Hochschule
Ravensburg-Weingarten
Technik | Wirtschaft | Sozialwesen

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

1-2 Seiten

1.1 Motivation

Wir haben uns für dieses Thema entschieden, weil wir in Kombination mit Hard- und Software arbeiten wollten. Wir haben mitbekommen, wie umständlich z.B. der Roboter "Volksbot" im Robotiklabor der Hochschule mit einem zusätzlichen, auf dem Roboter platzierten Laptop gesteuert werden muss, der unnötig Platz und Gewicht beansprucht. Deshalb haben wir uns entschlossen, dieses Problem mithilfe des kleinen und leichten Raspberry Pi zu lösen. Die Idee stieß auch auf großes Interesse bei den Labormitarbeitern und bringt viel praktischen Nutzen.

Unser Ziel möchten wir mithilfe des Software-Frameworks ROS (Robot Operating System) realisieren. Für ROS haben wir uns entschieden, weil es zur Zeit die aktuellste und inovativste Methode ist, ein flexibles Netzwerk von Komponenten für z.B. die Robotersteuerung zu realisieren. Es wird weltweit eingesetzt und könnte auch in Zukunft sehr nützlich werden.

Durch die Kombination mit der Mächtigkeit und Flexibilität von ROS und der geringen Größe und dem niedrigen Energieverbrauch des Pi eröffnen sich außerdem neue und effizientere Einsatzmöglichkeiten.

Sehr gut finden wir auch die Tatsache, dass wir mit Linux arbeiten können und dieses Projekt viel Konfiguration des Betriebssystems voraussetzt, da wir auch gerne mehr Praxisanwendung mit Linux haben möchten.

1.2 Zielstzung

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

Zunächst möchten wir ein passendes, auf Linux basiertes Betriebssystem auswählen und auf dem Raspberry Pi aufsetzen. In diesem Fall haben wir uns für Raspbian entschieden, da es außerdem am kopatibelsten zu ROS ist.

Als nächstes wählen wir einen geeigneten Roboter aus. Die Kriterien dafür erarbeiten wir uns aus einer vorangegangenen ROS-Recherche.

Im Anschluss setzen wir ROS auf dem Raspberry Pi und mehreren Linux-Rechnern auf und stellen ein WLAN-Netzwerk zwischen den verschiedenen ROS-Komponenten her. Dieses Netzwerk wird mithilfe von Testausgaben überprüft.

Damit ist die Grundlage zum Anwenden von ROS auf dem Pi geschaffen, an die man weiter Projekte knüpfen kann.

optionale Ziele:

Bleibt am Ende noch genug Zeit, implementieren wir die Steuerung des "Volksbots" auf dem Raspberry. Zur Usabilitysteigerung integrieren wir ein Smartphone ins ROS-Netzwerk. Die Bewegungsrichtung und -geschwindigkeit des Roboters soll durch das Schwenken des Smartphones bestimmt werden.

Haben wir damit Erfolg und wiederum genügend Zeit zur Verfügung, sprechen wir die integrierte Kamera des Roboters an und streamen das Bild mithilfe des Netzwerkes auf einen externen Bildschirm.

Mit diesen Optionen wollen wir zeigen, wie Flexibel und Umfangreich ROS in der Anwendung sein kann.

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, sowie einem Leistungs- bzw. Eignungstest des Raspberry im Aufgabenbereich Robotersteuerung.

1.4 Aufbau der Arbeit

- Recherche über Linux Systeme auf dem Raspberry Pi
- Passendes Linux System auf Raspberry Pi gesetzt
- Recherche über ROS

- Geeigneten Roboter gewählt (Recherche Verfügbarkeit, Komponenten etc.)
- Komponenten geschafft (Wlan Stick, Akkupack, etc.)
- ROS auf Raspberry Pi aufgesetzt
- Netzwerk zwischen mehreren ROS-Komponenten hergestellt
- Netzwerk mithilfe von Testausgaben überprüft

optional

- Robotersteuerung wird zur Zeit implementiert

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 leistungsstärkere 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

Für Test und Kontrollausgaben benutzen wir einen am HDMI-Ausgang angeschlossenen Monitor.

Als Stromquelle steht ein Akkupack .....(Firma,Modell) zur Verfügung. Hierbei ist wichtig, dass der Raspberry mind. 700 mA, besser 1 A zur Stromversorgung bekommt. Bei niedrigerer Amperzahl arbeitet er oft nicht zuverlässig.

Ein schneller und großer RAM Speicher ist für unsere Zwecke wichtig, da viele Signale, teils sogar synchron, verarbeitet werden müssen.

Für die Datenübertragung über das Netz benutzen wir einen Standard 300Mbit/s Wlan-Stick.

Der Raspberry Pi arbeitet mit einer ARM-Prozessorarchitektur. Diese Architektur wird gerne für embedded Systems, wie PDAs oder Router, eingesetzt.

Sie ist auch auf jedem Smartphone zu finden, da sie den Vorteil einer sehr geringen Leistungsaufnahme bietet. Erwähnenswert ist die Architektur deshalb, weil wir im Laufe des Projektes Schwierigkeiten hatten, auf die wir später eingehen werden.

Wir benutzen das auf Linux basierende Betriebssystem Raspbian. Dabei handelt 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 Features wie etwa eine GUI.

Das System braucht 3GB Speicherbedarf unserer 16GB großen SD-Karte. Der zusätzliche Speicherplatz wäre nötig, falls man vorhat, Log-Dateien und Ähnliches direkt auf dem Raspberry Pi zu speichern. In unserem Fall übernehmen das jedoch die leistungsstärkeren PC's über das ROS Netzwerk.

