Mobile Roboter und Einführung in das Robot Operating System

Erik

3. April 2017

Inhalt

Zielstellung

- Überblick Bau von mobilen autonomen Roboter
- Vermittlung Komplexität ROS
- Keine Roboter- oder ROS-Anleitung

Wieso Roboter bauen?

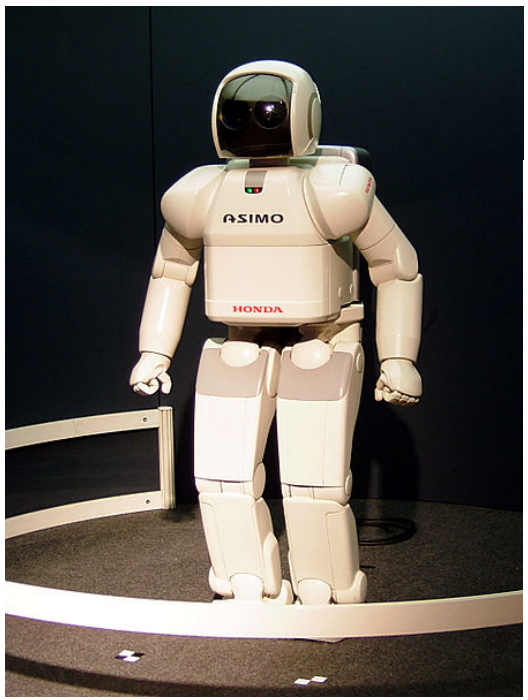
- Bier holen
- Handtücher falten
- Technik lernen
- Spaß am Gerät

Roboter


OMG ITS R2D2!
I LOVED HIM IN
STAR TREK
Mike Cole © 2015 OffWorldDesigns.com

Roboterarten

Industrieroboter



Humanoide Roboter

Autonome mobile Roboter

Transportroboter, Serviceroboter, ...

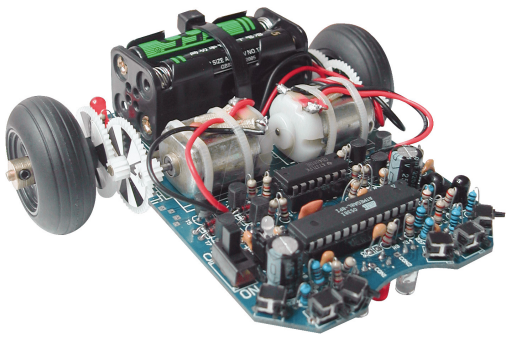
ROS-Roboter

PR2: 280.000$

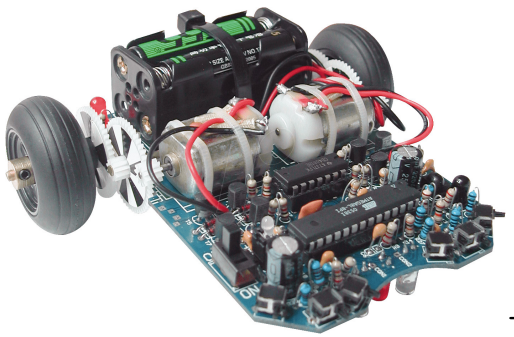
Husky: 20.000$

Turtlebot 2: 2.700$

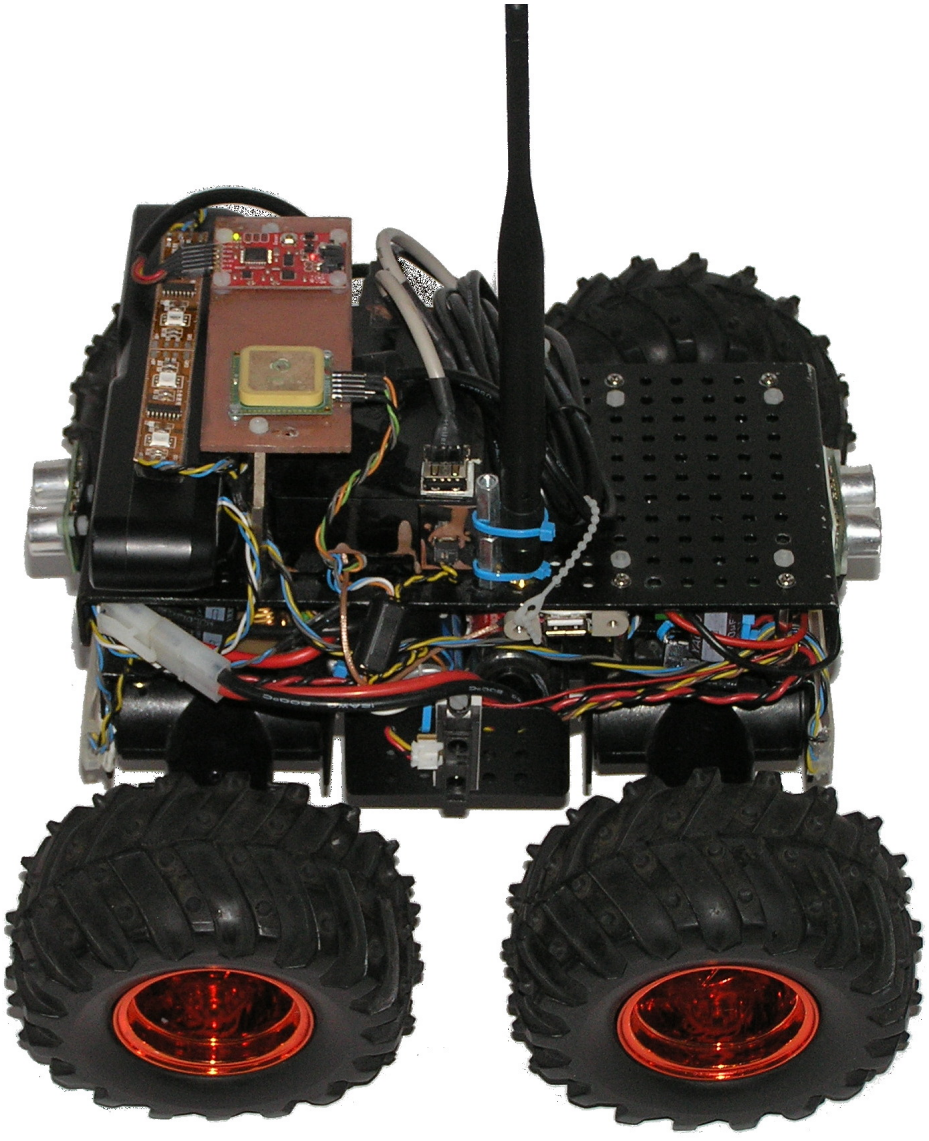
Roboter im Hobbybereich



Roboter im Hobbybereich

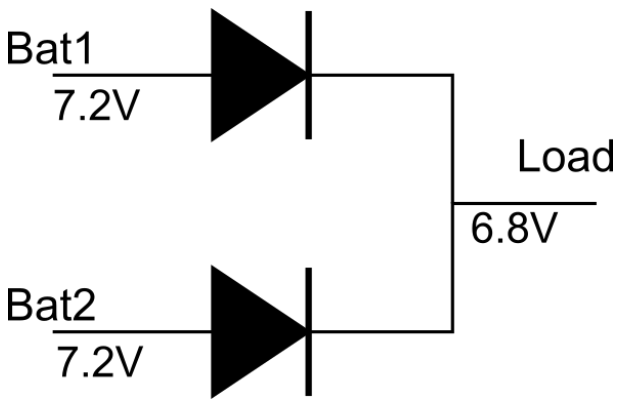


Wild Thumper

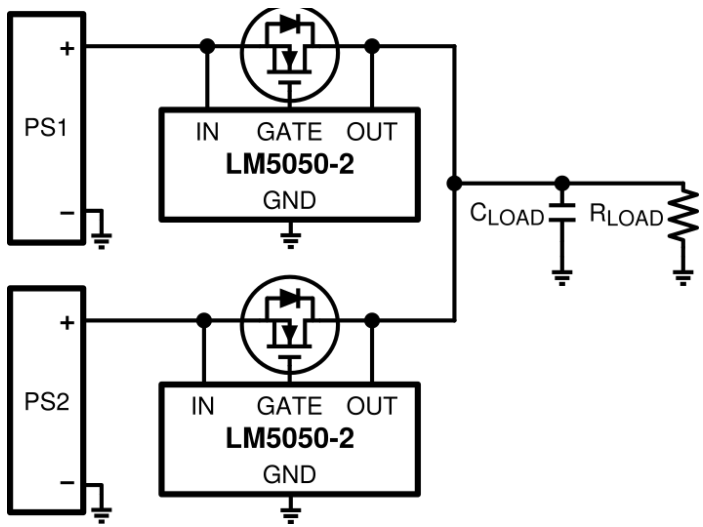


Spannungsversorgung

- Zwei NiMh Akkus 7.2V
- Umschaltung Akkus mittels Active ORing (LM5050+MOSFET)



ORing mit Dioden



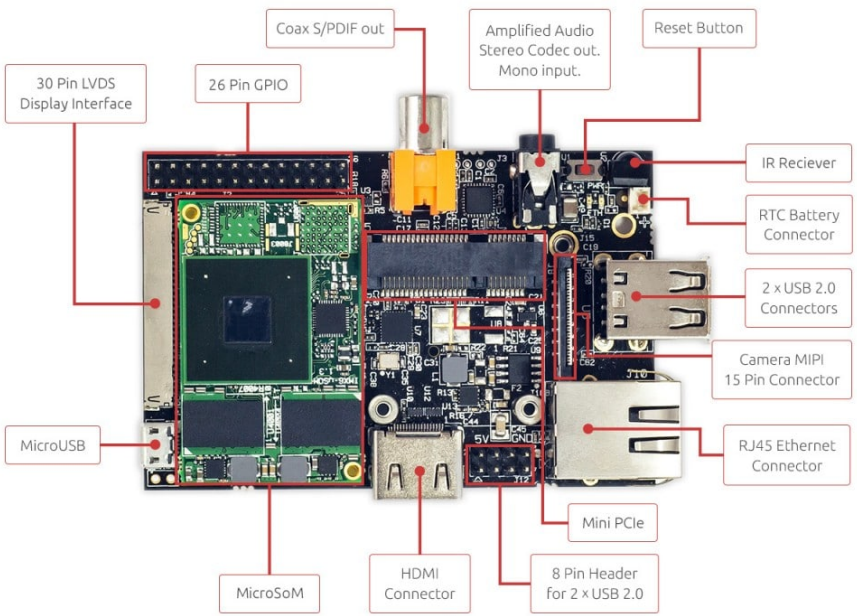
Active ORing

- Optional: Ein Akku + Dockingstation
- 5V via LM2576 Spannungsregler 3A
- Sicherungen!

