs-rm.de/~kaiser>

E-Mail: [robert.kaiser@hs-rm.de](mailto:robert.kaiser@hs-rm.de))

Sommersemester 2021

# 1. Einführung und Grundlagen



<http://gymnasium-blomberg.de/wp-content/uploads/2013/05/Wegweiser.jpg>

# Inhalt



## 1. Einführung und Grundlagen

1.1 Motivation

1.2 Einführung

1.3 Begriffe

# Motivation: ESP in der Presse



Quelle: Tagesanzeiger (Schweiz) vom 12.11.1997

## Auslieferungsstopp für A-Klasse

**Mercedes unterbricht für drei Monate die Auslieferung des kompakten A-Klasse-Fahrzeugs. In dieser Zeit sollen erhebliche Änderungen vorgenommen werden.**

Von Marcel Waeber

100 000 Vorbestellungen lagen vor, als vor gut zwei Wochen beim Mercedes der A-Klasse gravierende Sicherheitsmängel publik wurden. Das Fahrzeug war bei einem sogenannten Elchtest (einem abrupten Ausweichmanöver) gekippt, und eine Testperson war dabei verletzt worden (TA vom Montag). In der Folge wurden die ersten Verkaufsverträge rückgängig gemacht. Jetzt reagiert der Daimler-Benz-Konzern auf die Stornierungen.

### Neue technische Lösungen

„Wir nehmen die öffentlich geäusserte Kritik und vor allem die Sorgen unserer Kunden sehr ernst und bedauern die gezeigte Schwäche der A-Klasse in einem Extremtest“, sagte am Dienstag der Vorstandsvorsitzende Jürgen E. Schrepp. Gleichzeitig kündigte er Lösungen für die technischen Probleme des „Baby-Benz“ an: neue Stabilisatoren, neue Feder- und Dämpferabstimmung an den Achsen, die Karosserie wird tiefergelegt, und die Reifen werden neu dimensioniert.

Wie Jürgen Hubbert, verantwortlich für den Bereich Personenwagen, ergänzend ausführte, soll die A-Klasse zusätzlich serienmässig mit dem Electronic Stability Program (ESP) ausgerüstet werden. Rechnergesteuert greift das ESP gezielt in das Bremssystem ein und hält das Fahrzeug bei Ausweichmanövern und Kurvenfahrten stabil. Das Schleuderrisiko wird dadurch massiv reduziert. Hubbert: „Situationen bei Eis, Schnee und Nässe lassen sich mit dem ESP beherrschen.“

Copyright ©TA-Media AG

# Motivation: ESP in der Presse



Quelle: Tagesanzeiger (Schweiz) vom 12.11.1997

## Auslieferungsstopp für A-Klasse

**Mercedes unterbricht für drei Monate die Auslieferung des kompakten A-Klasse-Fahrzeugs. In dieser Zeit sollen erhebliche Änderungen vorgenommen werden.**

Von Marcel Waeber

100 000 Vorbestellungen lagen vor, als vor gut zwei Wochen beim Mercedes der A-Klasse gravierende Sicherheitsmängel publik wurden. Das Fahrzeug war bei einem sogenannten Elchtest (einem abrupten Ausweichmanöver) gekippt, und eine Testperson war dabei verletzt worden (TA vom Montag). In der Folge wurden die ersten Verkaufsverträge rückgängig gemacht. Jetzt reagiert der Daimler-Benz-Konzern auf die Stornierungen.

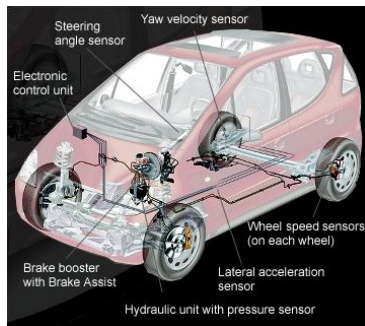
### Neue technische Lösungen

„Wir nehmen die öffentlich geäusserte Kritik und vor allem die Sorgen unserer Kunden sehr ernst und bedauern die gezeigte Schwäche der A-Klasse in einem Extremtest“, sagte am Dienstag der Vorstandsvorsitzende Jürgen E. Schrepp. Gleichzeitig kündigte er Lösungen für die technischen Probleme des „Baby-Benz“ an: neue Stabilisatoren, neue Feder- und Dämpferabstimmung an den Achsen, die Karosserie wird tiefergelegt, und die Reifen werden neu dimensioniert.

Wie Jürgen Hubbert, verantwortlich für den Bereich Personenwagen, ergänzend ausführte, **soll die A-Klasse zusätzlich serienmässig mit dem Electronic Stability Program (ESP) ausgerüstet werden. Rechnergesteuert greift das ESP gezielt in das Bremssystem ein und hält das Fahrzeug bei Ausweichmanövern und Kurvenfahrten stabil.** Das Schleuderrisiko wird dadurch massiv reduziert. Hubbert: „Situationen bei Eis, Schnee und Nässe lassen sich mit dem ESP beherrschen.“

Copyright ©TA-Media AG

# ESP-Hardware



## Beispiel: ESP

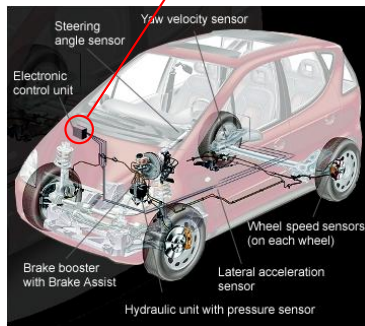
(Elektronisches Stabilitäts-Programm)

- Sensoren erfassen Beschleunigung und Winkelgeschwindigkeiten
- Einflussnahme auf Bremsen (einzeln ansteuerbar)
- Steuerrechner implementiert Funktionalität

# ESP-Hardware



## Steuerrechner



## Beispiel: ESP

(Elektronisches Stabilitäts-Programm)

- Sensoren erfassen Beschleunigung und Winkelgeschwindigkeiten
- Einflussnahme auf Bremsen (einzeln ansteuerbar)
- Steuerrechner implementiert Funktionalität

# ESP-Software



## Sensor-Überprüfung alle 20 Millisekunden

Das zentrale ESP-Steuergerät besteht aus zwei Rechnern mit einer Speicherkapazität von jeweils 56 kByte. Zum Vergleich: ABS benötigt nur ein Viertel dieser Rechnerleistung. ESP® nutzt diese großen Rechnerkapazitäten, um unter anderem laufend die einzelnen System-Komponenten zu überprüfen. So wird beispielsweise der wichtige Sensor zur Erfassung der Drehgeschwindigkeit des Fahrzeugs nach jedem Messvorgang zusätzlich kontrolliert - im Rhythmus von 20 Millisekunden.

