Roboter Orchestrierungssoftware

Exposé für die Lehrveranstaltung Systemadministration

Stefan Geiring, 30658
Marvin Müller, 32850
Nicolas Baumgärtner, 32849

Wintersemester 22/23
05.November 2022

Betreuer: M.Sc. Aykan D. Inan
Ravensburg-Weingarten University of Applied Sciences

Inhaltsverzeichnis

1 Thema

Entwicklung einer Orchestrierungssoftware mit Weboberfläche auf Basis von ROS.
Roboter werden immer häufiger eingesetzt egal ob in Spezial Industrial bereichen oder auch einfach zuhause. Die einfache und übersichtliche Orchestrierung vieler Roboter ist deshalb besonders wichtig. Roboter Schwärme spielen außerdem, in der heutigen Zeit immer öfters eine wichtige Rolle. Ein Beispiel für große Roboter Schwärme die bereits eingesetzt werden sind Lieferdrohen, Drohnenshows als ersatz für Feuerwerk oder die simplere Variante davon, eine Lagerverwaltung.

2 Motivation

Wir interresieren uns alle für die Welt der Robotik. Das Thema bietet unglaublich viel Lernpotential.
Desweitern ist das Thema fächerübergreifend, wir arbeiten in insgesamt 4 Sektoren: der reinen Softwareentwicklung, Webentwicklung sowie der Hardware/Elektronik Entwicklung und natürlich der Robotik Entwicklung/Forschung.

3 Ziel

Ziel des Projektes soll es sein eine Lauffähige Orchestrierungssoftware für Roboter auf ROS Basis zu entwicklen. Mit deren Hilfe man mehrere Roboter überwachen und auch steuern kann.
Desweitern eine selbst designde Hardware Platform für unsere kleinen Beispiel Roboter zu designen, welche im nachhinein durch diverse Komponenten erweitert werden können. Wie zum Beispiel Kamera, Bumber, Laser etc.

4 Eigene Leistung

Entwickung einer einfachen Weboberfläche auf ROS Basis welche als Orchestrierungssoftware für Roboter dienen soll. Die Weboberfläche der Orchestrierungssoftware soll in erster Linie als einfache Kontroll und Debugging Schnittstele dienen.
Um unsere Orchestrierungssoftware sinngemäß demonstrieren zu können sollen außerdem kleine Roboter auf ESP32 Basis gebaut werden. Diese Roboter sollen aus einem 3D-Gedruckten Gehäuse bestehen. Auf den ESP32 soll Micro-ROS auf Basis von FreeRTOS ausgeführt werden.
Unsere kleinen Beispiel Roboter sollen außerdem im Idealfall mit modular austauschbaren Erweiterungen ausstattbar sein. Diese Erweiterungen sollen Simple Sensorik und Aktorik zur Verfügung stellen.

5 Aufbau der Arbeit

- ROS in Docker Container lauffähig bekommen.
- Micro-ROS auf ESP32 oder anderer Hardware lauffähig bekommen. Hardware ist noch nicht final definiert, deshalb muss erst noch evaluiert werden ob der ESP32 unseren Anforderungen und Wünschen gerecht wird.
- Frontend für ROS auf Web-Basis programmieren.
- Evaluation des Frameworks für Web-Frontend (evtl. Software zwischenlayer nötig).
- Design des Roboter Gehäuses entwerfen, welches 3D-Gedruckt wird.
- Antriebsstrang des Roboters entwerfen.
- Evaluation und Testing welcher Antrieb unseren Anforderungen entspricht.

6 Grundbegriffe

Docker:

Software für die Container Verwaltung.

ROS:

Das Acronym ROS steht für Robot Operating System, es bietet eine art Framework um Roboter leichter einbinden und steuern zu können.

React:

Web Frontend Framework was ursprünglich von Facebook ins leben gerufen wurde.

Arduino:

Entwicklungsplatform auf Basis von Atmel AtMega Prozessoren. Wurde entworfen um Leien den Einstieg in die Microcontroller Welt stark zu vereinfachen. Findet heutzutage Weltweit Anwendung in der Maker Szene.

ESP32:

ESP32 bezeichnet eine Microntroller Familie auf Basis der ARM Architektur. Ursprünglich wurde der ESP32 von Expresif entworfen und gefertigt.

RaspberryPi:

Toolchain:

Als Toolchain wird die Kombination von Micro-ROS und ESP-IDF bezeichnet. Mit Hilfe dieser beiden Frameworks wurde die Firmware für unseren Roboter programmiert und kompiliert.

