Phase 1: Anwendungsanalyse

Echtzeitsysteme 2 - Vorlesung/Übung

Fabian Scheler
Michael Stilkerich
Wolfgang Schröder-Preikschat

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

http://www4.cs.fau.de/~{scheler,mike,wosch}
{scheler,mike,wosch}@cs.fau.de

Übersicht

- Zielsetzung
- Problematik
- Anforderungen & Fakten

Zielsetzung

- physikalisches Objekt
  - Welche Größen sind relevant?
  - Wie hängen diese Größen zusammen?
- Echtzeitsystem
  - Welche Ereignisse gilt es zu behandeln?
  - Welche Zeitschranken gilt es einzuhalten?
  - Welche Beziehung Zeitschranke ↔ physikalischen Objekt gibt es?
- Wie sieht das physikalische Model aus?
  - Welche Größen des physikalischen Models muss man abbilden?
  - Wie bildet man diese Größen ab?

Problematik

- selbst einfach erscheinende Objekte sind aus physikalischer Sicht äußerst komplex

➔ Vereinfachungen sind unabdingbar

- Beispiel: Hau den Lukas

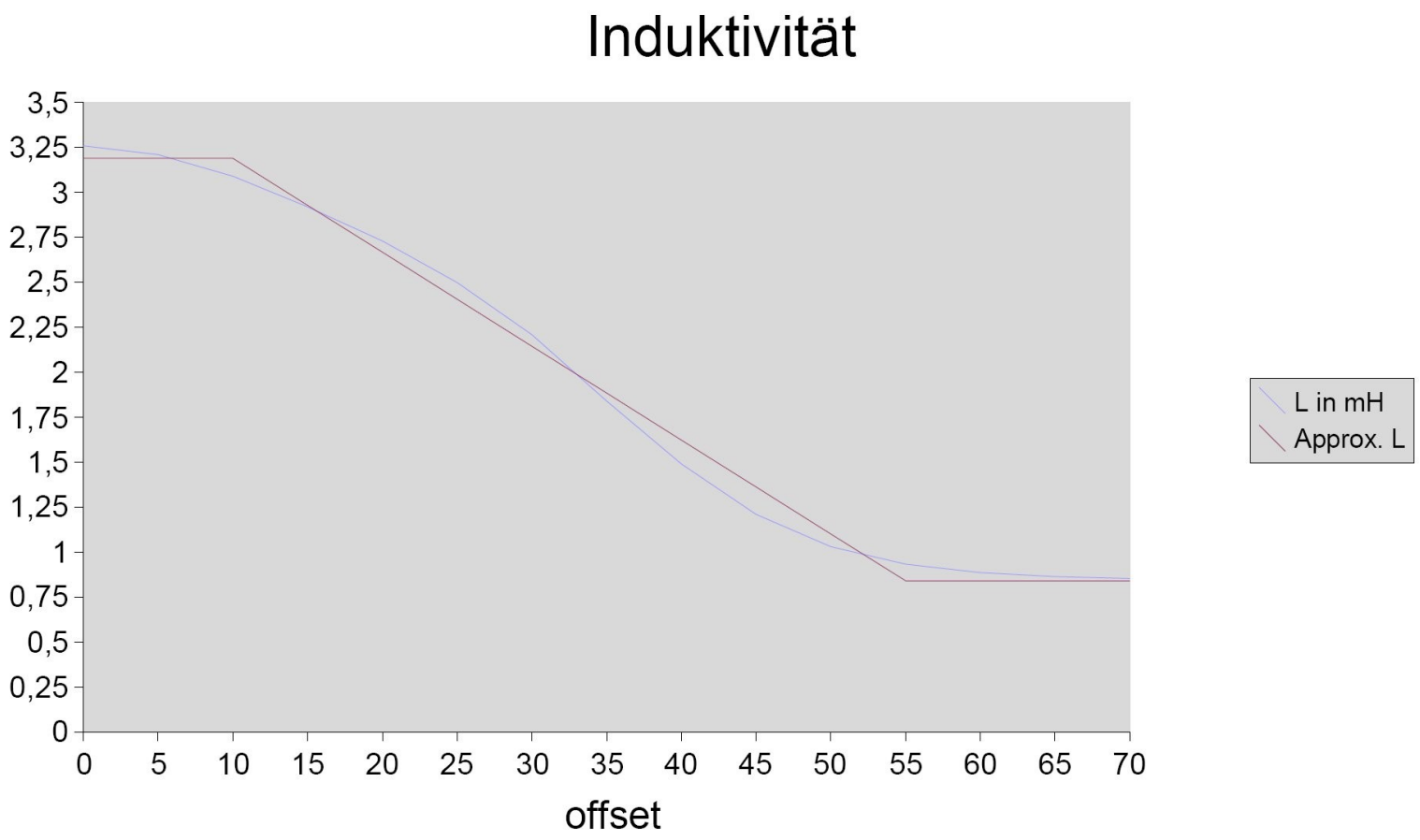
Problematik: Hau den Lukas

- Schaltvorgänge an den Spulen sind nicht konstant!
  - Ein- und Ausschaltvorgänge und die Induktionsgesetze
- Induktivität der Spule ist nicht konstant!
  - Wo ist der Eisenkern in der Spule?
- Kraft auf den Eisenkern in der Spule ist nicht konstant!
  - Wo ist der Eisenkern in der Spule?
  - Welche Geschwindigkeit hat der Eisenkern?
- Beschleunigung des Eisenkerns ist nicht konstant!
  - Kraft auf den Eisenkern ist nicht konstant

→ komplexe, zeitabhängige Vorgänge
  - macht Integration notwendig
  - häufig nicht mehr analytisch darstellbar
  - numerische Lösung werden erforderlich

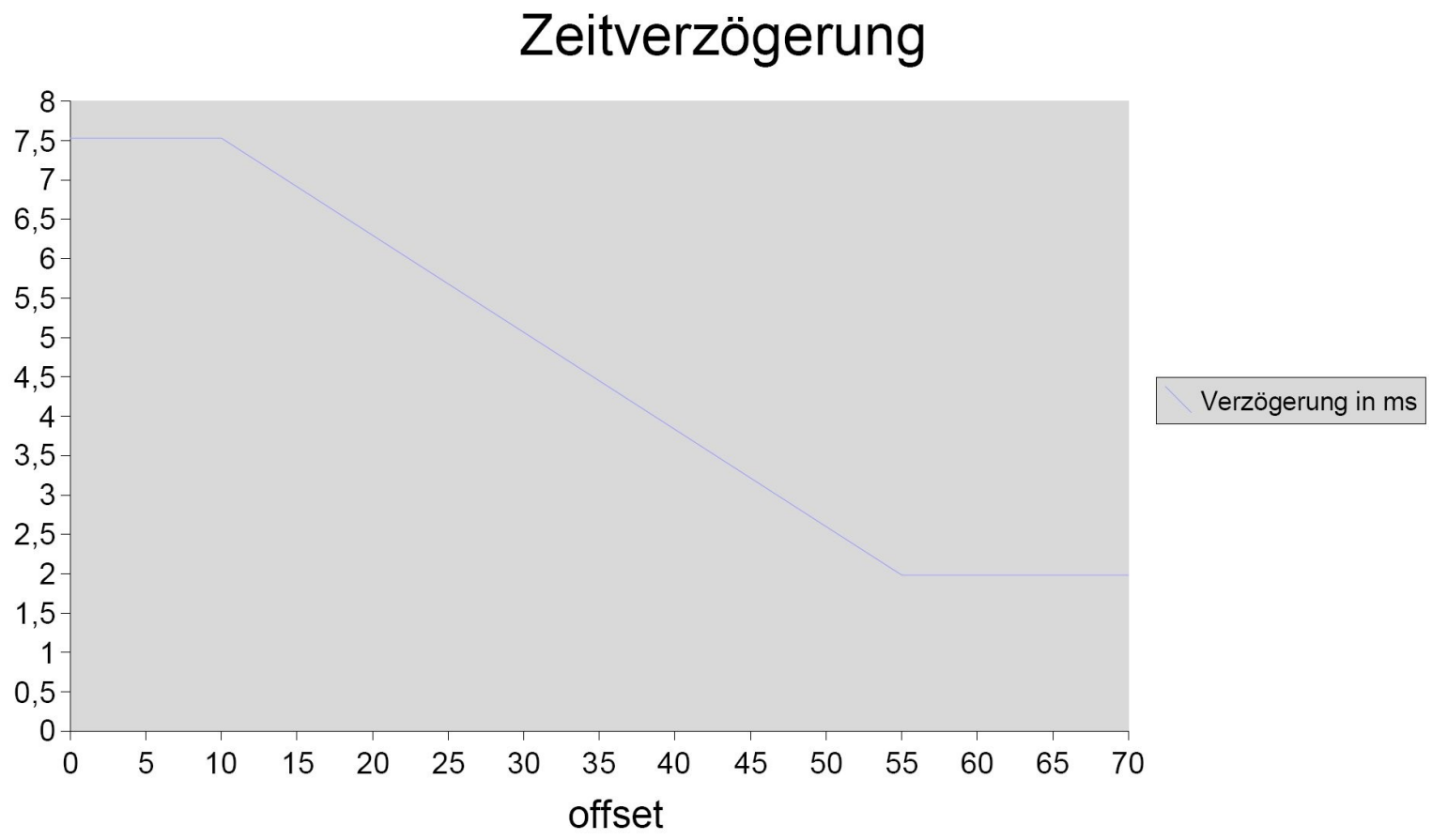
Ergebnisse: Hau den Lukas (SS 2006)

