

## Erreichen eines Ziels (ohne Kamera) Teil II: Realisation

Vortrag im PSBV Lego 2003, PDF-Version  
Christoph Sommer <DeltaDev@GMX.net>

Zielfindung / Hindernisvermeidung

## Aufgabenstellung

- Anfahren einer Lampe hinter einem kleinen Hindernisparcours



# Zielfindung / Hindernisvermeidung

## Anforderungen

- Autonomer Roboter
- Beliebiger Start- und Zielpunkt
- Beliebige Form und Anordnung der Hindernisse
- Unabhängigkeit von wechselnden Faktoren
  - Lichtverhältnisse
  - Batteriespannung
  - Untergrund

## Übersicht

- Realisation
  - Teil I
    - Verwendete Hardware
      - Aufbau
      - Probleme
    - Verwendete Software
      - Entwicklung
      - Probleme
  - Teil II
    - Programmablauf
    - Umsetzung der Konzepte
  - Teil III
    - Ausnahmebehandlung

# Zielfindung / Hindernisvermeidung

## Hardware: Aufbau

- Chassis
  - Vier Räder, dadurch Drehung um die eigene Achse möglich
  - Genauer als Ketten oder Dreirad



# Zielfindung / Hindernisvermeidung

## Hardware: Aufbau

- Sensoren
  - Touchsensoren an der Front
  - Rotationssensor an linkem Rad
  - Lichtsensor, fest nach vorne gerichtet



# Zielfindung / Hindernisvermeidung

## Hardware: Aufbau

- Touchsensor zur Hinderniserkennung
  - Breiter Bumper mit Über- und Unterfahrerschutz
  - Sehr sensibel
  - Erkennt auch schiefes Anfahren des Hindernisses

## Hardware: Probleme

- **Rotationssensor zur Drehwinkel- und Streckenmessung**
  - Unabhängigkeit von Batteriespannung und Untergrund
- **Problem: Messen an nur einer Achse**
  - Motoren drehen vor- und rückwärts unterschiedlich schnell
  - Rotationssensor bremst
- **Lösung: Einführen von Konstanten**
  - 2 Umrechnungskonstanten Rad-Drehwinkel in Roboter-Drehwinkel
  - Ausgleichsgetriebe

## Hardware: Probleme

- Übersetzung des Rotationssensors
  - Zu langsam
    - Ungenau
    - Leichte Variationen beobachtet
  - Zu schnell
    - RCX kommt nicht mehr mit

Zielfindung / Hindernisvermeidung

Software: Entwicklung

- NQC
  - Übersichtlichstes System
  - Verwendet Standard-Firmware
  - Programm läuft vollständig im Brick

## Software: Probleme

- Probleme bei der Entwicklung
  - Keine Verbindung zum PC
  - RCX arbeitet verhältnismäßig langsam
  - Eingeschränkter Speicherplatz für Variablen (32 Stück), keine Rückgabewerte
  - Eingeschränkte Rechenleistung, insbesondere auf 16bit signed integer
  - Eingeschränkter Programmspeicher, Probleme mit Codegröße

