Studienarbeitsthema

**STUDIENARBEIT**

des Studienganges Informationstechnik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Sebastian Barz

Simon Buttke

Sebastian Hantelmann

Yusuf Peker

Ferdinand Frank

Abgabedatum

Bearbeitungszeitraum 12 Wochen

Matrikelnummer, Kurs Matrikelnummer, TINF11ITNS

Ausbildungsfirma Firmenname, Stadt

Betreuer der Dualen Hochschule Prof. Jochem Poller

# Erklärung

Gemäß §5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011 versichere ich hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Ort, Datum | Unterschrift |

# Aufgabenstellung

In dieser Studienarbeit soll Anhand von RC – Modellbaufahrzeugen, das „Autonome fahren“ sogenannter „Roboter-Autos“ erprobt werden. Durch das „Autonome Fahren“ soll es den Fahrzeugen möglich sein, ohne menschliche Eingriffe fortbewegen zu können und dabei mit Hindernisse zu umfahren.

Dabei sollen zwei verschiedene Modellbaufahrzeuge mit jeweils unterschiedlicher Steuerungshard- und Software umgesetzt werden.

Das erste Modellfahrzeug besteht aus einem Scania R470 Highline Modelltruck der mit einem Tinkerforge Microcontroller ausgestattet ist. Hierbei soll die Steuerung mit der Programmiersprache Java bewerkstelligt werden.

Das zweite Fahrzeug besteht aus einem Fahrgestell eines Roboters, auf dem ein Raspberry Pi nebst Komponenten wie Kamera und Abstandssensor installiert werden soll. In diesem Fall soll die Ansteuerung mittels der Programmiersprache Python umgesetzt werden.

Abschließend sollen die Hard- und Software Lösungen beider Fahrzeuge analysiert und miteinander verglichen werden.

Weiterhin sollen die bisher im Studium gesammelten Kenntnisse zu Programmiersprachen und Elektrotechnik weiterentwickelt und neue gesammelt werden.

# Vorwort

Die Motivation hinter dieser Studienarbeit geht aus der Idee hervor, zu zeigen, dass selbst in kurzer Arbeitszeit Roboter mit bestimmtem Verhalten auf zwei unterschiedlichen Plattformen umgesetzt werden können. Roboter finden in vielen Bereichen unseres Lebens, zumeist in der Produktion, große Anwendung. Als Beispiel sind hier Roboterarme zum Zusammenbau von Autokomponenten zu nennen. Dadurch gewinnen sie zunehmend an Wichtigkeit, weshalb sich die in dieser Studienarbeit behandelten Projekte diesem Thema annehmen. Initial wurde hierbei eine gemeinsame Planung des Vorgehens und der künftigen Arbeitsteilung vorgenommen, daraufhin arbeiteten beide Teams getrennt an den Robotern. Schlussendlich sind für Vergleich und Fazit die beiden Projekte wieder zusammengelaufen, um hier die Differenzen aufzeigen zu können.

Dabei konzipierten Sebastian Hantelmann, Yusuf Peker und Simon Buttke eine Lösung für den Raspberry Pi und der Programmierspreche Python.

Dem zweiten Subteam gehörten Sebastian Barz und Ferdinand Frank an. Diese hatten eine Lösung mit der Programmierschnittstelle Tinkerforge zu erarbeiten und umzusetzen.

Ein Dank geht an Herrn Professor Poller für die Betreuung dieser Studienarbeit. Nebst diesem hat uns Herr Hargesheimer mit dem Laborinventar bei Hardwarefragen stets unterstützt.

# Inhaltsverzeichnis

[Erklärung II](#_Toc389038679)

[Aufgabenstellung III](#_Toc389038680)

[Vorwort IV](#_Toc389038681)

[Inhaltsverzeichnis V](#_Toc389038682)

[Abkürzungsverzeichnis VII](#_Toc389038683)

[Abbildungsverzeichnis VIII](#_Toc389038684)

[Einleitung 1](#_Toc389038685)

[1.1 Verwendete Bricks und Bricklets 5](#_Toc389038686)

[1.2 Modelltruck Scania R470 Highline 8](#_Toc389038687)

[1.3 Erste Schritte 9](#_Toc389038688)

[1.4 Vorbereitung 14](#_Toc389038689)

[1.5 Kommunikation zwischen Computer und Tinkerforge 14](#_Toc389038690)

[1.6 Ansteuerung des Servo-Bricks 15](#_Toc389038691)

[1.7 Auswerten der Signale vom Ultraschall-Bricklet 18](#_Toc389038692)

[1.8 Erste Fahrten 18](#_Toc389038693)

[1.9 Benutzeroberfläche 20](#_Toc389038694)

[1.10 Parallele Verwendung von Servo- und Ultraschall-Bricks 21](#_Toc389038695)

[1.11 Nutzung von Threads 22](#_Toc389038696)

[Hardware 24](#_Toc389038697)

[1.12 Raspberry Pi 24](#_Toc389038698)

[1.13 Kamera 26](#_Toc389038699)

[1.14 RP5-Roboter 26](#_Toc389038700)

[1.15 Schrittmotor 27](#_Toc389038701)

[1.16 Ultraschallsensor 28](#_Toc389038702)

[Zusammenbau 29](#_Toc389038703)

[Verdrahtung 34](#_Toc389038704)

[WLAN Einrichtung 36](#_Toc389038705)

[1. Versuch Web-Service für Kamera 40](#_Toc389038706)

[JavaScript & Ajax 43](#_Toc389038707)

[WebSockets 45](#_Toc389038708)

[Steuerung Basics 47](#_Toc389038709)

[Versuch 1 50](#_Toc389038710)

[Versuch 2 50](#_Toc389038711)

[Versuch 3 52](#_Toc389038712)

[Fazit und Ausblick 53](#_Toc389038713)

[Appendix 53](#_Toc389038714)

[Glossar 53](#_Toc389038715)

[Literaturverzeichnis 54](#_Toc389038716)

# Abkürzungsverzeichnis

|  |  |
| --- | --- |
| Brickv | Brick Viewer (Tinkerforge Software) |
| Brickd | Brick Daemon (Tinkerforge Software) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Abbildungsverzeichnis

[Abbildung 1: Das Tinkerforge Baukastensystem 2](#_Toc388960961)

[Abbildung 2: Verwendeter Tinkerforge-Stapel und US Bricklet 4](#_Toc388960962)

[Abbildung 3: Scania R470 Highline Orange Edition 7](#_Toc388960963)

[Abbildung 4: Brickv Homescreen 9](#_Toc388960964)

[Abbildung 5: Brick Viewer Updates / Flashing 10](#_Toc388960965)

[Abbildung 6: Truck von oben 11](#_Toc388960966)

[Abbildung 7: Fahrerloge mit US-Distance Bricklet 12](#_Toc388960967)

[Abbildung 8: Raspberry Pi 13](#_Toc388960968)

[Abbildung 9: Raspberry Pi Gehäuse mit Kamerahalterung 14](#_Toc388960969)

[Abbildung 10: Offizielle Raspberry Pi Kamera 15](#_Toc388960970)

[Abbildung 11: Robby RP5 16](#_Toc388960971)

[Abbildung 12: Schrittmotor 17](#_Toc388960972)

[Abbildung 13: HC-SR04 Ultraschallsensor 18](#_Toc388960973)

[Abbildung 14: Robby RP5 mit angeschlossenem Schrittmotor 19](#_Toc388960974)

[Abbildung 15: Holzplatte für Hardware 19](#_Toc388960975)

[Abbildung 16: Akkupack und Blockbatterie 20](#_Toc388960976)

[Abbildung 17: Holzplattenkonstruktion auf dem Robby RP5 21](#_Toc388960977)

[Abbildung 18: Verbindung durch Raspberry Pi Gehäuse zu den GPIO-Anschlüssen 21](#_Toc388960978)

[Abbildung 19: Raspberry Pi und Kamera im Gehäuse 22](#_Toc388960979)

[Abbildung 20: Installation des Ultraschallsensors 23](#_Toc388960980)

[Abbildung 21: GPIO-Anschlüsse des Raspberry Pi 24](#_Toc388960981)

[Abbildung 22: Schaubild zur Verdrahtung von Schrittmotor und Raspberry Pi 24](#_Toc388960982)

[Abbildung 23: Installieren des DHCP-Servers 26](#_Toc388960983)

[Abbildung 24: Installieren des HostAPD-Treibers 26](#_Toc388960984)

[Abbildung 25: Veränderungen in der DHCP-Konfiguration 26](#_Toc388960985)

[Abbildung 26: Konfiguration des DHCP-Dienstes bezüglich Netzwerkdaten 27](#_Toc388960986)

[Abbildung 27: Ändern der IP-Adresse des WLAN-Adapters 28](#_Toc388960987)

[Abbildung 28: Starten des Netzwerkdienstes 28](#_Toc388960988)

[Abbildung 29: Aufnahme des Netzwerkdienstes in die Autostart-Routine 29](#_Toc388960989)

[Abbildung 30: Installieren des Web-Services aus GitHub 29](#_Toc388960990)

[Abbildung 31: Benutzeroberfläche des Web-Services im Internetbrowser 30](#_Toc388960991)

[Abbildung 32: Python Script in PHP ausführen 31](#_Toc388960992)

[Abbildung 33: Beispiele für das Senden von Anweisungen 31](#_Toc388960993)

[Abbildung 34 : Einbinden von JavaScript 33](#_Toc388960994)

[Abbildung 35 : Aufruf der Steuerung per Ajax 34](#_Toc388960995)

[Abbildung 36: Websocket mit Tornado 35](#_Toc388960996)

[Abbildung 37 : Adresse um „Index.html“ zu sehen 35](#_Toc388960997)

[Abbildung 38 : Clientseitige Öffnung eines Websockets 36](#_Toc388960998)

[Abbildung 39 : Buttons für die Steueurng 36](#_Toc388960999)

[Abbildung 40 : Zuweisung der Tasten W und A 37](#_Toc388961000)

[Abbildung 41 : Entscheidung der Fahrtrichtung 38](#_Toc388961001)

[Abbildung 42 : Aufnahme des Videos 39](#_Toc388961002)

[Abbildung 43 : JavaScript Player „JW Player“ 40](#_Toc388961003)

[Abbildung 44 : Installieren und Starten von MJPEG Streamer 41](#_Toc388961004)

# Einleitung

1. Tinkerforge

[[1]](#footnote-1)

Die Firma “TinkerForge“ entwickelte 2011 ein gleichnamiges open Source Baukastensystem um Ideen schnell realisieren zu können. Dies wird durch „die aufeinander abgestimmten Tinkerforge Module“[[2]](#footnote-2) realisiert. Die Hersteller plakatieren, dass es Könnern erlaubt sich auf die Software zu konzentrieren, während es Anfängern die Möglichkeit bietet „mittels spannenden Anwendungen das Programmieren zu lernen.“[[3]](#footnote-3)

Im universitären Umwelt ist die Tinkerforge Plattform eine großartige Möglichkeit Prototypen und Demonstratoren zu entwickeln, die zur Lösung von Problemstellungen in Bachelor- und Studienarbeiten verwendet werden können.

