Robotikpraktikum - GPS auf Rädern A

Max Hartmann, Philipp Gernandt & Tobias Buck
(Physik)

Betreuer: Gero Plettenberg & Thomas Kloepfer

13.4.2015

Betreuer: Benjamin Reh & Thomas Kloepfer

Projektziel

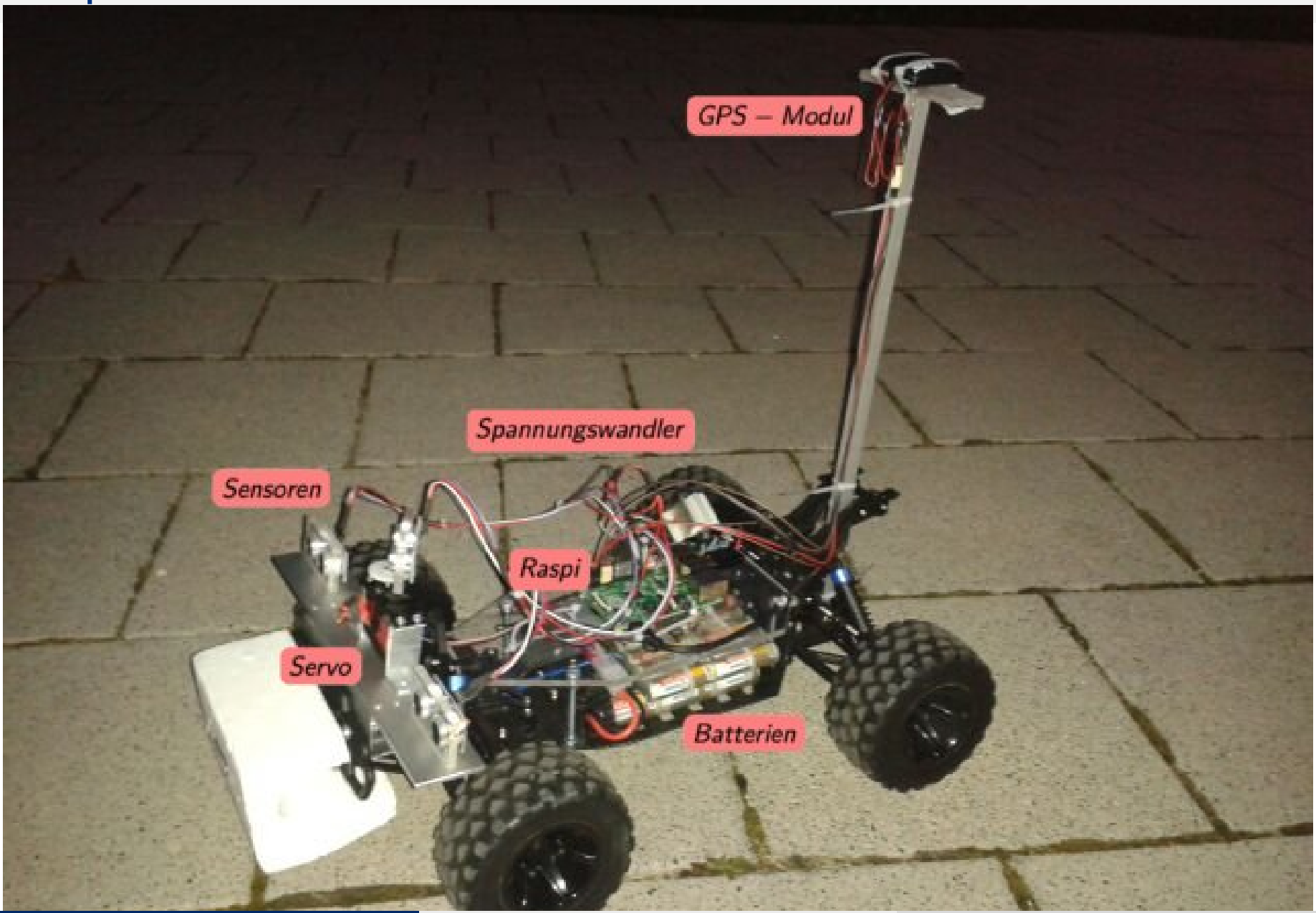
Zielvorgaben

- Konstruktion eines autonomen Fahrzeugs
  - Fahrt nach GPS-Daten
  - Ausweichen kleinerer Hindernises

