
# 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: ROS: React: ESP32: RaspberryPi: Toolchain: micro-ROS, ESP-IDF Micro-ROS

# 7 Zielsetzung und Anforderungen

Entwicklung einer Weboberfläche mit der Roboter aus 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 eine Schnittstelle zwischen Web und ROS geschaffen werden.
(evtl. als 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:*
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

Micro ROS ist eine fork von ROS und damit auch lauffähig auf schwächeren Systemen. Als Microcontroller für unsere Roboter Platform fiel die Entscheidung auf einen ESP32. Der ESP32 bietet für seine Kosten und seine Größe ausreichend Rechenpower und desweiteren unterstützt das Micro-ROS Framework bereits den ESP32. Als alternativen für den ESP32 könnten ansonsten noch ein Rapsberry PI oder ein Arduino bzw. ATmega Microcontroler eingesetzt werden.
Für das Front-End kommen viele Frameworks in Frage aber wir wollten unsere Auswahl vorerst auf React oder Vue beschränken. Mit diesen Frameworks haben bereits alle unsere Gruppenteilnehmer gearbeitet und somit hatten wir alle schon Erfahrung damit gesammelt. (evtl. plain html,css,js)

Als Notlösung für unser Frontend kämme ansonsten Plain HTML und CSS in frage, was aber höchstwahrschienlich in einem sehr großen Aufwand enden würde.
Unter anderem soll zur Demonstration unserer Orchestrierungssoftware ein kleiner Roboter gebaut werden. Der Roboter sollte einfach und schnell gebaut werden können durch 3D-Druckbare Teile. Für den Roboter kommen viele verschiedene steuermechanismen in Frage. Zum einen wäre ein Kettenantrieb möglich, eine Ackermann Lenkung oder ein Differentialantrieb. Knicklenkung

# 10 Evaluation der Lösungsideen anhand der Anforderungen

## 10.1 Wahl des Frontend Frameworks

GUI soll angelehnt an Mainsail OS sein.
React finaler Kanididat weil:?

## 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 Prototyp und Designenscheidungen

Die Anforderungen an die Roboter 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.

Die 3D-Druckbarkeit der Teile sollte auf alle Fälle kein Problem darstellen. Die Elektronik für unseren Roboter basiert auf einem ESP32 welches kostengünstig und recht einfach zu beschaffen ist. Die restlichen elektronischen Bauteile wie z.B. Lineare Spannungswandler können in diversen Elektronik Versandshops gekauft werden.
Ein weiterer wichtiger Punkt ist die Antriebsart des Roboters. Für unsere Anforderungen eignet sich der sogenannte Differentialantrieb sehr gut. Es werden nur 2 Motoren für den Antrieb benötigt und mindestens eine um 360° drehbare Rolle.

# 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

Der ESP32 ist der Microcontroller der den ganzen Roboter steuert. 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.

### 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