Tinkerforge-Komponenten finden auch in der Industrie Ihren Platz, so unterstützen sie in der Prototypenentwicklung, zur Evaluation von neuen Konzepten, die Automatisierung von Produktionsabläufen und weiteren Projekten mit dem Ziel, „Persönliche Lösungen kostengünstig, schnell und einfach“ umzusetzen.[[4]](#footnote-4)

Die Tinkerforge Module bestehen aus Platinen, die als sogenannte Bricks (Mikrocontrollerbausteine) bezeichnet werden. Auf diesen Platinen sind die verschiedensten elektronischen Bauteile, abhängig von der benötigten Kombination des Bricks, sind Schalter, LEDs, Kondensatoren und viele weitere Bauteile verbaut.[[5]](#footnote-5)

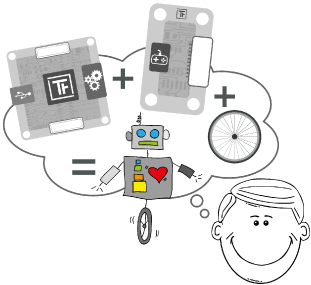
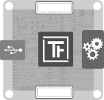


Abbildung : Das Tinkerforge Baukastensystem[[6]](#footnote-6)

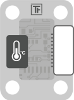
Der Zusammenbau der Module ist intuitiv und ohne Elektronikkenntnisse durchführbar. So müssen die Module nur zusammengesteckt und können anschließend bei Bedarf über Schrauben stabilisiert werden. Dadurch ist es möglich sich bei Lösungsansätzen voll auf die Software und Programmierung zu konzentrieren.

Vom Hersteller werden 3 Modul-Arten unterschieden Bricks, Bricklets und Master-Extensions.

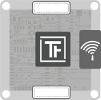
**Bricks**

Bricks sind die Grundbausteine des Tinkerforge, diese verfügen über eine Mini-USB Schnittstelle zur Steuerung. Grundsätzlich dient jeder Brick für die Durchführung einer bestimmten Aufgabe, „zum Beispiel das Steuern eines DC-, Schritt- oder Servomotors“[[7]](#footnote-7). Das Verbauen eines Masterbrick ermöglicht es ganze Tinkerforge-Stapel zu bauen, die nur noch eine USB-Verbindung benötigen.

**Bricklet**

Briklets sind keine eigenständigen Bauteile, sie erweitern vielmehr die Funktionen von Bricks. Einerseits stehen der Produktplattform von Tinkerforge unterschiedlichste Servo-Bricklets zur Messung verschiedenster Größen, wie zum Beispiel der Temperatur zur Verfügung. Andererseits wurden Steuerungs-Bricklets entworfen für die Steuerung von Ein-und-Ausgaben.[[8]](#footnote-8)

**Master-Extensions**

Master Extensions dienen ebenso wie Bricklets der Erweiterung „von einzelnen Master Bricks oder ganzen Stapeln von Bricks“ um neue Funktionen.

## Verwendete Bricks und Bricklets

Zur Steuerung des Modelltruck Scania R470 Highline wird, im Rahmen dieser Studienarbeit, ein Tinkerforge Stapel bestehend aus einem Master-Brick 2.0, zur Kommunikation innerhalb des Stapels, eine WLAN-Masterextension mit einer 2,4 GHz RP-SMA Antenne, als drahtloses Verbindungsmodul zwischen Tinkerforge und gewünschten Steuergerät, sowie ­­­­­einem Servo-Brick, zur Ansteuerung der Servo-Motoren, verwendet. Zusätzlich dient ein Distance US Bricklet zur Messung von Entfernungen unter der Zuhilfenahme von Ultraschall.



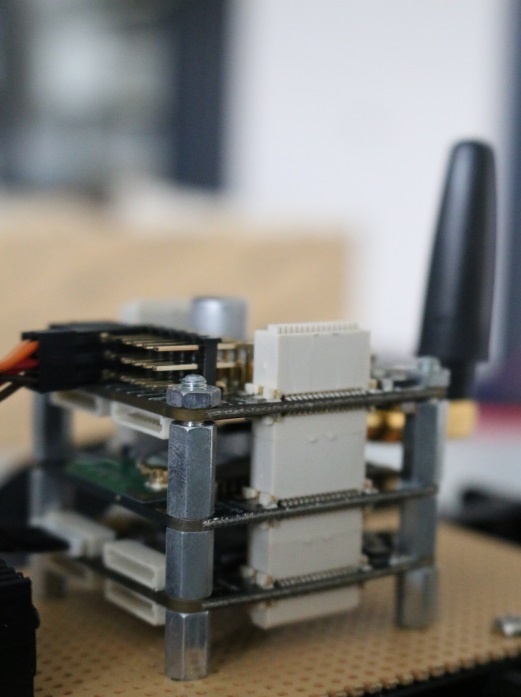
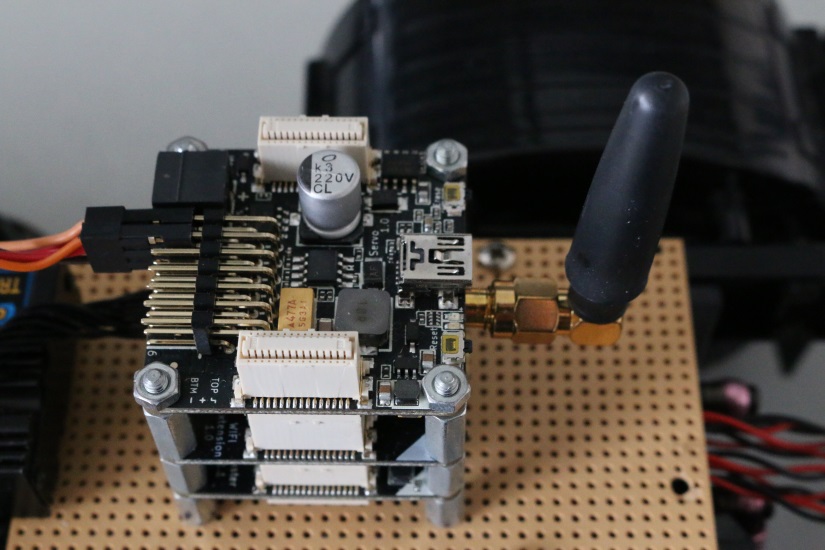
 

Abbildung : Verwendeter Tinkerforge-Stapel und US Bricklet

**Servo-Brick**

Der Servo-Brick dient der Steuerung von bis zu 7 RC-Servomotoren die mit einem Strom von maximal 3A ansteuerbar sind. Er liefert dabei einige Möglichkeiten der Regelung und Steuerung:

* Servospannung, Periode und Pulsweite
  + - * Position, Geschwindigkeit und Beschleunigung

Dazu bietet er einen USB- und 2 Bricklet Anschlüsse[[9]](#footnote-9)

**Wifi-Master-Extension**

Die Wifi-Master-Extension verschafft dem Problem der Kabelgebundenen Steuerung Abhilfe indem es eine WLAN-Verbindung zum Tinkerforge bereitstellt. Dadurch kann der Tinkerforge über Smartphone, Laptops und viele weitere Endgeräte mit Wifi-Modul angesteuert werden. Dabei sind verschiedene Antennen, Verbindungsvarianten und Verschlüsselungen konfigurierbar.

* DHCP, Static IP, 802.11b/g/n Access Point, AdHoc
* WEP, WPA, WPA2 Personal und Enterprise Verschlüsselung
* 2,4Ghz RP-SMA Antennen in verschiedenen Größen

Dazu ist sie mit einem 18dBm Verstärker ausgestattet welche Verbindungen über große Reichweiten und Hindernisse ermöglicht.[[10]](#footnote-10)

**Master Brick 2.0**

„Der Master Brick ist mit einem 32 Bit ARM Microkontroller ausgestattet und besitzt 2 Aufgaben.“[[11]](#footnote-11) Einerseits weist es 4 Briklet Anschlüsse zur Erweiterung durch viele Bricklets.

Als weitere Aufgabe dient ein Master Brick als Fundament und Kommunikator eines Stapels. Dabei wird er als unterster Brick, sozusagen als Basis des Stapels verwendet, sendet und empfängt daten über die USB-Schnittstelle von PCs und weiteren Steuergeräten an die im Stapel enthaltenen Bricks weiter.

**Distance US Bricklet**

„Das Distance US Bricklet ist mit einem Ultraschall-Entfernungsmesser ausgestattet“[[12]](#footnote-12) und dient dem Messen von Entfernungen von 2-400 cm. Die gemessenen Entfernungen werden in Form eines einheitslosen Wertes zurückgegeben.

Dazu ist es mit einem HC-SR04 Sensor ausgestattet und besitzt einen Messwinkel von 15°.[[13]](#footnote-13)

**Stromversorgung**

Der Tinkerforge benötigt eine eigene Stromversorgung mit einer Spannung von 9V. Um ein Optimum an Mobilität zu erreichen wird eine Powerbank verwendet, die in die Fahrerloge gelegt und mit einem USB- zu Mini-USB-Kabel mit dem Tinkerforge verbunden werden kann.

## Modelltruck Scania R470 Highline

Der Tinkerforge-Stapel wird zur Steuerung eines „Scania R470 Highline“ Modelltrucks verwendet.

Dieser wird Grundsätzlich als Modellbaufahrzeug über eine RC-Fernsteuerung betrieben und gesteuert. Dazu ist auf dem Truck ein Modelcraft „Truck 90 – WP“ Fahrtregler verbaut. Da die Steuerung aber über den Tinkerforge bewerkstelligt werden soll, bleibt dieser Fahrtregler aber unbenutzt.

