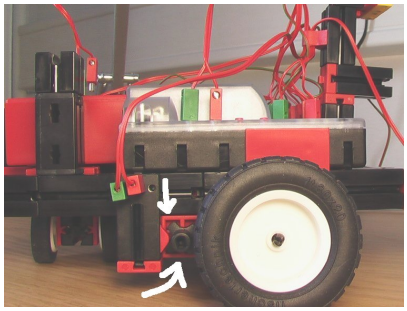
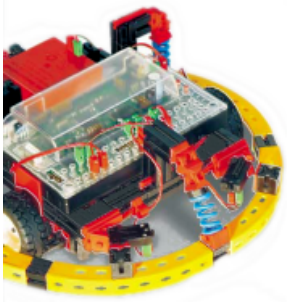
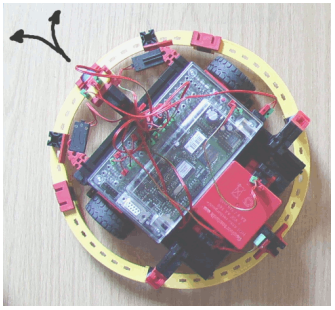
Technical Report RoboPrAnleitung

Versuchsanleitung
Praktikum Robotik & visuelle Sensorik

Holger Friedrich
Andreas Fürtig
Tobias Weis

email: holger.friedrich@vsi.cs.uni-frankfurt.de

tobias.weis@vsi.cs.uni-frankfurt.de

andreas.fuertig@vsi.cs.uni-frankfurt.de

Sommersemester 2011

joint robotics lab

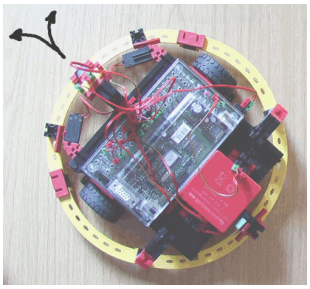
GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Visual Sensorics and Information Processing, Institute for Computer Science
Johann Wolfgang Goethe-Universität Frankfurt am Main, Germany
http://www.vsi.cs.uni-frankfurt.de

Kapitel 1

Einführungsversuch: Motorregelung

Material: Roboter, Widerstand zum Abbremsen
Literatur: [Brä03, N.04, Fri07]



1.1 Motorregelung

Gleichstrommotoren gleicher Bauart können – je nach Herstellungstoleranzen und je nach aufzuwendender Kraft – bei gleicher Ansteuerung durchaus unterschiedliche Geschwindigkeiten liefern. Deshalb ist die Geradeausfahrt mit einem Roboter, den zwei Räder mit verschiedenen Motoren antreiben, durchaus ein Problem. Der Roboter hat deshalb Sensoren, die die Raddrehung messen. Mit dieser Information kann eine Rückkopplung erfolgen. Mit Hilfe einer *Regelung* der Steuergröße (eingestellte Potentialdifferenz) kann nun die gewünschte Drehzahl eingestellt, bzw. korrigiert werden.

In diesem Versuch soll eine Regelung entwickelt werden, die eine Geradeausfahrt des Roboters selbst dann noch ermöglicht, wenn einer der beiden Motoren stark gedrosselt wird (z.B. durch einen ohmschen Widerstand).

Für diesen Versuch sind zwei Teilprobleme zu lösen:

- Geschwindigkeitsmessung
- Regelung

Diese werden im Folgenden beschrieben.

Hinweis:

Weiterführende Informationen zur Regelung von Kleinrobotern gibt es beispielsweise in einem Paper zum RoboCup: [GGE+05].

1.1.1 Geschwindigkeitsmessung

Wir benötigen zwei Motoren und zwei Sensoren für die Raddrehung. Die Sensoren bestehen aus einem einfachen Schalter und einem Zahnrad, das den Schalter bei Motordrehung auslöst (Abb. 1.1). Das Drehen des Rades erzeugt eine `0-1` bzw. `1-0` Flanke am digitalen Eingang. Leider kann durch diese einfache Schaltung die Drehrichtung des Rades nicht erkannt werden. Man muss sich deshalb bei diesem Versuch auf die an den Motorausgängen eingestellte Drehrichtung verlassen.

