Robotikwerkstatt DokuWiki

Das Projekt wurde im SS2017 im Zuge eines FWP2 Faches begonnen.
Bei Fragen an: Robert Willeit [mailto:willeit@hm.edu]

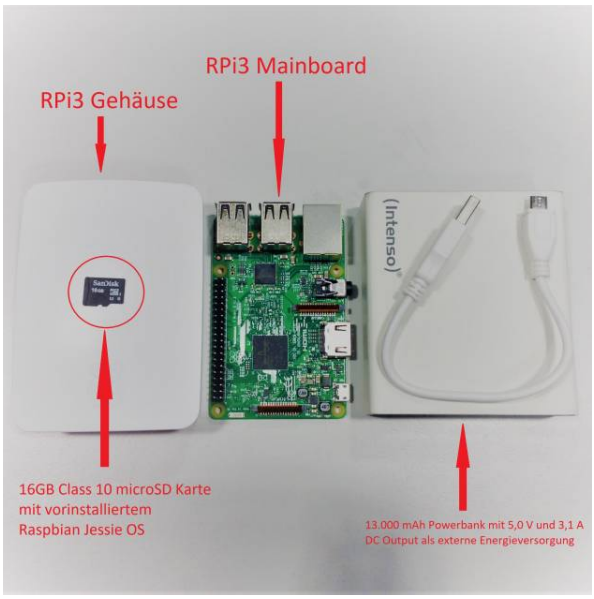
Source Code

src WLAN und BLE
src Visualisierung der Position mittels MATLAB TCP/IP Server
Hinweis zum Source Code. Bitte durchlesen!

Aufgabenstellung

Ziel des Projektes war es, ein System zu entwickeln, mit welchem man sich anhand von Wlan oder Bluetooth in einem Raum orientieren kann. Hierbei sollten speziell die Signalstärken der genannten Funknetzwerke ermittelt werden, um daraus die Position zu bestimmen.

Hardwareübersicht



Als Empfänger und zur Datenverarbeitung wurde ein Raspberry Pi 3 (RPi3) gewählt, da dieser mit dem direkt auf der Platine verlöteten BCM43143 [http://www.cypress.com/file/298756/download] Chip ausgestattet ist und somit sowohl Wireless LAN als auch Bluetooth LE (Low Energy) unterstützt.

Ein weiterer Grund für die Wahl des RPi's ist die Möglichkeit Linux als Betriebssystem zu installieren und die dadurch bereitgestellten Treiber für Wireless LAN/Bluetooth LE zu nutzen. Die Benutzung von Linux Funktionen und Treibern bringt zudem eine Kompatibilität zu anderen Systemen auf denen Linux läuft, wodurch eine Fortführung des Projektes nicht auf die Verwendung des RPi's angewiesen wäre.

Für die Positionsbestimmung mit Bluetooth wurden sogenannte Bluetooth Beacons [https://de.wikipedia.org/wiki/IBeacon] verwendet. Diese arbeiten mit der stromsparenderen Bluetooth Low Energy (BLE) [https://de.wikipedia.org/wiki/Bluetooth_Low_Energy] Technologie und senden in regelmäßigen Abständen Signale aus, welche analysiert werden können. Die Beacon können u.a. mit der Beacon CFG (Google Android) [https://play.google.com/store/apps/details?id=com.yunliwuli.beaconcfg] oder BeaconSET (Apple iOS) [https://itunes.apple.com/de/app/beaconset/id1052655664?mt=8] App konfiguriert werden (z.B. Reichweite,ID, Name, Sendeintervall etc.).

Hardwareeinrichtung

- Raspberry Pi 3 (RPi3) als Empfänger und zur Datenverarbeitung

Der Raspberry Pi 3 wurde mit vorinstalliertem Raspbian Jessie Betriebssystem auf mitgelieferter microSD Karte ausgestattet. Damit kann man die microSD Karte gleich einstecken und den RPi3 sofort hochfahren. Falls man Raspbian lieber auf der eigenen microSD Karte (Class 10 microSD ist für ein ruckelfreie System notwendig) haben will, gibt es dann die passende Anleitung dazu:

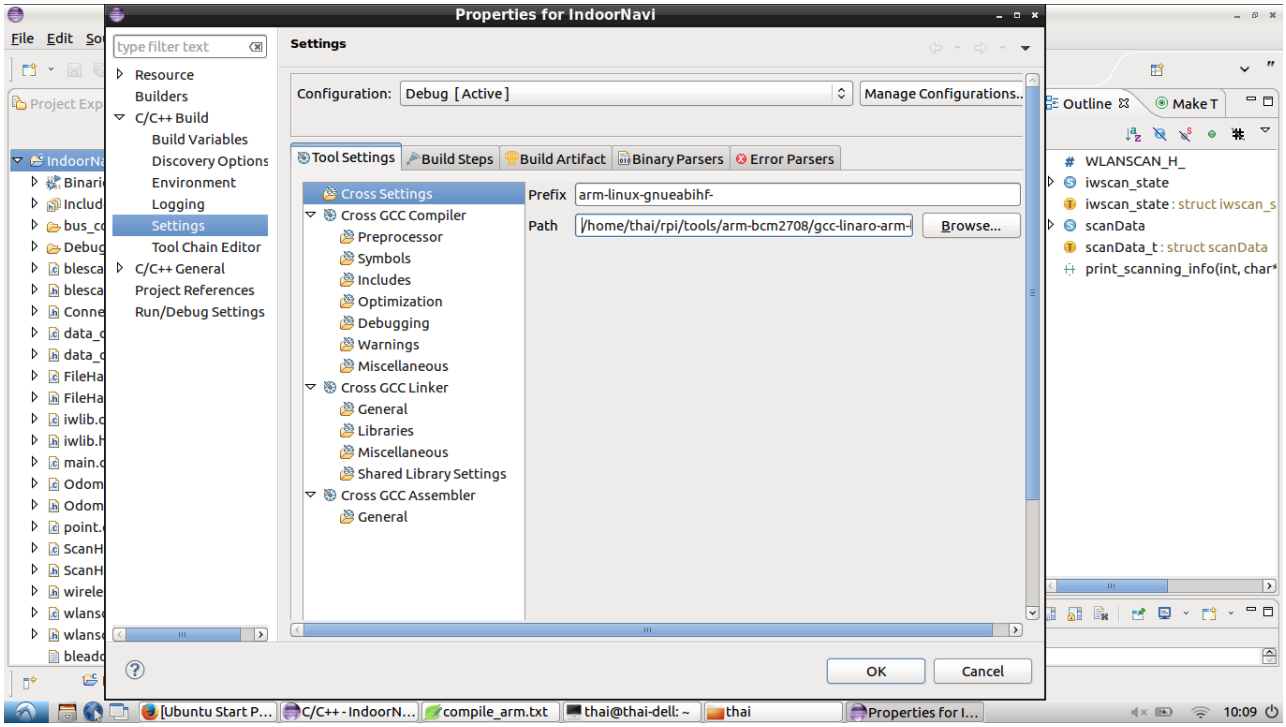
1. Jessie Abbild herunterladen (empfohlen) [http://downloads.raspberrypi.org/raspbian/images/raspbian-2017-07-05/]
2. Abbild auf microSD installieren (auf Englisch) [https://www.raspberrypi.org/documentation/installation/installing-images/]

Cross Compile

Für die Softwareentwicklung für den RaspberryPi wurde Lubuntu 16.04 LTE Betriebssystem auf ein Laptop installiert. Auf dieser kann dann die Toolchain aus der github Seite für den RPi installiert werden (Achtung, github muss vorher auf Linux installiert werden!):

```
git clone git://github.com/raspberrypi/tools.git
```

In Eclipse kann nun *arm-linux-gnueabihf-* als Compiler-Präfix und Linker (statt normal gcc) und */pfad_des_werkzeugs/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/* als Compiler-Path unter *Rechtsklick Projekt/**Properties/ C/C++ Build /Settings** eingestellt werden:



Die so erzeugten Executables sollten nun auf dem RaspberryPi ausführbar sein(aber **nicht** mehr auf dem Laptop/Rechner!).