Zum Antrieb des Trucks werden zwei Typ 540 Baukastenmotore eingesetzt, diese sind unterhalb der Fahrerloge verbaut und dienen als Frontantrieb.

Der Truck ist 452 mm lang, 187 mm breit und 293 mm hoch. Ist er voll aufgebaut und sind alle Teile enthalten misst er ein Gewicht von 2450 Gramm.[[14]](#footnote-14)



Abbildung 3: Scania R470 Highline Orange Edition[[15]](#footnote-15)

**Stromversorgung**

Die Stromversorgung des Trucks wird über einen 9V starken, handelsüblichen Lithium-Ionen-Akku bewerkstelligt. Dieser befindet sich unterhalb des Trucks, wo er jederzeit durch lösen zweier Spangen, nach vorherigem umdrehen des Trucks entnommen werden kann. Damit sich dieser Akkumulator nicht dauerhaft im eingeschalteten Zustand befindet, wurde an der Verkabelung ein Schalter angebracht, welcher das Aktivieren und Deaktivieren ermöglicht.

## Erste Schritte

**Initiale Installation benötigter Software**

Die adäquate Steuerung und Benutzung des Trucks wird grundsätzlich über das Wifi-Bricklet oder den USB-Eingang des Tinkerforge ermöglicht.

Initial muss hierfür aber eine USB-Verbindung hergestellt werden, worüber Einstellungen zur Einrichtung, aber auch erste Tests durchgeführt werden können.

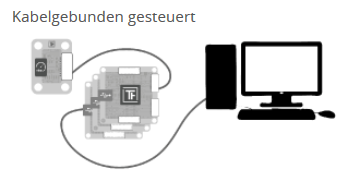


Abbildung : Tinkerforge: Kabelgebunden gesteuert[[16]](#footnote-16)

Zur Benutzung des Tinkerforge über einen PC muss zuvor ein Dienst, der BrickDeamon (Brickd) und eine Benutzeroberfläche zur Durchführung von ersten Tests und Firmware-updates der sogenannte BrickViewer (BrickV) installiert und gestartet werden.

Diese Softwareprodukte können auf der Internetseite der Firma Tinkerforge unter <http://www.tinkerforge.com/de/doc/Downloads.html> heruntergeladen werden.

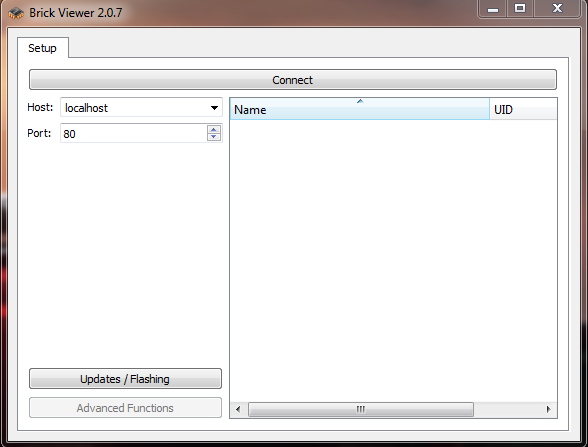


Abbildung : Brickview Homescreen

**Aktualisieren der Firmware der Tinkerforge-Komponenten und Software**

Bevor jedoch der Zugriff über den Brickviewer möglich ist, muss ein Update der Firmware der Bricklets durchgeführt werden. Dies ist durch die Schaltfläche „Update / Flashing“ des Brickviewers möglich. Nach betätigen dieser Schaltfläche öffnet sich ein neues Fenster.

Besteht eine Verbindung zum Internet, so ermittelt der Brickviewer unterschiede zwischen der Firmware der verwendeten Tinkerforge-Komponenten und der aktuell verfügbaren Version. Sind Diskrepanzen vorhanden markiert der Brickviewer diese farblich, wie folgend zu sehen. Orange markierte Komponenten, können aktualisiert werden, während rot markierte aktualisiert werden müssen.

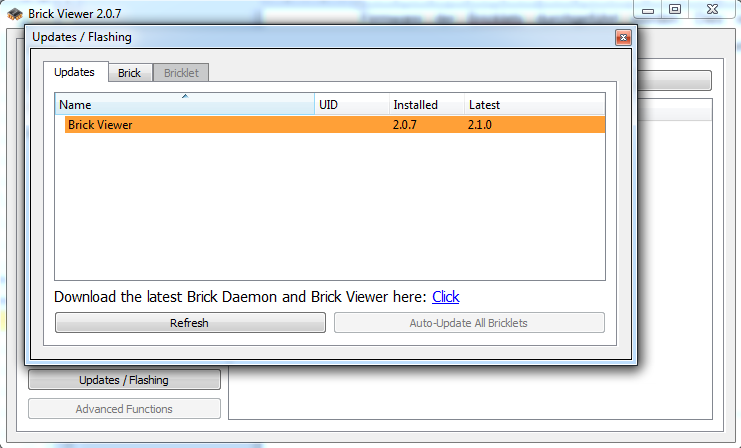


Abbildung : Brick Viewer Updates / Flashing

**Einrichten der WLAN-Verbindung**

Über den Brickviewer wird die Einrichtung des WLAN-Bricklets durchgeführt. Das Team entschied sich für die Einrichtung eines Access Points mit einer statischen IP. Die Wahl fiel auf diese WLAN-Einstellung da sie die konsistenteste und stabilste bei einigen Versuchen zu sein scheint.

Um nun eine schnurlose Verbindung zum Tinkerforge WLAN-Bricklet herzustellen sind die Internetoptionen des Computers anzupassen und eine, zur statischen IP des Access Points des Trucks passende, IP zu konfigurieren.



Abbildung : Tinkerforge: Drahtlos gesteuert[[17]](#footnote-17)

Durch den Brickviewer können über die WLAN- oder USB-Verbindung Servo-Motoren Tests durchgeführt und ein Verständnis für die einzelnen Parameter und Einstellmöglichkeiten erlangt werden.[[18]](#footnote-18)

**Befestigen der Tinkerforgebauteile**

Der Tinkerforge Stapel wurde anschließend mithilfe einer Platte und Schrauben an der Querstrebe des Trucks befestigt. Das Befestigen des Tinkerforge Stapels sorgt für Stabilität und Sicherheit der Hardware, welche zuvor nur durch die verbunden Kabel an dem Truck gehalten wurden.



Abbildung : Truck von oben

Die Befestigung des Ultraschall Entfernungsmessers wird mit einer Metallstrebe und einigen Schrauben bewerkstelligt. Er befindet sich auf der rechten Seite der roten Fahrerloge des Trucks und ist in Fahrtrichtung ausgerichtet. Einen optischen Eindruck dieser Installation liefert das folgende Bild.



Abbildung : Fahrerloge mit US-Distance Bricklet

1. Programmierung

Die Tinkerforge-Plattform kann unter anderem mit folgenden Programmiersprachen angesteuert werden: C/C++, C#, Delphi, Java, JavaScript, Perl, PHP, Python, Ruby und Shell.[[19]](#footnote-19) Aufgrund der vielseitigen bisher im Studium gesammelten Erfahrungen mit Java, wurde diese Programmiersprache zur Kommunikation mit den Bricks ausgewählt.

## Vorbereitung

Damit der Tinkerforge über Java angesprochen werden kann, müssen als erstes die Java-Bindings heruntergeladen und in die Entwicklungsumgebung eingebunden werden. Diese liegen auf der Website der Tinkerforge GmbH zum Download bereit. Zusätzlich stehen Quelltexte und Beispielanwendungen zur Verfügung. So können schnell erste Testprogramme erstellt werden.

Im Anschluss wurde die Programmierschnittstellen (API) der verwendeten Bricks und Bricklets untersucht. In den nachfolgenden Kapiteln wird erklärt, wie die Kommunikation mit dem Tinkerforge funktioniert und wie die einzelnen Elemente bedient werden können.

## Kommunikation zwischen Computer und Tinkerforge

Der Tinkerforge kann wie bereits angesprichen entweder über die USB-Verbindung oder über den WLAN-Brick erreicht werden. In beiden Fällen wird eine TCP/IP-Verbindung hergestellt. Dafür muss das zugehörige Java-Object com.tinkerforge.IPConnection erzeugt werden. Wie von einer Verbindung über IP gewohnt, werden ein Host und ein Port benötigt. Aus der Dokumentation geht hervor, dass der USB-Host „localhost“ und der Port „4223“ lauten müssen. Host bei der WLAN-Variante ist die IP, die der Tinkerforge sich entweder selbst zuweist, oder von einem externen Route via DHCP zugewiesen bekommt; als Port wird der http-Port 80 benutzt. Die hergestellte Verbindung muss im letzten Schritt noch an die entsprechenden Bricks gebunden werden. Dazu besitzt jeder Brick eine eindeutige Nutzer ID. Dies dient der Unterscheidung, denn es können in einem Stapel Typen des gleichen Bricks mehrfach vorkommen.

Nur mit der Verwendung von Masterbrick und gegebenenfalls der WLAN-Extension kann nun schon ein erstes Testprogramm aus der Entwicklungsumgebung gestartet werden:

String host = "localhost";

**int** port = 4223;

String UID = "Q4rOfG"

IPConnection ipcon = **new** IPConnection();

BrickMaster master = **new** BrickMaster(UID, ipcon);

ipcon.connect(host, port);

**int** voltage = master.getStackVoltage();

System.*out*.println("Stack Voltage: " + voltage/1000.0 + " V");

ipcon.disconnect();

Die Ausgabe dieses Programmes lautete in unserem Fall „Stack Voltage: 8,988 V“ und gab somit die Spannung aus, mit der der Masterbrick von unserer Powerbank versorgt wurde. Zu beachten ist die einfache Syntax, mit der der Tinkerforge angestuert werden kann. Im obigen Beispiel wurde die Spannung über die USB-Verbindung abgefragt. Für die Benutzung des WLAN-Moduls müssen lediglich Host und Port geändert werden. Ein neues Objekt zu erstellen, wie bei dem BrickMaster, ist nicht notwendig.

