

Fakultät für Elektrotechnik und Informatik

ROS auf dem Raspberry Pi

14 November, 2013

Systemadministration Projekt in Angewandter Informatik
von Denis Herdt und Almin Causevic

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Entscheidung für dieses Thema fiel dadurch, dass es zukünftig genutzt werden kann und die Arbeit mit Soft-/Hardware kombiniert wird.

Im Robotiklabor der Hochschule Weingarten wird außerdem zur Zeit noch umständlich ein zusätzlicher Laptop zur Steuerung der Roboter verwendet. Ein Beispiel dafür ist der "Volksbot". Durch den Raspberry Pi ergeben sich etliche Vorteile.

Mithilfe des Software-Frameworks ROS (Robot Operating System) kann ein flexibles Netzwerk von Komponenten realisiert werden, in welches der Raspberry eingebunden wird. Dies ist eine einfache und schnelle Methode, eine Robotersteuerung realisieren zu können. ROS wird weltweit eingesetzt und kann auch in Zukunft sehr nützlich sein.

Durch die Kombination mit der Mächtigkeit und Flexibilität von ROS und der gerinen Größe und niedrigen Energieverbrauch des Pi eröffnen sich neue und effiziente Einsatzmöglichkeiten.

Zudem ist dieses Projekt für das Fach Systemadministration sinnvoll, weil mit Linux gearbeitet wird und viel Konfiguration des Betriebssystems vorausgesetzt wird.

1.2 Zielsetzung

Unser Hauptziel besteht darin, ROS auf dem Raspberry Pi aufzusetzen und zu nutzen. Dieses Ziel wird wie folgt unterteilt:

- Recherche über Linux Systeme auf dem Raspberry Pi
- Passendes Linux System auf Raspberry Pi aufsetzen
- Recherche über ROS
- Geeigneten Roboter wählen (Recherche Verfügbarkeit, Komponenten etc.)

- Komponenten geschaffen (Wlan Stick, Akkupack, etc.)
- ROS auf Raspberry Pi aufsetzen
- Netzwerk zwischen mehreren ROS-Komponenten herstellen
- Netzwerk mithilfe von Testausgaben überprüfen

Mit diesen Zielen ist eine Grundlage für weitere Anwendungen mit ROS auf dem Raspberry geschaffen.

optional

Da uns am Ende noch genügend Zeit zur Verfügung stand, haben wir folgende Punkte bearbeitet:

- Steuerungspaket für "Volksbot" einbinden
- Verbesserung der Usability durch Smartphone-Steuerung
- Optimierung der WLAN-Verbindung am Raspberry

Bleibt am Ende noch genug Zeit, wird die integrierte Kamera des Roboters angesprochen und das Bild mithilfe des Netzwerkes auf einen externen Bildschirm übertragen. ers soll durch das Schwenken des Smartphones bestimmt werden.

1.3 Eigene Leistung

Das "neue" an diesem Projekt bezieht sich auf die Ablösung eines Laptops durch den kleineren und leichteren Raspberry Pi als ROS-Komponente am ausgewählten Roboter. Desweiteren möchten wir herausfinden, ob die Leistung des Pi für diese Aufgabe ausreicht.

1.4 Aufbau der Arbeit

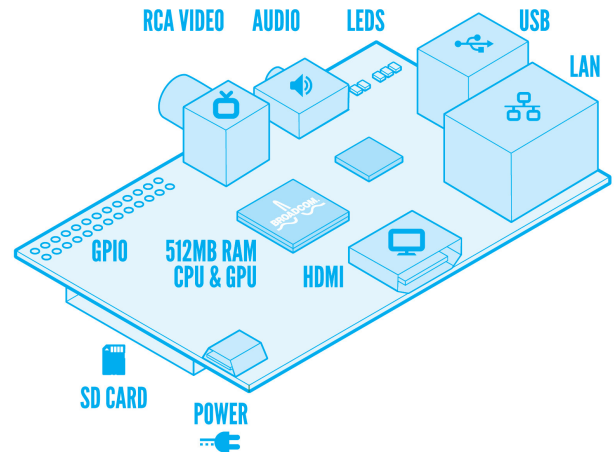
In den Grundlagen werden alle wichtigen Komponenten und Aspekte genauer erklärt. Anschließend wird das Grundproblem, welches wir lösen wollen, erläutert und verwandte Arbeiten aufgelistet. Darauf folgend werden die dadurch entstandenen Anforderungen und Lösungsvorschläge, sowie die Implementation beschrieben.