Umsetzung

- Umbau eines Modellautos
- Steuerung durch einen Raspberry Pi
- Erkennen der Hindernisse durch Ultraschallsensoren
- GPS Modul mit Kompass
- Programmierung in Python

Komponenten



Aufbau des Autos

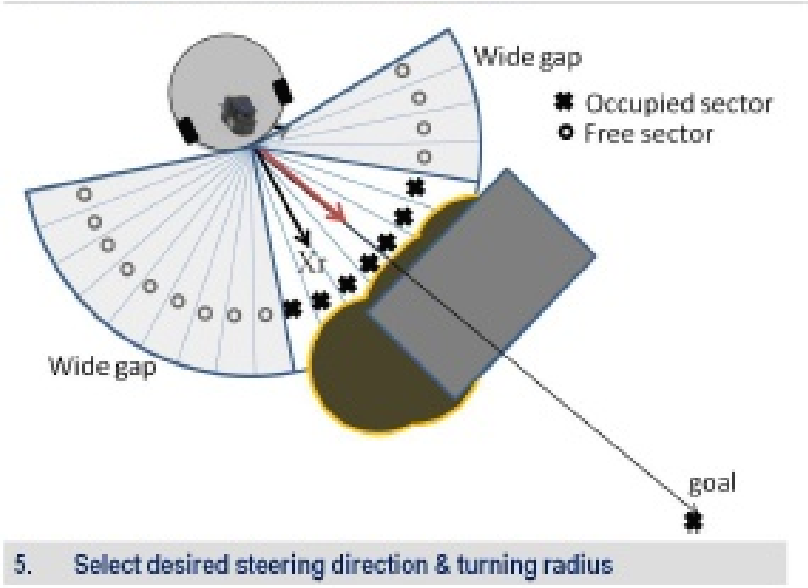
- Bausatz des Modellautos
- Plexiglasplatte für Raspi, Spannungswandler, usw.
- Ansteuerung des Fahrtreglers und der Servos
- Auslesen und Ansteuern von GPS und Ultraschallsensoren
- Erarbeitung eines Navigationskonzepts
- Entwicklung und Optimierung des Ausweichalgorithmus
- Implementierung von GPS und Kompass ins Fahrtkonzept
- Optisches Tuning
- Fehlerbehebung/Verbesserungen

Features

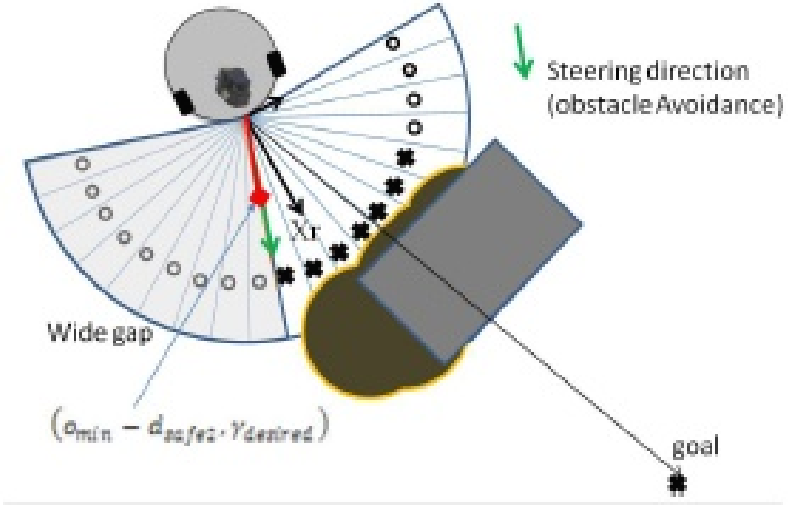
- Kommunikation via WLAN
- Verschiedenen Output-Level zum Debuggen
- Erstellen von Logfiles (GPS + Output) für Plots in Google Maps
- GPS und Sensoren in Background-Threads
- Verschiedene Modi für den schwenkbaren Sensor
- Findet günstigsten Weg bei vielen Hindernissen
- Getrennte Spannungsversorgung