7 Zielsetzung und Anforderungen

Entwicklung einer Weboberfläche mit der Roboter auf ROS Basis gesteuert und überwacht werden können. Oberfläche zeigt alle Roboter an und stellt Basic tools zur verfügung um mit diesen zu kommunizieren und diese zu steuern. Außerdem soll damit eine Schnittstelle zwischen Browsern und ROS geschaffen werden.

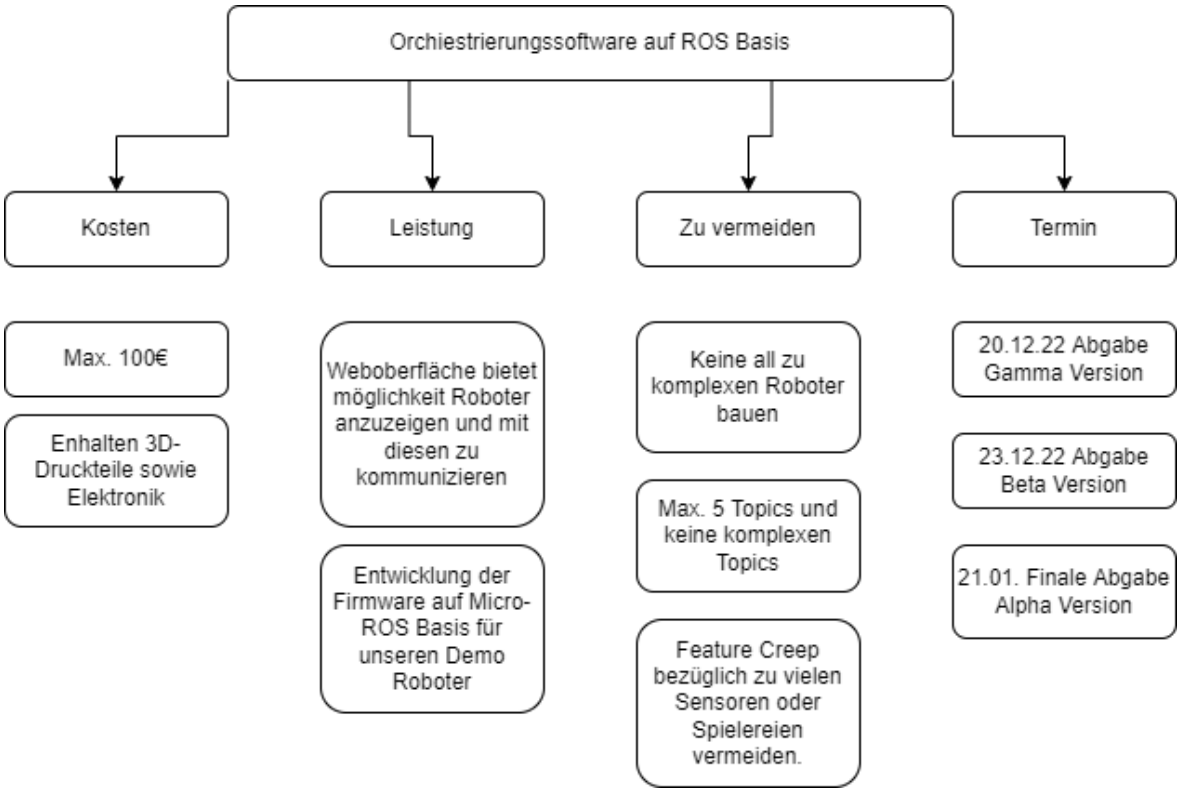

Abbildung 1: Anforderung Diagram

Kosten:

- Endkosten belaufen sich auf weniger als 100€.
- In diesen Kosten soll der Microcontroller inkl. 3D-Druck und gesamter Elektronik inkludiert sein.

Leistung:

- Weboberfläche ist lauffähig.
- Roboter kann mit ROS-Messages gesteuert werden.
- In Eigenleistung kleine fahrbare Roboterplatform auf ESP32 basis kreieren.

Zu vermeiden:

- Keinen zu komplexen Roboter designen. (Vollständig autonom fahrender Roboter)
- Feature creep mit all zu vielen Sensoren gilt zu vermeiden.

- Roboter Antrieb mit zu viel Technik ausstatten.
- Vorerst sollte nur ein Testroboter entwicklet werden.
- Webinterface nicht zu stark mit ROS Datentypen befüllen.