2 Grundlagen

2.1 Raspberry Pi Hardwarekomponenten

”Der Raspberry Pi ist ein kreditkartengroßer Einplatinencomputer, der von der Raspberry Pi Foundation entwickelt wurde”[?] Wir benutzen für dieses Projekt das leistungstärkere Modell B. Technische Details: [?]

- Preis: ca. 35 Euro
- Prozessor: ARM1176JZF-S(700 MHz)
- Broadcom VideoCore IV
- SDRAM: 512 MB
- Bis zu 16 GPIO-Pins
- USB-Anschlüsse: 2
- FBAS, HDMI



[?]

- 3,5-mm-Klinkenstecker (analog), HDMI (digital)
- Kartenleser für SD (SDHC und SDXC)/MMC/SDIO
- 10/100-MBit-Ethernet-Controller
- 5 V, 700 mA (3,5 Watt)
- 5-V-Micro-USB-Anschluss (Micro-B), alternativ 4 x AA-Batterien

Stromquelle Als Stromquelle steht ein Akkupack zum Einsatz. Hierbei ist wichtig, dass der Raspberry mind. 700 mA, besser 1 A zur Stromversorgung bekommt. Bei niedrigerer Amperzahl arbeitet der PC oft nicht zuverlässig.

RAM Das Model B bietet einen schnelleren und größeren RAM Speicher. Dieser ist wichtig, da viele Signale, teils sogar synchron, verarbeitet werden müssen.

Datenübertragung Für die Datenübertragung über das Netz wird ein Standard 300Mbit/s Wlan-Stick benutzt.

Prozessor Der Raspberry Pi arbeitet mit einer ARM-Prozessorarchitektur. Diese Architektur wird gerne für embedded Systems, wie PDAs oder Router, eingesetzt. Diese ist auch auf jedem Smartphone zu finden, da sie den Vorteil einer sehr geringen Leistungsaufnahme bietet. [?]

Betriebssystem Das auf Linux basierende Betriebssystem Raspbian wird verwendet. Dabei handelt es sich um ein für Raspberry Pi optimiertes Open-source Debian-System. Es enthält viele für die ARM-Architektur vorkompilierte Pakete (über 35.000), dazu auch weitere Features, wie etwa eine GUI. [?]

GPIO GPIO (General Purpose Input/Output) ist ein weiteres interessantes Feature des Raspberry Pi. Es gibt die Möglichkeit, jegliche Hardware-Funktionalität anzusteuern. Beispielsweise können LED-Leuchten oder der Start-Knopf einer Kaffeemaschine damit über ROS gesteuert werden. Dies bietet Anreiz für noch mehr Ansätze und Umsetzungen mithilfe des Raspberry Pi und ROS.

2.2 ROS

ROS ist kein eigentliches Betriebssystem im herkömmlichem Sinne, sondern eine Art strukturierte Kommunikationsschicht. Die Ziele von ROS können zusammengefasst werden zu:

- Peer-to-peer (System mit vielen laufenden Prozessen auf verschiedenen Hosts)
- Tool-based (Microkernel-Design bestehend aus vielen Komponenten)

- multi-lingual (unterstützt viele Computersprachen, z. B. Python, C++, etc.)
- thin (Wiederverwendbarkeit von Code steht im Mittelpunkt)
- open source and free (unter der BSD Lizenz) [?]

Es ist das einzig existierende Framework, welches sich auf diese Kriterien spezialisiert. Den Entwicklern von ROS ging es vor allem um die Vereinfachung, Software für eine hohe Zahl an unterschiedlichen Robotern zu schreiben.

ROS stellt dazu Bibliotheken, Werkzeuge, Hardware Abstraktion, Gerätetreiber, Bibliotheken, Visualisierungen, Nachrichtenvermittlung, Paketverwaltung und andere Komponenten zur Verfügung. [?] Im Nachfolgenden werden die für dieses Projekt wichtigsten ROS Komponenten kurz beschrieben.