GPIO (General Purpose Input/Output) ist ein weiteres interessantes Feature des Raspberry Pi. Es gibt uns die Möglichkeit, jegliche Hardware-Funktionalität anzusteuern. Beispielsweise können LED-Leuchten oder der Start-Knopf der Kaffeemaschine damit über ROS gesteuert werden. Wir

konnten uns leider jedoch zeitlich bedingt nicht mehr mit dieser Thematik beschäftigen, es bietet aber Anreiz für noch mehr Ansätze und Umsetzungen mithilfe des Raspberry Pi und ROS.

2.2 Smartphone

Es wird ein aud 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.

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 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, ähnlich zu Linux)
- 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 die hohe Zahl an verschiedenen Roboter zu schreiben. ROS stellt dazu Bibliotheken und Werkzeuge, Hardware Abstraktion, Gerätetreiber, Bibliotheken, Visualisierungen, Nachrichtenvermittlung, Packetverwaltung und andere Komponenten zur Verfügung.

Im Nachfolgenden werden die wichtigsten ROS Core 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 oben genannte 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 Paramater gespeichert werden können und mit welchen gearbeitet werden kann.

Message Eine Message kann als Sprache angesehen werden, in welcher Nodes kommunizieren. Es ist eine Struktur aus simplen Parametern. Beispiel:

```
# This represents a vector in free space.
float64 x
float64 y
float64 z
```

Eigene Messages mit den uns bekannten Standardtypen wie int, bool, etc können leicht definiert werden.

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

Manifest Sie enthält Meta Informationen über ein ROS Package, wie beispielsweise Abhängigkeiten oder Lizenzinformationen.

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.

catkin Catkin ist das seit Groovy in ROS verwendete Workspace-Verwaltungssystem. 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.

3 Problem

Hauptproblem??

3.1 Verwandte Arbeiten

Steffen, Marc ROS-Tut

4 Anforderungen

- Datenpakete über das ROS-Netzwerk verschicken
- Beliebige Datentypen verarbeiten
- Beliebige Linux-Systeme einbinden
- Überprüfung von Log-Dateien, Kontrollausgaben etc..
- Robuste und zuverlässige Kommunikation
- Modulares und flexibles Netzwerk

- Repositories sollen leicht in Projekte eingebunden werden

optional:

- Roboter soll bei Steuerung in richtige Richtungen fahren
- Roboter soll sofort auf Richtungsanweisungen reagieren
- Sensibilität der Steuerung sollte einstellbar sein
- Steuerung über WLAN
- Raspberry Pi auf Roboter ohne Kabelzgewirr
- Node des Raspberry beim Hochfahren automatisch starte
- Leichte Bedienbarkeit des Roboters

- Streaming-Bild ruckelfrei verarbeiten
- unkonvertiert Echtzeitübertragung
- Ausgabe am externem Bildschirm

5 Lösungsvorschläge

Es wird ein ROS-Master aufgesetzt. Dieser sorgt für ein bestehendes ROS-Netzwerk und ist für die Komponentensynchronisation und Paketverwaltung zuständig. Über Topics und Nodes werden Datenpakete verschickt und empfangen.

Die Verarbeitung von verschiedenen Datentypen ist durch msg und srv realisierbar

Die Unterstützung beliebiger Linuxsysteme wird von ROS grundsätzlich unterstützt.

Kontrollausgaben und ähnliches werden durch Horchen an Topics mithilfe von echo und durch Ausgabenbereiche im Quellcode durch das Netzwerk realisiert.

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

Ein Modulares und flexibles Netzwerk, sowie das einfache Einbinden von Repositories ist gernerell durch ROS in kombination mit Catkin gegeben.

opt: abhängig vom richtigen Code der Motorsteuerung

abhängig von Raspberry Pi Hardware und effizientem Code und guter Hardware

abhängig vom Code

Hardware nötig Wlan-Stick, Akku-Pack

sollte in Linux beim booten konfiguriert werden

gegeben durch gute Smartphone App

opt: abhängig von Raspberry Pi Hardware und Übertragungs-Codierung

abhängig vom Codex Hardware Komponenten

nötige Hardware und Netzwerk legen

6 Bewertung der Lösungen

alles gut

7 Implementation

ROS Netzwerk (topics, nodes, catkin...) Linux SD Karte Pi Datenübertragung an Volksbot Konfiguration Motorsteuerung-Paket und Compilieren Kontrollausgaben Netzwerkkomponenten zusammen arbeiten lassen (Hard/-Software) Wlan -¿ Teilproblem Kamera -¿ Teilproblem Zeit

8 Fazit

1 Seite

8.1 Zusammenfassung

sehr knapper Zeitrahmen hohe Anforderungen für diesen Zeitraum interessantes Projekt anspruchsvolles Projekt viel über Linux und Ros und Hardware und Netzwerken gelernt Spaß

9 Ausblick

Benutzung des Pi's als Hardware Nodes Lab Benutzung des Pi's für kleine Roboterprojekte Lab Immer mehr binary packages für ROS Pi verfügbar -¿ mehr Möglichkeiten z.B.: GPIO Interface -¿ Knöpfe Kaffemaschine, Relays, alle möglichen Signale auf Hardwareebene

10 Eigene Leistung

anstatt Linux PC wird Pi benutzt

11 Quellen

http://de.wikipedia.org/wiki/Raspberry_Pi
http://www.amazon.de/Raspberry-Pi-RBCA000-Mainboard-1176JZF-S/dp/B008PT4GGC
http://www.raspbian.org/
http://www.softwareok.de/?seite=faq-System-Allgemein&faq=13
http://de.wikipedia.org/wiki/ARM-Architektur
http://www.rn-wissen.de/index.php/Raspberry_PI:_GPIO
http://www.volksbot.de/index-de.php
http://wiki.ros.org/
http://wiki.ros.org/groovy/Installation/Raspbian