Die komplette Anleitung zum Cross-Compile gibt es auf hertaville.com (auf Englisch): Einrichtung [http://www.hertaville.com/development-environment-raspberry-pi-cross-compiler.html]

Auf den RPi kriegt man die Datei u.a. mit dem Kommando scp [http://www.hypexr.org/linux_scp_help.php]

```
scp /pfad_zur_datei/datei_name pi@ip_address:/pfad_wo_es_hin_soll/
#Beispiel
scp /home/user/workspace/my_project/Debug/my_exe pi@10.179.1.13:/pi/home/
```

Um Raspberry Pi in dem selben Netzwerk wie bei dem Laptop zu verbinden, muss die *wpa_supplicant.conf* Datei mit der folgenden Einstellung umgeändert werden:

Security: **WPA&WPA2 Enterprise**
Authentification: **Protected EAP (PEAP)**
Anonymous identity: hm.edu Hochschul-Mail-Passwort
CA certificate: **no certificate required**
PEAP Version: **Version0**
Inner authentification: **MSCHAPv2**
Username: **user@hm.edu**
Password: **hm.edu Hochschul-Mail-Passwort**

Die komplette Anleitung stellt LRZ auf ihrer Webseite zur Verfügung: eduroam unter Linux (wpa_supplicant) [http://man7.org/linux/man-pages/man2/ioctl.2.html]

Wlan und Bluetooth Treiber

Linux stellt bereits Treiber für Wlan und Bluetooth bereit. Mittels eines IOCTL [http://man7.org/linux/man-pages/man2/ioctl.2.html] Systemaufrufes kann mit dem Treiber kommuniziert werden:

```
// ioctl Syntax
int ioctl(int fd, unsigned long request, char* argp);
```

fd: offener File Descriptor
request: Gerätespezifischer Code
argp: Pointer auf Speicherbereich

```
#include <sys/ioctl.h>
//...
int skfd; //File Descriptor für Socket
struct iwreq wrq; //Struktur für Treiberparameter
skfd = iw_sockets_open(); //Öffnet einen neuen Socket
char  buffer[buflen]; /*buflen = IW_SCAN_MAX_DATA = 4096 (sh. wireless.h*/

/* wrq wird mit Parametern beschrieben */
wrq.u.data.pointer = buffer; //Für Scan Ergebnisse
wrq.u.data.flags = 0; //Spezielle Flags
wrq.u.data.length = buflen;
strcpy(wrq.ifr_name,"wlan0");

ioctl(skfd, SIOCGIWSCAN, &wrq);
// ...
```

Das obige Beispiel zeigt den ioctl-Aufruf um die Wlan Scan Ergebnisse zu bekommen.

```
// Definition der requests in wireless.h
#define SIOCSIWSCAN     0x8B18          /* trigger scanning  */
#define SIOCGIWSCAN     0x8B19          /* get scanning results */
```

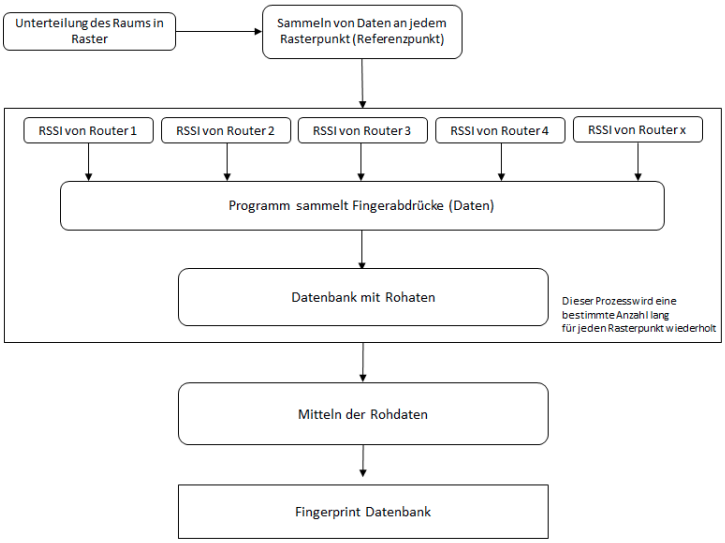
Der im Projekt geschriebene Code ist zum Teil eine Abwandlung von Linux's iwlist [https://linux.die.net/man/8/iwlist] und hcitool [https://linux.die.net/man/1/hcitool].