©DaimlerChrysler, 1998

## Beispiel: ESP (Elektronisches Stabilitäts-Programm)

- Digitale Regelung
- Zeitverhalten muss vorhersagbar („deterministisch“) sein.
- „Eingebettetes System“ (*embedded System*)

⇒ Mit klassischen PCs wohl kaum machbar.



# ESP-Software



## Sensor-Überprüfung alle 20 Millisekunden

Das zentrale ESP-Steuergerät besteht aus zwei Rechnern mit einer Speicherkapazität von jeweils 56 kByte. Zum Vergleich: ABS benötigt nur ein Viertel dieser Rechnerleistung. ESP® nutzt diese **großen Rechnerkapazitäten**, um unter anderem laufend die einzelnen System-Komponenten zu überprüfen. So wird beispielsweise der wichtige Sensor zur Erfassung der Drehgeschwindigkeit des Fahrzeugs nach jedem Messvorgang zusätzlich kontrolliert - **im Rhythmus von 20 Millisekunden**.

©DaimlerChrysler, 1998

## Beispiel: ESP (Elektronisches Stabilitäts-Programm)

- Digitale Regelung
  - Zeitverhalten muss vorhersagbar („deterministisch“) sein.
  - „Eingebettetes System“ (*embedded System*)
- ⇒ Mit klassischen PCs wohl kaum machbar.

# Einführung

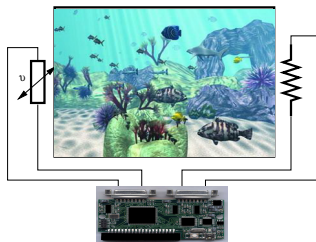


## Typische Anwendungsgebiete für Echtzeitsysteme:

- oft Problemstellungen der Mess- / Steuer- / Regelungstechnik
- Interaktion von Computern mit physikalischen/ technischen Systemen (mithilfe von Sensoren und Aktoren)
- phys./ techn. System gibt Zeitanforderungen vor
- Computer muss diesen Zeitanforderungen unter allen Umständen gerecht werden

### Beispiel: Aquarium

- Technisches System: Fischtank
- Sensor: Temperaturfühler
- Aktor: Heizung
- Embedded System (Regel-Algorithmus)

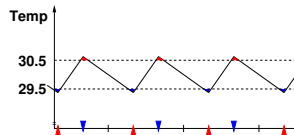


# Beispiel: Aquarium



## Einfacher Algorithmus

```
while TRUE do
  read(&WaterTemp);
  if WaterTemp < 29.5 then
    heater(TRUE);
  else
    if WaterTemp > 30.5 then
      heater(FALSE);
    end
  end
  wait(DELAY);
end
```



- In festen Zeitabständen prüfen:
  - ▶ Temp. zu hoch → Heizung aus
  - ▶ Temp. zu niedrig → Heizung an

**Wichtig: Physikalisches System bestimmt Echtzeitanforderung**

- Zeitabstände zu groß -> Fehler

„Zeitgesteuertes Echtzeitsystem“

# Beispiel: Aquarium



## Einfacher Algorithmus

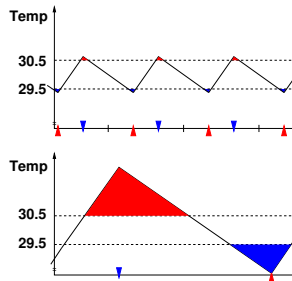
```
while TRUE do
  read(&WaterTemp);
  if WaterTemp < 29.5 then
    heater(TRUE);
  else
    if WaterTemp > 30.5 then
      heater(FALSE);
    end
  end
  wait(DELAY);
end
```

- In festen Zeitabständen prüfen:
  - ▶ Temp. zu hoch → Heizung aus
  - ▶ Temp. zu niedrig → Heizung an

**Wichtig: Physikalisches System bestimmt Echtzeitanforderung**

- Zeitabstände zu groß -> Fehler

„Zeitgesteuertes Echtzeitsystem“



# Beispiel: Aquarium



## Einfacher Algorithmus

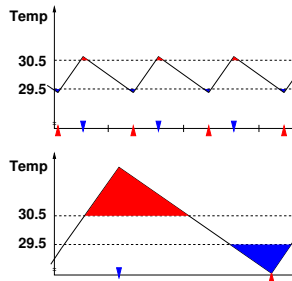
```
while TRUE do
  read(&WaterTemp);
  if WaterTemp < 29.5 then
    heater(TRUE);
  else
    if WaterTemp > 30.5 then
      heater(FALSE);
    end
  end
  wait(DELAY);
end
```

- In festen Zeitabständen prüfen:
  - ▶ Temp. zu hoch → Heizung aus
  - ▶ Temp. zu niedrig → Heizung an

**Wichtig: Physikalisches System bestimmt Echtzeitanforderung**

- Zeitabstände zu groß -> Fehler

**„Zeitgesteuertes Echtzeitsystem“**



# „Echtzeit“ heißt nicht „echt schnell“!



- Zu beachten: „Echtzeitverhalten“ beinhaltet keine Aussage über absolute Geschwindigkeit
- Anwendung gibt Zeitschranken vor
- Beispiel: Airbag
  - ▶ Zeitschranken im Millisekunden-Bereich
  - ▶ „Ereignisgesteuert“ (Ereignis: Aufprall)
  - ▶ Sowohl zu frühes als auch zu spätes Auslösen kann fatal sein
- (Zugleich ein Beispiel für ein sicherheitskritisches System)