Die Geschwindigkeitsmessung besteht ja darin, über einem kurzen Zeitintervall T_S die Zahl der Flanken zu zählen. Da pro Radumdrehung nur sehr wenige Flanken gemessen werden können, wird eine genaue Geschwindigkeitsmessung unmöglich. Deshalb ist es erforderlich, ein Verfahren anzuwenden, das die Flanken in einem bestimmten Zeitintervall T_S zählt.[1] Ferner benötigen Sie den Zeitpunkt der einzelnen Messungen. Lesen Sie dazu die Echtzeituhr aus (Funktion `gettimeofday` aus `sys/time.h`, siehe Manpage).

1.1.2 Regelung

Grundlagen: Wir verwenden jeweils einen PID-Regler zur Regelung der Motorgeschwindigkeit. Weitere Informationen entnehme man [Brä03, Kap. 7] und den Folien zur Vorlesung „Introduction to Robotics“ [Mes05].

Simulation: Mit dem theoretischen Modell aus der Vorlesung kann das Zusammenspiel des Reglers mit den Motoren simuliert werden. Die CMU bietet ein Tutorial [MT97] an, das zeigt, wie dies in Matlab geschehen kann.

Hier soll das kostenfreie `octave` [Eat05] genutzt werden. Die Simulation geschieht mit folgenden Befehlen:

[1] Denken Sie daran, was passiert, wenn Sie weniger als n Datensätze vorliegen haben. Ein dynamisch wachsendes Array aus der STL [sgi05] (z. B. `deque`) könnte Ihnen dabei helfen!

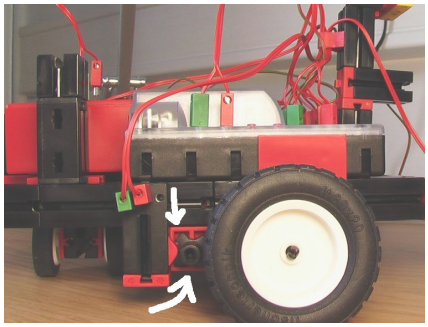


Abbildung 1.1: Einfacher Sensor für die Erfassung der Raddrehung: Mikroschalter und Zahnrad.

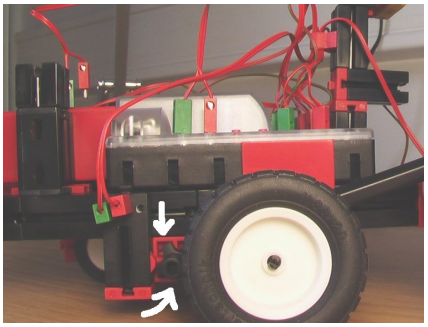


Abbildung 1.2: Das Drehen des Rades erzeugt eine `0-1` Flanke am digitalen Eingang.

```
Kp=1;
Ki=1;
Kd=1;

R=4;
tau=.5;
k=1/4.5;

num=[R*Kd R*Kp R*Ki];
den=[tau+Kd k+Kp Ki];

mfdsys=tf(num,den);
sysout(mfdsys,'tf')
step(mfdsys)
```

Beispieldatei `EsrMotorsteuerungControlEx1.m` für die Simulation mit `octave`. Das Laden geschieht aus der `octave`-Kommandozeile durch tippen des Dateinamens `EsrMotorsteuerungControlEx1`.
Hinweis: Bei einigen Versionen von `octave` ist es notwendig, den Befehl `sysout(mfdsys,'tf')` durch `sysout(mfdsys)` zu ersetzen.

1.2 Vorbereiten der Roboter

Die Motoren und Sensoren sind wie in der Bauanleitung [N.04, S. 14] beschrieben anzuschließen. Davon abweichend soll ein anderes Zahnrad benutzt werden, um die Mikroschalter zur Messung der Raddrehung auszulösen. Mit dem Zahnrad aus der Bauanleitung erhalten wir nur 8 Flanken pro Umdrehung – deutlich zu wenige für eine genaue Geschwindigkeitsmessung! Deshalb nutzen wir ein Zahnrad, das 20 Flanken pro Umdrehung erzeugt; bei der Montage ist auf die leichte Verschiebung zu achten, damit der Schalter korrekt auslöst (siehe Abb. 1.3). Dies kann z. B. mit dem Programm `fttestinterface` aus [Fri07] oder dem Programm `testrobo` aus der mitgelieferten Build-Umgebung erfolgen.