Hauptcomputer

Hummingboard

- ARM Cortex-A9 (i.MX6) Solo/Dual/Quad @1GHz: Linux
- 2GB RAM
- Raspberry Pi Formfaktor & I/O-Header
- WLAN & Bluetooth



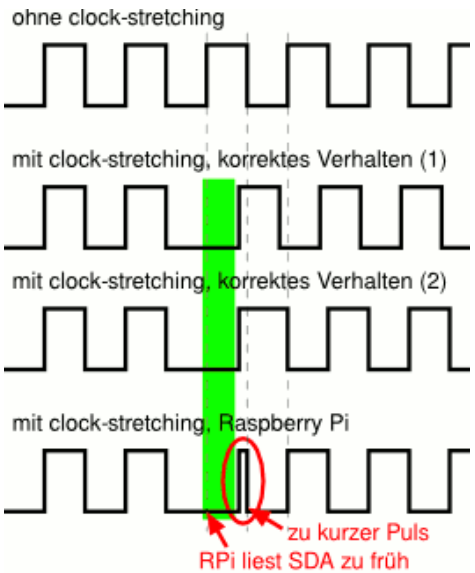
Hauptcomputer Alternative: Raspberry Pi

- ARM Cortex-A53

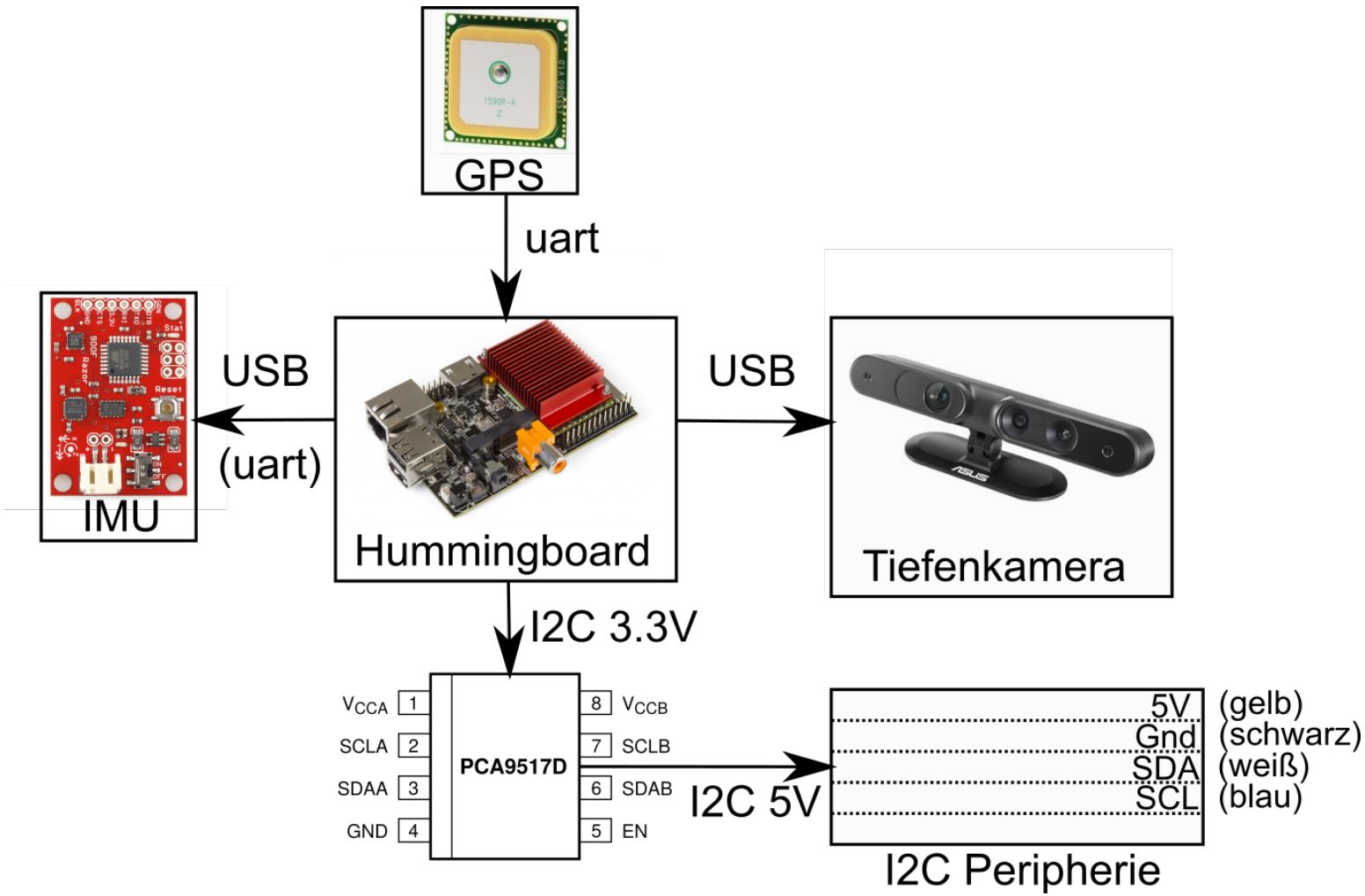
Stromverbrauch (inkl. USB HW)

	Idle	**Navigation**
Hummingboard	0.65A	0.80A
RPi3	0.58A	0.71A

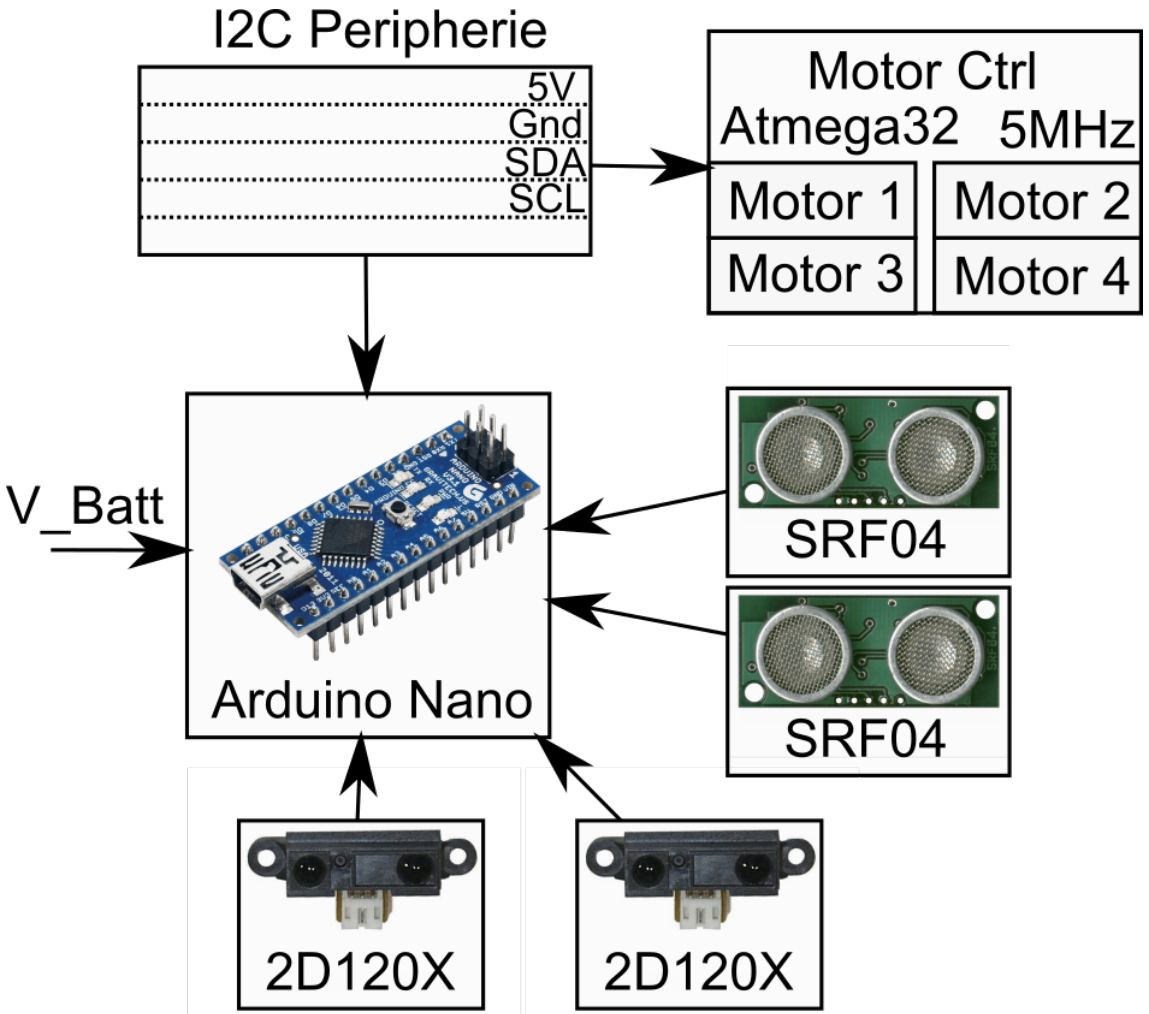
- Problem: I2C Clock-Stretching Bug
- Alternative: Odroid C2? (ARM Cortex-A53)



Peripherie I



Peripherie II (I2C-Bus)



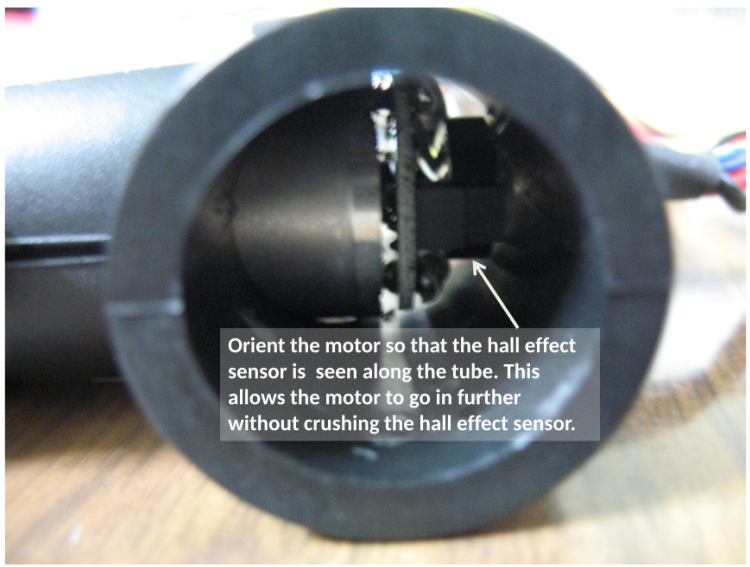
Antrieb

- 4x Motoren mit Encoder 6V, 6.5 A (stall)
- 4x H-Brücke VNH2SP30 $\leq 30A/20kHz$ (Vorher: TLE5205 $\leq 5A$)
- Geschwindigkeitsregelung: PID implementiert, nur PI verwendet
- Atmega32: $PWM = F\_CPU/256 \Rightarrow F\_CPU = 5MHz$

PWM freq (Hz)	Speed (Ticks/s)	Strom (A)
15625	2160	0.110
17021	2150	0.104
18018	2181	0.103
19531	2162	0.101

- 4x PWM output (3 Timer)

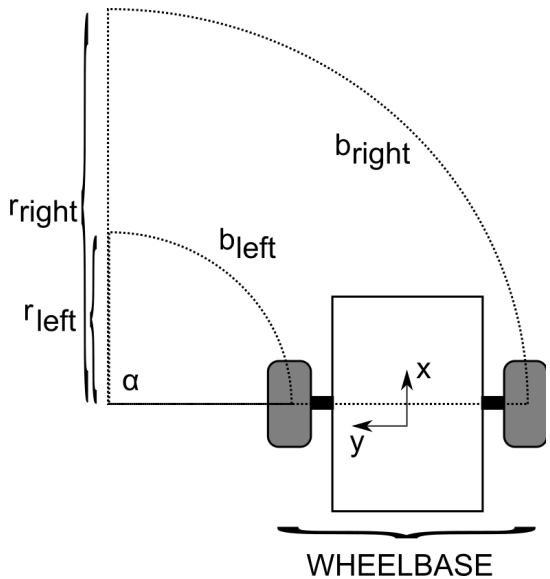
Umrüstung Motoren mit Encodern



Odometrie I

„Odometrie (Wegmessung) bezeichnet eine Methode der Schätzung von Position und Orientierung (Lageschätzung) eines mobilen Systems anhand der Daten seines Vortriebsystems.“ (Wikipedia)

Differentialantrieb:

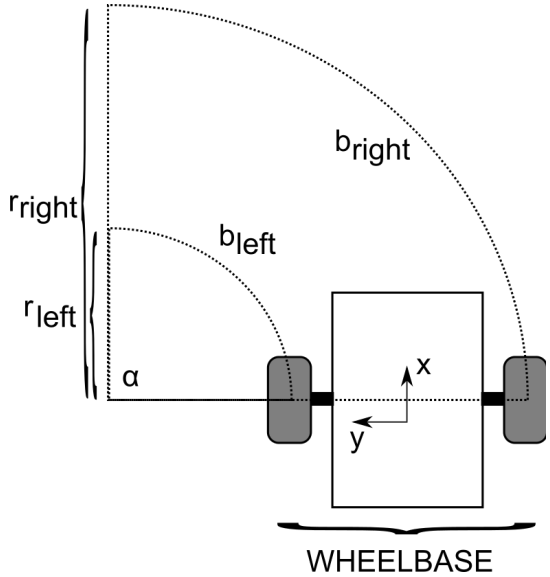


Rotation:
Kreisbogen $b = \alpha \cdot r$
und $r_{right} = r_{left} + WHEELBASE$

$$\Rightarrow \alpha \mathrel{+}= \frac{b_{right} - b_{left}}{WHEELBASE}$$

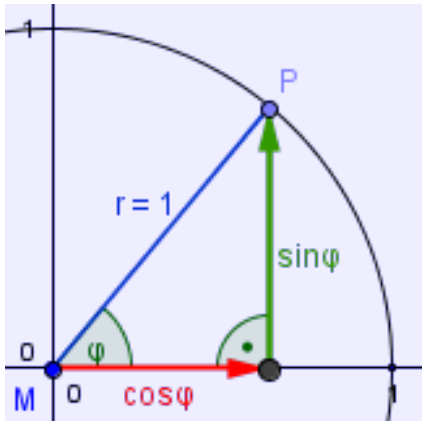
Odometrie I

Differentialantrieb:



Rotation:
Kreisbogen $b = \alpha \cdot r$
und $r_{right} = r_{left} + WHEELBASE$

$$\Rightarrow \alpha \mathrel{+}= \frac{b_{right} - b_{left}}{WHEELBASE}$$



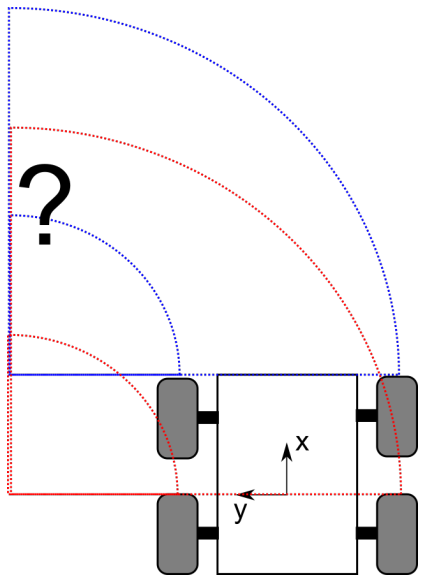
Translation:

$$r = \frac{b_{left} + b_{right}}{2.0}$$

$$pos \mathrel{+}= \begin{pmatrix} cos(\alpha) \\ sin(\alpha) \end{pmatrix} \cdot r$$

Odometrie II

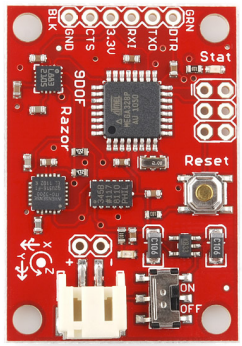
- Differentialantrieb mit 4 Rädern



- Wie 2 (3) Räder: Mittelwert vorne/hinten
- Genaues Zentrum der Rotation abhängig vom Untergrund

Odometrie III

- Lösung: Translation mit Odometrie, Rotation mittels IMU

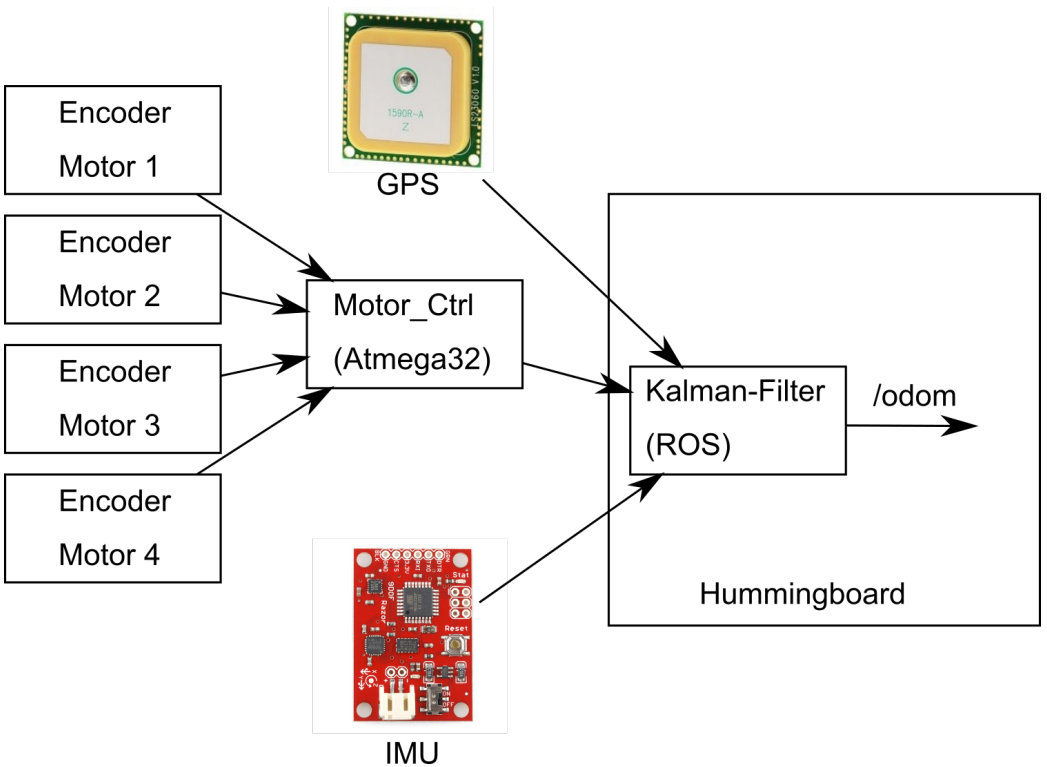


Razor IMU

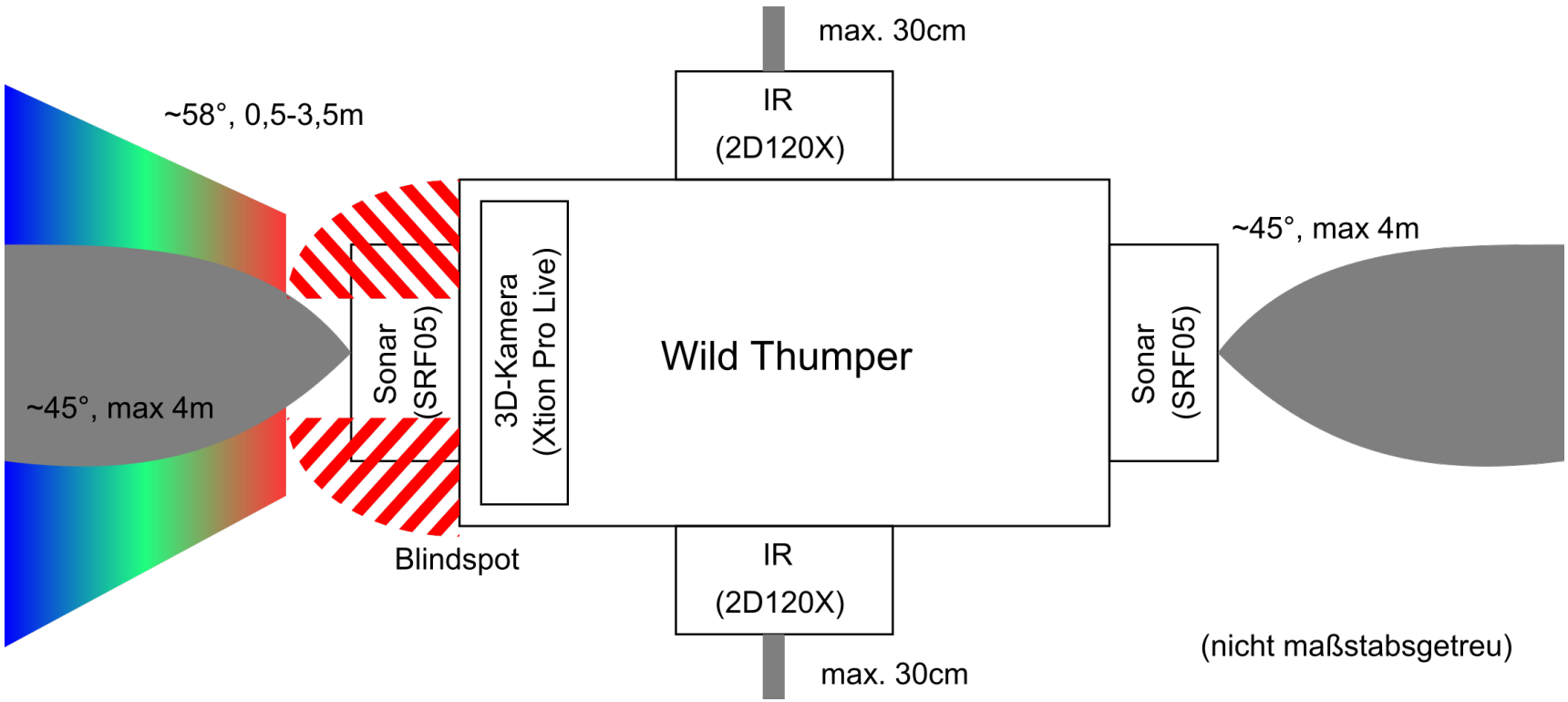
- 9DOF Inertial Measurement Unit
- 3x Magnetometer: Kompass (langzeitstabil, kein Drift, langsam)
- 3x Beschleunigungssensor: Erdbeschleunigung (langzeitstabil, kein Drift)
- 3x Gyroskop: Rotationen (Drift, schnell)
- Sensorfusion in Atmega328 via Direction Cosine Matrix (DCM)
- Kalibrierung erforderlich

Odometrie IV

- Berechnung Radumdrehung-Odometrie im AVR
- Fusion der Daten mit Kalman Filter (Rekursiver Schatzalgorithmus)
- GPS: Interferenz vom Hummingboard (HDMI)



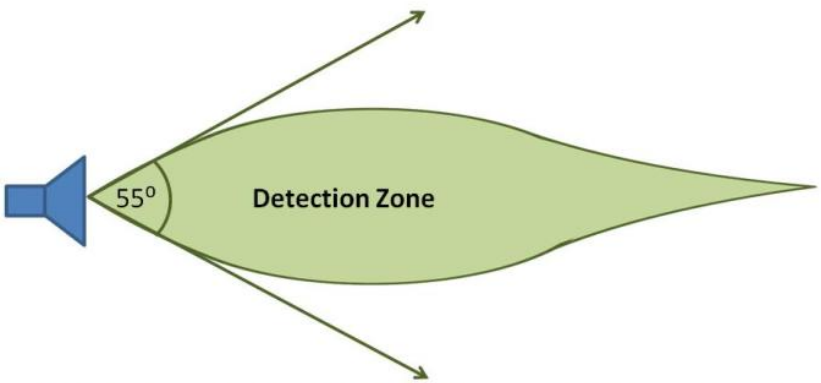
Distanzsensoren I



Distanzsensoren II

US-Sensor SRF05:

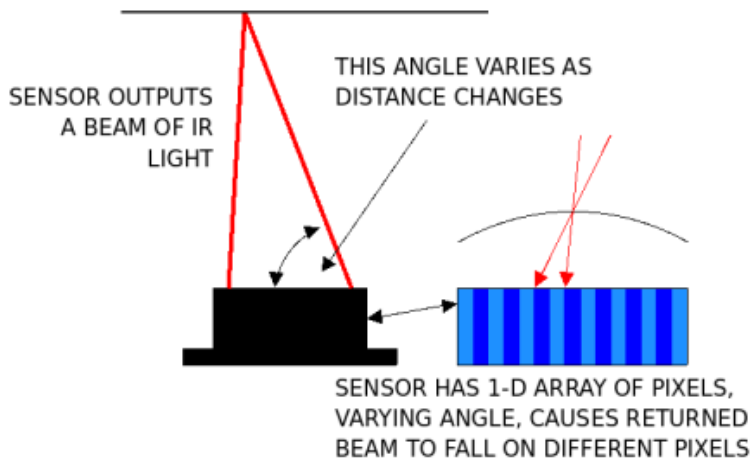
- Breiter Öffnungswinkel
- Reichweite: 3cm - 4m
- Echo Impuls: TTL-Pegel Signal, Impulsweite proportional zur Entfernung



The detection zone of the SRF05 is about 1 meter across at its widest point and just over 4 meters long.