## Ansteuerung des Servo-Bricks

Um den Servo-Brick ansteuern zu können, muss wieder ein entsprechendes Java-Objekt erstellt werden und mit der IPConnection verknüpft werden. Zusätzlich muss dem Brick aber noch mitgeteilt werden, welche Spannung es auf die Ausgänge legen soll. Wir verwenden die volle dem Brick zur Verfügung stehende Spannung von 9V:

String SERVO\_UID = "6JoKit";

BrickServo servo = **new** BrickServo(SERVO\_UID, ipcon);

servo.setOutputVoltage(9000);

Der Brick ist nun fertig konfiguriert. Es müssen allerdings pro verwendeten Ausgang noch verschiedene Einstellungen vorgenommen werden. Der Truck verfügt über insgesamt drei Servo-Anschlüsse für die Lenkung, die elektronische Fahrzeugsteuerung und für die Gangschaltung. Da wir die Gangschaltung vorerst nicht benötigen, müssen zwei Servos eingestellt werden. Die nachfolgend zu sehenden Werte haben sich als praktikabel erwiesen und werden beim Programmstart gesetzt:

servo.setPulseWidth((**short**)0, 1000, 2000);

servo.setAcceleration((**short**)0, 30000);

servo.setPosition((**short**)0, (**short**)0);

servo.enable((**short**)0);

Zu beachten ist, dass bei allen Methoden im ersten Parameter (Typ short) der Ausgang angegeben werden muss, für den die Änderung vorgenommen werden soll.

Da die Servos über Pulswellenmodulation angesteuert werden, müssen eine minimale und eine maximale Pulsweite eingestellt werden. Die Werte entsprechen einer Pulsweite von 1ms bis 2ms. Damit die Befehle schnell umgesetzt werden können, wurde die maximale Beschleunigung von 30.000 gesetzt. Bevor wir die Servos mit enable dann einschalten, werden sie noch in die Ausgangsposition „0“ gesetzt.

Dabei bedeutet die Ausgangsposition bei der Lenkung „Geradeaus“ und bei der Fahrzeugsteuerung „Stillstand“. Akzeptiert werden Werte von -9000 bis 9000, wobei negative Werte „Lenken nach links“ bzw. „Rückwärts fahren“ bedeuten und positive Werte „Lenken nach rechts“ bzw. „Vorwärts fahren“.

Die elektronische Fahrzeugsteuerung, die Strom von einem unter dem Fahrzeug montierten Akku bezieht, wertet also nur die anliegende Spannung aus und gibt die entsprechenden Signale selbst an den Motor weiter. Im Gegensatz dazu wird der Lenkungs-Servo direkt vom Tinkerforge angesteuert.

Die Implementierung der Methoden aller Bricks wurde relativ umständlich umgesetzt, was daran liegt, dass sie alle Exceptions werfen können. Das bedeutet, keine Funktion kann ohne try-catch-Blöcke genutzt werden, was den Quelltext sehr aufblähen würde. Ein kleiner Workaround gliedert die genutzten Funktionen in selbst definierte Methoden aus, die dann Exceptionhandling nur an einer Stelle betreiben und nicht über den gesamten Code verteilt:

**private** **static** **void** setPosition(**int** ser, **int** pos) {

**if** (ser == 0)

speedslider.setValue(pos);

**else**

steerslider.setValue(pos);

**try** {

servo.setPosition((**short**)ser, (**short**)pos);

} **catch** (Exception e) {

}

}

Das verhindert zwar einen unübersichtlichen Quelltext, ist aber nicht im Sinne der Fehlerbehandlung

## Auswerten der Signale vom Ultraschall-Bricklet

Auch Bricklets besitzen eine eindeutige ID, die bei der Objekterzeugung angegeben werden muss. Um den gerade gemessenen Abstandswert zu erhalten, gibt es folgende einfache Methode:

dir = **new** BrickletDistanceUS(BRICKLET\_UID, ipcon);

int d = getDistance();

Das US-Bricklet kann auch kontinuierlich über eine Listener-Funktion abgefragt werden, die jedes Mal ausgeführt wird, sobald sich der Wert ändert. Es werden Werte von 0 bis 4095 zurückgeliefert, die je nach Betriebsspannung auch etwas schwanken können[[20]](#footnote-20). Da der Messbereich des Ultraschalls von 2cm bis 400cm reicht, kann eine Division durch 10 als guter Näherungswert für die Entfernung in Zentimetern angesehen werden.

## Erste Fahrten

Um das Fahrverhalten des Trucks, also Beschleunigung, Lenkung, Geschwindigkeit, aber auch Reaktionszeit von Computer zur Plattform zu testen, wurde eine einfache „Fernsteuerung“ programmiert. Die Pfeiltasten der Tastatur werden dabei abgefragt und entsprechende Befehle an den Tinkerforge gesendet. Nachfolgend die wesentlichen Zeilen Quelltext der JavaFX-Anwendung:

scene.setOnKeyPressed(**new** EventHandler<KeyEvent>() {

**public** **void** handle(KeyEvent event) {

KeyCode code = event.getCode();

**if** (code == KeyCode.UP) //Antrieb

setPosition(0, 3000);

**else** **if** (code == KeyCode.DOWN)

setPosition(0, -3000);

**else** **if** (code == KeyCode.LEFT) //Lenkung

setPosition(1, -9000);

**else** **if** (code == KeyCode.RIGHT)

setPosition(1, 9000);

}

});

scene.setOnKeyReleased(**new** EventHandler<KeyEvent>() {

**public** **void** handle(KeyEvent event) {

KeyCode code = event.getCode();

**if** (code == KeyCode.UP) //Antrieb

setPosition(0, 0);

**else** **if** (code == KeyCode.DOWN)

setPosition(0, 0);

**else** **if** (code == KeyCode.LEFT) //Lenkung

setPosition(1, 0);

**else** **if** (code == KeyCode.RIGHT)

setPosition(1, 0);

}

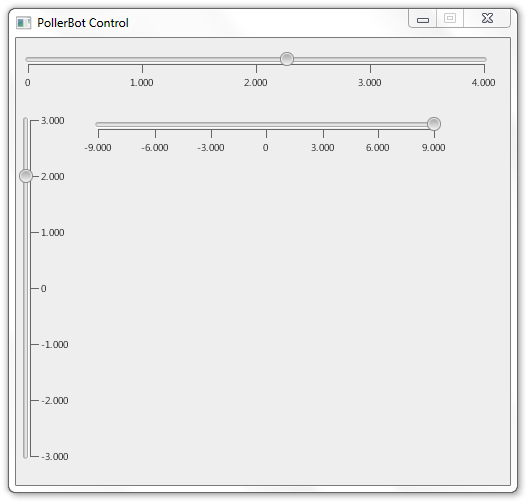
});

Immer wenn eine Pfeiltaste gedrückt wird, wird die Position des jeweiligen Servos verändert. Der Truck lenkt dann und/oder fährt. Wird die Taste losgelassen, kehren die Seros in ihren ursprünglichen Zustand zurück.

Die Steuerung des Trucks gestaltet sich relativ träge. Das liegt daran, dass der Tinkerforge selbst keine Berechnungen vornimmt, sondern nur die Befehle eines Computers an die einzelnen Bauteile umleitet. Aus diesem Grund wurden auch die Werte für den Antrieb von 9000 auf 3000 herabgesetzt. Bei der Verwendung der Maximalwerte kann der Truck wegen seiner hohen Geschwindigkeit nicht immer rechtzeitig gestoppt werden. Außerdem drehen auf glattem Untergrund die Reifen auf dem ersten Stück durch.

## Benutzeroberfläche

Da die Ausgabe von Entfernungswerten auf der Konsole nicht sehr anschaulich ist und Veränderungen nur schwer nachvollzogen werden können, wurde eine grafische Benutzeroberfläche programmiert. Es wurde ein Schieberegler implementiert für den gemessenen Entfernungswert und jeweils einen für die beiden Servos, die anzeigen, welche Signale zum Fahren und Lenken an den Tinkerforge gesendet werden.



Der obere horizontale Schieberegler gibt die gemessene Entfernung an (im Beispiel gut 2 Meter). Der untere zeigt den Zustand der Servolenkung an (lenkt nach rechts). Der Regler am linken Bildrand zeigt die Geschwindigkeit an (langsames Fahren).

## Parallele Verwendung von Servo- und Ultraschall-Bricks

Da der Truck aber nicht vom Menschen gesteuert werden soll, sondern das Ziel das autonome Fahren ist, muss der Ultraschallsensor kontinuierlich abgefragt werden und den Messwerten entsprechende Aktionen an die Servos gesendet werden. Die IPConnection sieht vor, dass sie von mehreren Bricks genutzt werden kann.[[21]](#footnote-21) Der dazugehörige Code:

IPConnection ipcon = **new** IPConnection();

servo = **new** BrickServo(SERVO\_UID, ipcon);

dir = **new** BrickletDistanceUS(BRICKLET\_UID, ipcon);

ipcon.connect(HOST, PORT);

Der erste Ansatz bestand darin, die Listener-Funktion des US-Bricklets zu verwenden und dann beim Über- oder Unterschreiten von Schwellwerten entsprechende Aktionen einzuleiten. Allerdings kann der Tinkerforge Bricks nie wirklich gleichzeitig verwenden. Bei Tests offenbarte sich, dass er entweder die Abstandswerte ausgeben oder die Servos ansteuern konnte. Genau dementsprechend ist auch der BrickViewer aufgebaut. Die Bricks können nur separat in Tabs angesprochen und bedient werden. Wir konnten sogar mit dem BrickViewer beeinflussen, welche der beiden Bricks in unserem Java-Programm funktionieren soll und welche nicht.

Es liegt nahe, dass je nach geöffnetem Tab ein Brick aktiviert und ein zugehöriger Listener gestartet wird, während die anderen Bricks pausiert werden.

## Nutzung von Threads

Da keine echtparallele Nutzung der Bricks möglich ist, bleibt nur folgende Alternative: Es muss dauerhaft erst ein Abstandswert ermittelt und ausgewertet werden und dann direkt eine Anweisung an den Tinkerforge geschickt werden.

Da eine Endlosschleife direkt im JavaFX-Programmteil dazu führt, dass das Programm nicht richtig starten kann, wurde die Schleife, die beim Fahren ausgeführt wird, in einen Thread in der Klasse ausgegliedert. Dazu wurde das Interface Runnable implementiert und eine Methode run() definiert, die folgenden Inhalt hat:

**public** **void** run() {

**int** d;

**while** (**true**) {

d = getDistance();

distanceslider.setValue(d);

**if** (d < 500)

setPosition(0, 0);

**else**

setPosition(0, 3000);

}

}

In dieser einfachen Schleife wird dem Tinkerforge gesagt, der Truck soll so lange geradeaus fahren, bis die gemessene Distanz des Ultraschallbricklets 500, also etwa 50cm, unterschreitet. Nach dem Testlauf kann das Programm einfach beendet werden, indem das GUI Fenster geschlossen wird.