<http://www.citroenet.org.uk/publicity-brochures/schiffer/schiffer.html>

# Software-Entwicklung



## Sicherheitskritische Bereiche erfordern oft den Einsatz von Echtzeitsystemen

- Prozessleittechnik
  - ▶ Walzstraßen, Chemiewerke, (Kern-)Kraftwerke, ...
- Fahrzeugtechnik
  - ▶ ABS, ESP, Steer-by-wire, Motorsteuerung, ...
- Avionik
  - ▶ Autopilot, Fly-by-wire, ...
- Fehleranfälligkeit wächst mit steigender Komplexität der Hard- und Software
- Gefährdungspotenzial eines Systems  $\Rightarrow$  „safety“-Anforderungen

$\Rightarrow$  Systematisches Testen gefordert  
(z.B. DO-178, Richtlinien der Flugzeugindustrie)

# Software-Entwicklung



## Sicherheitskritische Bereiche erfordern oft den Einsatz von Echtzeitsystemen

- Prozessleittechnik
  - ▶ Walzstraßen, Chemiewerke, (Kern-)Kraftwerke, ...
- Fahrzeugtechnik
  - ▶ ABS, ESP, Steer-by-wire, Motorsteuerung, ...
- Avionik
  - ▶ Autopilot, Fly-by-wire, ...
- Fehleranfälligkeit wächst mit steigender Komplexität der Hard- und Software
- Gefährdungspotenzial eines Systems  $\Rightarrow$  „safety“-Anforderungen

$\Rightarrow$  **Systematisches Testen** gefordert  
(z.B. DO-178, Richtlinien der Flugzeugindustrie)



# Beispiel: Sicherheitsstandard DO-178B



- Anforderungen der US-amerikanischen Bundesbehörde für Flugsicherheit (FAA) zur Zulassung von Systemen für den Einsatz in der zivilen Luftfahrt
- Definiert 5 „Integrity Levels“:

Level	Beschreibung	Anforderung
A	Failure would cause or contribute to a catastrophic failure of the aircraft	Level B + 100 % Modified Condition Decision Coverage
B	Failure would cause or contribute to a hazardous/severe failure condition	Level C + 100 % Decision Coverage
C	Failure would cause or contribute to a major failure condition	Level D + 100 % Statement (or line) Coverage
D	Failure would cause or contribute to a minor failure condition	Level D + 100 % Requirement Coverage
E	Failure would no effect on the aircraft or on pilot workload	No Coverage Requirements

⇒ Forderung nach systematischen Verfahren des Software Engineering entsprechender Systeme

# Beispiel: Sicherheitsstandard DO-178B



- Anforderungen der US-amerikanischen Bundesbehörde für Flugsicherheit (FAA) zur Zulassung von Systemen für den Einsatz in der zivilen Luftfahrt
- Definiert 5 „Integrity Levels“:

Level	Beschreibung	Anforderung
A	Failure would cause or contribute to a catastrophic failure of the aircraft	Level B + 100 % Modified Condition Decision Coverage
B	Failure would cause or contribute to a hazardous/severe failure condition	Level C + 100 % Decision Coverage
C	Failure would cause or contribute to a major failure condition	Level D + 100 % Statement (or line) Coverage
D	Failure would cause or contribute to a minor failure condition	Level D + 100 % Requirement Coverage
E	Failure would no effect on the aircraft or on pilot workload	No Coverage Requirements

⇒ Forderung nach systematischen Verfahren des Software Engineering entsprechender Systeme

# Definitionsversuch „Embedded System“



## Was ist ein Embedded System ?

- Ein Rechner, der aufgrund seiner Programmierung die Funktion eines bestimmten Gerätes erfüllt (*embedded* = *eingebettet*)
- Rechner ist i.d.R physisch in das Gerät mit Sensoren und Aktoren integriert
- Beispiele: Telefon (Smartphone: Grenzfall), CD-Player, Mpeg/MP3-Player, Spielautomat, Scannerkasse, Motorsteuerung, ABS, Drohne, ...
- Falls User Interface vorhanden, auch aktueller Begriff „Cyber-Physical System“ genutzt



# Definitionsversuch „Embedded System“ (2)

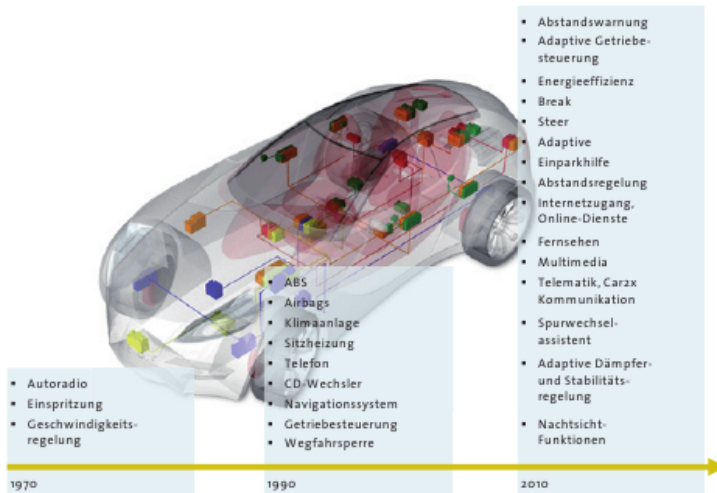


## Anforderungen an Embedded Systems

- **Häufig feste, statische Software („Firmware“)**
- robust (Verhalten bei störenden Einflüssen)
- kompakt, portabel, häufig keine bewegten Teile
- häufig Massenprodukte → geringer Preis
- batteriebetrieben → geringer Energieverbrauch
- „24/7“-Betrieb → zuverlässig, fehlertolerant
- u.U. lange Lebensdauer (z.B. 30+ Jahre)
- u.U. Sicherheitskritische („safety-critical“) Anwendungen