IR-Sensor 2D120X:

- Reichweite: 4cm-30cm
- Spannung proportional zur Entfernung



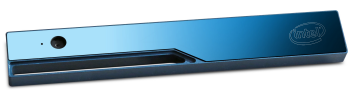
Distanzsensoren III

Hokuyo URG-04LX 1100€

RPLIDAR 500€

Neato XV-11 >110€(ebay)

Kinect 150$



Intel RealSense 169$

Distanzsensoren IV: Asus Xtion Pro Live

- RGB + Depth + Mikrofon
- USB2
- Tiefenkamera:
  - 320x240/60 fps
  - 640x480/30 fps
- RGB-Kamera: 1280x1024/30 fps
- Distanz: 0.8..3.5m

- Probleme: Fenster & Spiegel

Das Robot Operating System

- Open Source Framework, Seit 2007
- Bestandteile: Werkzeuge, Bibliotheken, Konventionen

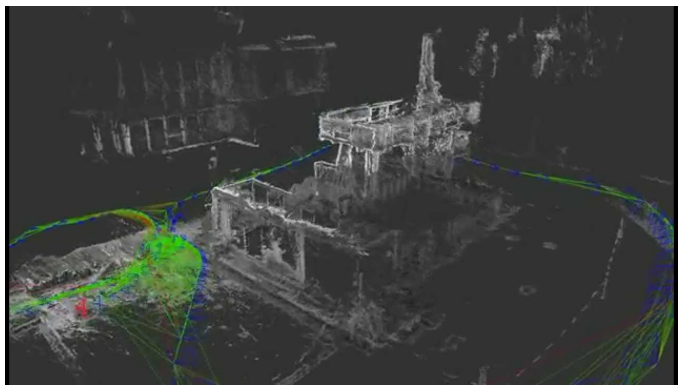
- Indigo Igloo
  - long-term support
  - Ubuntu Trusty (14.04 LTS)

- Kinetic Kame
  - Ubuntu Wily (15.10) & Ubuntu Xenial (16.04 LTS)

- Video: ROS Three Years

Wieso ROS?

- De-facto-Standard
- Wiederverwendbarkeit
- Ausgereiftere Software
- Viele Softwarepakete
- Nachvollziehbarkeit von Forschungsergebnissen..
  - Universitäten
  - Industrie (ROS-Industrial)
  - Hobby



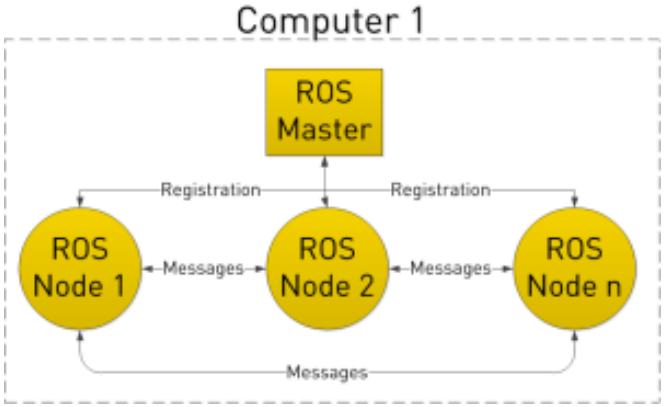
Large-Scale Direct Monocular SLAM

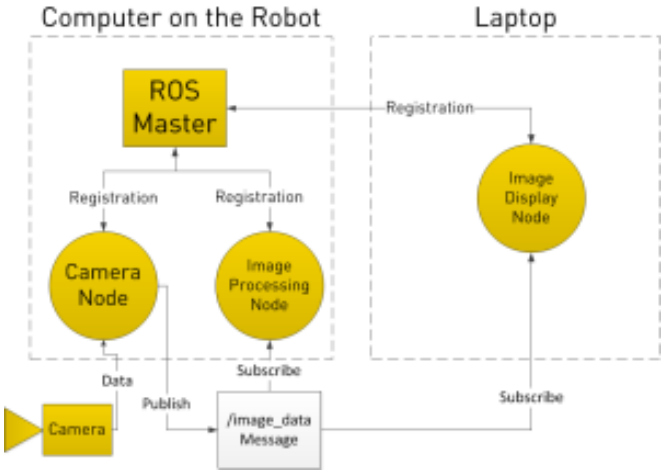
Installation

- ~~Installation from source~~
- Ubuntu Linux amd64
  - Virtuelle Maschine
  - Container
- Ubuntu Linux armhf
  - Raspberry Pi & etc.

Aufbau

- Middleware
- Interprozesskommunikation zwischen Knoten (Node)
- Jeder Knoten ist eine ausführbare Datei (C++, Python, ...)
- Modulares Design, über mehrere Computer
- Es gibt einen Master „roscore“ auf einem PC





- Nodelets: Node im selben Prozess (Zero-copy, nur C++)
- ROS2: DDS/RTPS

Kommunikationsart: Topics

- Publish-subscribe
- Für fortlaufende Datenströme (Sensoren, ...)
- Jeder zu jedem, jederzeit
- Callback
- Meistverwendete Kommunikationsart

Topic: geometry_msgs/Twist.msg

```
Vector3  linear
Vector3  angular
```

geometry_msgs/Vector3.msg

```
float64 x
float64 y
float64 z
```

Kommunikationsarten: Services & Actions

Services:

- Remote Procedure Calls
- Blockierend
- Für schnelle Berechnungen
- Beispiel: Addiere zwei Zahlen, initialisieren Daten

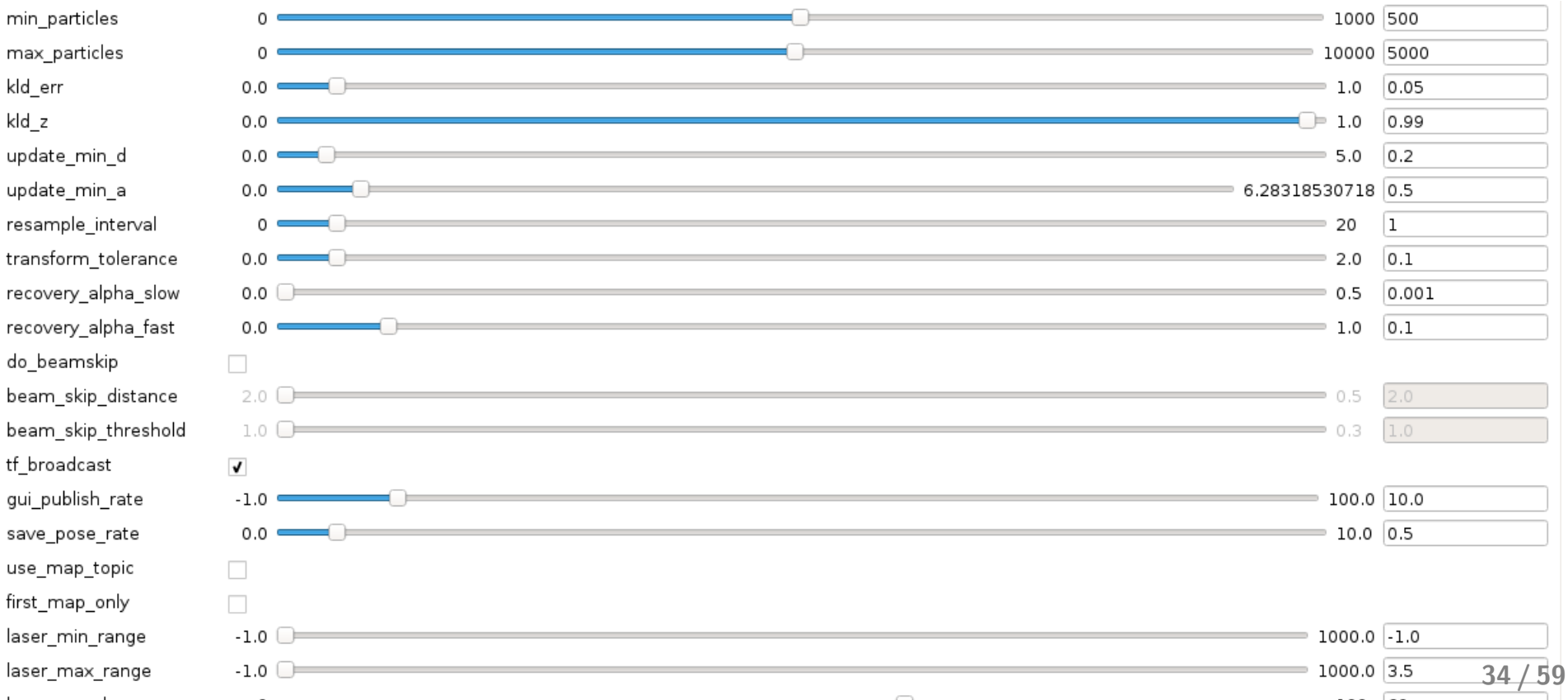
Actions:

- Längerlaufende Verhaltensweisen
- Können unterbrochen werden
- Stetige Rückmeldungen
- Beispiel: Bewege Roboter von A nach B

Viele Standard-Topics/Services/Actions definiert

Parameter

- rosparam für unveränderliche Werte
- dynamic_reconfigure für Werte die sich nur Laufzeit ändern
- rqt-GUI:

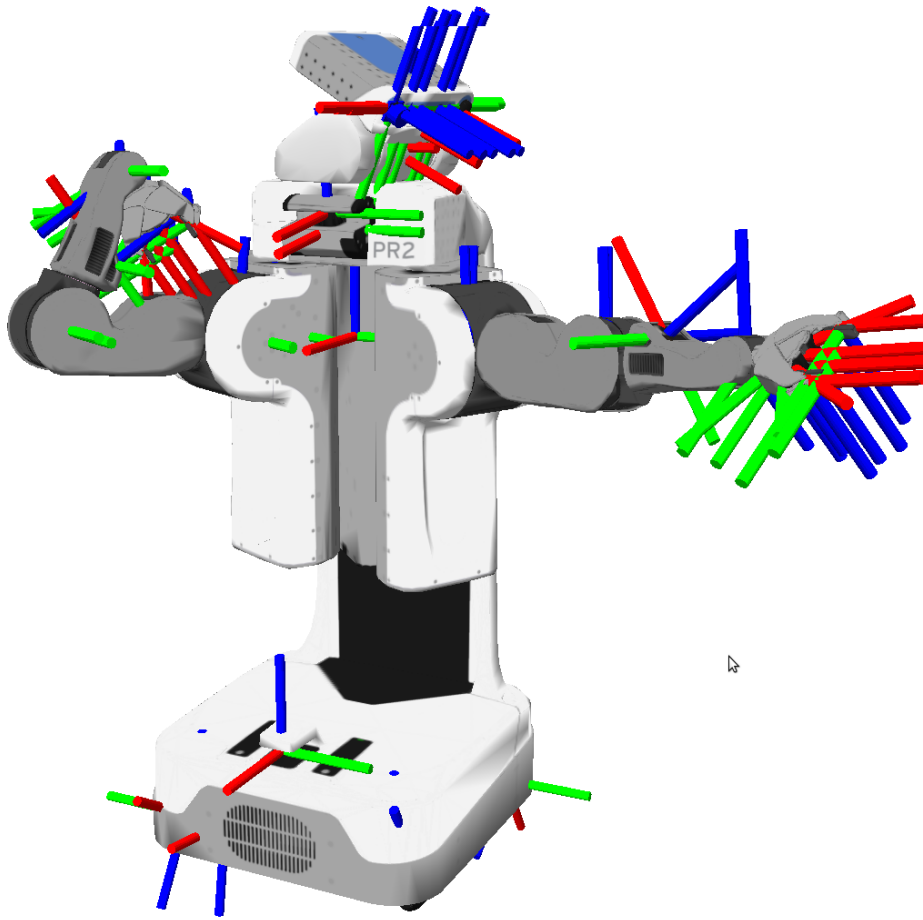


Einheiten & Konventionen

- Festgelegt in ROS Enhancement Proposal (REP) 103
- SI-Einheiten!
- Winkel in Radiant, mathematische Drehrichtung
- Drehungen im dreidimensionalen Raum durch Quaternionen
- Koordinatensystem lokale Karte „map“:
  - X: Osten
  - Y: Norden
  - Z: Aufwärts
- Koordinatensystem Körper „base_link“:
  - X: Vorwärts
  - Y: Links
  - Z: Aufwärts

Transform Library (TF)

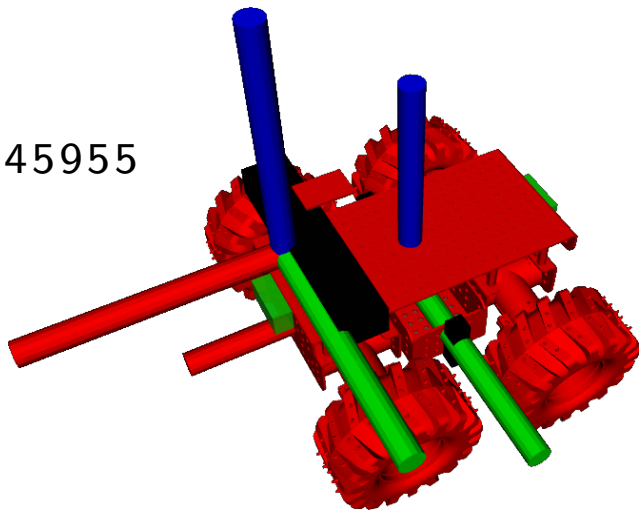
- Bibliothek zur 3D-Koordinatentransformation
- Jeder Roboter enthält mehrere Koordinaten
- Aktuelle Koordinaten werden als Nachricht versendet
- Beispiel: Wo befand sich mein Greifer vor 5s in bezug zur Karte?



