Von der Theorie zur Praxis: Echtzeitplanung in der Informatikausbildung

Fachtagung des GI/GMA/ITG-Fachausschuss Echtzeitsysteme 2016

Andreas Stahlhofen, Dawid Bijak, Dieter Zöbel

Institut für Softwaretechnik, Arbeitsgruppe Echtzeitsysteme

astahlhofen@uni-koblenz.de
dbijak@uni-koblenz.de
zoebel@uni-koblenz.de

17. November 2016

Agenda

Motivation

MARTOP (Mapping Real-Time to POSIX)

Anwendungsbeispiel

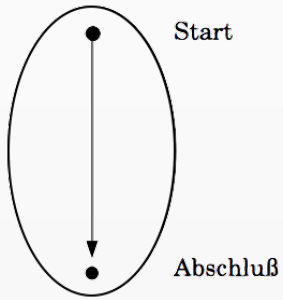
Fazit und Ausblick

Motivation

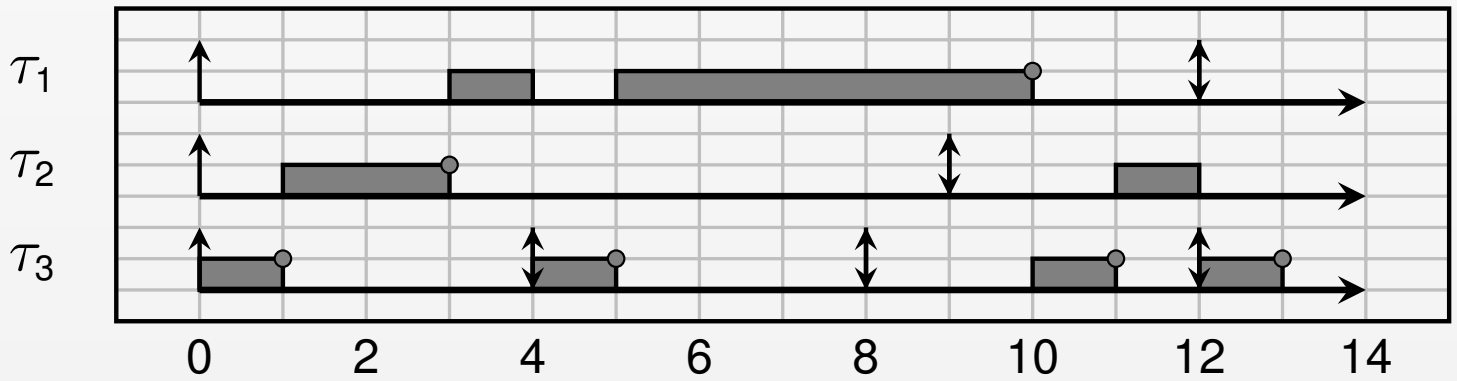
- Echtzeitsysteme bilden oftmals einen Themenbereich im Studiengang Informatik
- Durch Alltagsbezug ergibt sich die Lehrberechtigung
- Beispiele finden sich in folgenden Bereichen:
  - Automotiven Bereich
  - Mobile Kommunikation
  - Anwendungsentwicklung für Smartphones
  - ...

Motivation

- Fundamentale Teilgebiete sind insbesondere das **Taskmodell** und die Theorie der **Echtzeitplanung**
- Ausgehend vom allgemein bekannten Taskmodell ...



- ... bis hin zur Verplanung einer gegebenen Menge periodischer Tasks



Motivation

- Neben der Theorie spielen auch praktische Kompetenzen eine Rolle
- ... denn die richtigen Probleme tauchen erst im Rahmen der praktischen Umsetzung auf
- Möglichkeiten fehlen zum Abbilden der Modelle auf Ebene der Implementierung

Problematik

Studierende haben zu wenig Erfahrung im Bereich paralleler Programmierung und insbesondere mit low-level Konstrukten wie z.B. Threads, Systemaufrufen oder Timern.

Motivation

Unser Lösungsansatz

Entwurf einer objektorientierten Softwarebibliothek in der Programmiersprache C++.