Termine:

- 20.11. Abgabe Gamma Version.
- 23.12. Abgabe Beta Version.
- 30.12. Feedback zu Beta Versionen.
- 21.01. Finale Abgabe.

8 Stand der Technik und Forschung

Technologischer Standpunkt Software:
WEBROS TODO ROS bietet bereits ein starkes und recht einfach nutzbares Framework, allerdings vermisst man eine schöne und einfach bedienbare "graphischeÖberfläche. Es existieren kleinere Projekte welche sich dieser Frontend entwicklung annehmen. Allerdings hat uns keines dieser Projekte zufrieden gestellt. Vor allem was die Orchestrierungsmöglichkeit mehrerer Roboter angeht.

Technologischer Standpunkt Hardware:
Es existieren bereits viele Forschungen und Beispiele für diverse Roboter und deren verschiedensten Antriebskinematiken. Für unser Projekt werden wir allerdings eine eigene Roboter Platform designen und 3D-Drucken. Viele der bereits vorhandenen Roboter sind entweder zu Groß und teuer oder viel zu klein und deshalb ebenfalls nicht gut für eine demonstration geeignet. Unter anderem wollen wir das unser Roboter den Vorteil bietet weitestgehend 3D-gedruckt zu sein. Die ETH Zürich hat bereits eine kleine Zusammenfassung über die wichtigsten Antriebarten mobiler Roboter zur verfügung gestellt. Für unser Projekt haben wir uns vorerst für einen simplen Diferentialantrieb entschieden.

9 Lösungsideen

9.1 Frontend

Das Frontend könnte durch normales plain HTML, CSS und JS realisiert werden. Da dies aber nicht mehr dem heutigen technischen standart entspricht kämen viele verschiedene Frontend Frameworks in frage. Zur Auswahl stehen React, Vue oder Angular.

9.2 Roboter

Zur Demonstration unserer Orchestrierungssoftware soll ein kleiner Roboter gebaut werden. Der Roboter sollte, wie in unseren Anforderungen bereits aufgelistet, einfach und schnell gebaut werden können. Und damit unser kleiner Roboter auch problemlos in der Welt navigieren kann muss dieser auch mit einer Art Lenkung ausgestattet werden.

Für solch einen Steuermechanismus kommen viele verschiedene Lösungen infrage:
Ein Kettenantrieb ähnlich wie in Baggern wäre denkbar. Er benötigt nur 2 Motoren und bietet viel Bewegungsfreheit ohne komplizierte Mechanik.
Die Ackermann Lenkung wie Sie heute in allen PKW und LKW vorkommt wäre ebenfalls denkbar. Hierbei müsste man nur einen Motor einbauen für den allgemeinen vortreib und einen kleinen Servo um die tatsächliche Lenkbewegung der Vorder- oder Hinterache zu realisieren.
Der Differentialantrieb wäre eine weitere Methode unseren Antrieb zu realisieren, hierbei werden wie beim Kettenantrieb ebenfalls zwei Motoren benötigt. Allerdings entfällt hierbei die Notwendigkeit einer Laufkette.
Unsere letze Idee wäre eine Knicklenkung wie sie oft in Baustellenfahrzeugen eingesetzt wird. Es bräuchte nur einen Motor als Antrieb und die etwas kompliziertere Mechanik von der Ackermann Lenkung könnte durch eine simplere ersetzt werden.

Unser Roboter kommt natürlich nicht nur mit einem einfachen Antrieb aus sondern muss diesen auch entsprechend ansteuern können. Ein Arduino könnte sehr gut dafür geeignet sein.
Ein ESP32 käme auch in frage. Er bietet weitaus mehr Rechenpower als ein Arduino bei minimaler Preiserhöhung. Außerdem unterstützen die meisten ESP32 eine Verbindung über WLAN.
Als teuerste aber auch Leistungsstärke Kontrolleinheit könnte auch ein Raspebrry Pi dienen. Da wir unsere Roboter mit hilfe vom ROS Framework ansteuern wollen, sollte unsere gewählte Steuereinheit Micro-ROS unterstützen. Micro ROS ist eine fork von ROS selbst und dadurch auchauf schwächeren Systemen lauffähig.

10 Evaluation der Lösungsideen anhand der Anforderungen

10.1 Wahl des Frontend Frameworks

Da es unglaublich viele Web Frontend Frameworks gibt, die sich leider oft nur in minimalen Punkten unterscheiden, haben wir uns Schlussendlich für React aus den unten genannte Gründen entschieden.
Da Ein Teammitglied bereits viel Erfahrung mit React sammeln konnte sahen wir das als starken Vorteil für unser Team an. Außerdem verfügt React über eine enorm gute Dokumentation, da es ständig von seiner eigenen Community verbessert wird.