Topic Topics können als "named bus" gesehen werden, über welche Nodes Messages verteilen können. Es ähnelt dem Multicasting Prinzip. An Topics kann subscribed (zugehört) oder published (geredet) werden.

Node Eine Node ist ein simples Programm, welches jede Funktionalität ermöglicht. Sie kommunizieren direkt über Topics.

Roscore Der Roscore kann als Rückgrat des ROS Systems beschrieben werden. Es verwaltet die Registrierung der Nodes und Funktionen in einer Art lookup service für andere Nodes. Es bietet zusätzlich einen Parameter Server, in welchem im laufenden Betrieb Variablen und Parameter gespeichert werden können.

Package Software in ROS wird mithilfe von Packages verwaltet. Sie enthalten Nodes, datasets, configuration files, etc.

Launchfile Ein Launchfile ist eine simple XML Datei, welche Informationen über die zu Beginn zu startenden Nodes mit entsprechenden Parametern enthält. Dies bietet eine komfortablere Alternative, als jedes Node einzeln zu starten. Ein Beispiel für ein Launchfile ist in der Implementation bei der Steuerung eines Roboters (Volksbot) gegeben. [?]

catkin Catkin wird in ROS seit der Distribution Groovy als Workspace-Verwaltungssystem verwendet. Es ist in 4 Spaces unterteilt:

- Source Space

- Build Space
- Devel Space
- Install Space

Catkin bietet uns eine komfortable Möglichkeit über cmake ROS Packages zu compilieren, Tests an ihnen durchzuführen und sie schließlich zu installieren.

2.3 Volksbot

”VolksBot ist ein Roboterbaukastensystem[...] Mit Hilfe des Baukastens können sehr schnell und preiswert unterschiedlichste Varianten mobiler Roboter hergestellt werden.”[?] Der Volksbot wurde an der Hochschule Weingarten für den RoboCup verwendet.



[?]

2.4 Smartphone

Es wird ein auf Android basierendes Smartphone verwendet. Wir benutzen auf dem Smartphone eine in einer Projekt-Arbeit entstandene App. [?] Sie bindet sich an einen ROS-Master (später näher erläutert) und übergibt die X- und Y Koordinaten relativ zu sich weiter. Diese Werte können für die Steuerung des Motors verwendet werden, indem jeweils ein Wert an eine Radachse geliefert wird. Dieser Wert bildet die Geschwindigkeit des Rades ab. Zur erleichterten Bedienung stellt uns die App eine GUI zur Verfügung.

3 Problem

Viele der heutigen Roboter mit ROS-Systemen besitzen einen leistungsstarken und klobigen PC an Ihrer Seite. Doch für viele Roboteranwendungen würde auch ein kleinerer, leistungschwächerer PC ausreichen. Dadurch ergeben sich neue Vorteile:

- weniger Gewicht
- weniger Kosten
- Energieverbrauch sinkt → Akku hält länger
- Die Größe des Roboters nimmt ab

Der Roboter Volksbot wird mit einem für diesen Zweck überdimensionierten Laptop gesteuert. Diese Probleme sollen behoben und die obigen Vorteile verschafft werden, indem ROS auf einem Raspberry Pi aufgesetzt wird.

4 Verwandte Arbeiten

Master-Thesis:

1. Robot Navigation in Unknown Environment Based on Combined 2D and 3D Information, Marc Götz, August 2013
2. Semi-Autonomous Grasping of Unknown Objects, Steffen Pfiffner, November 2013

5 Anforderungen

Dieses Projekt hat das Ziel, dass ROS auf dem Pi getestet und an einem gegebenen Roboter angewendet wird. Aus diesem Ziel, sowie den vorangegangenen Punkten, erschließen sich folgende Anforderungen:

- Datenpakete über das ROS-Netzwerk verschicken

- Robuste und zuverlässige Kommunikation
- Modulares und flexibles Netzwerk
- packages sollen leicht in Projekte eingebunden werden

optional

- Roboter soll bei Steuerung in die richtige Richtungen fahren
- Roboter soll sofort auf Anweisungen reagieren
- Sensibilität der Steuerung sollte einstellbar sein
- Steuerung über WLAN
- ROS-Core des Raspberry beim Hochfahren automatisch starten
- Leichte Bedienbarkeit des Roboters
- Streaming-Bild ruckelfrei verarbeiten
- unkonvertiert Echtzeitübertragung
- Ausgabe am externem Bildschirm

6 Lösungsvorschläge

Datenpakete über das Netzwerk verschicken:

Es wird ein ROS-Master aufgesetzt. Dieser gilt als Knotenpunkt (vergleichbar mit einem Defaultgateway) und sorgt für die Kommunikationsmöglichkeit im ROS-Netzwerk.

Zuverlässige Kommunikation:

Die Netzwerkstabilität und -zuverlässigkeit könnte durch Begrenzung der zeitgleichen Datenübertragung umgesetzt werden. Zusätzlich kann man eine vorangehende Überprüfung der verschickten und erwarteten Datentypen einbinden.

Flexibles Netzwerk, Packageeinbindung:

Ein modulares und flexibles Netzwerk, sowie das einfache Einbinden von packages ist durch ROS in Kombination mit Catkin gegeben.

optional

Steuerung eines Roboters:

Die Reaktionszeit und Steuerung wird durch zuverlässige und leistungstarke Hardware, sowie durch einen effizient geschriebenen Code bestimmt. In diesem Fall kann die Hardware (mit Ausnahme des Pi) aufrüsten. Der Code lässt sich gar nicht oder nur mühsam verändern und austauschen.

ROS-Core automatisch starten:

Um bequem mit dem Raspberry arbeiten zu können, ohne jedes mal den ROS-Core starten zu müssen, muss dieser in den Bootvorgang des Raspbian eingebunden werden.

Leichte Bedienbarkeit:

Diesen Punkt wird mit einer Smartphone App realisieren. Der Roboter soll dabei auf das Schwenken des Smartphones reagieren und seine Geschwindigkeit/Richtung entsprechend anpassen.

Videoübertragung:

Ein sauberes und störungsfreies Übertragungsbild hängt hauptsächlich von der Hardware des Raspberry Pi, aber auch von der Codierung ab. Der Stream soll durch Umleiten des Bildes über das Netzwerk an einen PC-Monitor realisiert werden. Falls dabei Probleme auftreten sollten, kann versucht werden, die Codierung zu bearbeiten oder leistungsverstärkende Hardware an den Pi anzuschließen.

7 Bewertung der Lösungen

Bis jetzt sind die Lösungsansätze relativ erfolgreich gewesen. Abgesehen von kleineren Leistungsproblemen durch Datenübertragung über das WLAN-Netzwerk, konnten alle Anforderungen gut erfüllt werden.

8 Implementation

8.1 Raspbian

Zuerst wird das Betriebssystem Raspbian auf dem Pi aufgesetzt. Hierzu gibt es 2 mögliche Vorgehensweisen:

1. Installation mit NOOBS
2. Flashen der SD-Karte (Linux mittels dd)

Es wurde die 2. Methode gewählt, da NOOBS 6 verschiedene Images anbietet, doch in diesem Fall nur Raspbian benötigt wird. Zudem erfordert die 2. Vorgehensweise mehr Arbeit mit dem Linux Betriebssystem, was in dieser Vorlesung nur vorteilhaft sein kann. Die genauen Installationsschritte werden dem Unterpunkt "Using the Linux command line" entnommen. [?]

Um mögliche Fehlerquellen schon beim Herunterladen der Image Datei zu vermeiden, wird empfohlen, wie in der Anleitung beschrieben, die sha1sum zu vergleichen.

Nun muss nur noch die SD-Karte in den Raspberry eingesteckt werden. Falls eine grüne Leuchte am Raspberry blinkt oder eine Bildschirmausgabe an einem Monitor erscheint, ist dieser Schritt gelungen.

8.2 ROS aufsetzen

Der Hersteller Willow Garage stellt seit der Entstehung von ROS mehrere Distributionen zur Verfügung. Für den Raspbian ist die momentan zweitaktuellste Distribution Groovy zu empfehlen. Die aktuelle Version Hydro wird noch nicht unterstützt.