**Oftmals (aber nicht immer): echtzeitfähig**

# Immer mehr Anwendungen



Quelle: BITKOM – Eingebettete Systeme – Ein strategisches Wachstumsfeld für Deutschland (2010)

# Embedded Systems als Wachsender Markt



## Beispiel Automobilindustrie

- Teilweise mehr als 100 Prozessoren in einem Fahrzeug
- Haupt-Einsatzgebiet: Steuerung/Regelung
  - ▶ früher Oberklasse-Modellen vorbehalten
  - ▶ heute: z.B. ABS / ESP für die meisten Modelle erhältlich
- 25% der Gesamtkosten, bald bis zu 50%
- Erheblicher Teil der Kosten liegt in Software
  - ▶ z.B. ABS: 10.000 Zeilen C-Code
  - ▶ z.B. Klimaanlage: 25.000 Zeilen C-Code
  - ▶ Bereits > 100 MB Software auf allen Controllern

# Klassifikation Embedded SW: Low-End



## Low-End Embedded Control Anwendungen

- basierend auf 4/8-bit Prozessoren,  $\mu$ Controller oder Festpunkt-Signalprozessoren
- geringe Anwendungskomplexität
- i.d.R. keine Notwendigkeit für Betriebssystem
- steigender Software-Umfang in Lösungen
- Über lange Zeit steigender Markt für Software (1995: 4 Mrd\$, 2010: 749 Mrd\$, 2011: 713 Mrd\$)\*
- Marktanteil der low-end Anwendungen sinkt gegenüber High-End Anwendungen

\* Quelle: IDC-Studie „Intelligent Systems: The Next Big Opportunity“ (2011)

# Klassifikation Embedded SW: High-End



## High-End Embedded Control Anwendungen

- „Intelligente Systeme“:
- Ein oder mehrere 16/32/64-bit Prozessoren,  $\mu$ Controller oder Gleitpunkt-Signalprozessoren
- (64-bit Anteil noch gering)
- mittlere bis hohe Anwendungskomplexität
- Einsatz von Echtzeitbetriebssystemen, Softwareentwicklungswerkzeugen, etc.
- Markt für Software-Zulieferer
- stark wachsender Markt (2010: 865 Mrd\$, 2011: 1075 Mrd\$)\*
- Marktanteil der High-End Anwendungen steigt stark an (Migration aus dem low-end Bereich)

\* Quelle: IDC-Studie „Intelligent Systems: The Next Big Opportunity“ (2011)



# Klassifikation Embedded SW: High-End

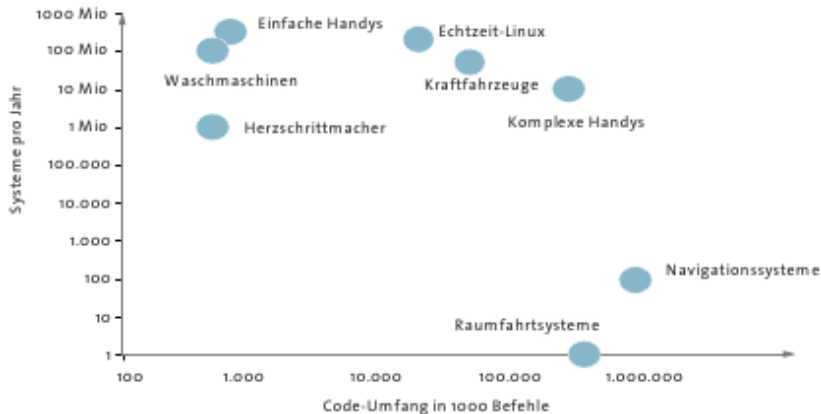


## High-End Embedded Control Anwendungen

- „Intelligente Systeme“:
- Ein oder mehrere 16/32/64-bit Prozessoren,  $\mu$ Controller oder Gleitpunkt-Signalprozessoren
- (64-bit Anteil noch gering)
- mittlere bis hohe Anwendungskomplexität
- Einsatz von Echtzeitbetriebssystemen, Softwareentwicklungswerkzeugen, etc.
- Markt für Software-Zulieferer
- stark wachsender Markt (2010: 865 Mrd\$, 2011: 1075 Mrd\$)\*
- Marktanteil der High-End Anwendungen steigt stark an (Migration aus dem low-end Bereich)

\* Quelle: IDC-Studie „Intelligent Systems: The Next Big Opportunity“ (2011)

# Komplexität von Embedded Software

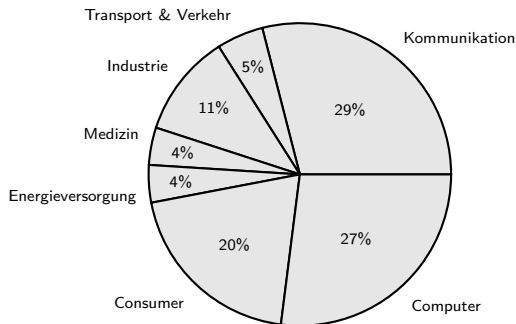


\* Quelle: BITKOM – Eingebettete Systeme – Ein strategisches Wachstumsfeld für Deutschland (2010)

# Märkte der Embedded Systems



## Anteile am Gesamterlös im Jahr 2011 (1.789 Mrd \$)\*



\* Quelle: IDC-Studie „Intelligent Systems: The Next Big Opportunity“ (2011)

# Anwendungsgebiete



## Hauptanwendungen im Segment Office Automation:

- Drucker (Laser, Tinte)
- Digitale Kopierer, Fax-Geräte, Dokumenten-Scanner
- Terminals (X, NCs, ThinClients)

## Hauptanwendungen im Consumer-Segment:

- Unterhaltung (CD-/MP3-Player), Spielekonsolen
- Digitale Audio- und Video-Systeme, Interaktives TV (Set-Top-Boxen)
- Haushaltsgeräte (z.B. Mikrowelle)
- Haus-Management (Smart Home) (z.B. Heizungssteuerung, Alarmanlage)

## Militärische Anwendungen:

- Große Investitionen, Förderung von Innovationen (ohne Wertung)