Topic-Beispiel

Nachrichtenart:
sensor_msgs/LaserScan

```
header:
  seq: 12456
  stamp: secs: 1487871687, nsecs: 824445955
  frame_id: camera_depth_frame
angle_min: -0.509
angle_max: 0.509
angle_increment: 0.006
scan_time: 0.0329
range_min: 0.45
range_max: 10.0
ranges: [2.92, 2.91, ...]
```



- Nachricht wird unter dem Topic **scan** versendet
- tf-Frame der Nachricht ist **camera_depth_frame**

Roboter-Beschreibungssprache URDF

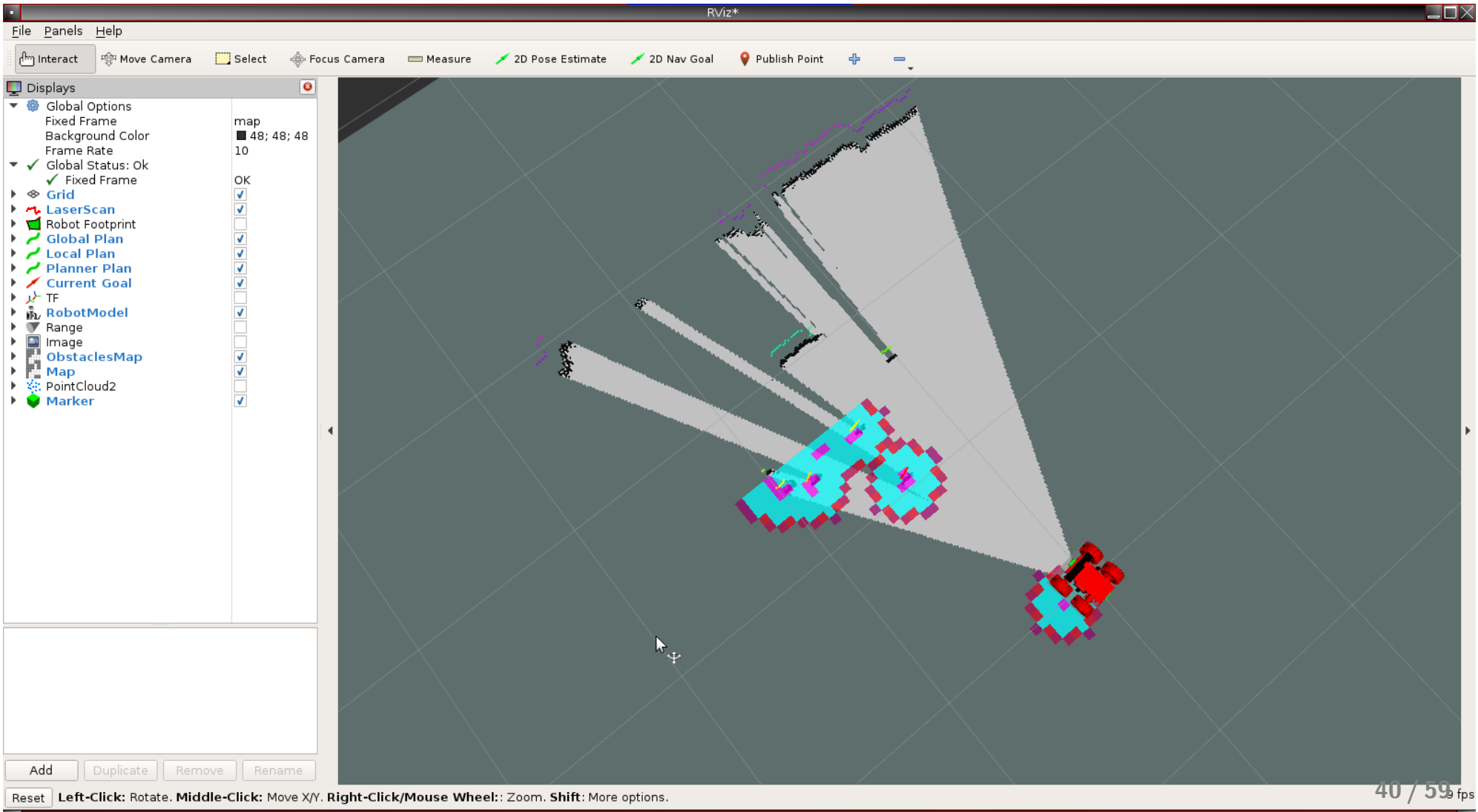
- Unified Robot Description Format
- XML-Format
- Link: Elemente des Roboters
- Joins: Bewegliche und starre Verbindungen zwischen Links
- robot_state_publisher: urdf-joins nach tf

```
<?xml version="1.0"?>
<robot name="myfirst">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2" />
      </geometry>
    </visual>
  </link>
</robot>
```

Werkzeuge (Auszug)

- rostopic
- rosbag: Aufzeichnung und Wiedergabe von Topics
- rviz: Grafische Darstellung: Roboter und Umgebung
- rqt: Div. GUI-Werkzeuge
  - rqt_console: Log Ausgaben
  - rqt_graph: Node-Diagramm
  - rqt_image_view
  - rqt_dynamic_reconfigure
  - rqt_plot: 2D Diagramm von Nachrichten

rviz


RViz*
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 10
Global Status: Ok
Fixed Frame OK
Grid
LaserScan
Robot Footprint
Global Plan
Local Plan
Planner Plan
Current Goal
TF
RobotModel
Range
Image
ObstaclesMap
Map
PointCloud2
Marker
Add Duplicate Remove Rename
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
9 fps

Launch-Dateien

- Starten mehrerer ROS-Nodes über eine Datei
- Laden von Konfigurationen aus YAML-Dateien
- Beispiel move_base.launch (Auszug):

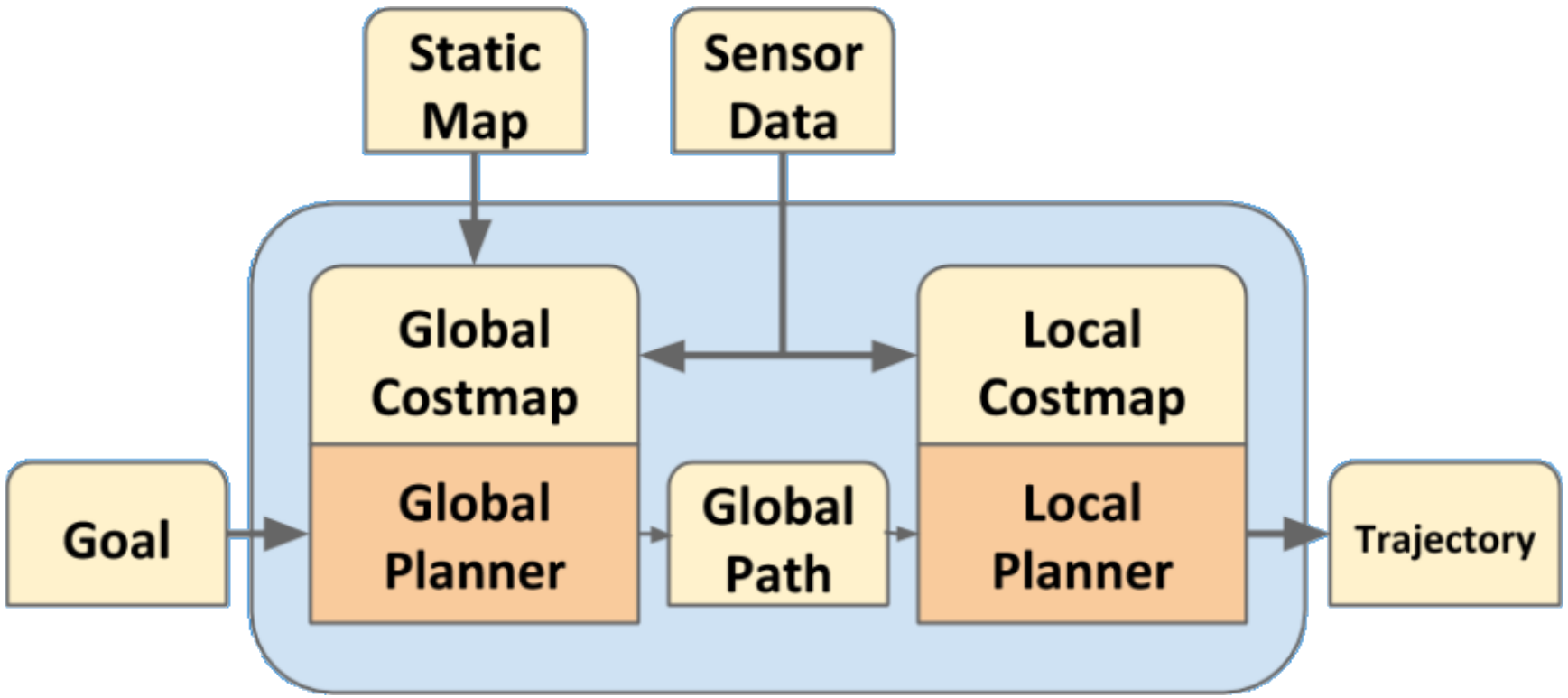
```
<?xml version="1.0"?>
<launch>
    <include file="launch/gmapping.launch" if="$(arg slam_gmapping)" />

    <node pkg="move_base" type="move_base" name="move_base" output="screen">
        <param name="controller_frequency" value="10.0" />
        <rosparam file="config/common_params.yaml" command="load" ns="global_costmap" />
        <rosparam file="config/common_params.yaml" command="load" ns="local_costmap" />
        <rosparam file="config/global_costmap_params.yaml" command="load" />
        <rosparam file="config/local_costmap_params.yaml" command="load" />
        <rosparam file="config/base_local_planner_params.yaml" command="load" />
        <remap from="/cmd_vel" to="move_base/cmd_vel"/>
    </node>
</launch>
```

MoveIt!