Algorithmik

Grundidee - WLAN-Positionierung

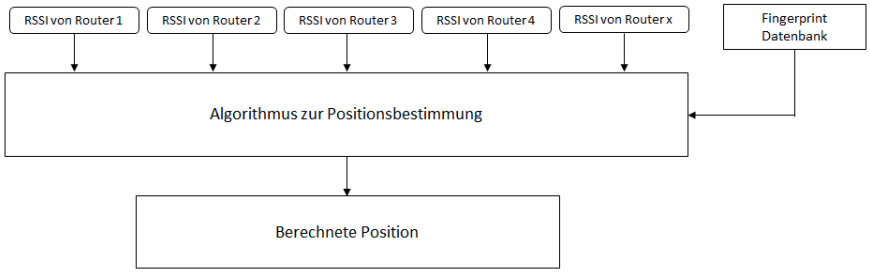
Für die Positionierung mittels WLAN wurde das sogenannte Fingerprinting-basierte Verfahren eingesetzt. Dieses Verfahren unterteilt sich in zwei Phasen.

In der ersten Phase, der Offlinephase bzw. Kalibrierungsphase werden alle empfangbaren Router sowie deren Signalstärken an eigen festgelegten Referenzpunkten gemessen und diese mit der zugehörigen Position des Punktes in einer Datenbank, in unserem Fall, das Textdokument gespeichert.



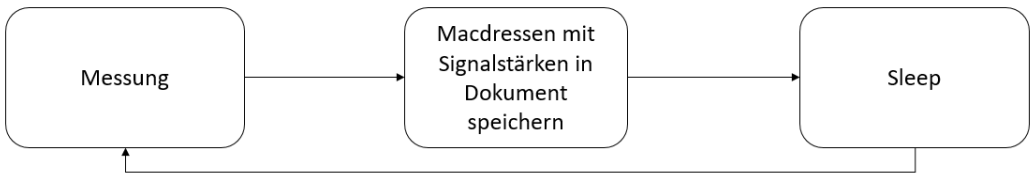
In der zweiten Phase, der Onlinephase bzw. Lokalisierungsphase werden aktuelle Signalstärkewerte gemessen und diese mit der Datenbank verglichen. Somit kann die Position mit Hilfe der Fingerabdrücke bestimmt werden.

Da die Genauigkeit hier stark vom verwendeten Raster abhängt, wurde anschließend zwischen den Referenzpunkten interpoliert. Somit ist die Berechnung der Position nicht mehr direkt vom Raster abhängig.



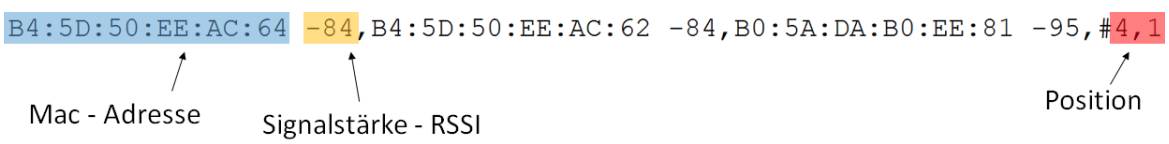
Vermessung

Um eine Positionierung zu garantieren, müssen eindeutige Daten an den jeweiligen Standorten (Referenzpunkten) des Raumes erhoben werden. Dies geschieht mittels Signalstärkenmessung über den WLAN beziehungsweise den Bluetooth Empfänger des Raspberry Pi`s. Dabei werden am jeweiligen Standort (Eingabe der zur vermessenden Position beim Start) alle empfangbaren Geräte mit deren Macadresse und Signalstärke gemessen und anschließend mit der zugehörigen Position in einer Reihe in einem Textdokument abgespeichert.



Format der Datenbank

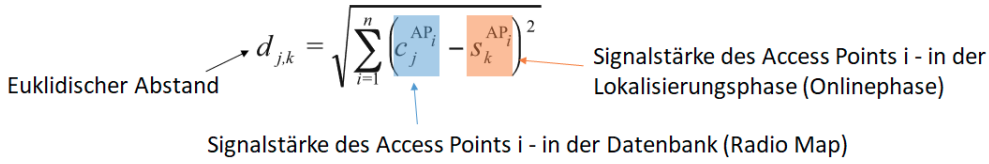
Nach der Vermessung hat jeder Eintrag der Datenbank folgendes Format:



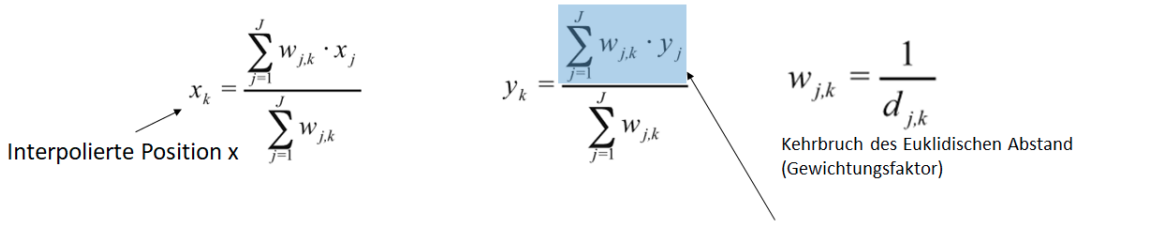
Positionierung

Nachdem nun auch eindeutige Positionsdaten vorhanden sind, kann jetzt die Lokalisation in dem zuvor vermessenen Raum stattfinden. Dafür werden zuallererst die Daten aus dem Textdokument geladen und an ein Struct mit einem Array übergeben. Nun kann die Navigation erfolgen. Hierbei wird zuerst, wie bei der Vermessung eine Scan der W-Lan/Bluetooth Signale gestartet. Diese werden anschließend mit den Daten aus dem Textdokument verglichen.

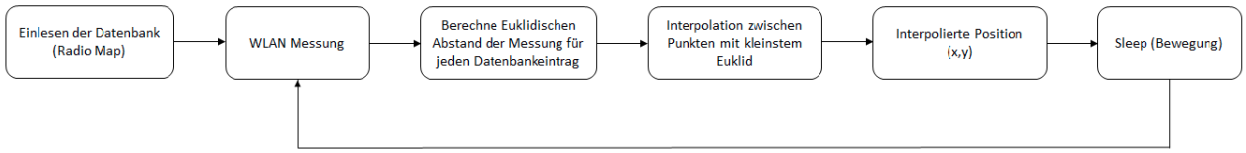
Hierzu wird der euklidische Abstand für jeden Datenbank-Eintrag berechnet. Dies ist ein Maß für die Differenz zwischen den Signalstärkewerten aus der Lokalisierungsphase und denen aus der Kalibrierungsphase.



Nach der Berechnung, gibt es Positionen mit minimalen Euklidischen Abstand. Nur wird zwischen diesen Positionen interpoliert.

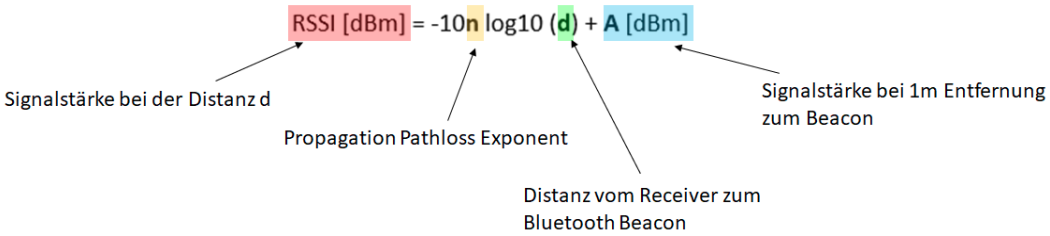


Das Ergebnis ist nun eine interpolierte Position des Empfängers. Nachdem die Position berechnet worden ist, wird wiederum eine Bewegung durch einen Sleep simuliert. Dieser Vorgang wird bis zum Durchlaufen der Schleife wiederholt.

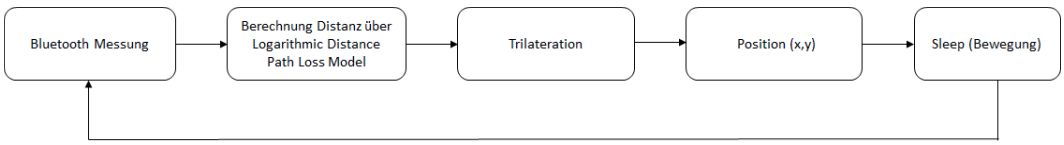


Grundidee - Bluetooth-Positionierung

Bei der Positionierung mit Bluetooth wird nicht wie bei WLAN mit dem Fingerprint Verfahren vorgegangen, sondern hier wurde mit Trilateration gearbeitet. Um die Entfernungen zu den Beacons zu bestimmen, wurde das Logarithmic Distance Path Loss Model verwendet.