Achtung: Bitte achten sie auf die Polung der Batterie!

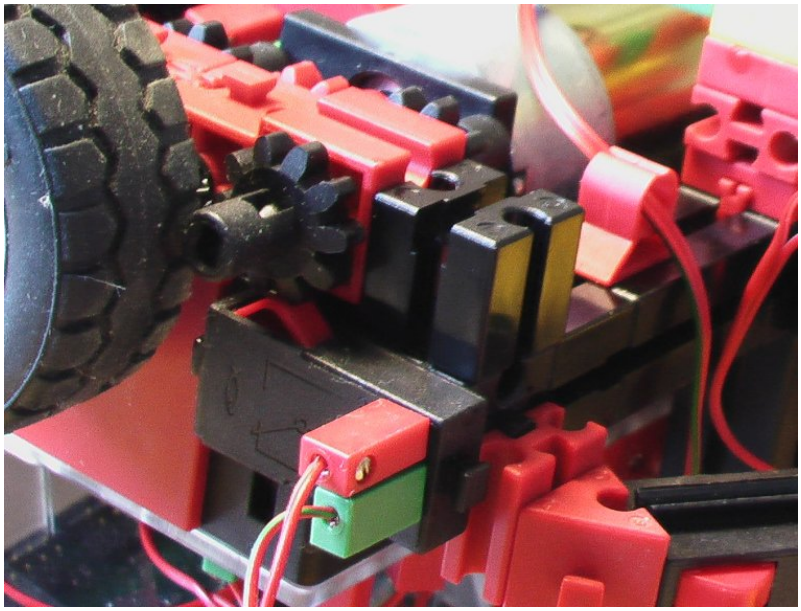


Abbildung 1.3: Modifizierter Sensor für die Erfassung der Raddrehung: 20 Flanken pro Umdrehung. Damit der umgebaute Schalter funktioniert, muss der richtige Abstand zwischen Schalter und Zahnrad eingestellt werden!

1.3 Aufgaben

1. Machen Sie sich mit der bereitgestellten Fernsteuerung für den Roboter vertraut, die es erlaubt, den Roboter per Tastendruck zu steuern. Achten Sie insbesondere auf die Methode ßtep", die in jedem Durchlauf der Hauptschleife aufgerufen wird, diese werden Sie in diesem Versuch selbst für die Messung implementieren.

2. Implementieren Sie ein Verfahren zur Geschwindigkeitsmessung, das die von den Raddrehungssensoren erzeugten Flanken nutzt. Verwenden Sie die Motorsteuergröße zur Bestimmung der Laufrichtung.

Hinweis

Achten Sie darauf, dass Ihr Verfahren für vorwärts und rückwärts laufende Motoren funktioniert.

Wählen Sie als Zeitkonstante für die Mittelung $T_S = 100\text{ms}$.

Kapseln Sie diese Komponente wie beim letzten Versuch in ein Objekt, das eine Funktion `step(...)` bereitstellt, die die Verarbeitung für einen Zeitschritt ausführt. Darüber hinaus sollte es eine Funktion `getSpeedInTicksPerSecond` geben, die sie Anzahl der Flanken im Messzeitraum T_S berechnet (steigende *und* fallende Flanken sollen als „Tick“ berücksichtigt werden).

Hinweis

Stellen Sie unbedingt sicher, dass dieser Teil fehlerfrei funktioniert, bevor Sie zur nächsten Teilaufgabe übergehen!

3. Bestimmen Sie die Drehgeschwindigkeit eines Rades ohne Belastung (Leerlauf) auf Geschwindigkeitsstufe 4 in Umdrehungen pro Sekunde. Diese Größe dient als „Soll-Drehzahl“ in den nächsten Aufgabenteilen.

4. Simulieren Sie in `octave` die Impulsantwort des Systems beim Einschalten für $K_p = 1$, $K_i = 1$, $K_d = 1$, $\tau = 1$, $k = 1/4.5$, $R = 4$.

5. Implementieren Sie in C++ einen PID-Regler *für einen Motor*, der eine einstellbare konstante „Soll-Drehzahl“ einhält, auch wenn der Motor gebremst oder gedrosselt wird (natürlich geht dies nur bis zur Maximalansteuerung). Nutzen Sie den soeben entwickelten Algorithmus zur Geschwindigkeitsmessung zur Berechnung der „Ist-Drehzahl“.