## Teil II: Der Programmablauf

- Programm

- Orientierung auf Lampe

- Orte Lampe
    - Drehe dich zur Lampe

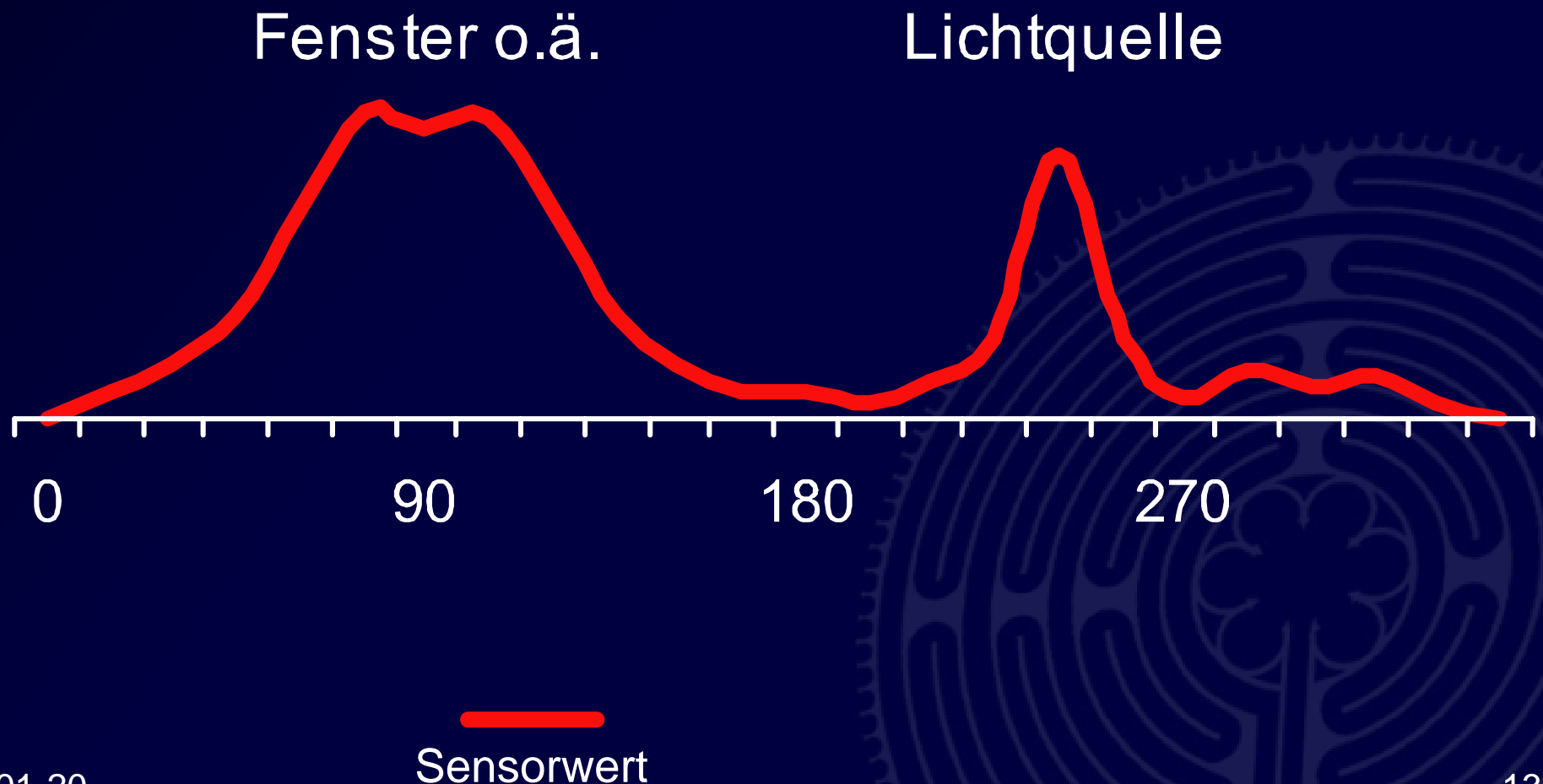
- Fahrt zur Lampe

- Lampe erreicht?
      - Dann fertig
    - Hindernis gefunden?
      - Weiche Hindernis aus
      - Orientiere dich neu auf die Lampe