- Induktivität der Spule
  - in Abhängigkeit der Position des Eisenkerns



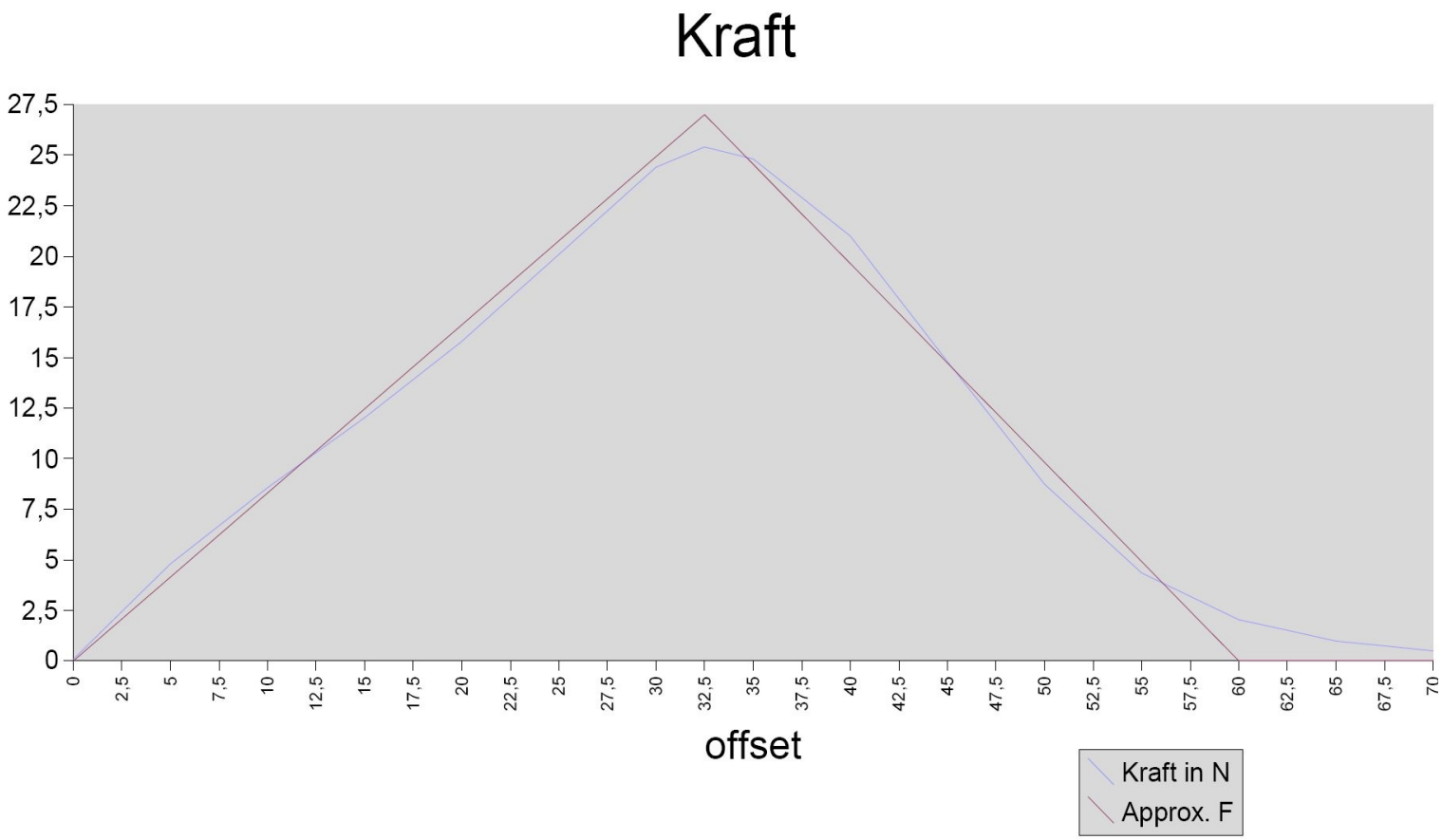
Ergebnisse: Hau den Lukas (SS 2006)