Generell reagiert das Programm bei der Verwendung von Threads schneller, als noch mit den Listener-Funktionen der Bauteile.

# Hardware

Dieses Projekt beinhaltet verschiedenste Hardware-Komponenten. Diese werden verbaut und verdrahtet, und daraufhin von Seiten der Software angesteuert.

## Raspberry Pi

Der Raspberry Pi Model B ist ein komplettes Rechensystem, welches vor allen Dingen für seine geringe Größe und einen niedrigen Preis bekannt geworden ist. Bei Maßen von 8,6cm in der Breite und 5,4cm in der Tiefe ist er zu einem Preis von aktuell ca. 35€ erhältlich.



Abbildung 10: Raspberry Pi

Dabei bringt der Raspberry Pi eine für viele Einsatzmöglichkeiten ausreichende Ausstattung an Hardware mit. Zu dieser Hardware gehören 512 Megabyte RAM-Speicher und ein Prozessor mit einer Rechenleistung von 700 Megahertz. Peripherie wie z.B. Maus und Tastatur, oder auch WLAN-Adapter können über die zwei USB-Anschlüsse verbunden werden. Mittels LAN-Anschluss wird er mit einem Netzwerk verbunden und über HDMI-Output und S-Video werden die Bildsignale ausgegeben. Durch einen Micro-USB-Anschluss wird die Energieversorgung sichergestellt. Zur Kommunikation mit anderen Geräten, wie z.B. Platinen, besitzt er außerdem noch GPIO-Pins, an welche sich Jumper-Kabel anschließen lassen. Einer der Besonderheiten dieses Rechensystems ist der Fakt, dass es ohne Betriebssystem ausgeliefert wird. Da es keinen Festplatten- oder Flashspeicher besitzt, muss das Betriebssystem auf einer SD-Karte installiert werden, um es so auf dem Raspberry Pi zu benutzen. Dieser bootet dann direkt von der SD-Karte. Um den Raspberry Pi nicht äußeren Einflüssen auszusetzen, ist es ratsam, ihn durch ein Gehäuse zu schützen.



Abbildung 11: Raspberry Pi Gehäuse mit Kamerahalterung

Bei diesem Gehäuse werden alle Anschlüsse freigehalten und zudem lässt sich eine Kamera installieren. Dadurch ist der Raspberry Pi gegen Beschädigungen wie Stöße geschützt. Aufgrund des guten Preis-/Leistungsverhältnisses gibt es mittlerweile eine große Fangemeinde des Raspberry Pi, in welcher ein Groß an Anleitungen und Ideen für neue Einsatzmöglichkeiten zu finden ist.

## Kamera

Die offizielle Raspberry Pi Kamera ist eine Videokamera, welche eine Videoauflösung von 1080p, also Full HD, bei 30 Bildern pro Sekunde liefert. Es ist auch möglich, mit der Kamera Bilder zu schießen, dies geschieht mit 5 Megapixeln. Die Kamera verfügt über ein 150mm langes Flachbandkabel, das zu dem 15-poligen MIPI-Kamera-Anschluss des Raspberry Pi führt. Angebracht werden kann sie z.B. an speziellen Gehäusen wie dem bereits gezeigten Raspberry-Pi-Gehäuse. Hier kann es dann auf der Oberseite eingesetzt werden und ist dadurch schwenkbar. Einsatzbereiche für die Kamera sind z.B. die Benutzung als Web- oder auch als Überwachungskamera.



Abbildung 12: Offizielle Raspberry Pi Kamera

## RP5-Roboter

Der Roboter Robby RP5 ist als Produkt ein autonomer, mobiler Roboter. In seinem eigentlichen Verwendungszweck kann er mittels des Mikrocontrollers, welcher auf dem Board auf der Oberseite installiert ist, programmiert werden. Da in dieser Arbeit mit dem Raspberry als Mikrocontroller gearbeitet werden soll, wird dieses Board deinstalliert und durch eine eigene Konstruktion ersetzt. In diesem Fall wird der Roboter also einzig als Fahrgestell genutzt, welches durch seine zwei getrennt steuerbaren Elektromotoren angetrieben wird.

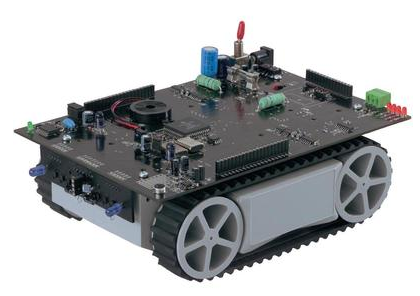


Abbildung 13: Robby RP5

## Schrittmotor

Der Schrittmotor dient zur Ansteuerung der beiden Elektromotoren des RP5 Roboters. Er ist ein Synchronmotor und sorgt dafür, dass beide Antriebe gleichmäßig und synchron geschaltet werden. Dabei hat er zwei Ausgänge, an die diese dann angeschlossen werden können. Zudem gibt es vier Anschluss-Pins für die Steuerung der Energieabgabe auf diese Motoren. Die Energieversorgung vom Schrittmotor selbst geschieht über eine 9V Blockbatterie. Die ganzen Anschlussmöglichkeiten sind auf der nächsten Abbildung als Vogelperspektive dargestellt.

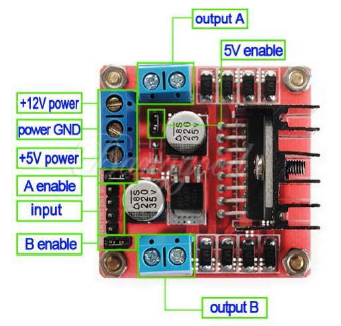


Abbildung 14: Schrittmotor

## Ultraschallsensor

Der HC-SR04 ist ein Ultraschallsensor, um Distanzen zu Objekten zu messen. Das Vorbild aus der Natur ist dabei die Umgebungsorientierung von Delphinen oder auch Fledermäusen. Dies ist dabei in einer Entfernung von zwei Zentimetern bis zu vier Metern bei einer Genauigkeit von drei Millimeter möglich. Laut Angabe des Herstellers wird die Arbeit dabei nicht Sonnenlicht oder zu dunkle Stellen beeinträchtigt. Der Sensor hat vier Pins für Anschlüsse: Eine Energieversorgung Vcc, eine Erdungsmöglichkeit GND, einen Trigger und einen Echokanal. Während sich Vcc und GND um die Grundversorgung kümmern, werden Trigger und Echo zur Ansteuerung benötigt. Indem Eingangsspannung auf den Trigger-Pin gegeben wird, startet der Sensor seinen Messvorgang und schickt dafür ein Ultraschallsignal in Richtung des Sensors ab. Dieser Ultraschall wird von sich frontal vor dem Sensor befindenden Objekten zurückgeworfen und dann vom Sensor wieder als Reflektion aufgefasst. Daraufhin wird der Echo-Pin auf eine feste Spannung gesetzt und verbleibt für eine bestimmte Zeit in diesem Zustand. Wie lange, das hängt von der gemessenen Entfernung ab. Je größer die Entfernung, umso länger verbleibt der Pin in diesem Zustand. Diese Zeit, welche in Mikrosekunden angegeben wird, lässt sich dann im operierenden Programm für die Berechnung verwenden.



Abbildung 15: HC-SR04 Ultraschallsensor

# Zusammenbau

Da der RP5-Roboter nur als Fahrgestell benutzt werden soll, wird das Board mit dem dazugehörigen Mikrocontroller abgebaut. Dadurch ist der Zugriff auf die Elektromotoren des Roboters mittels einfacher Leitungen möglich. Der Schrittmotor soll diese dann ansteuern und findet seinen Platz im Inneren des Robotergehäuses, wo er auch bleiben und bei Fertigstellung nicht mehr sichtbar sein soll. Zu diesem Stand sieht der Roboter wie in folgender Darstellung gezeigt aus.

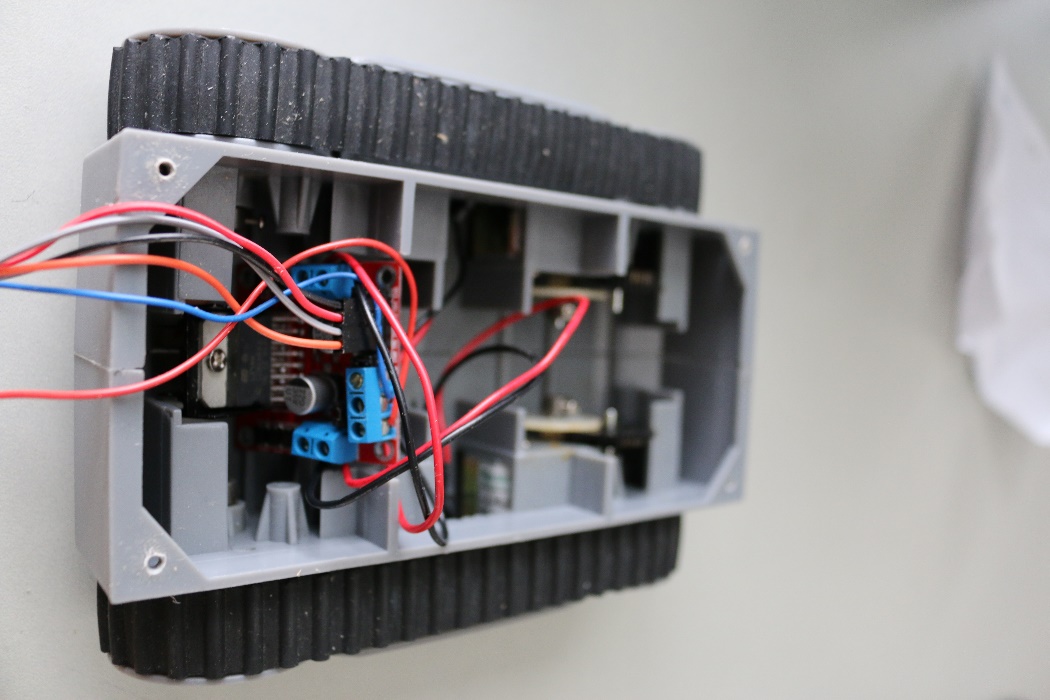


Abbildung 16: Robby RP5 mit angeschlossenem Schrittmotor

Um die weiteren Komponenten installieren zu können, wird eine ebene Fläche benötigt. Hier wird anstelle des vorher installierten Mikrocontroller-Boards eine dünne Holzplatte mit den Maßen 14cm x 18cm angebracht. Diese wird mittels eines M3-Bohrers an den Ecken mit Löchern vom Durchmesser 3mm ausgestattet. Sie sollen deckungsgleich zu den Löchern an den Ecken der Oberseite des RP5-Roboters sein. Daher wurden die Positionen der Löcher am Roboter ausgemessen und dann an den entsprechenden Punkten am Holzbrett die Bohrungen vorgenommen. Um die Leitungen vom Schrittmotor aus dem Inneren des Roboters einfach nach außen leiten zu können, wird zudem ein Spalt in die Holzplatte eingebaut. Die Holzplatte ist in der folgenden Abbildung dargestellt.