# Zielfindung / Hindernisvermeidung

## Orte Lampe

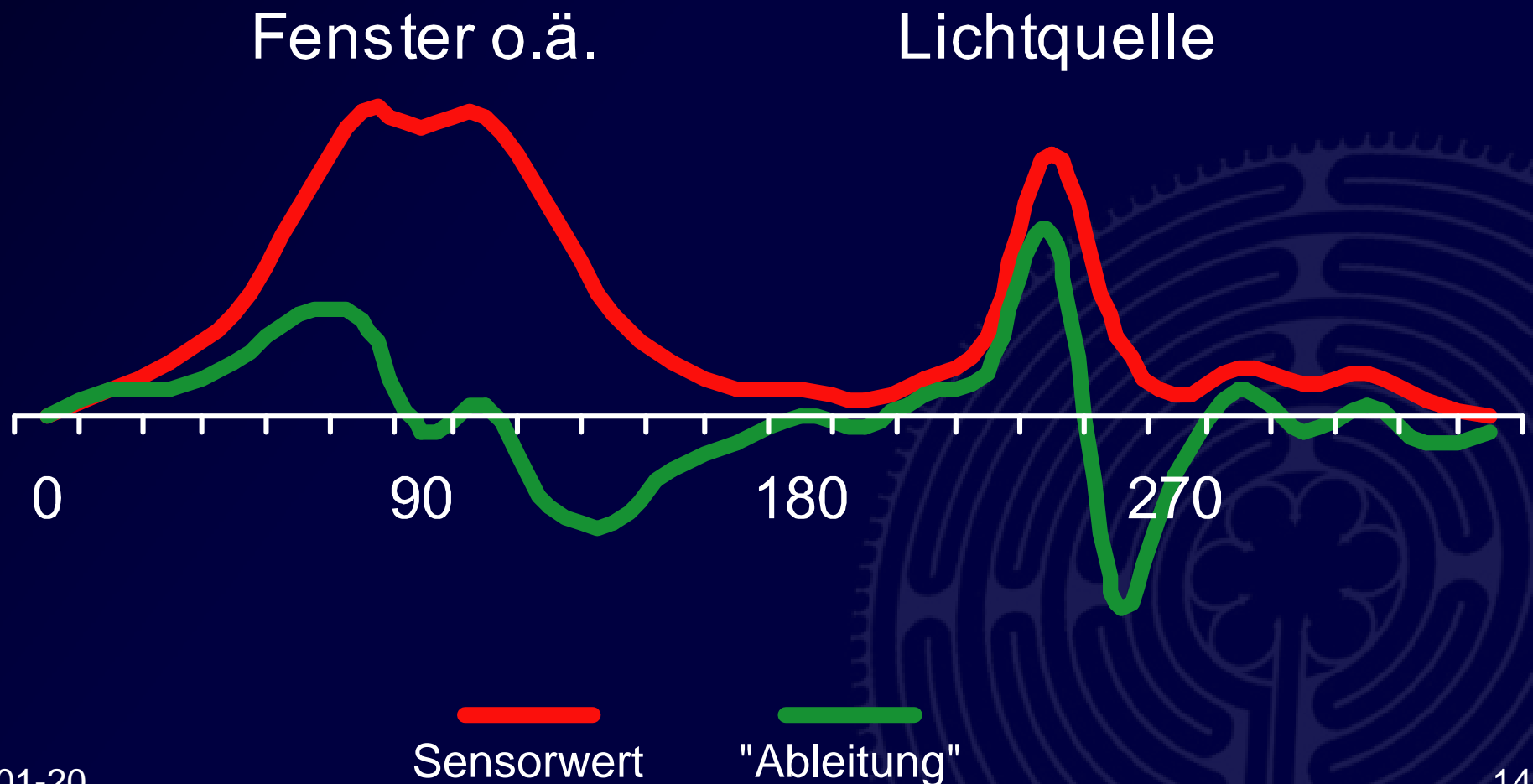
- 360°-Schwenk (Drehwinkelmessung)
- Problem: Identifizieren der Lampe