Es gibt 2 sinnvolle Wege, ROS Groovy auf dem Raspbian System aufzusetzen.

1. Source packages des ROS-Kerns kompilieren
2. binary packages für die ARM-Architektur installieren

Da die Installation und das Kompilieren von Source zwar lehrreich, doch auch relativ zeitaufwendig ist, werden die vorhandenen binary packages für ARM benutzt. Hierfür nach Punkt 2 der Anleitung vorgehen [?]

Leider werden dadurch nur die grundlegendsten Packages von ROS unterstützt. Da ein großer Vorteil von ROS die Modularität ist, können leicht benötigte packages über catkin nachinstalliert werden. Dazu wird später die Vorgehensweise am Beispiel des packages zur Steuerung des Roboters erklärt.

ROS integriert sich sehr gut in die Shell des Linux Systems. Unterstützt

werden mehrere Shell-Typen. In diesem Projekt wird die Bourne Again Shell verwendet. ROS bietet für seine Systemumgebung Äquivalente zu den wichtigsten Shell Befehlen, beispielsweise `roscd` für `cd` oder `rosls` für `ls`.

Dadurch ergibt sich der Vorteil, nicht den vollständigen Pfad zu einem ROS package oder Verzeichnis eingeben zu müssen. Beispielsweise ist es möglich sofort in ein ROS Verzeichnis mit `roscd` zu wechseln, ohne den vollständigen Pfad mittels `cd` eingeben zu müssen.

Damit man auf die ROS-typischen Shell Befehle Zugriff hat, muss zuerst `source` werden.

```
source /opt/ros/groovy/setup.bash
```

Der Source Befehl bewirkt, dass die Pfade zu den ROS Verzeichnissen dem Linux System bekannt gemacht werden.

Es empfiehlt sich, sich mit den typischen ROS Befehlen etwas vertraut zu machen. Siehe dazu [?].

Mit diesen Einstellungen sollte es kein Problem mehr sein, ein Netzwerk zwischen Systemen aufzusetzen.

8.3 Netzwerk aufsetzen

Damit ein Netzwerk aufgesetzt werden kann, muss ein Linuxsystem zum Master werden. Im Grunde ist es wichtig, in allen Systemen des Netzwerks in `export` die IP-Adresse und die Port-Nummer (Standard: 11311) des Systems anzugeben, welches als Master dienen soll. Hierzu kann man mithilfe des Befehls

```
export ROS_MASTER_URI="http://IP-ADRESSE:11311"
```

die IP-Adresse des Masters angeben.

Eine der Möglichkeiten sich die IP-Adresse des Systems anzeigen zu lassen ist der Shell Befehl `ifconfig`.

ROS Workspace anlegen:

Für ROS sollte noch ein Workspace mit `catkin` erzeugt werden. Anleitung dazu unter Punkt 4 der Quelle [?]. Auch das Sourcen des Workspace sollte am Besten in die `.bashrc`, aufgenommen werden.

Es soll noch zusätzlich ein Beispiel package erzeugt, welches später zum Testen des Netzwerks erforderlich ist. Dazu in das Source Verzeichnis des catkin Workspace wechseln

```
cd ~/catkin_ws/src
```

und ein package erzeugen.

```
catkin_create_pkg beginner_tutorials  
std_msgs rospy roscpp
```

Das package sollte noch compiliert werden. Erneut in das catkin Workspace wechseln

```
cd ~/catkin_ws
```

und

```
catkin_make
```

ausführen.

8.4 Testen des Netzwerks

Um das bestehende Netzwerk zu testen, können Datenpakete zwischen den Systemen verschickt werden. Zuerst sollte der Master ansprechbar sein. Hierfür wird der Kern von ROS mit dem Shell Befehl

```
roscore
```

gestartet.

Falls bei Eingabe des Befehls

```
roscpp list
```

als Antwort mindestens ein Node (/rosout) erscheint, läuft der Kern ordnungsgemäß.