- ROS-Framework für mobile Manipulatoren
- Planung von Bewegungsabläufen
- Vorwärts- und Rückwärtskinematik
- 3D-Erkennung

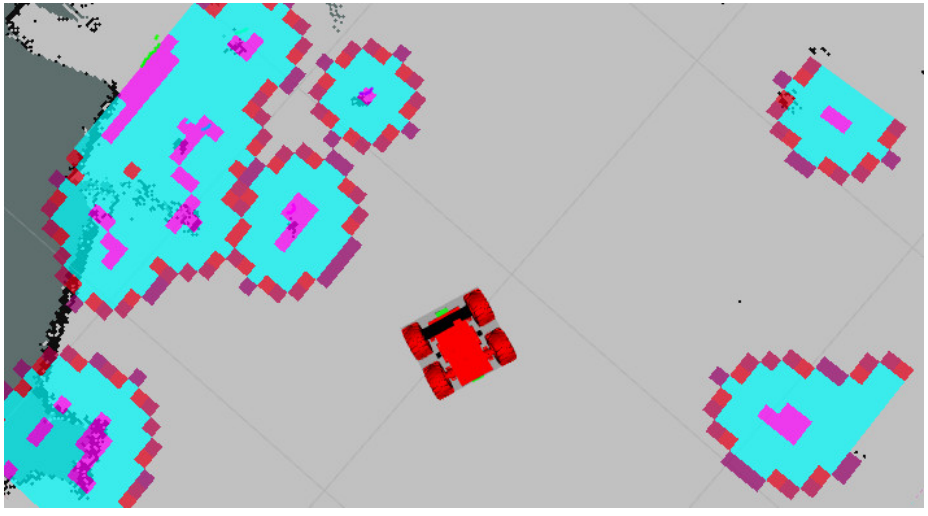
Navigation Stack: Überblick

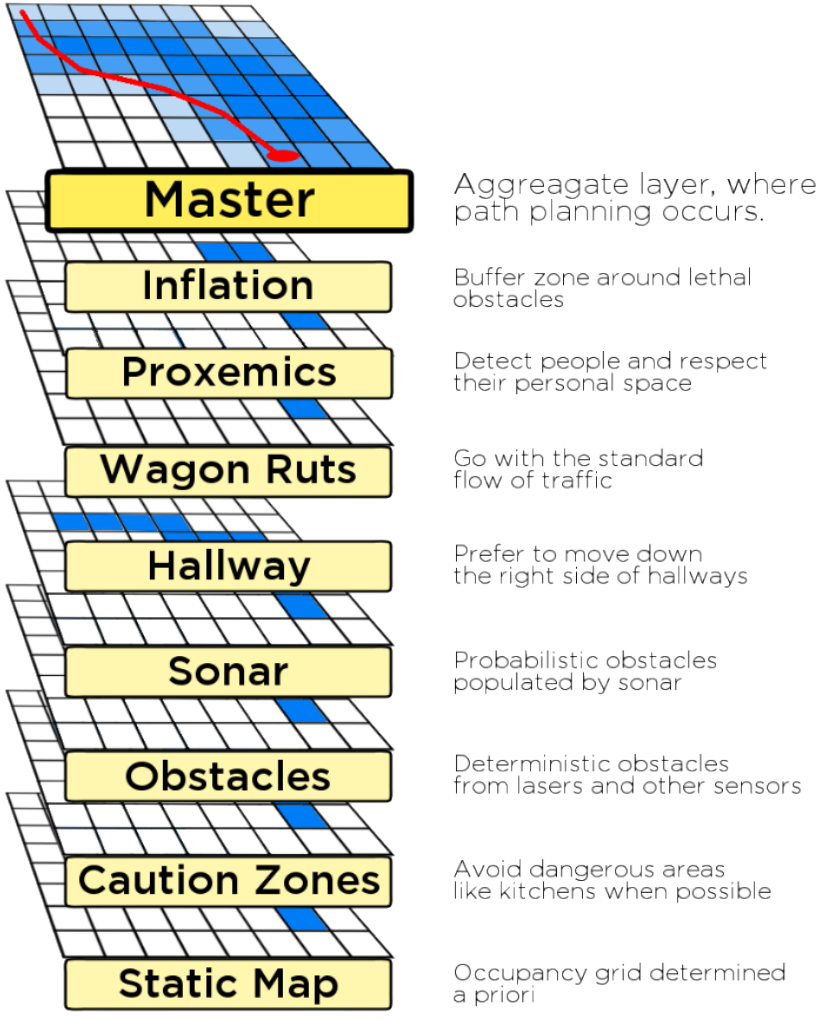
- Video: ROS Five Years



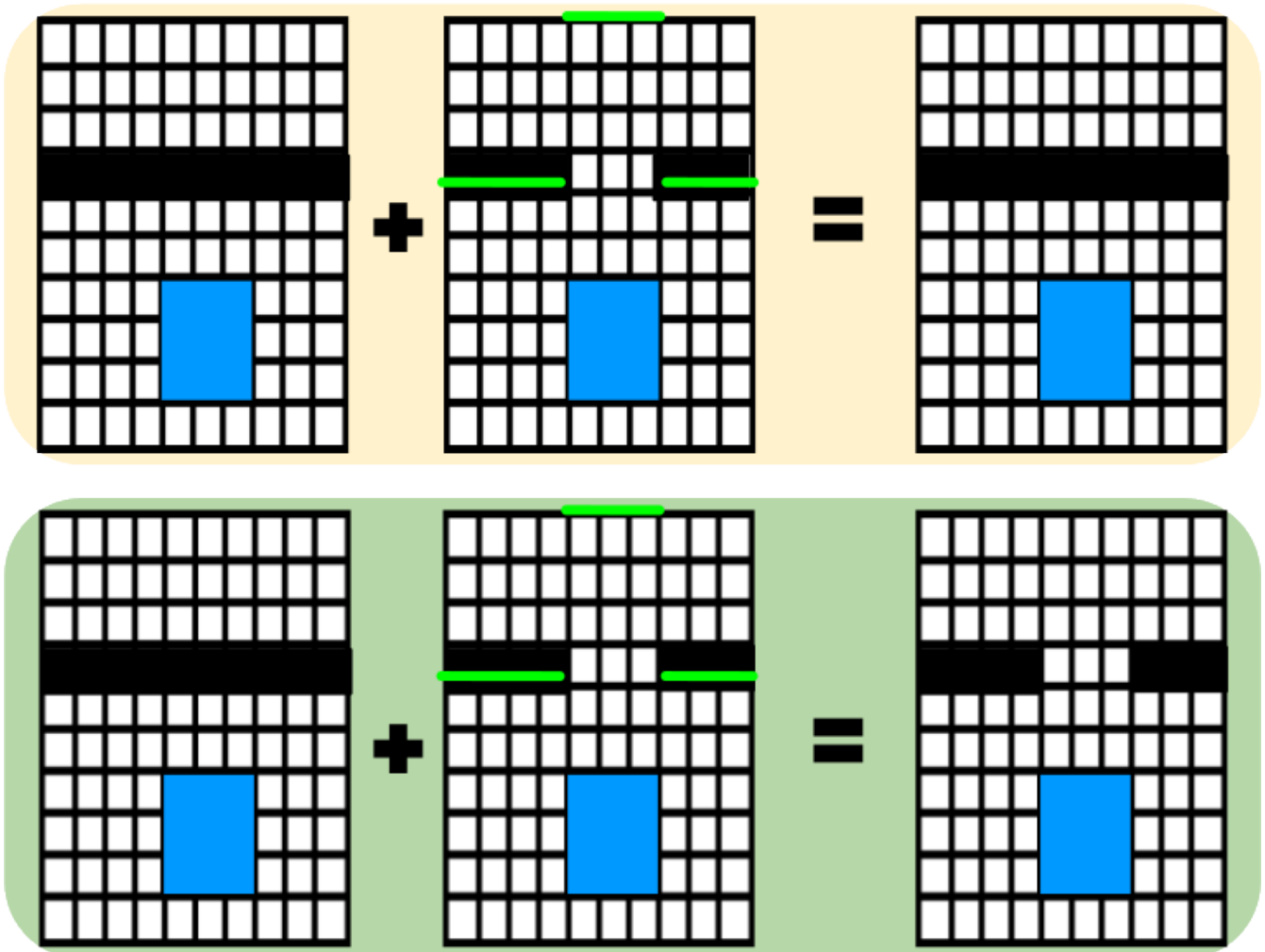
Layered Costmap: Kombinationen

- Global:
  - Static Layer
  - Inflation Layer
- Lokal:
  - Obstacle Layer (Laser)
  - Range Layer (US & IR)
  - Inflation Layer