# Zielfindung / Hindernisvermeidung

## Orte Lampe

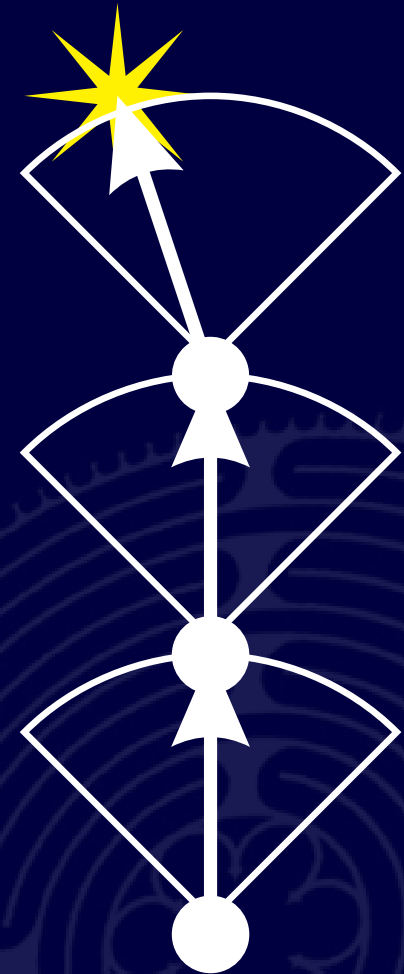
- Lösung: Differenz zum letzten Messwert
- Identifiziert Lichtquelle sehr sicher



# Zielfindung / Hindernisvermeidung

## Fahrt zur Lampe

- Problem:  
evtl. keine genaue Zielpfeilung
- Lösung:  
regelmäßiges Scannen



Zielfindung / Hindernisvermeidung

Zielerkennung

- Methode
  - Überschreitung eines Grenzwerts am Lichtsensor
- Implementierung
  - Bei Vollausschlag: Ziel erreicht
  - Einrichtung eines watcher-tasks

Zielfindung / Hindernisvermeidung

Hinderniserkennung

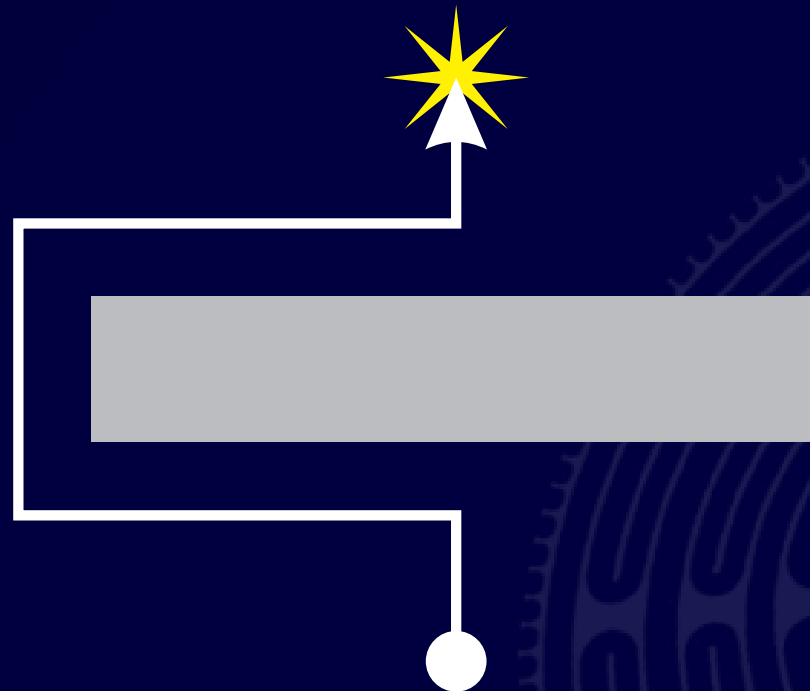
- **Bumperkontakt**

- Parallelschaltung der Touchsensoren an der Front
- Sehr sichere Erkennung von Hindernissen bis knapp über dem Boden