# Anwendungsgebiete (2)



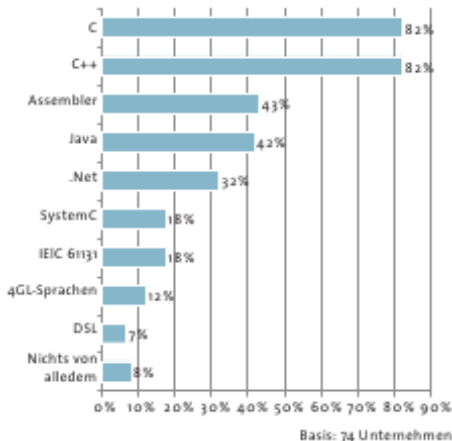
## Hauptanwendungen im Transport-Segment:

- Luftfahrt (Flugzeug, Air Traffic Control)
- Automobil-Elektronik
- Schienenfahrzeuge (ICE)
- Verkehrssteuerungen (Schienenverkehr, Ampelanlagen, Toll Collect)

## Hauptanwendungen im Kommunikations-Segment:

- Nebenstellenanlagen
- IP- und sonstige Telefone, digitale Anrufbeantworter
- Anruf-Verteilsysteme (Automatic Call Distribution)
- Voice Messaging
- Netzwerkgeräte zur Datenübertragung (z.B. Router)

# Arbeitsmarkt



Quelle: BITKOM Studie zur Bedeutung des Sektor Embedded-Systeme in Deutschland

- Embedded Markt in DE:  
> 19 Mrd € in 2010
  - ▶ 8% konstantes Wachstum
  - ▶ 85% des Umsatzes in EU
  - ▶ 69% des Umsatzes in DE
- Know-How wird in EU/DE gehalten
- Hardwarenahe Programmiersprachen gefordert

# Begriffe



## Echtzeitbetrieb (Def. nach DIN 44300, 1985):

- Ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig bereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.

⇒ korrektes Systemverhalten erfordert damit auch, dass zeitliche Vorgaben eingehalten werden.

## Deadline (Frist):

- Zeitpunkt, zu dem die Verarbeitungsergebnisse vorliegen müssen.

# Begriffe



## Echtzeitbetrieb (Def. nach DIN 44300, 1985):

- Ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig bereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.

⇒ korrektes Systemverhalten erfordert damit auch, dass zeitliche Vorgaben eingehalten werden.

## Deadline (Frist):

- Zeitpunkt, zu dem die Verarbeitungsergebnisse vorliegen müssen.



# Begriffe: Deutsch / Englisch



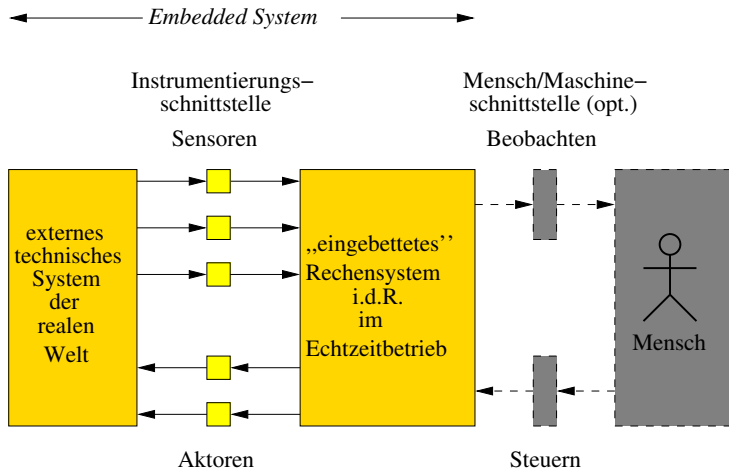
## Deutsch:

Echtzeitsystem  
Echtzeitbetriebssystem  
eingebettetes Rechensystem  
Frist  
Rechtzeitigkeit  
Ausführungsplanung, -plan  
Sensoren  
Aktoren  
Mensch/Maschine-Schnittstelle  
Steuerung

## Englisch:

- real-time system (RTS)  
- real-time operating system (RTOS)  
- embedded (computer) system  
- deadline  
- timeliness  
- scheduling, schedule  
- sensors  
- actuators, actors  
- man/machine (operator) interface  
- control

# Grundmodell eines Embedded Systems



# Spezialfälle



## Monitoring:

- nur Sensoren, keine Aktoren

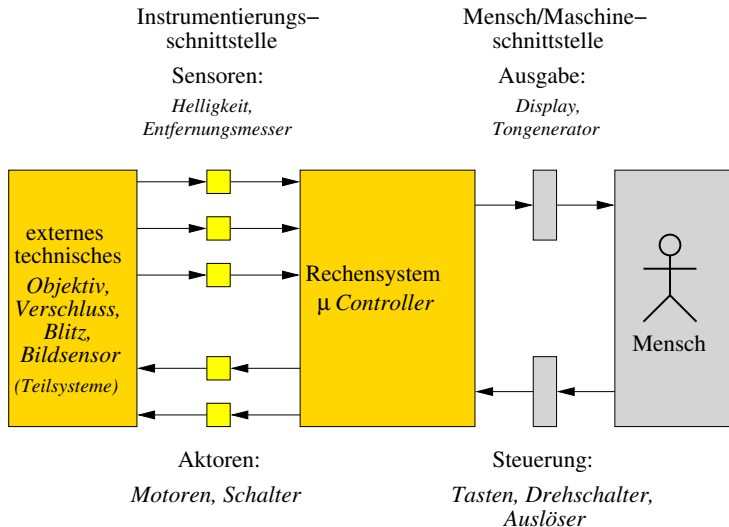
## Open Loop:

- nur Aktoren, keine Sensoren

## Feedback Control:

- Sensoren und Aktoren

# Beispiel: Digital gesteuerte Kamera



# Reaktives System



## Das Verhalten des Systems ist ereignisgetrieben.

- Auf jedes Ereignis (Stimulus) der externen Welt, das an den Sensoren sichtbar wird, erfolgt eine spezifische Aktion (Reaktion).