Folgende **Anforderungen** wurden erhoben:

Anforderung 1: Abbilden der Modelle und Notation auf Programmierebene

Anforderung 2: Hohes Abstraktionsniveau

Anforderung 3: Erweiterbarkeit

Anforderung 4: Adressierung relevanter Plattformen (z.B. Raspberry Pi)

Agenda

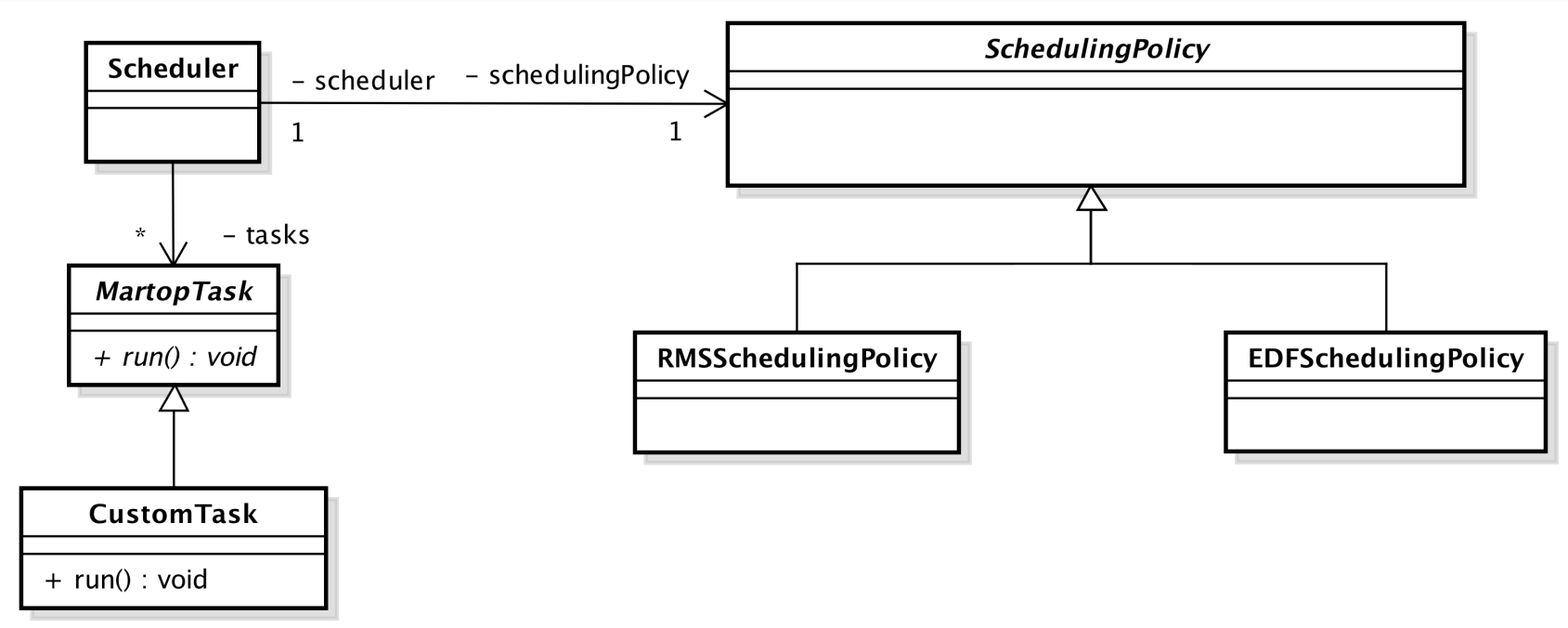
Motivation

MARTOP (Mapping Real-Time to POSIX)