- Schaltverzögerung der Spule
  - in Abhängigkeit der Position des Eisenkerns



Ergebnisse: Hau den Lukas (SS 2006)

- Kraftwirkung der Spule auf den Eisenkern
  - in Abhängigkeit der Position des Eisenkerns



Ergebnisse: Hau den Lukas (SS 2006)

- selbst das Modell ist bereits stark vereinfacht
- verschiedene Größen werden nicht berücksichtigt
  - Gegeninduktion
  - Wirbelströme
  - Luftwiderstand
- gewisse Randparameter müssen experimentell bestimmt werden

Anforderungen & Fakten

- Hau den Lukas
- Eisenbahn
- Robertino

Anforderungen: Hau den Lukas

- Programmierbarkeit
  - ein Programm wird durch eine Datenstruktur beschrieben, die durch Prozeduraufrufe erzeugt/initialisiert wird

```
lift_continous(my_prog,7);
fall_stepwise(my_prog,2);
lift_continous(my_prog,4);
...
```

  - schließlich wird dieses Programm ausgeführt

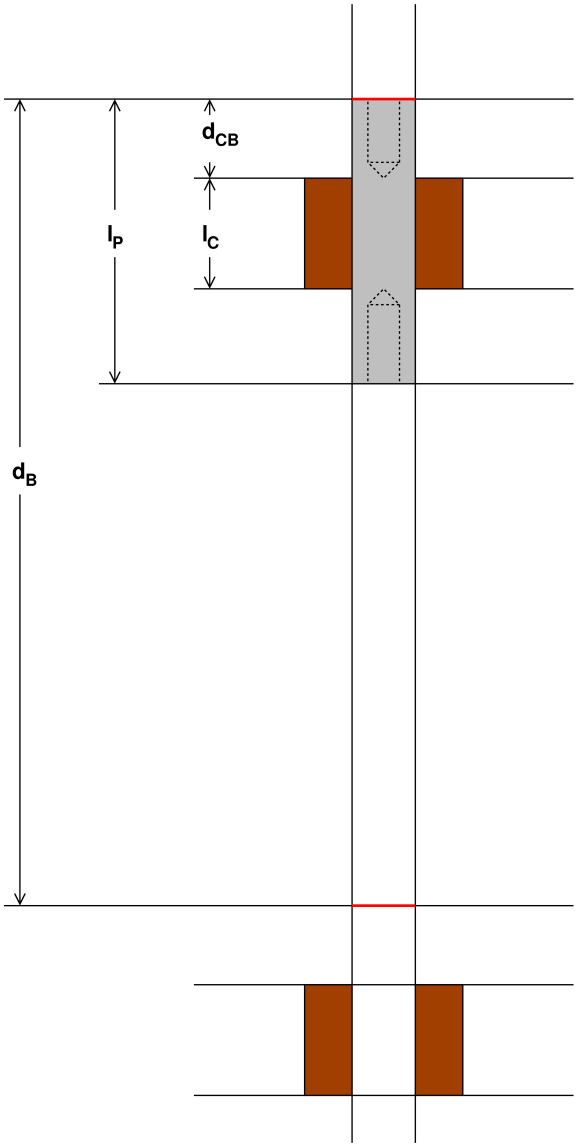
```
start_program(my_prog);
```

- folgende Primitive sollen beherrscht werden
  - schrittweise anheben/fallen lassen
  - kontinuierlich anheben/fallen lassen
- Primitive sind kombinierbar