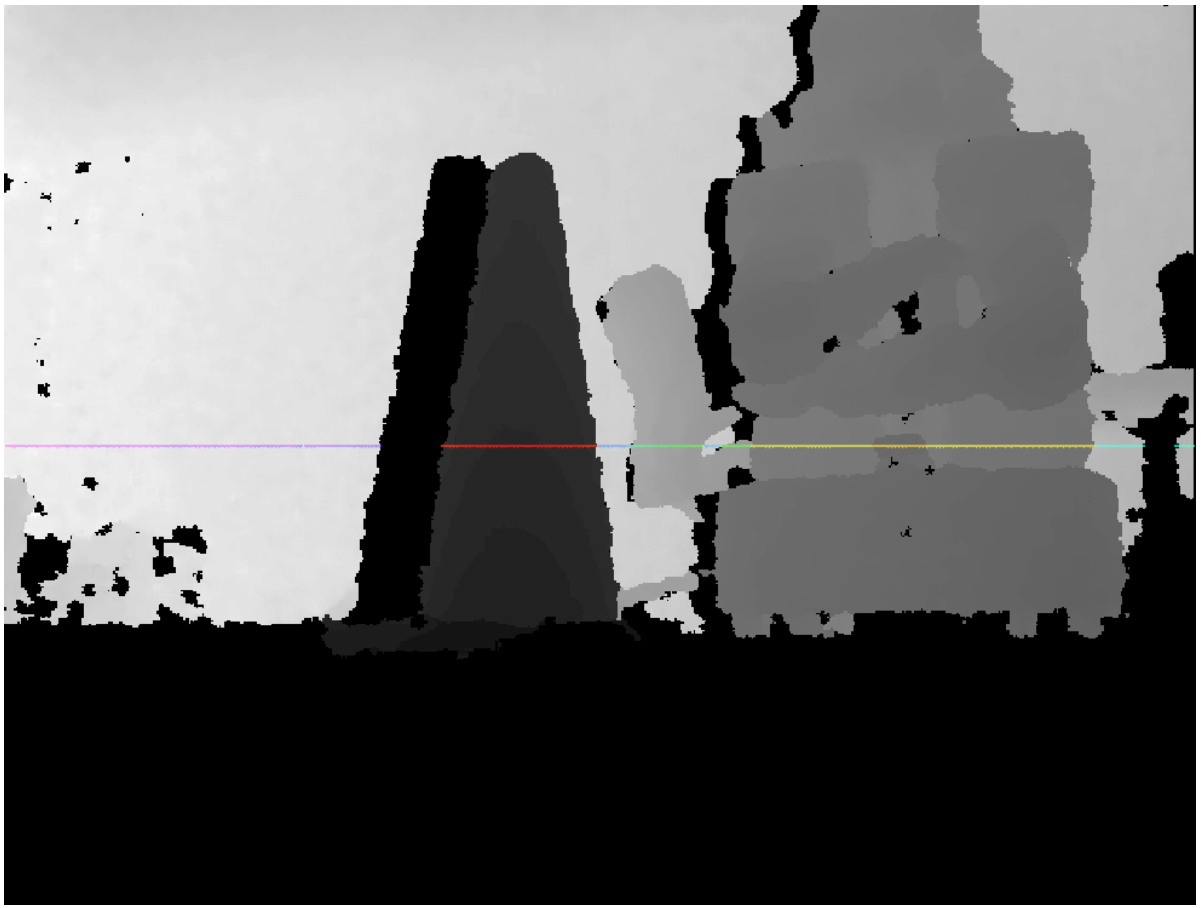
Layered Costmap: Kombinationen



Sensor source: depthimage_to_laserscan

- Erzeugt „Fake“-Laserscan aus 3D-Bildern

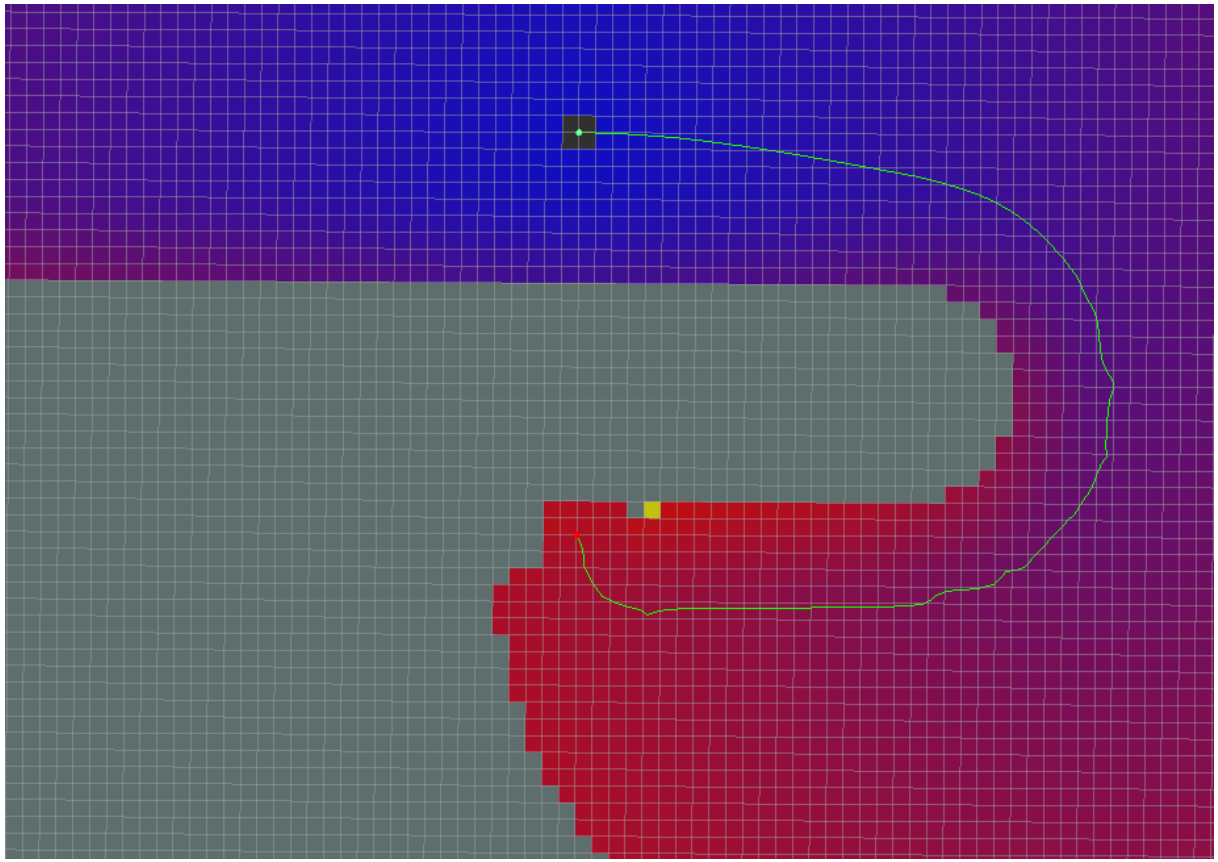
Motiv



Depthimage mit Laserscan

global_planner

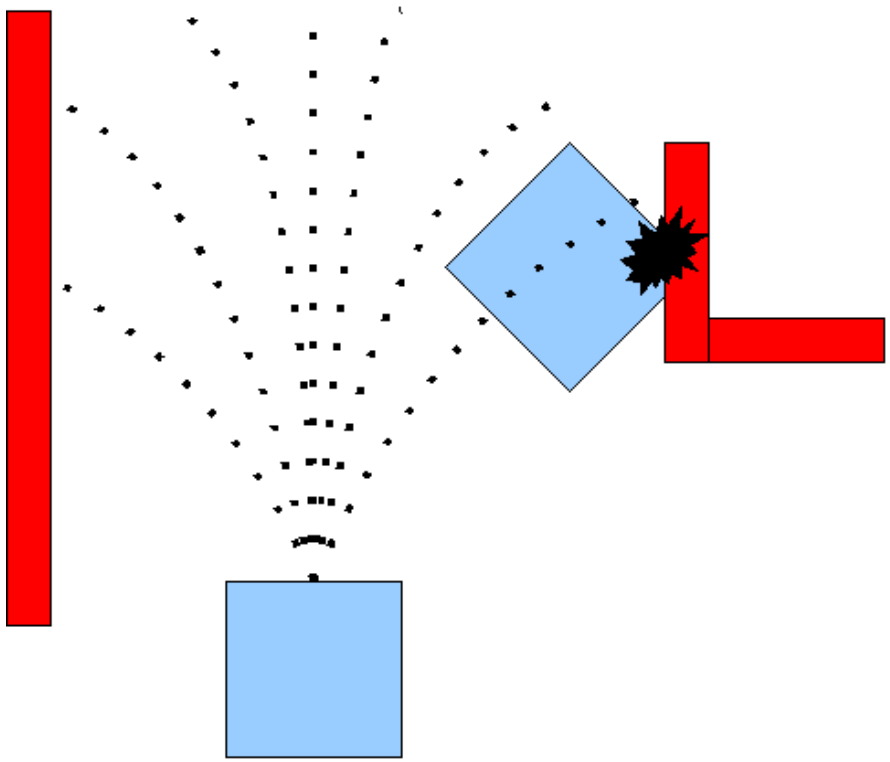
- Dijkstra oder A*
- Erzeugt „Global Plan“

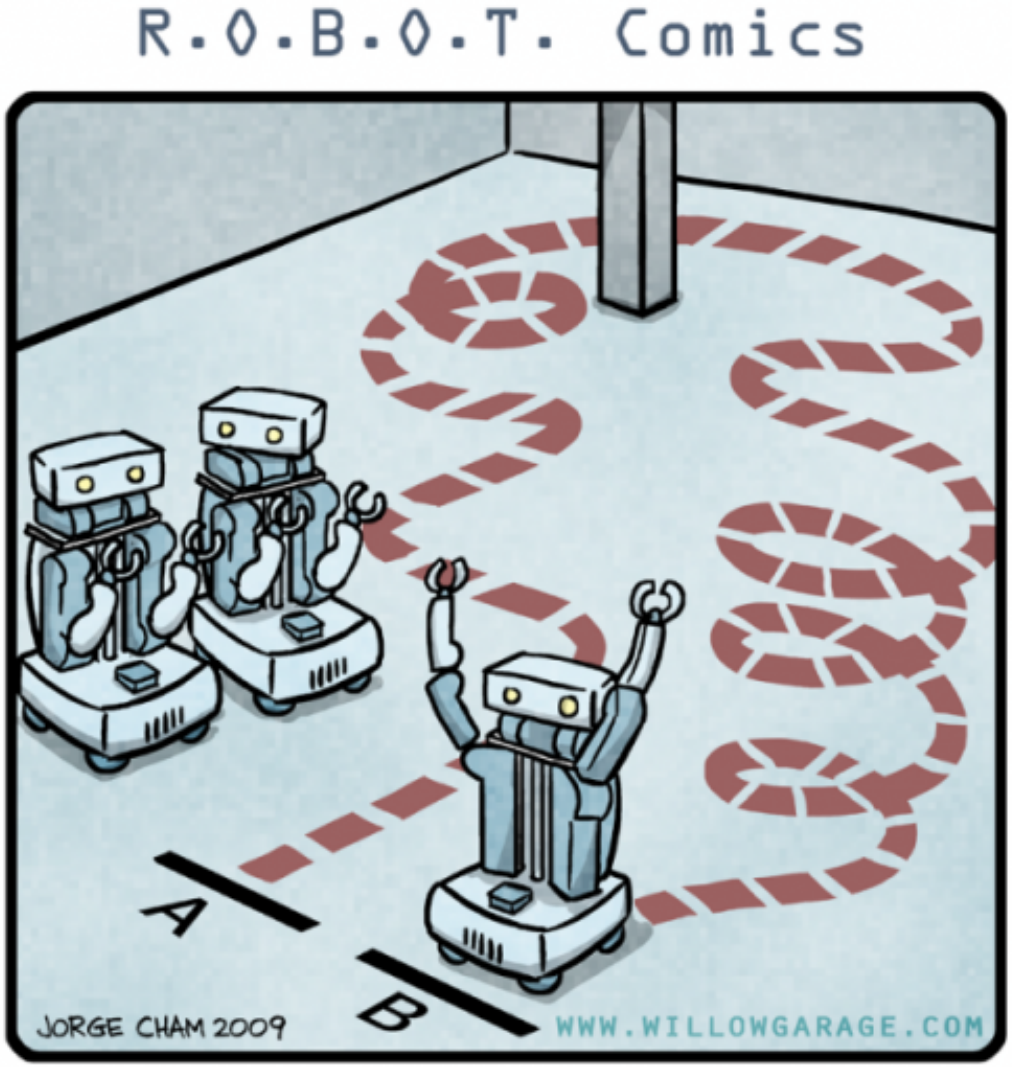


base_local_planner

- Trajectory Rollout oder Dynamic Window Approach

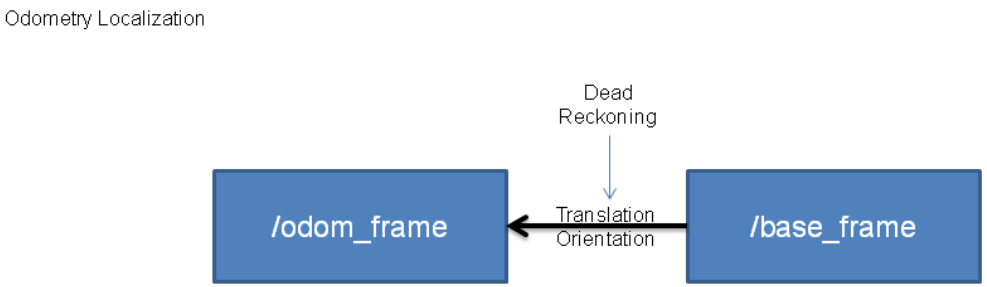
1. Stichprobenweise die möglichen Geschwindigkeiten im Steuerraum abtasten
2. Vorwärtssimulation für jeden Weg berechnen
3. Wege bewerten (Nähe zu Hindernissen/Ziel) und illegale Wege verwerfen
4. Besten Weg verwenden

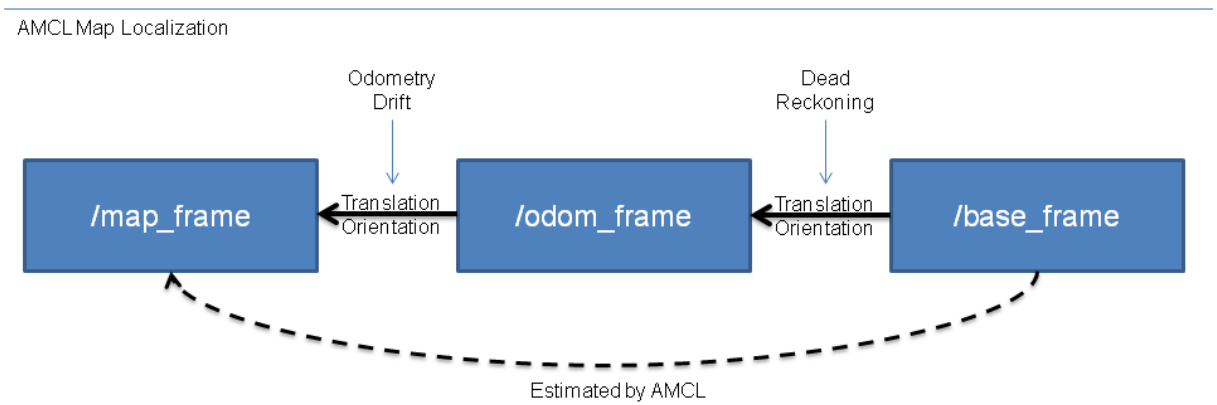



R.O.B.O.T. Comics
A
B
JORGE CHAM 2009
WWW.WILLOWGARAGE.COM
"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

amcl

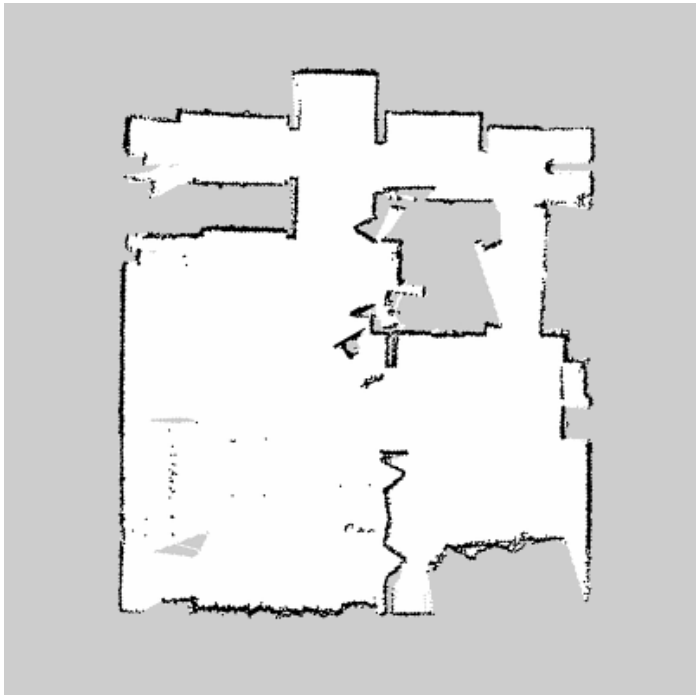
- Adaptive Monte Carlo Localization
- Verfolgt die Position des Roboters anhand einer bekannten Karte