Anwendungsbeispiel

Fazit und Ausblick

Anforderung 1: Abbilden der Modelle und Notationen



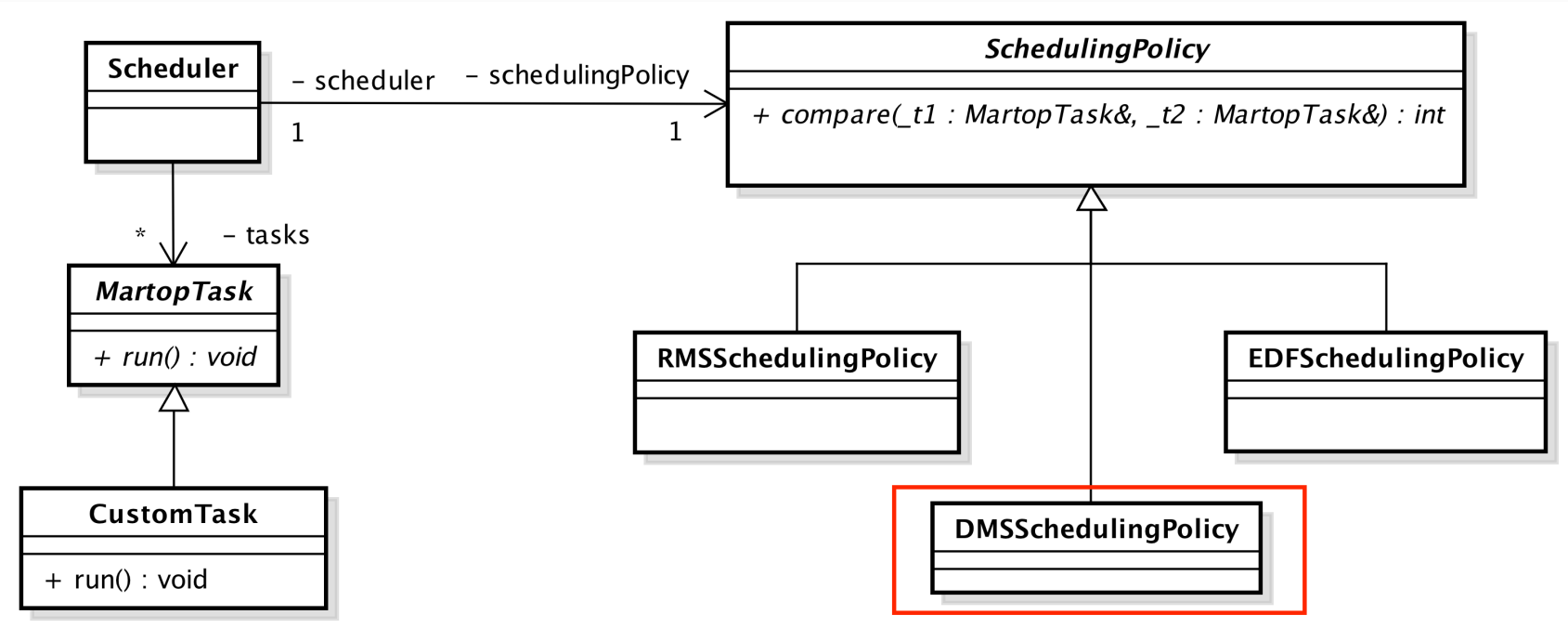
MARTOP Klassendiagramm

Anforderung 2: Hohes Abstraktionsniveau

MARTOP Code-Beispiel:

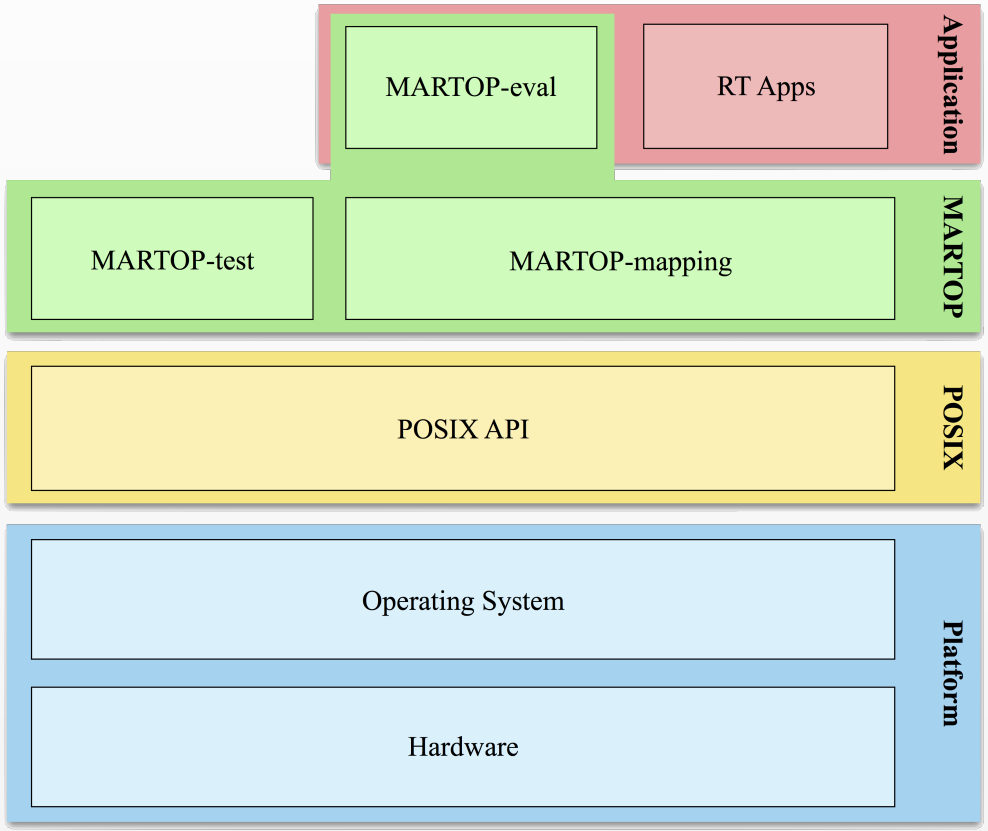
```
1  /* Create two tasks t1 and t2. */
2  MartopTask *t1 = new CustomTask(std::chrono::milli(4),
       std::chrono::milli(10));
3  MartopTask *t2 = new CustomTask(std::chrono::milli(2),
       std::chrono::milli(7));
4
5  /* Choose an according scheduling policy. */
6  EdfSchedulingPolicy edfSchedulingPolicy;
7
8  /* Initialize the scheduler and add the tasks to
       schedule. */
9  Scheduler scheduler(edfSchedulingPolicy);
10 scheduler.add(t1);
11 scheduler.add(t2);
12
13 /* Start the scheduler. */
14 scheduler.start();
```

Anforderung 3: Erweiterbarkeit



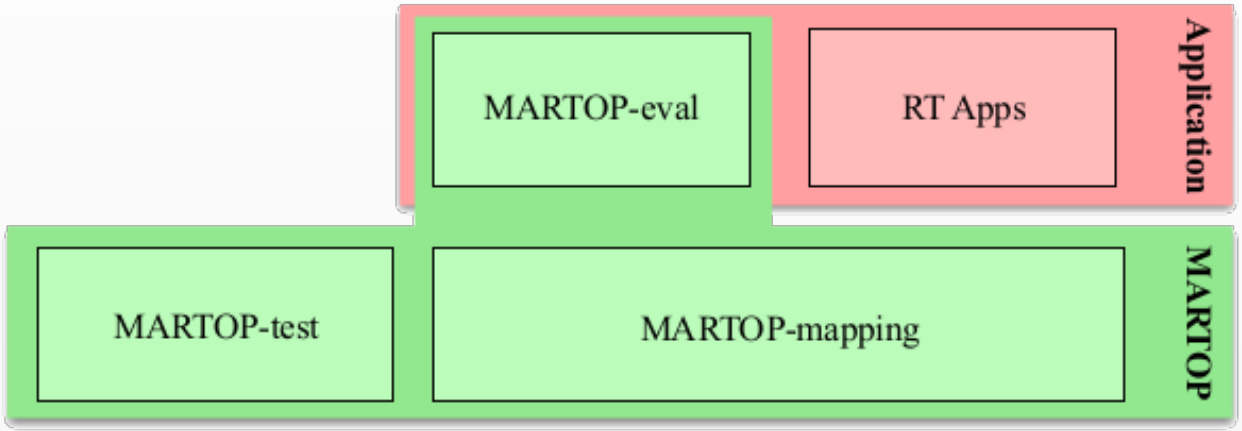
MARTOP Klassendiagramm

Anforderung 4: Plattformkompatibilität



MARTOP Softwarearchitektur als Schichtenmodell.