Abbildung 17: Holzplatte für Hardware

Da die Löcher von Platte und Roboter nun deckungsgleich sind, können sie miteinander verbunden werden. Dafür werden Abstandsbolzen, ebenfalls mit 3mm Durchmesser, verwendet. Diese halten durch ihr Schraubgewinde die Holzplatte am Roboter, und werden dafür an allen vier Ecken installiert. Die Abstandsbolzen haben eine Länge von 3cm, wodurch Akkupack und 9V Blockbatterie hier Platz finden.



Abbildung 18: Akkupack und Blockbatterie

Für den weiteren Teil der Komponenten wird eine zweite Holzplatte benötigt, welche mit den gleichen Löchern ausgestattet wird. Auch hier wird wieder ein Spalt für die Weiterführung der Leitungen eingebaut. Diese zweite Platte wird dann auf den vier Abstandsbolzen aufgelegt und von der anderen Seite mit Schrauben fixiert. Dadurch erhält der Roboter eine zweite Etage zum Befestigen der übrigen Komponenten.

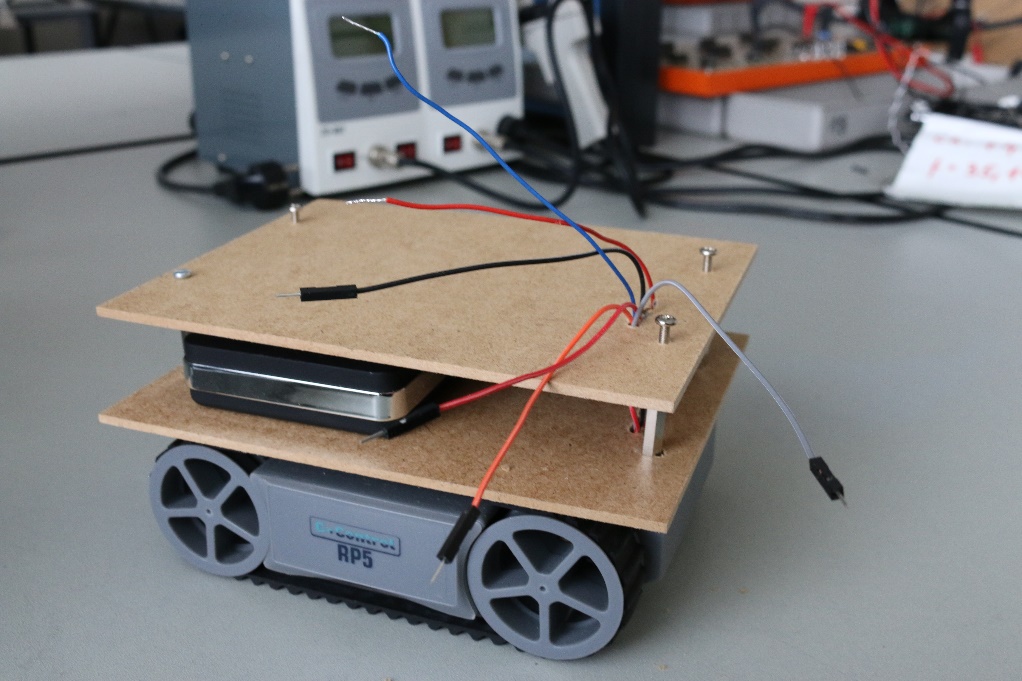


Abbildung 19: Holzplattenkonstruktion auf dem Robby RP5

Hier findet dann der Hauptbestandteil der Arbeit Platz, der Raspberry Pi, welcher als Mikrocontroller den Roboter steuern soll. Um hier Platz für die Verdrahtung zu schaffen, wurden über den GPIO-Pins ebenfalls Löcher in das Gehäuse gebohrt, die die Leitungen nach außen leiten.

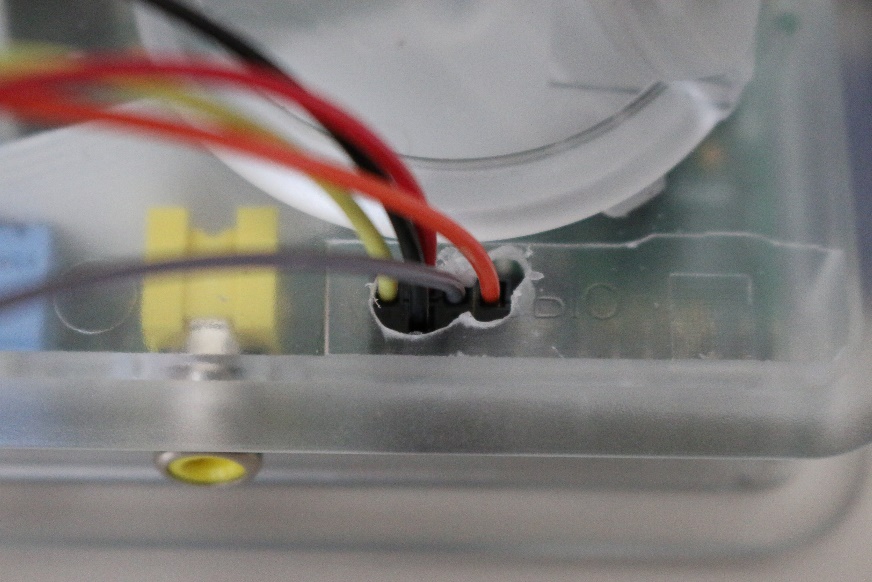


Abbildung 20: Verbindung durch Raspberry Pi Gehäuse zu den GPIO-Anschlüssen

Damit der Roboter visuelle Rückmeldungen geben kann, wird die offizielle Raspberry Kamera installiert. Diese findet auf der oberen Seite des Raspberry Pi ihren Platz, um hier frei drehbar ihre Aufnahmen zu machen. An dieser Position kann sie zudem auch aufrecht gestellt werden, um so nach vorne zu blicken.

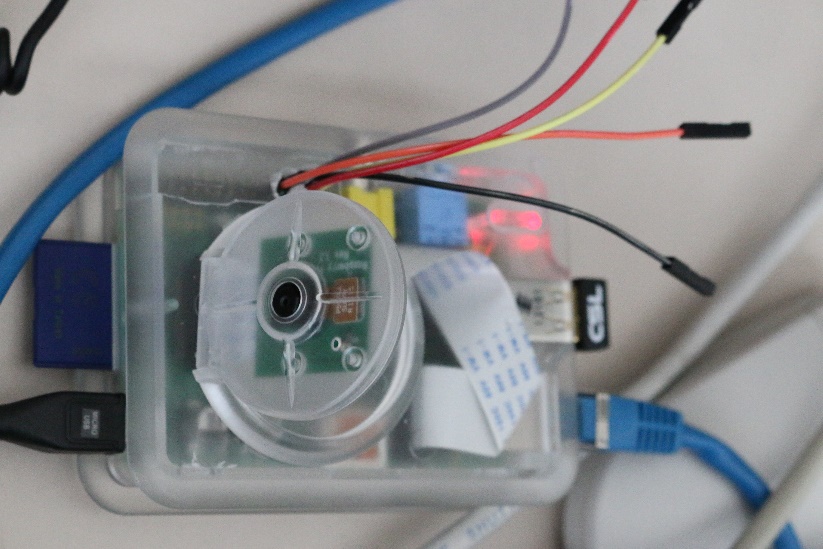


Abbildung 21: Raspberry Pi und Kamera im Gehäuse

Zur Distanzmessung findet der Ultraschallsensor HC-SR04 an der Vorderseite des Roboters seinen Platz. Hier wird er an einer möglichst tiefen Position installiert, um so auch tiefliegende Objekte erkennen zu können, wie z.B. Kanten des Bodens. Einen optischen Eindruck dieser Installation liefert das folgende Bild.

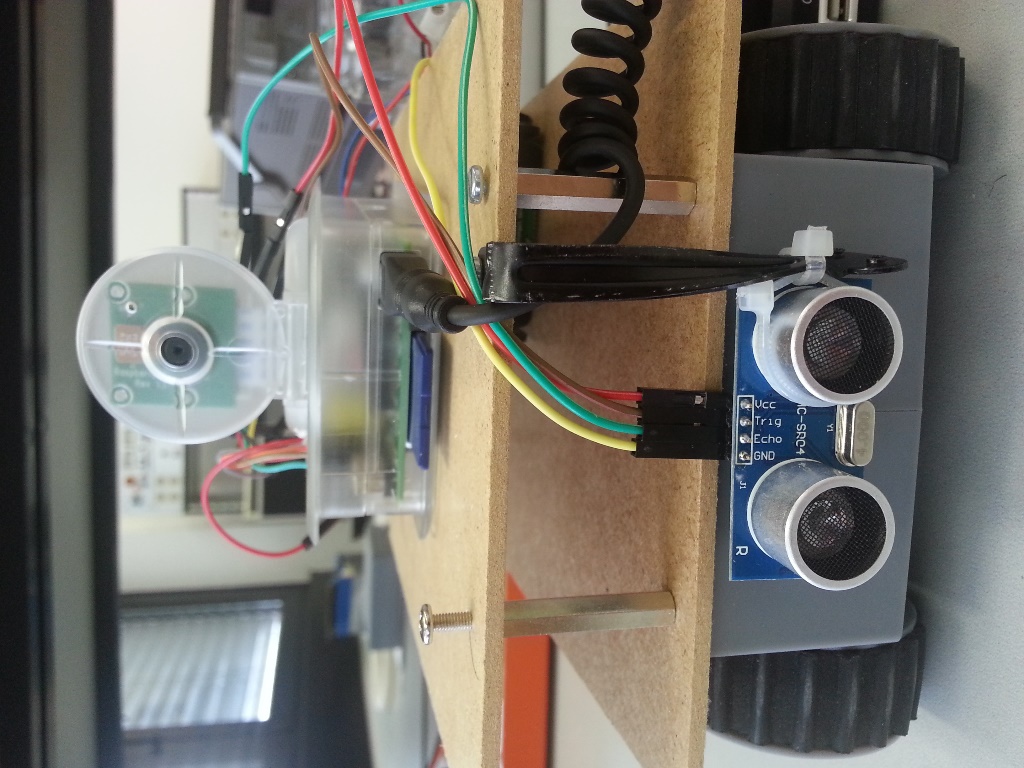


Abbildung 22: Installation des Ultraschallsensors

Zur Halterung wird hier eine Metallstange verwendet, welche passend verbogen worden ist, sodass sie auf der Oberseite der Holzplatte mit einer Schraube und Mutter Halt findet. Am Sensor selbst hingegen ist sie mit einer Kombination aus zwei Kabelbindern befestigt. Die Anschlüsse des Sensors sind nach oben ausgerichtet, sodass auch hier eine einfache Verdrahtung in Richtung des Raspberry gewährleistet ist.