# Zielfindung / Hindernisvermeidung

## Hindernisvermeidung

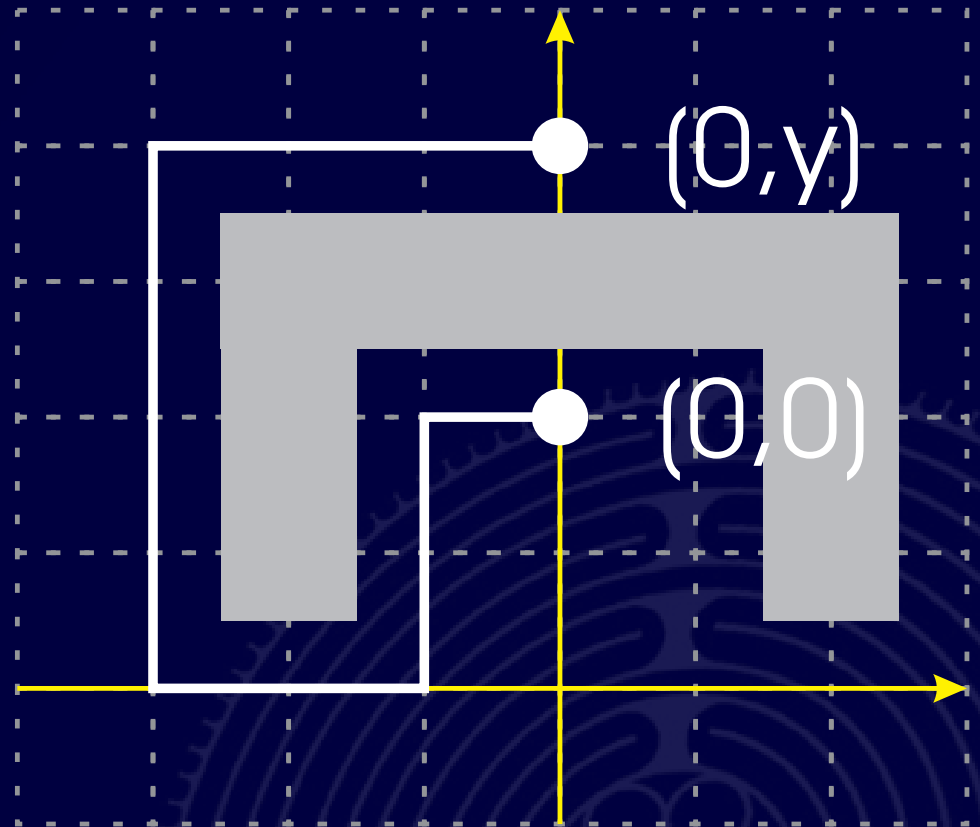
- Theorie:
  - Algorithmus „Bug2“ (Lumelsky / Stepanov)
  - Inspiriert durch Beobachtung von Insekten