Anforderung 4: Plattformkompatibilität

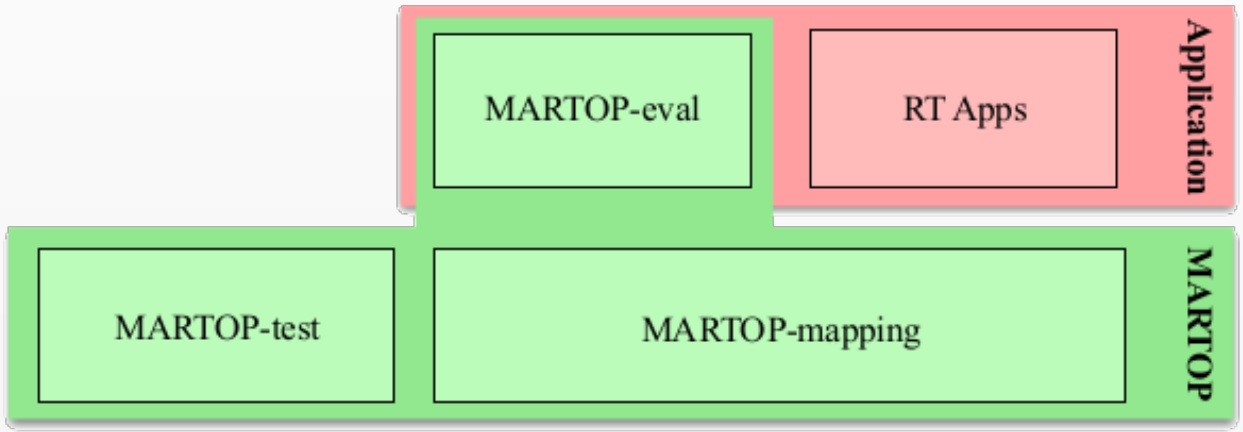


MARTOP-test Testet Plattform bezüglich POSIX-Kompatibilität

Problematik: Es werden 3 POSIX Kompatibilitätslevel unterschieden:

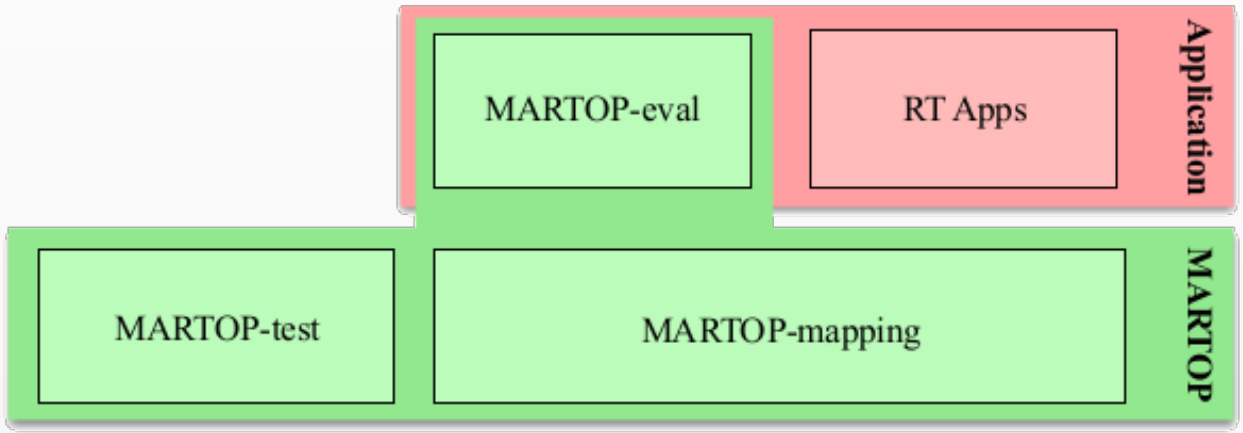
1. POSIX kompatibel
2. POSIX konform
3. POSIX zertifiziert

Anforderung 4: Plattformkompatibilität



MARTOP-mapping Implementierung der Programmierschnittstelle

Anforderung 4: Plattformkompatibilität



MARTOP-eval Scheduled zufällig generierte Menge von Tasks unter Verwendung eines gewählten Planungsverfahren und validiert die Korrektheit dieser Ausführung