Anforderungen: Hau den Lukas

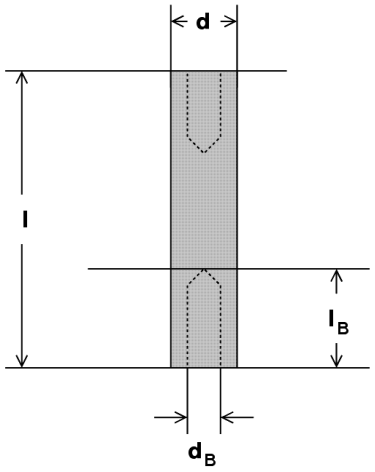
- physikalisches Modell
  - soll Geschwindigkeit des Eisenkerns berücksichtigen
    - → Schaltzeitpunkte der Spulen bestimmen
  - soll die Beschleunigung/Verzögerung durch die Spulen annähern
    - → Rückschlüsse auf die Änderung der Geschwindigkeit
- Benutzerschnittstelle
  - Programme werden unveränderlich gespeichert
  - Programme werden geladen [optional]
    - → Interaktion über die serielle Schnittstelle
- verteiltes System [optional]
  - zusammen mit der Eisenbahnsteuerung
  - Verteilung auf zwei Knoten
  - Kommunikation über TTCAN

Fakten: Hau den Lukas

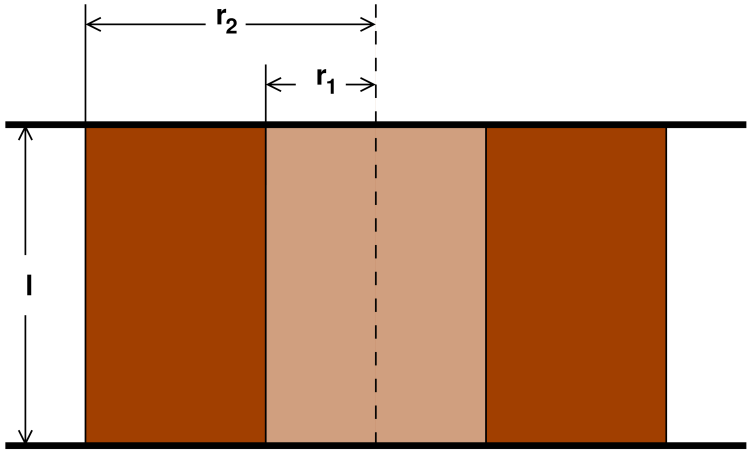


l_C	Länge: Spule	30 mm
l_P	Länge: Projektil	82 mm
d_B	Abstand: Lichtschranken	230 mm
d_{CB}	Abstand:Spule – Lichtschranke	24 mm

Fakten: Hau den Lukas



l	Länge: Projektil	82 mm
l_B	Länge: Bohrung	26 mm
d	Durchmesser: Projektil	18,6 mm
d_B	Durchmesser: Bohrung	10 mm
m	Gewicht	144 g
	Material: Eisen	



l	effektive Länge	23 mm
r_1	Innendurchmesse	28 mm
r_2	Außendurchmesse	mm
l	Induktivität	0,82 mH
R	Widerstand	0,52 Ohm
N	Windungszahl	178
	Toleranz	3,00%

Fakten: Hau den Lukas

- Ansteuerung der Spulen
  - Pin 0.5 – Pin 0.12 (x.y: x – Port, y – Pin)
- Abfrage der Lichtschranken
  - Pin 7.0 – Pin 7.7
- Unterbrechungen der Lichtschranken
  - Pin 1.0
- Dauer von Auslösung der Lichtschranke bis zur ISR
  - → vernachlässigbar

Anforderungen: Eisenbahn

- Steuerung mehrerer Züge
  - Züge können angehalten werden
  - Züge können weg genommen werden (~ in Parkstellung)
  - Züge können dazu kommen (~ aus der Parkstellung)
- Veränderliches Streckennetz
  - Gleisstrecken können stillgelegt werden
  - Gleisstrecken können wieder in Betrieb genommen werde
  - keine Partitionierung des Schienennetzes
- Programmierbarkeit
  - Fahrpläne werden unveränderlich gespeichert
  - Fahrpläne werden geladen [optional]
    - → Interaktion über die serielle Schnittstelle