# Verdrahtung

Der Raspberry verfügt über 26 GPIO-Pins, mit welchen sich zumeist Steuersignale verarbeiten und ausgeben lassen. Einige davon haben auch andere Aufgaben, die Aufteilung ist in der nächsten Abbildung gezeigt.

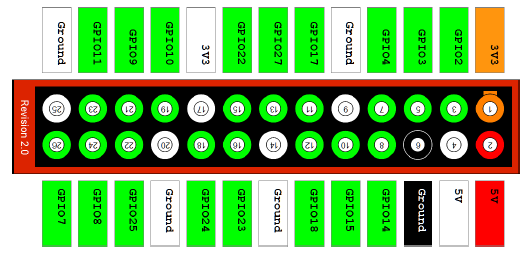


Abbildung 23: GPIO-Anschlüsse des Raspberry Pi

Die Pins 9 und 25 dienen als Ground-Verbindung. Pin 1 und 17 können eine Versorgungsspannung von 3.3V abgeben, welche zum Betreiben von Niederspannungs-Geräten verwendet werden kann. Für Geräte mit 5V Betriebsspannung werden die Pins 2 und 4 verwendet. Zur Arbeit mit dem Schrittmotor werden lediglich vier Pins zur Ansteuerung benötigt, die Pins 19, 21, 23 und 24. Diese werden dann mit den vier Input-Pins am Schrittmotor verbunden, wie das folgende Konzept zeigen soll.

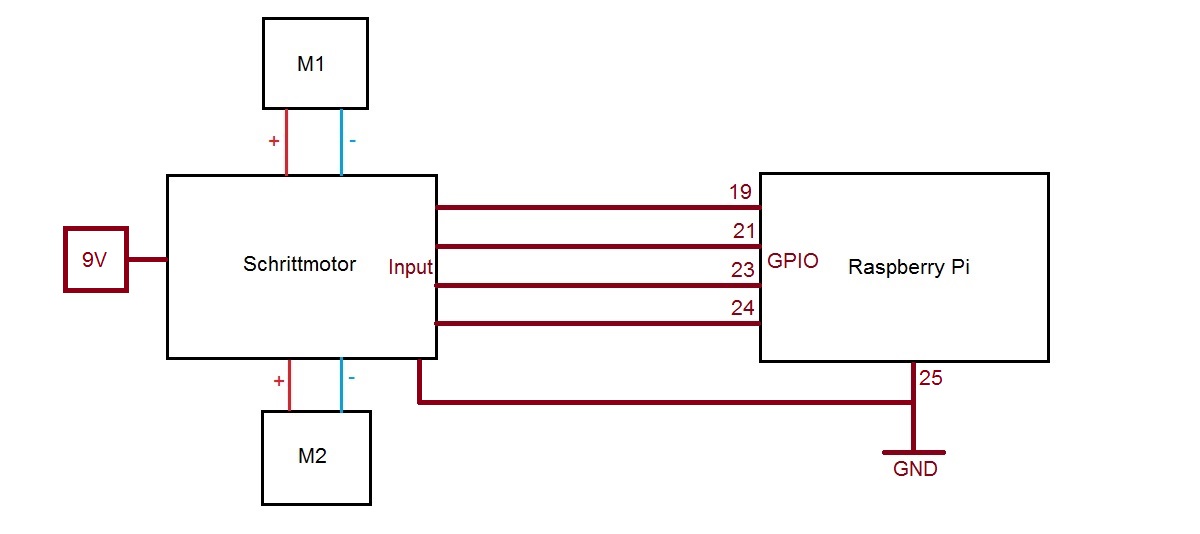


Abbildung 24: Schaubild zur Verdrahtung von Schrittmotor und Raspberry Pi

Die Leitungen für diese Verbindungen werden von der oberen Holzplatte, auf der sich der Raspberry Pi befindet, durch die Spalte auf beiden Holzstücken in das Innere des RP5 Roboters geführt. Dadurch ist gewährleistet, dass die Verdrahtungswege möglichst kurz bleiben und nicht die anderen Bauteile behindern. Der Ultraschallsensor erhält seine Versorgungsspannung Vcc von 5V über den Pin 2 des Raspberry Pi. Der Erdungsanschluss wird mit denen des Schrittmotors und Raspberries zusammengeführt. Das Triggersignal kommt von Pin 8 und das zurückkommende Echosignal wird über Pin 10 geliefert. Da das Echosignal den verbundenen Pin auf die angeschlossene Versorgungsspannung setzt, würde hier der GPIO-Pin auf 5V gesetzt werden. Diese arbeiten aber nur mit 3.3V, alles darüber führt zu einem Defekt des Raspberry. Deshalb wird in die Verbindung ein Widerstand von 1000 Ohm eingebaut, um die Spannung von 5V auf 3.3V zu reduzieren[[22]](#footnote-22). Für die Verbindung von der ebenfalls installierten Kamera kommt ein Flachbandkabel zum Tragen, welches im Inneren des Raspberry Pi – Gehäuses verläuft. Es hat genug Platz, um sich dort entsprechend den Bewegungen des drehbaren Kameraelements mitzubewegen. Durch das Gehäuse wird ebenfalls sichergestellt, dass all diese Verbindungen geschützt sind und sich nicht lockern können.

# WLAN Einrichtung

Der Raspberry Pi soll für dieses Projekt ein eigenes WLAN-Netz aufspannen, auf das sich Benutzer dann mit ihrem Gerät verbinden können sollen. Dafür wird der bereits präsentierte WLAN-Adapter verwendet, der den Raspberry Pi um eben jene Funkqualitäten erweitert. Damit dieser als Router fungieren kann, müssen einige Konfigurationsschritte durchgeführt werden. Dies beginnt bei der Installation der Router-Software, welche sich in zwei Programme aufgliedert: „HostAPD“ und DHCP. Zunächst wird letztere installiert, wobei es sich um eine Implementation des „Internet Systems Consortium“ (ISC) handelt. Als DHCP Server soll der Raspberry dafür zuständig sein, Adressen an neue Benutzer bei Eintritt ins Netz zu vergeben. Der Befehl für diese Installation ist auf nachfolgender Abbildung gezeigt.



Abbildung 25: Installieren des DHCP-Servers

Die „HostAPD“ – Software bringt die Funktionalität des Access Points mit sich. Zudem werden hier auch Sicherheitsaspekte behandelt, wie z.B. das Setzen eines Passwortes. Hierbei wird ein modifizierter Treiber für den WLAN-Adapter benötigt. Dieser kann aus GitHub geladen und als „HostAPD“ – Treiber mittels folgender Befehlsfolge initialisiert werden.

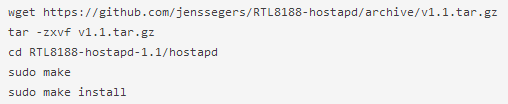


Abbildung 26: Installieren des HostAPD-Treibers

Der „wget“ – Befehl holt sich die benötigten Dateien aus dem angegebenen Pfad. Mit dem „tar“ – Kommando können diese Daten dann entpackt werden, woraufhin mit „cd“ direkt in dieses Verzeichnis gewechselt wird. „Sudo make“ und „sudo make install“ schließlich installieren die Software, was einige Momente lang dauern kann. Ist dies geschehen, muss der DHCP – Server eingerichtet werden. Hierfür wird die Datei „dhcpd.conf“ im „Nano“ – Texteditor geöffnet. Die beiden Optionen des Servers von folgender Abbildung müssen ungültig gemacht, d.h. auskommentiert werden.



Abbildung 27: Veränderungen in der DHCP-Konfiguration

Da es sich hierbei um den offiziellen DHCP – Server im lokalen Netzwerk des Raspberry Pi handelt, muss das Wort „authoritative“ gültig gemacht werden. Dadurch wird kenntlich gemacht, dass dieser DHCP – Server die Verantwortungen der Adressverwaltung im Netz übernimmt. Am Ende der Datei wird der Block von nachfolgender Abbildung eingefügt, um das Netzwerk und die dazugehörigen Adressen des DHCP – Servers zu definieren.

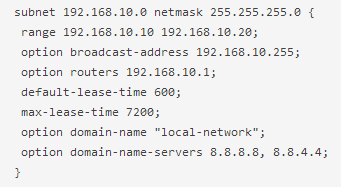


Abbildung 28: Konfiguration des DHCP-Dienstes bezüglich Netzwerkdaten

Es wird dadurch angegeben, dass das Subnetz die Adresse 192.168.10.0 mit der Subnetzmaske 255.255.255.0 erstellt wird. In diesem können Adressen von 192.168.10.10 bis .20 vergeben werden. Die Broadcast-Adresse ist .255, und die Router-Schnittstelle liegt auf IP-Adresse .1. Die Lease – Zeit beschreibt die Zeit, die ein Nutzer seine IP – Adresse behalten darf, ohne dass sie wieder freigegeben und evtl. an andere Nutzer vergeben wird. Standardmäßig liegt diese laut Konfigurierung bei 600, maximal bei 7200 Minuten. Der Name für dieses Netzwerk ist zunächst „local-network“ und als Namensserver werden die Google DNS Server benutzt, welche auf den Adressen 8.8.8.8 und 8.8.4.4 liegen.