Programmstruktur

Navigation

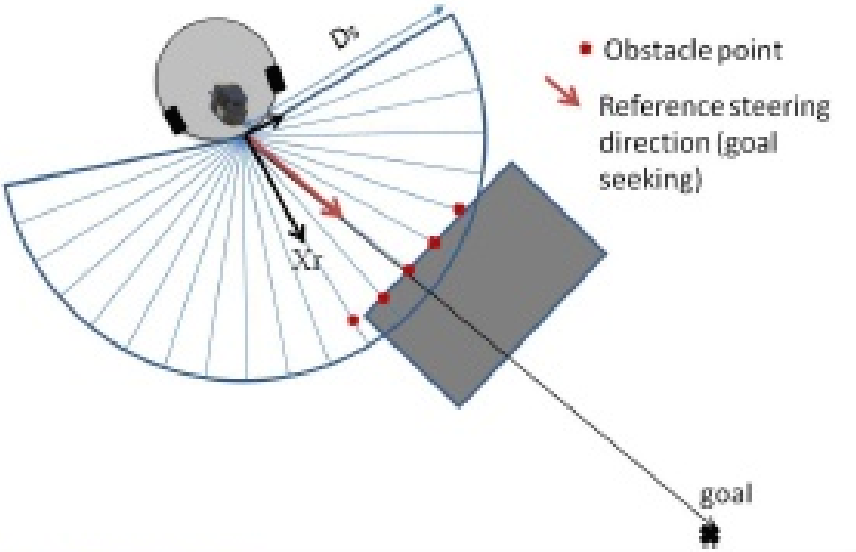


- Sensoren abfragen
- Richtung zum Ziel vom GPS/Kompass

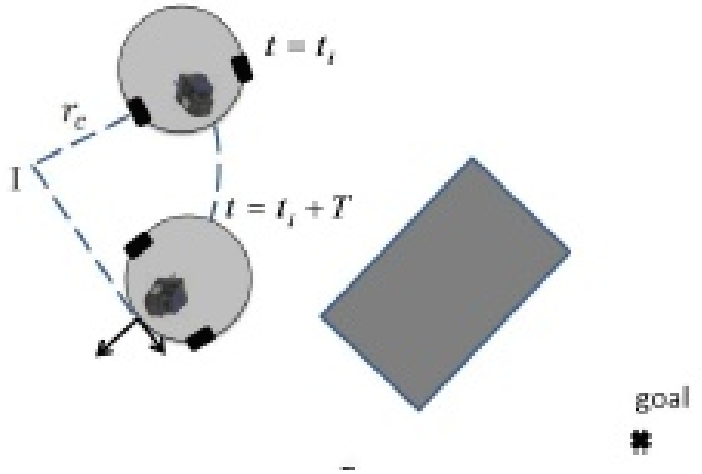


- freie Segmente finden
- Einteilung in große, mittlere und kleine Lücken

Navigation



6. Execute Action.



- Lenkrichtung anhand einer Kostenfunktion errechnen
- $Cost = c_1(r_{ref} - r_{gap}) + c_2 d_{gap}$
- in errechnete Richtung ausweichen

Sensor-Thread

- Abfragereihenfolge: links, oben, rechts, oben, links, ...
- Bei Abfrage oben, drehe den Servo weiter nach rechts
- Am rechten Rand, springe nach links

- Speichere aktuelle Sensordaten in Array
- Ignoriere Werte kleiner als 7cm
- Fünf Versuche für korrekte Messung

Sensor-Thread

```
def get_distance(i):
  GPIO.setmode(GPIO.BCM)

  TRIG = [15, 8, 7]
  ECHO = [14, 9, 11]

  GPIO.setup(TRIG[i], GPIO.OUT)
  GPIO.setup(ECHO[i], GPIO.IN)

  GPIO.output(TRIG[i], False)

  time.sleep(.04)

  GPIO.output(TRIG[i], True)
  time.sleep(0.00001)
  GPIO.output(TRIG[i], False)
```

```
while (GPIO.input(ECHO[i])!=1):
  start_time = time.time()

while (GPIO.input(ECHO[i])==1):
  dt = time.time() - start_time
  if(dt>0.05):
    return 10000000.0

dist = dt*17000.0
time.sleep(.04)
return dist
```

Probleme

- Spannungsversorgung
- defekte Bauteile und Lieferschwierigkeiten
- Ultraschallsensoren
- GPS-Genauigkeit und Kompass
- Testfahrten

Ausblick

- feinere Motor- und Sensoransteuerung, schnellere Sensorabfrage
- Verbesserung des GPS
- bessere Sensoren
- Verfeinerung des Navigationsalgorithmus
- Benutzeroberfläche (z.B. Webpage zur Steuerung)
- interne Karte bzw. Einbindung von OpenStreetMaps

Vielen Dank für Ihre Aufmerksamkeit

Es folgt GPS auf Rädern B