Anschließend wird mit Trilateration die Position des Receivers bestimmt. Die Koordinaten der Beacons kann im Code festgelegt werden. Nachdem die Position gefunden worden ist, wird wiederum eine Bewegung durch einen Sleep simuliert. Dieser Vorgang wird bis zum Durchlaufen der Schleife wiederholt.



Ergebnis

WLAN

Bei der WLAN Positionierung wurde als Testumgebung ein 3×7 Punkraster (28 Referenzpunkte) im Labor realisiert. Mit dem Raspberry Pi wurden im Schnitt 10-15 Access Points pro Referenzpunkt registriert. Die empfangenen Signalstärken lagen dabei zwischen -50 dBm (stärkste) und -90 dBm (schwächste).
Bei der Positionierung in der Testumgebung konnte mit Interpolation eine Genauigkeit von +- 4m erreicht werden. Gründe für solche Abweichungen liegen an den geringen Abständen zwischen zwei Referenzpunkten (1 Meter) sowie den Mehrwegeffekten (andere Objekte stehen auf dem Punktraster).

Bluetooth

Die Positionierung mittels Bluetooth Signalstärken und Trilateration erbrachte eine Genauigkeit von knapp 4 Meter. Leider funktioniert derzeit die Bluetooth Positionierung nur auf dem Rechner, da einige Bibliotheken beim Kompilieren nicht erkannt werden.

Aussichten

- Testumgebung

Um eine bessere Genauigkeit bei der WLAN Positionierung zu erreichen, sollte eine bessere Testumgebung gewählt werden, da die Signalstärken zwischen den Referenzpunkten bereits im Bereich von 1-2 dBm schwanken.
Aus dem Grund konnte mit diesem Testgebiet keine optimale Genauigkeit erreicht werden. Zur Steigerung der Positionierungsgenauigkeit sollte im Idealfall der Abstand zwischen den Referenzpunkten mehr als 3 Meter betragen. Dafür wäre eine andere Räumlichkeit zu empfehlen.

- Statistik

Durch Verwendung einer Maximum-Likelihood-Funktion kann die Genauigkeit verbessert werden. Zusätzlich wird zu den gemittelten Signalstärken, die Standardabweichung der Signalstärke aus einem Access Point vorab in der Kalibrierungsphase berechnet.

- Darstellung von aktuell berechneter Position und grafische Benutzeroberfläche

Derzeit wird die aktuell berechnete Position des Raspberry Pi im Programm nur über die Konsole ausgegeben. Duch eine Verbindung des RPi mit dem Rechner über TCP/IP, können die Daten live ausgewertet werden. Hierzu wurde bereits ein kleines Serverprogramm in Matlab realisiert. Die Verbindung zwischen der Applikation und dem Matlab Server wurde bereits erfolgreich hergestellt. Die berechneten Positionen können erfolgreich empfangen werden, jedoch muss der Plot der Positionen evtl. im Matlab Programm noch leicht angepasst werden. Zudem könnte im Plot ein Umgebungsplan eingebaut werden.

Damit die ganzen Befehle nicht über die Kommandozeile ausgeführt werden müssen, wäre eine grafische Benutzeroberfläche für das Programm sinvoll, mit der auch die entsprechenden Argumente/Parameter an das Programm übergeben werden können.

- Kopplung mit Turtlebot (siehe ROS a.k.a Robot Operating System)

Der Turtlebot liefert dann Odometriedaten (Distanz, Winkel) an den Raspberry Pi, auf dem dann eine neue Position errechnet wird und dann automatisch dem Textdokument angefügt wird. Gleichzeitig könnte der Roboter dann die errechneten Positionsdaten zurückbekommen. Mit einem autonomen System, welches ein Gerät zur Distanzmessung enthält, ergibt sich nun auch die Möglichkeit ein Slam-Verfahren durchzuführen. Hier kann man mit der Position und den Distanzen aus dem Entfernungsmesser die Wände kartografieren und schlussendlich eine Karte erstellen. Zudem könnte mit den aufgezeichneten Positionsdaten bei der Fahrt die Trajektorie des Roboters aufgezeichnet und visualisiert werden.

---

userwiki/robot/indoornavigation/index.txt · Last modified: 2018/01/18 10:12 by ahariyanto