Um nun einfache Testausgaben zwischen Systemen zu schicken, werden zwei ROS Nodes und ein Topic zur Kommunikation eingerichtet. Auf dem ROS Master wird ein listener Node (Subscriber) gestartet. Auf einem anderen ROS System wird ein talker Node (Publisher) gestartet.

Der Quellcode kann hier heruntergeladen werden [?].

Es kann auch der talker anstatt des listeners auf dem Master ausgeführt werden.

Es sollte nicht vergessen werden, die Quelldateien mittels

```
chmod +x Dateiname.py
```

ausführbar zu machen.

Alternative: C++ Code [?]

Die Nodes mit dem Befehl

```
roslaunch beginner_tutorials Dateiname.py
```

starten. Im Shell Fenster des Listener Nodes sollte nun die Ausgabe des Talker Nodes erscheinen.

Falls dies nicht passiert, sollte die obige Anleitung noch einmal genau befolgt werden und zusätzlich das Netzwerk auf Fehler überprüft werden. [?]

Die Nodes und Topics können zum besseren Verständnis visualisiert werden. Zuerst muss das rqt package installiert werden.

```
sudo get-apt install ros-groovy-rqt
```

Mit

```
rqt_graph
```

wird das Verhältnis der Nodes visualisiert.

8.5 Steuerung des Volksbot

Da der Roboter frei mit dem Smartphone gesteuert werden soll und sich der Raspberry losgelöst auf dem Volksbot befinden soll, werden hierfür Akkus für die Stromversorgung und ein Wlan-Stick für die Datenübertragung benutzt.

Für die Ansteuerung des Motorcontrollers des Volksbot benutzen wir ein in einer Projekt Arbeit entstandenes ROS package. [?]

Nach Befolgen der Installationsanleitung (weicht etwas von der Standard

catkin Installation ab) kann das package kompiliert werden. Anschließend können über eine roslaunch Datei die nötigen Nodes und Topics des Packages gestartet werden.

```
roslaunch tmc200ctrl_examples tmc200controller.launch
```

Wobei *tmc200ctrl_examples* das Verzeichnis darstellt und *tmc200controller.launch* das Launch File ist. Die Netzwerk-Funktionalität ist nach den obigen Tests gegeben. Ein Smartphone verbindet sich über eine ROS App [?] mit dem Master. Es sendet nun je nach Neigungswinkel des Smartphones entsprechend Koordinatendaten. Diese Daten werden durch das ROS package entsprechend verändert, sodass Sie als Geschwindigkeiten für den Volksbot verwendet werden können. Sie werden an den Motor des Volksbot gesendet. Der Raspberry wird mithilfe eines USB zu VGA Adapters an den Volksbot angeschlossen. Dieser wird per USB an den Raspberry Pi und per VGA an eine serielle Schnittstelle des Motors angeschlossen.

Nun kann der Roboter durch Schwenken des Smartphones gesteuert werden.

9 Fazit

9.1 Zusammenfassung

Sehr gut ist der Aspekt, dass man das Thema frei wählen konnte. Dadurch steigt das Interesse und die Motivation an der Arbeit. Durch die hohen Anforderungen und den knappen Zeitraum ist es aber auch sehr anspruchsvoll und arbeitsintensiv.

9.2 Ausblick

In Zukunft kann der Raspberry Pi durch die Tatsache, dass ROS benutzt wurde, vielseitig im Robotiklabor verwendet werden. Zum Beispiel bei Projekten, die wenig Spielraum bei Größe und Energieverbrauch zulassen. Durch die Tatsache, dass ROS immer mehr an Beliebtheit gewinnt, nimmt auch die Anzahl an binary-packages und Ausarbeitungen zu. Dies eröffnet wiederum

mehr Einsatzmöglichkeiten. Zum Beispiel kann man nicht nur Roboter, sondern durch das GPIO-Interface auch viele andere Hardwarekomponenten(wie die Bedienoberfläche einer Kaffeemaschine, Lichtschalter, etc.) steuern.

9.3 Eigene Leistung

Durch dieses Projekt wird der Laptop am "Volksbot" durch einen viel kleineren und leichteren Raspberry Pi ersetzt. Außerdem eröffnet sich die Möglichkeit, den Pi als Hardware-Node zu benutzen oder anderweitig einzusetzen.