10.2 Warum ROS?

Da wir alle bereits mit ROS gearbeitet haben, ist die Softwareentwicklung für einen Microcontroller deutlich einfacher, da ROS viele Standard-Aufgaben wie die Interprozesskomunikation erledigt. Außerdem ist es für uns leichter bekannte Hürden leichter zu umgehen. ROS stellt auch eine große Auswahl an Bibliotheken und Tools bereit, die die Entwicklung ebenfalls erleichtern und beschleunigen.

Ein weiterer Grund dafür, dass wir uns für die Entwicklung mit ROS entschieden haben ist, dass wir im Robotiklabor viel Hilfe bekommen können, da dort alle Roboter mit ROS entwickelt werden. Weitere Vorteile von ROS sind, dass es open-source ist und kostenlos.

10.3 Micro-ROS und RTOS

Die Entwicklung mit dem Mikrocontroller findet mit Micro-ROS statt, da das die Standard-Bibliothek für die Entwicklung mit Microcontrollern mit ROS2 ist.
Das Besondere an ROS2 und Micro-ROS im Vergleich zu ROS1 ist, dass es ermöglicht ein Real Time Operating System auf dem Microcontroller auszuführen. Ein RTOS wird standardmäßig mitinstalliert, ist aber für uns auch interessant, da wir bisher noch nicht mit RTOS gearbeitet haben und es eine gute Lernmöglichkeit mit beschränkten Aufwand ist.
Wie ROS selbst, ist Micro-ROS open-source und kostenlos.
Außerdem gibt es eine bereits bestehende Toolchain namens ESP-IDF, die das Cross-Compilieren, Flashen, Monitoring und Debugging erleichtert.

10.4 Roboter Platform

Die Anforderungen an die Roboter Demo Platform für dieses Projekt waren recht klein und einfach gehalten.
Das Hauptaugenmerk bei der Neuentwicklung und dem Design unserer Roboter Platform sollte zum einen die 3D-Druckbarkeit des Roboters sein aber auch das der Roboter vor allem in kurzer Zeit gedruckt und betriebsfertig gemacht werden kann.
In unserem Fall zeigten sich für die Antriebsart beim Differentialantreib die meisten Vorteile. Wir benötigen nur 2 Motoren die unabhängig voneinander angesteuert werden müssen. Es können normale Reifen verwendet werden und keine aufwendig zu bauende Ketten. Und beim Differentialantreib wird lediglich eine um 360° frei rotierbare Rolle benötigt. Somit entfällt auch ein großer Aufwand bei der Lenkmechanik.
Im allgemeinen sollte die 3D-Druckbarkeit der Teile somit kein Problem darstellen.
Für die Elektronik bzw. die Steuerung unseres Roboter fiel die Entscheidung auf einen ESP32. Dieser ist kostengünstig und recht einfach zu beschaffen. Zusätzlich bietet der ESP32 weitaus mehr Rechenleistung als ein Arduino und garantiert somit auch eine gewisse Skalierbarkeit. Die restlichen elektronischen Bauteile wie z.B. Lineare Spannungswandler können in diversen Elektronik Versandshops gekauft werden.

11 Implementierung

11.1 React Front-End

11.1.1 Aufbau des Frontends (Architektur)

11.2 ROS Back-End

11.2.1 Firmware Kompilierungs Toolchain

11.3 Roboter Platform

11.3.1 Elektronik

Wie in unserer Evaluation der Lösungsideen bereits beschrieben übernimmt der ESP32 Microcontroller, siehe Abbildung 2, die Steuerung unseres Roboters.

Abbildung 2: ESP32 Microcontroller

Da es sich beim Differentialantreib nur um eine Mechanische Anordnung der Motoren selbst handelt, musste noch ein geeignetes Bauteil als Motor selbst gefunden werden. Die Motoren müssen ein recht hohes Drehmoment aber eine kleine Umdrehungszahl pro Minute aufweisen. Initial war es deshalb die Idee für normale 5V Motoren wie sie auch in DVD Laufwerken verwendet werden ein kleines Getriebe zu konzeptionieren und zu bauen.