Agenda

Motivation

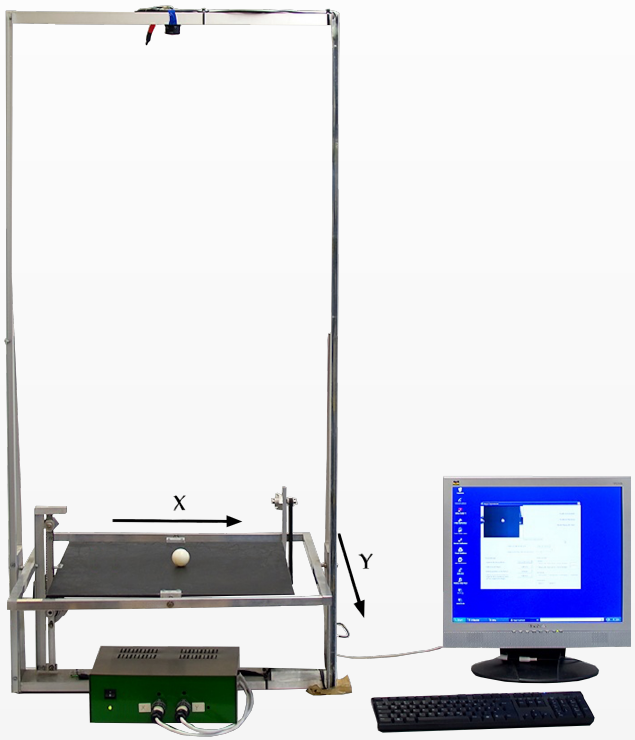
MARTOP (Mapping Real-Time to POSIX)

Anwendungsbeispiel

Fazit und Ausblick

Wippe-Experiment

- **Ziel:** Balancieren eines Balls
- Webcam dient zur Positionsbestimmung
- Ebene beweglich mittels Schrittmotoren
- Schrittmotorcontroller erlaubt Ansteuerung über RS232
- Computer dient als Rechensystem und realisiert Regelalgorithmus



Versuchsaufbau des Wippe-Experiments

Wippe-Experiment

Vorgehensweise bei Implementierung mittels MARTOP:

1. Identifikation der einzelnen Tasks
2. Auswahl eines passenden Planungsverfahrens
3. Implementierung der Tasks
4. Bestimmung der Taskparameter (inkl. *wcet*)
5. Implementierung der Hauptroutine mit Hilfe von MARTOP

Wippe-Experiment

Identifikation der Tasks und Bestimmen der Taskparameter:

1. *Aufnahme* des Bildes mittels Webcam
2. *Berechnungskomponente*, untergliedert in folgende Teilaufgaben:
   - 2.1 Bildverarbeitungsalgorithmus zur Bestimmung der Ballposition
   - 2.2 Berechnung der Stellgröße mittels PID-Reglers
   - 2.3 Senden der abgeleiteten Positionen an die Schrittmotoren
3. Senden von *Diagnosedaten* über eine asynchrone UDP-Schnittstelle

Task τ_i	C_i	T_i
τ_1	17.7 ms	35 ms
τ_2	33.6 ms	72 ms
τ_3	4.3 ms	250 ms

Planungsverfahren: EDF

$U \approx 0,99$

Testergebnisse

- Ausführung auf Desktop-Rechner mit Intel-Core 2 Duo, 2,6 GHz, 2 GB DDR2-SDRAM
- Betriebssystem Linux 16.04 Ubuntu inklusive installiertem *preemptive patch*
- Messungen ergaben folgende Werte:
  - Gemessener *Scheduling-Overhead*: Ø = 61 μs, *max* = 70 μs
  - Start-Verzögerung eines Tasks: Ø = 308 μs, *max* = 421 μs
  - Auslastung von nahezu 99% zu optimistisch, denn Fristverletzungsrate bei ca. 5%
  - Ab Auslastung von 91% sind keine Fristverletzungen mehr aufgetreten

Agenda

Motivation

MARTOP (Mapping Real-Time to POSIX)

Anwendungsbeispiel

Fazit und Ausblick

Fazit

MARTOP bietet ...

- ... **Selbsttest** auf gegebener Plattform
- ... eine **intuitive** Programmierschnittstelle
- ... hohes Maß an **Portabilität**
- ... Möglichkeit zur **Erweiterbarkeit**

Ausblick

- Tests und Evaluation
- Unterstützung von Mehrkern-Prozessoren
- Implementierung weiterer Planungsverfahren

Vielen Dank für die Aufmerksamkeit!

Fragen?