   Kapseln Sie diese Komponente wie beim letzten Versuch in ein Objekt, das eine Funktion `step(...)` bereitstellt, die die Verarbeitung für einen Zeitschritt ausführt (mit einem Zeitschritt ist die das einmalige Durchlaufen der Hauptschleife gemeint, d. h. `step(...)` wird einmal aufgerufen).

Hinweis

Für die Regelung ist es besonders wichtig, dass der Regler in einem konstanten Zeitraster aufgerufen wird, z. B. alle 10 Millisekunden. Dies können wir nicht direkt gewährleisten (ohne echtes Echtzeitbetriebsystem, ohne Trennung von Steuerung und Bildverabeitungskomponente und Threads / Prozesse, etc.) Achten Sie deshalb darauf, dass der Regler zwar oft aufgerufen wird, aber erst nach dem Ablauf einer bestimmten Zeitspanne T_C aktiv wird. T_C sollte so gewählt sein, dass $T_S \leq T_C$. Bewährt hat sich $T_S = T_C$.

Hinweis

Stellen Sie unbedingt sicher, dass dieser Teil fehlerfrei funktioniert, bevor Sie zur nächsten Teilaufgabe übergehen!

6. Verwenden Sie je einen PID-Regler pro Motor, um den Roboter geradeaus fahren zu lassen (beide PID-Regler erhalten gleiche Soll-Geschwindigkeit). Drosseln Sie einen Motor während der Fahrt (mittels eines Potentiometers) und testen Sie so die Qualität dieser Regelung. Die Aufgabe gilt als gelöst, wenn der Roboter nach kurzer Zeit wieder ungefähr geradeaus fährt.

7. Es ist deutlich sichbar, dass der Regler oftmals wegen der Diskretisierung zwischen zwei Ausgabegrößen hin und her schwankt. Lösen Sie dieses Problem, indem der Regler die Motorgröße ändert, auch wenn gerade keine Neuberechnung stattfindet.

Berechnet der Regler beispielsweise $u = 4.33$, soll die Steuergröße der Geschwindigkeit in ca. 1/3 der Fälle 4 und in 2/3 der Fälle 5 sein (Tipp: `man rand`).

1.4 Erforderliche Vorbereitung

- Theorie des PID.
- Druchführung der Octave-Simulation.

Literaturverzeichnis

[Brä03] BRÄUNL, T.: *Embedded Robotics*. Springer, 2003. – ISBN 3-540-03436-6

[Eat05] EATON, John W.: *Octave Home Page*. `http://www.octave.org/`. Version: 2005

[Fri07] FRIEDRICH, Holger ; J.W.GOETHE UNIVERSITAT (Hrsg.): *FtApi - fischertechnik C++ progamming api*. v0.4.1. Frankfurt am Main: J.W.Goethe Universitat, 2007. `http://www.vsi.cs.uni-frankfurt.de/fischer/`

[GGE+05] GLOYE, Alexander ; GÖKTEKIN, Cüneyt ; EGOROVA, Anna ; TENCHIO, Oliver ; ROJAS, Raúl: Learning to Drive and Simulate Autonomous Mobile Robots. In: NARDI, D. (Hrsg.) ; ET (Hrsg.) ; AL. (Hrsg.): *RoboCup 2004: Robot Soccer World Cup VIII*, Springer-Verlag Berlin Heidelberg, 2005 (LNAI 3276), S. 160–171

[Mes05] MESTER, Rudolf: *Introduction to Robotics*. Vorlesung. `http://www.vsi.cs.uni-frankfurt.de/teaching/`. Version: 2005

[MT97] MESSNER, Bill ; TILBURY, Dawn: *Control Tutorials for Matlab*. `http://www.engin.umich.edu/group/ctm/`. Version: 1997

[N.04] N., N. ; FISCHERTECHNIK GMBH (Hrsg.): *Robo Mobile Set - Bauanleitung*. 1. Waldachtal: fischertechnik GmbH, October 2004. – 111868

[sgi05] Silicon Graphics, Inc.: *SGI - Services & Support: Standard Template Library Programmer's Guide*. `http://www.sgi.com/tech/stl/`. Version: 2005