

Projektvorschlag: Steuerung des TeachRobot

Benjamin Lucke (CEE2)
benjamin.lucke@hs-furtwangen.de

Christian Weichel (AIB3)
christian.weichel@hs-furtwangen.de

8. Dezember 2009

1 Übersicht

Das Ziel des vorgeschlagenen Projekts ist die Konzeptionierung und Implementierung eines Hardware/Software Systems zur Steuerung des TeachRobot. Dies beinhaltet:

- Entwurf und Bau eines Hardware-Interfaces für die PC-TeachRobot Kommunikation (Ersatz für vorhandene ISA-Hardware)
- Entwurf und Implementierung einer rechnerseitigen API zur Steuerung des TeachRobot
- Entwurf und Implementierung eines TeachRobot-Simulators welcher mit obiger API programmiert werden kann
- Implementierung einer Beispiel-Anwendung mit den obigen Komponenten

2 Konkretisierung

2.1 Hardware-Interface

Das Hardware-Interface ist die Schnittstelle zwischen Computer und TeachRobot. Sie besteht aus zwei Komponenten:

1. **Treiber-Platine:** beheimatet die Treiber-ICs für die Motoren und deren Inkrementalgeber.
2. **Steuereinheit:** nimmt die Befehle des Computers über eine USB-Serielle Verbindung entgegen und setzt sie in Signale für die Treiber-Platine um. Ihre Implementierung erfolgt mit einem *Luminary Micro LM3S6965 (rev C) Evaluation Board*. Die Architektur der Steuereinheit ist Bild 2.1 zu entnehmen.

Das *Hardware-Interface* rechnet in einem absoluten Koordinatensystem, welches initialisiert werden muss und relativ zur Nullstellung arbeitet.

2.2 Rechnerseitige API

Definition 1 (Zustand). Es sei $Z = [0; 2\pi]^6$ ein Zustand des Roboterarms, wobei die Elemente von Z die Winkel der Achsen darstellen.

Die API dient zweierlei Dingen: der Abfrage und Veränderung des Zustandes. Für die Veränderung stehen vier Möglichkeiten zur Verfügung:



Abbildung 1: Architektur der Steuereinheit

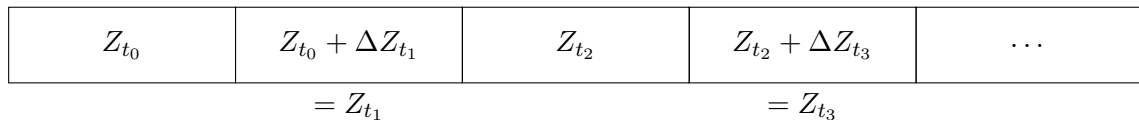


Abbildung 2: Die Zustandsqueue

- **Zeitdiskret** Die Gegenseite der API nimmt Zustände (oder Delta's zum letzten Zustand) entgegen, welche mit einem diskreten Zeitindex t_i versehen werden. Der Roboterarm strebt nun nacheinander diese Sollzustände an. Siehe Abb. 2.2
- **absoluter Sollzustand** (spez. vom Zeitdiskreten mit $i = 0$). Es gibt genau einen Sollzustand zu welchem der Roboterarm strebt. Dieser kann sowohl absolut als auch Relativ verändert werden.

Ferner unterstützt die API folgende Funktionen:

- **init** Initialisiert den Roboterarm
- **firstEndPosition** Fährt den Roboterarm auf die erste Endschalterposition
- **secondEndPosition** Fährt den Roboterarm auf die zweite Endschalterposition

Die API unterscheidet nicht ob gegen den Roboterarm oder einen Simulator programmiert wird. Ausserdem ist sie Sprach-Unabhängig (d.h. nicht festgelegt auf Java, C, etc.).

2.3 Simulator

Die Parameter des zu simulierenden Roboterarmes, lassen sich über eine Konfigurationsdatei bestimmen. Der Simulator lässt sich über die oben beschriebene API steuern und verhält sich wie der "physikalische" Roboterarm. Eine Kollisionserkennung, sowie ein physikalisches Modell sind nicht vorgesehen.

2.4 Anwendungsbeispiele

Es sind zwei Anwendungsbeispiele zu entwickeln. Zum einen ein Test-Client welcher als Frontend für die API dient und eine manuelle Steuerung des Armes ermöglicht. Zum anderen ein Programm, dass spez. Muster welche vor eine Kamera gehoben werden lokalisiert und den Roboterarm in diese Richtung greifen lässt.