Abbildung 3: Prototyp des eigenbau Motorgetriebe

Die Idee bewies sich aber recht schnell als zu schwierig und wurde deshalb fallen gelassen. Als Ersatz für unseren Misslungenen Versuch fiel die Entscheidung auf Modellbau Getriebemotoren. Siehe Abbildung 4.
Ein Anforderung an unsere Demo Roboter Platform war die 3D-Druckbarkeit, sowie deren Modularer Aufbau. Das Fahrgestell unseres Roboters setzt sich aus insgesamt 3 Modulen

Abbildung 4: Modelbau Getriebemotor

zusammen. Die Motorenhalterung woran die Motoren selbst und auch die H-Brücke befestigt sind. Die Zentrale Platform für unseren Microcontroller. Und die beiden Freirollen an der Front des Roboters.
Dieser muss natürlich auch mit Spannung versogt werden. Der Akku unseres Roboter sind sogenannte 18650 LiIon Akkus, diese sind wiederaufladbar und haben die perfekte Größe für unseren kleinen Roboter.
Die Versorgungsspannung für die Motoren kann direkt von unserem Akkupack abgegriffen werden und zur H-Brücke geführt werden. Die H-Brücke benötigt genau so wie der ESP32 eine Versorgungsspannung für die Logik. Diese sollte laut Datenblatt bei 5V liegen, erlaubt sind aber auch 3V. Was in unserem Fall ideal ist da unser ESP32 ebenfalls nur mit 3,3V maxmimal versorgt werden darf.
Die 3,3V Logikspannung werden auf unserer Platine von einem LM317T, einem Lineareren Spannungswandler zur Verfügung gestellt.

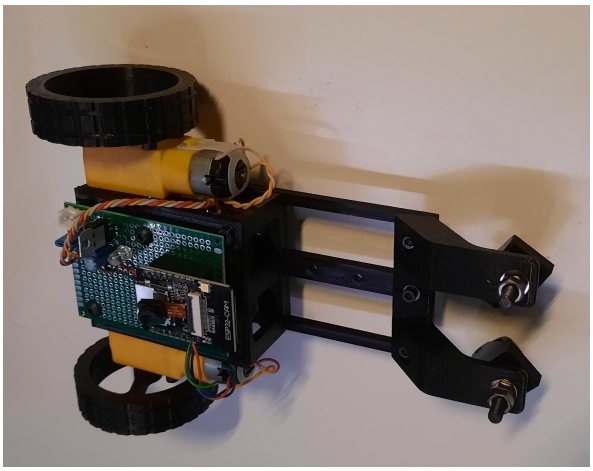

Abbildung 5: Roboter Demo Platform

11.3.2 3D-Druckteile

Die Wichtisgte Anforderung an die Platform war das diese Modular aufgebaut werden kann. Sprich man kann im Nachhinein neue Teile doer Module einfach anschrauben oder kaputte bzw. alte Teile einfach erneuern.

Die Roboter Platform wurde deshalb iterativ designed. Zu Begin wurde nur das Chassis designed und gedruckt. Sobald das Ergebniss zufrieden stellend war wurde das nächste Modul designed und gedruckt. So wurde ein art designkette erstellt. Beginnend beim Chassis inkl. Motorhalterung, danach die Freilaufrollen und Schlussendlich die Halterung für die Elektronik.
Unser finalles Design des Roboter ist unten ersichtlich:
(Bild folgt)

12 Evaluation der Implementierung

12.1 Fehler und Verbesserungen Frontend

12.2 Fehler und Verbesserungen Backend

12.3 Fehler und Verbesserungen Roboter Platform

Getriebe selber bauen sollte doch kein Problem sein, oder?: Initial sollten kleine 5V Motoren aus alten DVD-Laufwerken mit selbstgebauten getriebe als Antrieb dienen. Jedoch musste man recht schnell festellen dass das Design von einem Getriebe doch nicht so einfach ist. Es muss auf den perfekten Abstand der Zähne der Zahnräder geachtet werden. Das die Zahnräder konzentrisch und sich leichtgängig drehen und noch vieles mehr. Letzendlich war unser erster Getriebeversuch auch der letzte. Der Motor hatte einfach viel zu wenig Drehmoment um alle auftrettenden Reibungsverluste zu überkommen. Ein netter versuch um das ein oder andere zu lernen war er aber denoch.
(bild folgt)
Das Problem mit der Freilaufrolle: Der erste Versuch eine Freilaufrolle zu designen und zu drucken funktionierte nur zu 50(prozent). Die Rolle konnte zwar ohne Probleme gedruckt werden, allerdings konnte sie sich nicht frei und leichtgängig genug drehen. Das lag zum einen daran das
(bild folgt)

13 Fazit und Ausblick