## Beispiele:

- Parkscheinautomat (reaktiv, kein Echtzeitbetrieb)
- Airbag (reaktiv, Echtzeitbetrieb)

# Zeitgetriebenes System



**Das Verhalten des Systems wird durch zeitliche Größen kontrolliert, z.B.**

- Absolute Zeitpunkte (der realen Außenzeit UTC)
- Mission Time (relativ zum Beginn der Mission)
- Zeitintervalle, -dauern
- Periode (wiederkehrende Aktion)

## Beispiele:

- öffentlicher Nahverkehr (zeitgetrieben, kein Echtzeitbetrieb)
- ABS (zeitgetrieben, Echtzeitbetrieb):
  - ▶ „Schaue alle 5 ms, ob Bremse getreten. Wenn ja, bewirke einen Bremsvorgang innerhalb von 20 ms“.
  - ▶ „Schaue alle 5 ms, ob Räder blockieren. Wenn ja, öffne die Bremse sofort für 100 ms“.
  - ▶ ...

# Systemlast



## Lastannahme:

- definiert die angenommene Spitzenbelastung (z.B. Anzahl Ereignisse je sec), die durch das externe System erzeugt wird.

## Achtung:

- Mittelwerte sind unbrauchbar!
- Statistische Argumente bzgl. einer geringen Wahrscheinlichkeit des Auftretens unabhängiger Ereignisse sind unzulässig.
- In kritischen Situationen treten häufig sehr viele, zeitlich eng korrelierte Ereignisse auf (→ „Ereignissturm“).

# Fehler



## Fehlerannahme:

- definiert Art und Anzahl der angenommenen Fehler sowie welche Funktionalität das System unter diesen Annahmen aufrecht erhält.

## Achtung:

- Das System muss im schlimmsten Fall (Worst Case) die maximale Anzahl von Fehlern bei Spitzenlast handhaben können.

## Geltungsbereich der Annahmen (Assumption Coverage):

- Wahrscheinlichkeit, dass die gemachten Annahmen mit der Wirklichkeit übereinstimmen.



# Vorhersagbarkeit



## Vorhersagbarkeit (Predictability):

- bedeutet, dass das Systemverhalten bei gegebenen
  - ▶ Lastannahmen und
  - ▶ Fehlerannahmen
- in Hinblick auf
  - ▶ Funktionalität
  - ▶ zeitliches Verhalten (Rechtzeitigkeit) und
  - ▶ Verlässlichkeit (Dependability)
- für den worst case eingehalten wird.  
→ Es wird damit "vorhersagbar".
- Garantiertes Systemverhalten muss nicht optimal sein.
- Häufig nimmt man z.B. eine etwas schlechtere Antwortzeit in Kauf, wenn man sicher ist, dass diese bestimmt nicht überschritten wird.

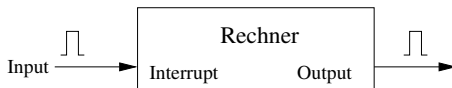
# Zeitverhalten



## Wesentlich: deterministisches Zeitverhalten

- zeitgesteuertes System: Zeitplan einhalten (kein „Jitter“)
- ereignisgesteuertes System: Antwortzeit einhalten

## Gedankenexperiment zur Antwortzeit:



T Konstant?

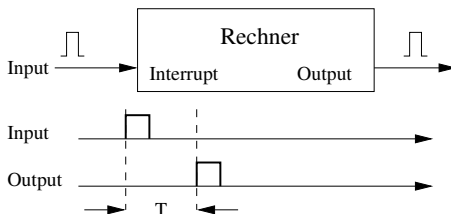
# Zeitverhalten



## Wesentlich: deterministisches Zeitverhalten

- zeitgesteuertes System: Zeitplan einhalten (kein „Jitter“)
- ereignisgesteuertes System: Antwortzeit einhalten

## Gedankenexperiment zur Antwortzeit:



## T Konstant?

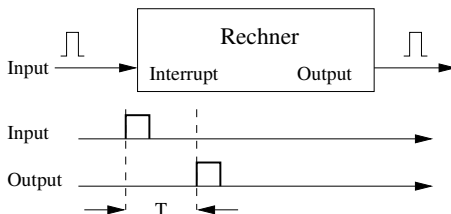
# Zeitverhalten



## Wesentlich: deterministisches Zeitverhalten

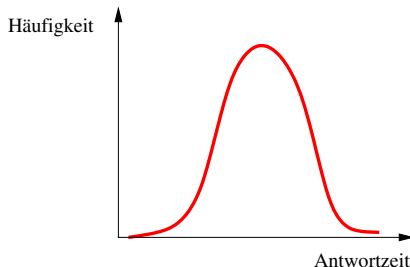
- zeitgesteuertes System: Zeitplan einhalten (kein „Jitter“)
- ereignisgesteuertes System: Antwortzeit einhalten

## Gedankenexperiment zur Antwortzeit:



**T Konstant? → Nein!**

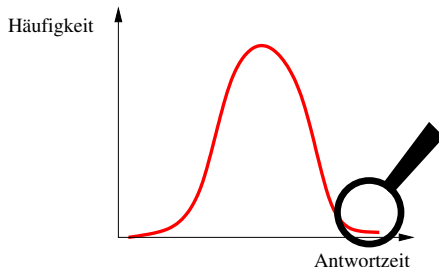
# Antwortzeit-Histogramm



## Wesentlich: deterministisches Zeitverhalten

- Gibt es eine maximale Antwortzeit, die niemals überschritten wird ?
- Ja → Echtzeitsystem

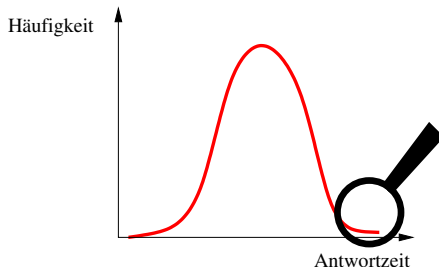
# Antwortzeit-Histogramm



## Wesentlich: deterministisches Zeitverhalten

- Gibt es eine maximale Antwortzeit, die niemals überschritten wird ?
- Ja → Echtzeitsystem