Anforderungen: Eisenbahn

- dynamische Routenberechnung
  - Fahrpläne sind statisch, Routen werden zur Laufzeit erstellt
  - Optimierung der Routen [optional]
- Ankoppeln und Abkoppeln von Waggons [optional]
- verteiltes System [optional]
  - zusammen mit dem „Hau den Lukas“
  - Verteilung auf zwei Knoten
  - Kommunikation über TTCAN

Fakten: Eisenbahn

- Fülle von Dokumentation
  - Ort: `/proj/i4ezs/experimente/Eisenbahn`

➔ Problematik
  - Was steht in der Dokumentation, was nicht?
  - Was ist überhaupt relevant?
  - Sichtung der Daten ist nicht trivial!

Fakten: Eisenbahn

- relevant sind unter anderem
  - Gleislängen
    - bestehend aus mehreren Gleisstücken
    - entsprechende Daten z.B. bei `www.trix.de`
  - Geschwindigkeiten der Loks
    - maximale Geschwindigkeit ist gedeckelt
    - Bestimmung (= Messung) der maximalen Geschwindigkeit
    - Unterstützung mehrerer Geschwindigkeitsstufen
  - Latenzen
    - Besetztmelder, Schranken, Weichen, ...
  - Trix-Bus
    - Übertragungsgeschwindigkeit, ...
  - ...

Fakten: Eisenbahn

- manche Größen kann man sicherlich vernachlässigen
  - Verhältnis zu anderen Größen
  - können diese Größen akkumuliert werden

- ➔ Hauptaufgabe
  - relevante Größen finden
  - wie Verhalten sich diese Größen zu einander
  - was sind bei der Steuerung die zeitkritischen Teile

Anforderungen: Robertino

- Bewegung des Robertino in der Ebene
  - Erkennung vorausliegender Hindernisse
  - Taumelnde Vorwärtsbewegung zur Erhöhung des Sichtfeldes
  - Kein festes Ziel
- Umgebung verändert sich
  - Hindernisse bewegen sich (Menschen, Hunde, ...)
  - Hindernisse werden hinzugefügt oder entfernt
- Festfahren des Robertino erkennen
  - Schmale Hindernisse (z.B. Stuhlbeine) werden u.U. nicht erkannt
  - Der Robertino fährt sich an diesen Hindernissen fest
  - Erkennung: Angestrebte stark über der tatsächlichen Motorgeschwindigkeit
  - In diesem Fall soll der Robertino zurückfahren und die Richtung ändern

Anforderungen: Robertino

- Programmierung
  - Implementierung in Java aufbauend auf einem KESO System
  - Knotenkommunikation durch den KESO Portalmechanismus
- Implementierung eines KESO CAN Treibers für den TC1796b

Fakten: Robertino

- Verteiltes System
  - Bestehend aus einem Tricore 1796b Controller und 3 mDSA
  - Kommunikation über einen CAN Bus
- Micro Digital Servo Amplifier (mDSA)
  - Atmel Atmega 8535 Controller
  - Intel 82527 Serial Communications Controller (CAN Bus)
  - H-Bridge zur Steuerung des Motors
- Fortbewegung
  - 3 Räder getrieben durch je einen Elektromotor
  - Jeweils ein Elektromotor ist an einen mDSA angebunden

Fakten: Robertino

- Erkennung der Umgebung
  - 6 Infrarotsensoren zur Entfernungsmessung
  - Jeweils 2 Sensoren an einen mDSA angebunden
  - Sichtfeld stark eingeschränkt
- Abweichung vom original Robertino
  - keine VGA Kamera
  - PC durch Tricore 1796b Controller ersetzt

Folgerung

- scheinbare Vorgänge sind physikalisch sehr komplex
- komplexe physikalische Vorgänge sind mathematisch häufig nicht analytisch lösbar
  - Vereinfachungen des Modells
  - numerische Lösungsansätze
- Massive Vereinfachungen sind notwendig
  - Was muss man wirklich wissen?
  - Was kann man wissen/messen?
  - Welche Einschränkungen sind damit verbunden?
- Designer des Echtzeitsystems muss sich mit dem physikalischen Objekt vertraut machen