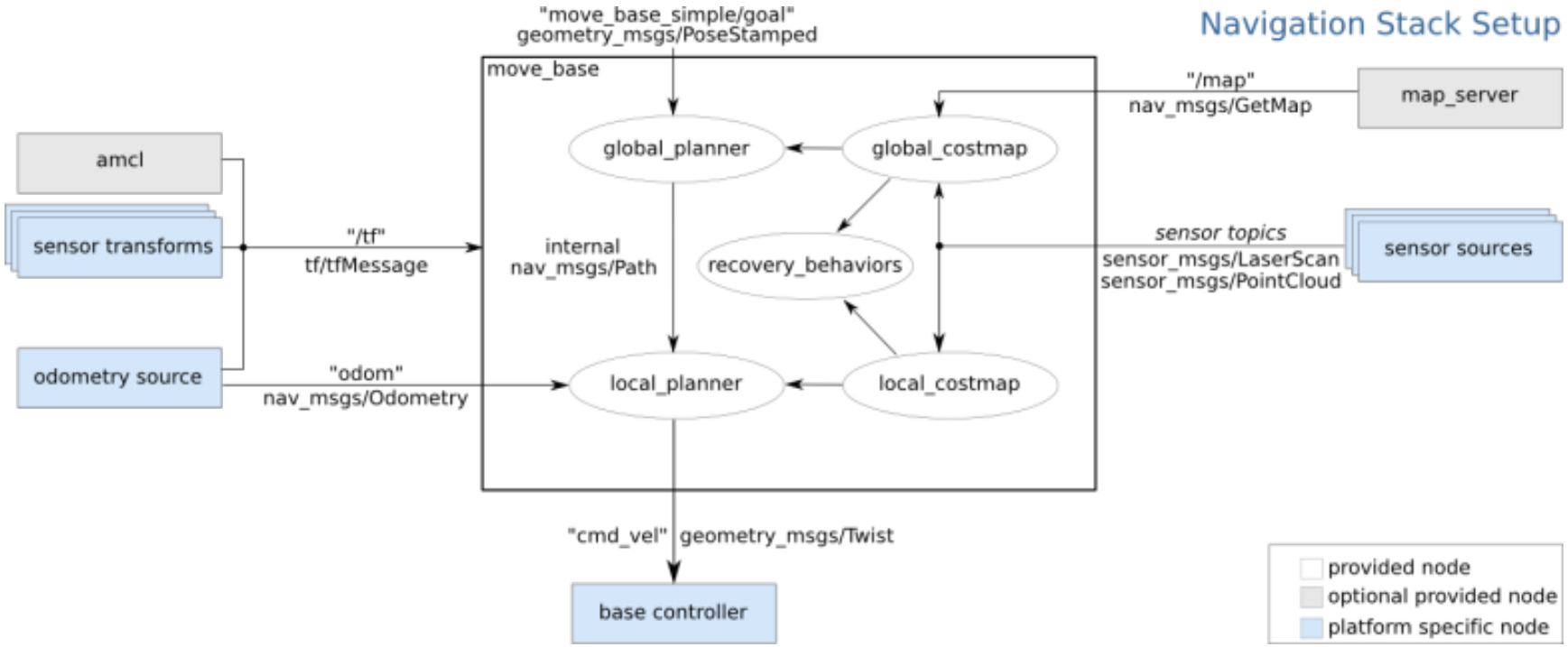
SLAM

- Simultaneous Localization and Mapping
- Erzeugt „Gebäudegrundrisse“
- OpenSlam Gmapping

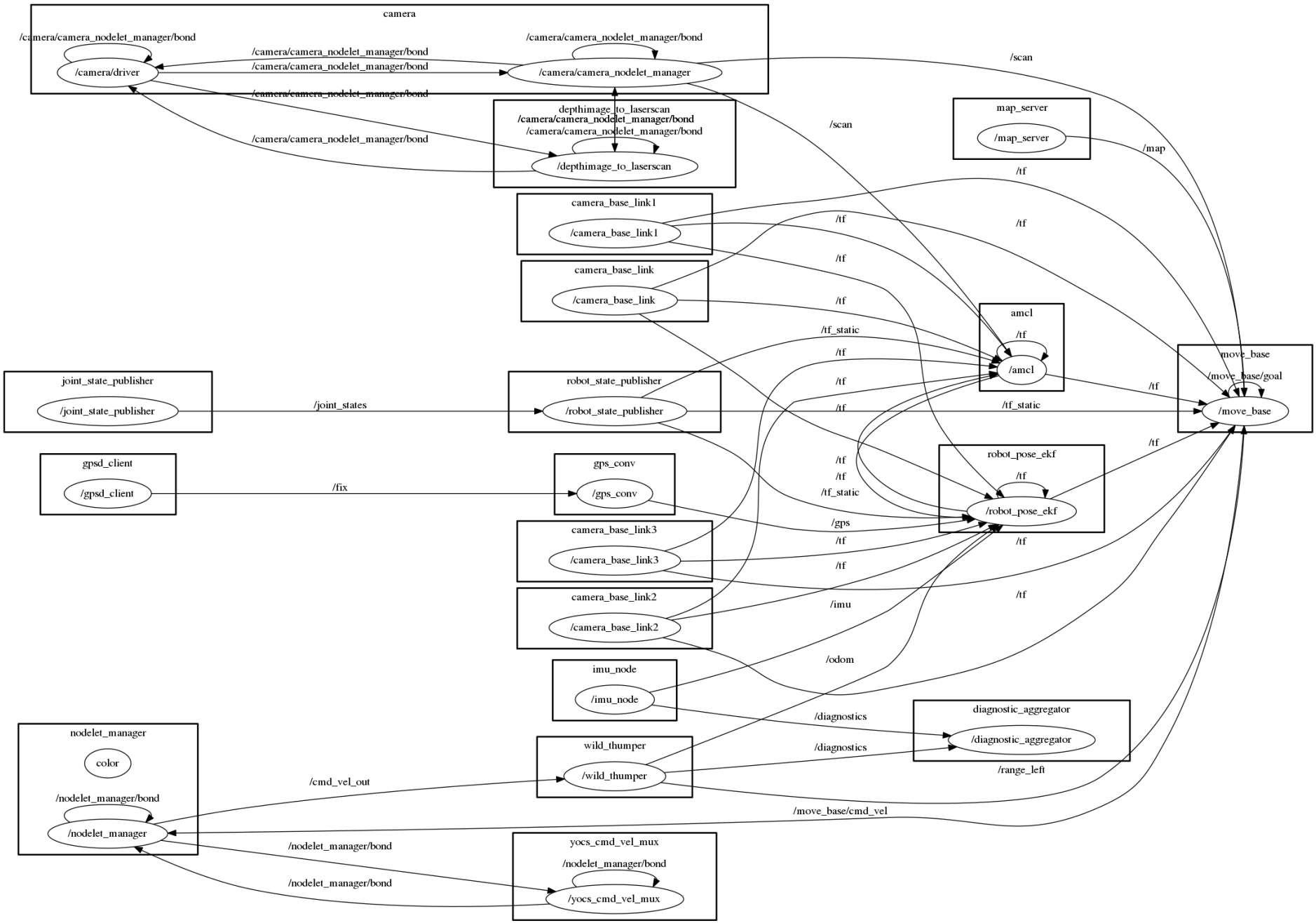


  - Hector Mapping
    - Schnelle Aktualisierungsrate (40Hz) von LIDARs
    - Ohne Odometrie
  - Google Cartographer

Navigation Stack: Aufbau



Node-Graph


camera
/camera/camera_nodelet_manager/bond
/camera/driver
/camera/camera_nodelet_manager/bond
/camera/camera_nodelet_manager/bond
/camera/camera_nodelet_manager/bond
/camera/camera_nodelet_manager/bond
/camera/camera_nodelet_manager/bond
/camera/camera_nodelet_manager
/scan
depthimage_to_laserscan
/camera/camera_nodelet_manager/bond
/camera/camera_nodelet_manager/bond
/depthimage_to_laserscan
/scan
map_server
/map_server
/map
/tf
camera_base_link1
/camera_base_link1
/tf
/tf
camera_base_link
/camera_base_link
/tf
/tf_static
amcl
/tf
/amcl
/tf
/tf
move_base
/move_base/goal
/move_base
/tf
/tf_static
joint_state_publisher
/joint_state_publisher
/joint_states
robot_state_publisher
/robot_state_publisher
/tf
/tf
gpsd_client
/gpsd_client
/fix
gps_conv
/gps_conv
/tf
/tf
/tf_static
robot_pose_ekf
/tf
/robot_pose_ekf
/gps
/tf
/tf
camera_base_link3
/camera_base_link3
/tf
/tf
camera_base_link2
/camera_base_link2
/imu
/tf
/odom
imu_node
/imu_node
/diagnostics
diagnostic_aggregator
/diagnostic_aggregator
/diagnostics
/range_left
wild_thumper
/wild_thumper
/cmd_vel_out
nodelet_manager
color
/nodelet_manager/bond
/nodelet_manager
/move_base/cmd_vel
/nodelet_manager/bond
/nodelet_manager/bond
yocs_cmd_vel_mux
/nodelet_manager/bond
/yocs_cmd_vel_mux

ROS-Hilfe

- Wiki & Tutorials: `http://wiki.ros.org`
- Bücher
- Fragen & Antworten: `http://answers.ros.org`

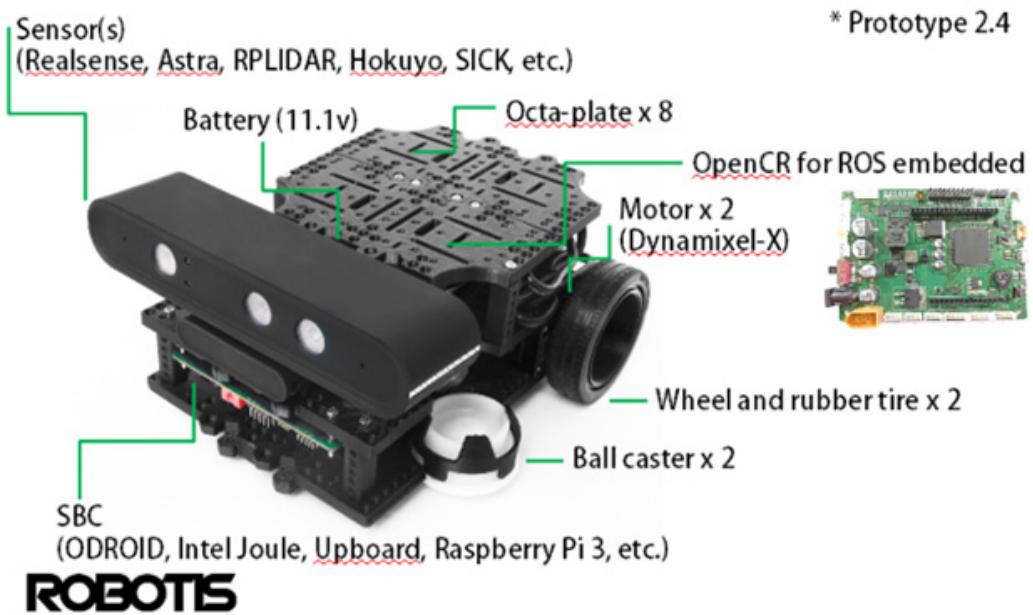
- Video: ROS Eight Years

Überlegungen zum eigenen Roboter

1. Antrieb und Grundgerüst dimensionieren
2. Akku dimensionieren
3. Sensorabdeckung
4. Verwendet ROS!
5. Hinweis: ROS kein Low-Level-Layer!

Ausblick: Turtlebot 3

- Open-Source Plattform
- 3D-Druckbare Bauteile
- April 2017
- Modular: `http://youtu.be/r3oNIWex8a0`
- `http://turtlebot3.robotis.com`



Fragen?