# Antwortzeit-Histogramm



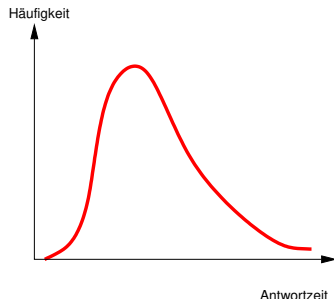
## Wesentlich: deterministisches Zeitverhalten

- Gibt es eine maximale Antwortzeit, die niemals überschritten wird ?
- Ja → Echtzeitsystem

# Optimierungsziel: Worst Case

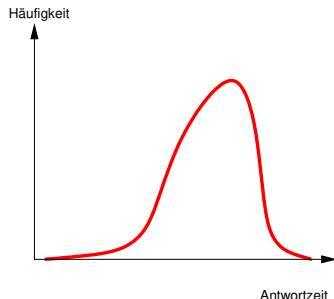


## Nicht-Echtzeitsystem



auf niedrige durchschnittliche  
Antwortzeit optimiert

## Echtzeitsystem



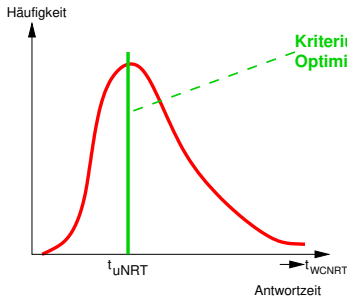
auf niedrige worst case Antwortzeit  
optimiert: evtl. höhere durchschnittliche  
Antwortzeit wird in Kauf genommen



# Optimierungsziel: Worst Case

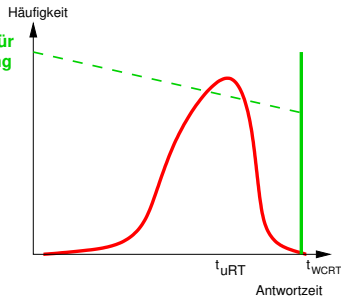


## Nicht-Echtzeitsystem



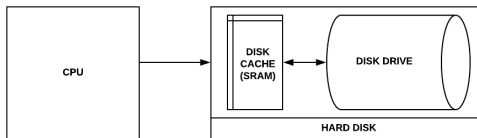
auf niedrige durchschnittliche  
Antwortzeit optimiert

## Echtzeitsystem



auf niedrige worst case Antwortzeit  
optimiert: evtl. höhere durchschnittliche  
Antwortzeit wird in Kauf genommen

# Optimierung auf worst-case: Beispiel



<https://medium.freecodecamp.org/the-hidden-components-of-web-caching-970854fe2c49>

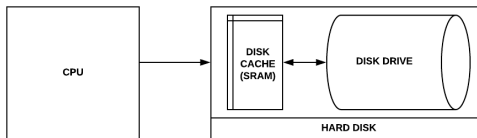
## Disk-Cache: RAM-Puffer hält häufig benötigte Daten

- + Verbesserung der mittleren Zugriffszeit
- Aber: Verschlechterung der worst case Zugriffszeit (wg. Cache-Verwaltungsoverhead)

→ Für EZ-System (zumindest) fragwürdig

- Entsprechende Design-Entscheidungen sind auch auf unterster (Betriebssystem-) Ebene erforderlich → „Echtzeitbetriebssystem“
- „Querschneidendes“ Thema (*cross-cutting concern*)

# Optimierung auf worst-case: Beispiel



<https://medium.freecodecamp.org/the-hidden-components-of-web-caching-970854fe2c49>

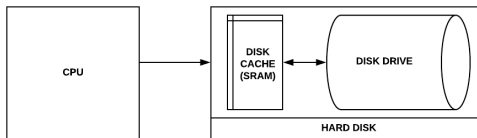
## Disk-Cache: RAM-Puffer hält häufig benötigte Daten

- + Verbesserung der mittleren Zugriffszeit
- Aber: Verschlechterung der worst case Zugriffszeit (wg. Cache-Verwaltungs-overhead)

→ Für EZ-System (zumindest) fragwürdig

- Entsprechende Design-Entscheidungen sind auch auf unterster (Betriebssystem-) Ebene erforderlich → „Echtzeitbetriebssystem“
- „Querschneidendes“ Thema (*cross-cutting concern*)

# Optimierung auf worst-case: Beispiel

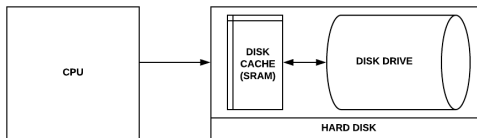


<https://medium.freecodecamp.org/the-hidden-components-of-web-caching-970854fe2c49>

## Disk-Cache: RAM-Puffer hält häufig benötigte Daten

- + Verbesserung der mittleren Zugriffszeit
  - Aber: Verschlechterung der worst case Zugriffszeit (wg. Cache-Verwaltungs-overhead)
- Für EZ-System (zumindest) fragwürdig
- Entsprechende Design-Entscheidungen sind auch auf unterster (Betriebssystem-) Ebene erforderlich → „Echtzeitbetriebssystem“
  - „Querschneidendes“ Thema (*cross-cutting concern*)

# Optimierung auf worst-case: Beispiel



<https://medium.freecodecamp.org/the-hidden-components-of-web-caching-970854fe2c49>

## Disk-Cache: RAM-Puffer hält häufig benötigte Daten

- + Verbesserung der mittleren Zugriffszeit
  - Aber: Verschlechterung der worst case Zugriffszeit (wg. Cache-Verwaltungsoverhead)
- Für EZ-System (zumindest) fragwürdig
- Entsprechende Design-Entscheidungen sind auch auf unterster (Betriebssystem-) Ebene erforderlich → „Echtzeitbetriebssystem“
  - „Querschneidendes“ Thema (*cross-cutting concern*)

# Charakterisierung



Ein Echtzeitsystem hat  
unter gegebenen Lastannahmen und  
Fehlerannahmen  
ein vorhersagbares Systemverhalten !