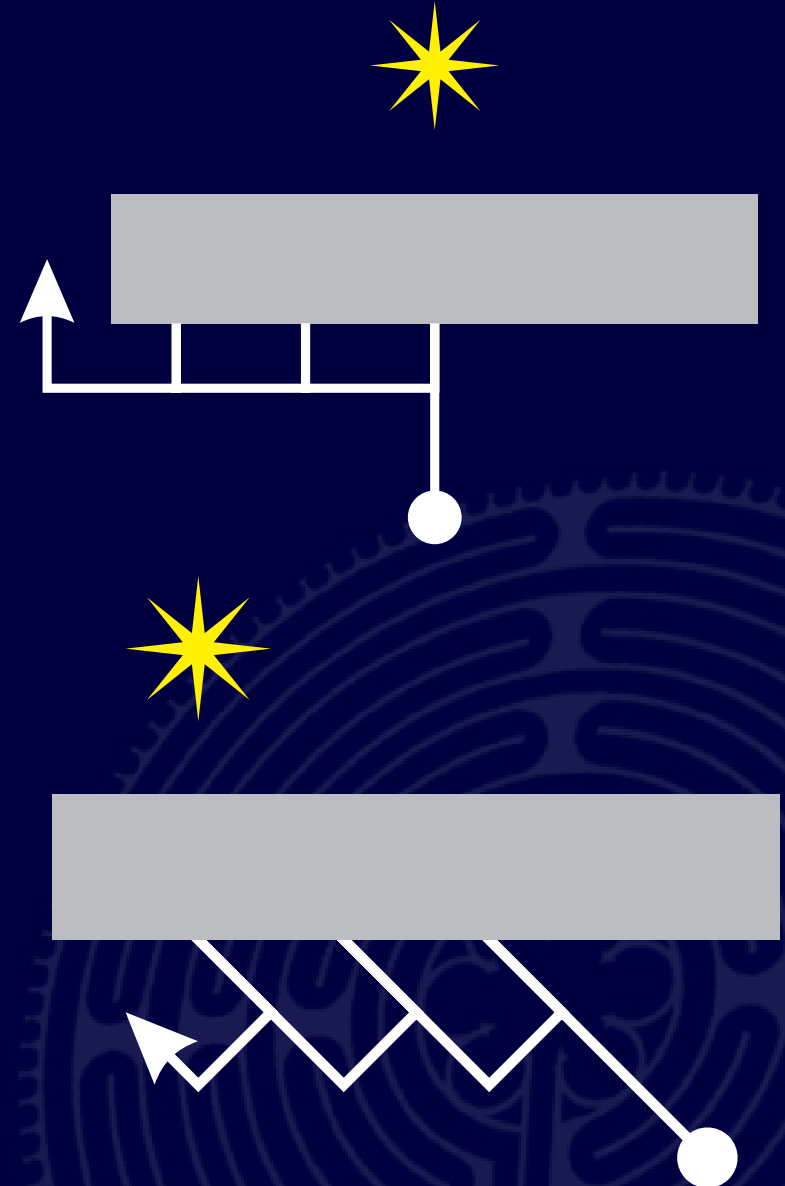
## Hindernisvermeidung: Implementierung

- Virtuelles Koordinatensystem
- Startpunkt am Hindernis ist  $(0,0)$
- Ziel liegt auf positiver  $y$  Achse
- Verlassen des Hindernisses bei  $x = 0, y > 0$



## Hindernisvermeidung: Implementierung

- Entlangfahren am Hindernis
  - Entlangtasten ohne Roboter zu verdrehen gestaltet sich zu schwierig
  - Deshalb: Einfach immer wieder versuchen, gegen das Hindernis zu fahren



# Zielfindung / Hindernisvermeidung

## Teil III: Ausnahmebehandlung

- Fehlerfälle

- Falsche Ortung der Lampe
- Hindernisse haben sich verändert
- Lampe zu weit weg oder verdeckt

## Zielfindung / Hindernisvermeidung

### Endlosschleifen

- Problem

- Situation

- Irrtümliche Annahme: Lampe ist hinter der Zimmerwand → Hindernis unendlich lang
    - Hindernis verschwindet → unendlich kurz

- Bug2 versucht dann erfolglos, dieses „Hindernis“ zu umfahren

- Lösung

- Abbruch von Bug2 nach festem Zeitintervall

## Zielfindung / Hindernisvermeidung

### Lampe verdeckt

- Problem
  - Lampe nicht ortbar, Bug2 kann nicht starten
- Vorgehen
  - Fahre in die Richtung, wo zuletzt die Lampe war
  - Bei Anstoßen an ein Hindernis:  
Umschalten auf alternativen Algorithmus
  - Versuche regelmäßig, die Lampe wieder zu orten



## Zielfindung / Hindernisvermeidung

### Implementierung

- Implementierung
  - Einfach, da Drehungen auf  $90^\circ$  beschränkt
- Vorteil
  - Algorithmus umfährt jede Art von Hindernis
  - Durch viele Scans sind Dreh- und damit Positionsinformation ungenau
  - Pledge braucht nur ungefähre Orientierung
- Nachteil
  - Sucht nicht im Inneren von Räumen

Zielfindung / Hindernisvermeidung

Zusammenfassung

- Erreichen eines Ziels (ohne Kamera)
  - Einsatz der Lego-Sensoren zur Lösung
  - Orten einer punktförmigen Lichtquelle
  - Ausweichen von Hindernissen