**Vorhersagbarkeit ist die Eigenschaft,  
die ein Echtzeitsystem von anderen Systemen  
unterscheidet.**

# Klassifizierung bzgl. Rechtzeitigkeit



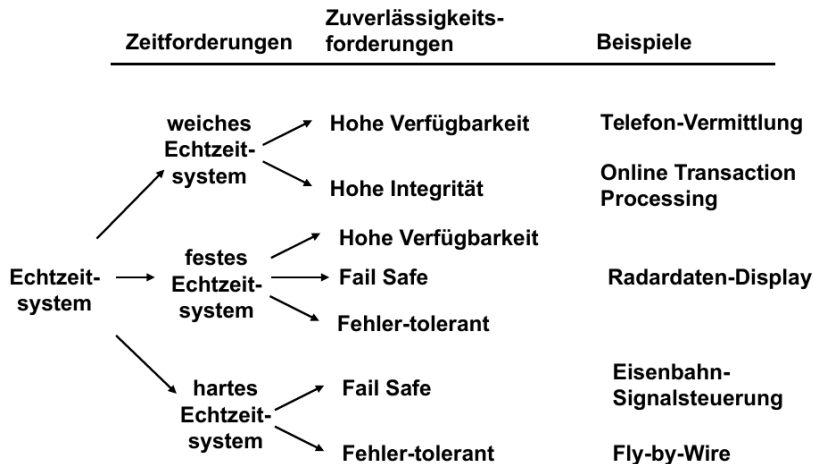
## Harte Echtzeitsysteme (Hard Real-Time Systems):

- Mindestens eine zeitliche Anforderung (Deadline) an das Systemverhalten muss immer und unter allen Last- und Fehlersituationen eingehalten werden (s.o.).
- Es werden „Garantien“ gegeben, die z.B. durch formale Beweise oder analytische Modelle belegt werden.
- Z.B.: garantierte Antwortzeiten zwischen Eingabe vom Sensor und reaktionsbedingter Ausgabe an den Aktoren.
- Jedes Verhalten außerhalb der Garantien wird als Systemversagen eingestuft.
- Unterklasse *Feste Echtzeitsysteme*: Versagen macht zunächst nur die aktuelle Operation wertlos, Versagen bei Wiederholung

## Weiche Echtzeitsysteme (Soft Real-Time Systems):

- Die zeitlichen Anforderungen werden in der Regel / statistisch eingehalten, gelegentliche Ausnahmen dürfen aber vorkommen.

# Beispiele

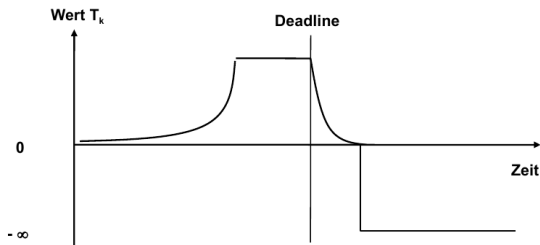




# Beispiele



- Modellierung von Echtzeitsystemen nach D. Jensen, 1990
- Auch *Value Functions* oder *Utility Functions* genannt
- Der Wert der Berechnung eines Ergebnisses eines Systems wird als Funktion der Zeit definiert.



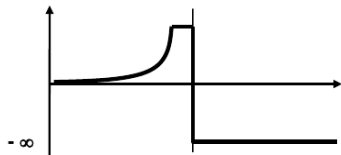
$T_k$ : Wert der k-ten Task

Wert der gesamten Berechnung:  $\sum_{k=1}^n T_k$

# Beispiele

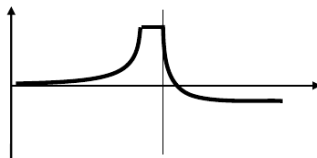


- Hard Real-Time: Verletzung der Zeitspezifikation kann katastrophale Folgen haben:



$$\sum_{k=1}^n T_k = -\infty$$

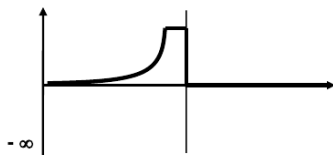
- Soft Real-Time: Zeitweise Verletzung der Zeitspezifikation kann mit gewissen Verlusten hingenommen werden:



# Beispiele



- Firm Real-Time: Verletzung der Zeitspezifikation macht Ergebnis wertlos



- Abgrenzung von Hard Real-Time:
  - ▶ Firm Real-Time-Systeme erzeugen keinen unmittelbaren (Personen-)Schaden bei Verletzung der Zeitschranke
  - ▶ Firm Real-Time in Literatur nicht immer betrachtet

# Fehleinschätzungen



## Common Misconceptions in RT-Computing (Stankovic, 1987):

- RT-Computing ist äquivalent zu „Fast Computing“.
- Fortschritte beim Supercomputing werden die RT-Probleme lösen.
- RT-Programmierung ist Assembler- und Gerätetreiber-Programmierung.
- RT-Systeme sind statische Systeme, angepasst an eine statische Umgebung.
- Die Probleme von RT-Systemen sind in anderen Gebieten der Informatik bereits behandelt und gelöst worden.

*„Necessary is a coherent treatment of correctness, timeliness, and fault-tolerance in large-scale distributed systems.“*

# Zukünftige Entwicklung



## Häufige Mängel heutiger Echtzeit-Systeme:

- teuer
- lange Lebensdauer
- ad-hoc-Methoden zur Validierung von Zeitbedingungen
- nicht skalierbar, keine inkrementellen Erweiterungen möglich
- schlechtes dynamisches und adaptives Verhalten

## Angestrebte Eigenschaften in den nächsten Generationen:

- komplex
- verteilt
- intelligente Sensoren und Aktoren
- adaptiv
- Verschärfung des Problems der Vorhersagbarkeit

→ **Responsive Systeme: echtzeitfähig, fehlertolerant, verteilt**