An der Textdatei „isc-dhcp-server“ im Verzeichnis „/etc/default“ muss die Zeile „INTERFACES“ abgeändert werden, sodass sie „INTERFACES=“wlan0““ angibt. Dadurch wird der DHCP-Server dazu veranlasst, seine Netzwerkadressen auf dem WLAN-Interface auszugeben. Der letzte Schritt ist nun das Vergeben einer festen IP-Adresse für den WLAN-Adapter. Dafür muss zunächst das WLAN-Interface heruntergefahren, da nur so Änderungen daran vorgenommen werden können. Dies geschieht mittels des Befehls „sudo ifdown wlan0“. Zur Bearbeitung der Adresse wird die Datei „interfaces“ im Pfad „/etc/network/“ im Texteditor geöffnet. Diese muss wie in folgender Abbildung gezeigt abgeändert werden.

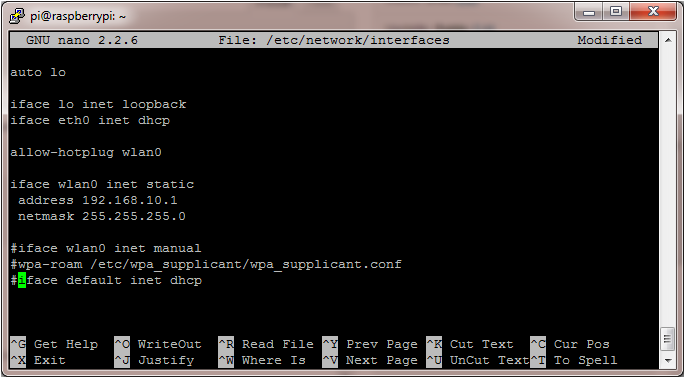


Abbildung 29: Ändern der IP-Adresse des WLAN-Adapters

Mit diesen Einstellungen verwendet der WLAN-Adapter in Zukunft immer die feste IP-Adresse 192.168.10.1. Diese feste Adresse ist wichtig, da so gewährleistet ist, immer mittels der gleichen Adresse über Programme wie Putty auf den Raspberry Pi gelangen zu können, um dort zu arbeiten. Für die Verbindung zum aufgebauten Netzwerk muss noch der Access Point konfiguriert werden. Dies erfolgt in der Datei „hostapd.conf“ im Systempfad „/etc/hostapd/“. Hier kann einerseits die SSID als Anzeigename für das Netzwerk geändert werden, als auch andererseits das wichtige Passwort, welches beim Verbinden zum Netz eingegeben werden muss. Somit ist der WLAN-Adapter korrekt konfiguriert und muss noch gestartet werden. Dies geschieht über die beiden Befehle aus der nachfolgenden Abbildung.



Abbildung 30: Starten des Netzwerkdienstes

Da sich das Netzwerk bei jedem Starten des Raspberries von selbst aufbauen soll, ohne dass es nach einem Login seitens des Benutzers erst gestartet werden muss, sollten diese beiden Befehle in die Autostart-Routine kopiert werden. Für den automatischen Start von Diensten nach Systemstart ist die Datei „rc.d“ zuständig. Diese wird mit dem Dienstnamen und einem „enable“-Flag aktualisiert, wie aus der nächsten Abbildung ersichtlich.



Abbildung 31: Aufnahme des Netzwerkdienstes in die Autostart-Routine

Damit ist die Einrichtung der WLAN-Funktionalität des Roboters beendet.

# 1. Versuch Web-Service für Kamera

Zum Streamen des Kamerabildes der Raspberry Kamera auf eine Benutzeroberfläche gibt es verschiedene Möglichkeiten. Eine simpel umzusetzende Möglichkeit dafür ist ein Web-Service. Als erster Versuch wird hier ein Projekt genutzt, für das bereits Vorarbeit geleistet worden ist, welche sich auf dem Sharing-Portal „GitHub“ befindet. Die in folgender Abbildung gezeigten Operationen laden die Dateien aus „GitHub“, wechseln in das Verzeichnis, geben dem Benutzer die Ausführungsrechte und installieren diesen Web-Service auf dem Raspberry Pi.

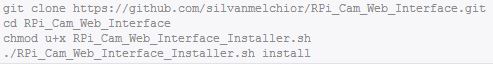


Abbildung 32: Installieren des Web-Services aus GitHub

Nach dieser Installation ist der Web-Service auf dem Raspberry Pi betriebsbereit. Sofern sich der Benutzer im Netzwerk des Raspberries befindet, kann er mittels einer einfachen Eingabe der IP-Adresse im Netzwerkbrowser den Web-Service starten. Mit der Adresse „192.168.10.1“ gelangt er so auf eine HTML-Seite, die in der nachfolgenden Abbildung dargestellt ist.

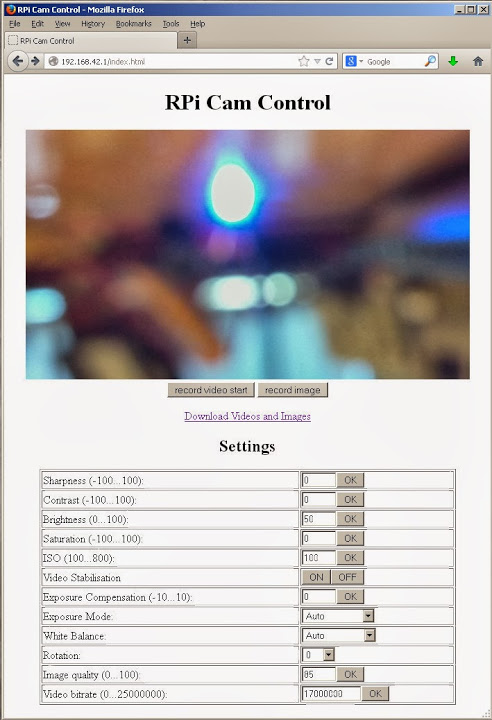


Abbildung 33: Benutzeroberfläche des Web-Services im Internetbrowser

Der Inhalt dieser Seite kommt dabei aus der Datei „index.html“ aus dem Web-Interface Ordner. Diese zeigt im oberen Teil das Kamerabild als Livestream an. Mittels der Knöpfe darunter lassen sich verschiedene Aktionen, wie z.B. Videoaufnahme oder Foto schießen ausführen. Diese werden dann auf der SD-Karte des Raspberries im Web-Service-Ordner abgelegt. Im Bereich Settings können zusätzliche Einstellungen für die Kamera vorgenommen werden, wie Kontrast und Helligkeit, aber auch tiefergehende wie Weißabgleich und ISO-Werte.

Zur technischen Realisierung werden die Fahranweisungen per Ajax an ein PHP Script gesendet. Dieser Code ist in Abbildung XXX zu sehen.

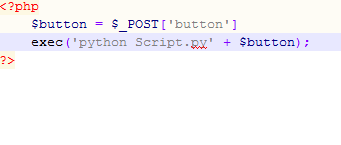


Abbildung 34: Python Script in PHP ausführen

Mit „exec“ kann das PHP Script einen Shell-Befehl ausführen. Das wird genutzt, um das Python Script für die Ausführung der Fahranweisungen zu starten. Die Fahrtrichtung erhält es als POST Variable. In JavaScript werden die Anweisungen wie in Abbildung XXX zu sehen gesendet.

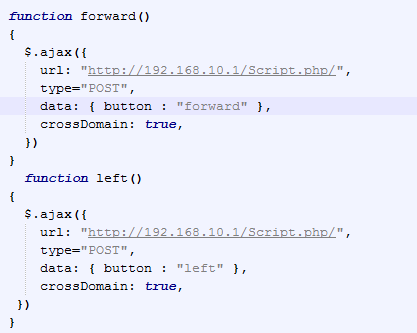


Abbildung 35: Beispiele für das Senden von Anweisungen

In diesem Versuch werden per AJAX die Variable „button“ mit der jeweiligen Richtung an das PHP Script per POST Methode vermittelt.

Aufgrund der Tatsache, dass die Kameraanwendung ein Komplettpaket ist, kommt diese mit einem eigenen Webserver der auf Port 8080 läuft. Da unser PHP Script auf einem anderen Port läuft kommt es aufgrund der Same-Origin-Policy (SOP) zu Problemen.

Für diesen Fall gibt es Cross-Origin Ressource Sharing (kurz CORS), dies ermöglicht es, die Same-Origin-Policy der Webbrowser zu ignorieren. Für diesen Mechanismus gibt es bei Ajax das Attribut „crossDomain“. Auf der anderen Seite muss der Webserver dem Client am anderen Port erlauben, die SOP zu brechen. Für alle anderen Webserver und Ports gilt die SOP noch.

Mangels Wissen, welcher Webserver die Kameraanwendung nutzt, kann der Eintrag für CORS nicht erledigt werden. Außerdem ist die Ausführung der Befehle mit einer hohen Latenz verbunden und dadurch nicht für die Steuerung geeignet.

# JavaScript & Ajax

JavaScript ist eine Clientseitige Skriptsprache. Sie wird meist zur Optimierung von Websites benutzt. Sie kann HTML und CSS dynamisch beeinflussen und ermöglicht dadurch das neu Laden von einzelnen Bereichen statt der kompletten Seite. JavaScript Code wird am Anfang der HTML Seite im Header definiert. Sie kann entweder wie in XXX im Header als In-Document[[23]](#footnote-23) oder extern eingebunden werden. Letzteres wird zur Nutzung von Bibliotheken und Plugins benutzt wie z.B. JQuery.



Abbildung 36 : Einbinden von JavaScript

JavaScript wird häufig neben dynamischen Laden von Inhalten dazu genutzt, um Formulardaten Clientseitig zu prüfen und den Server vor unnötigen Anfragen zu entlasten. Das wichtigste Einsatzgebiet ist DOM (Document Object Model). Dabei beeinflusst es HTML Elemente auf der Seite und kann die dazugehörige CSS oder Inhalt komplett verändern. Auch kann es den Inhalt eines Eingabefelds ermitteln und ihn an den Server zur weiteren Verarbeitung schicken, ohne dass die Seite neu laden muss.

Dies wird dann Ajax genannt. Ajax ist ein Apronym für AsynchronousJavaScriptandXML. Ajax ist ein Konzept zur Nutzung diverser Webtechnologien um eine asynchrone Datenübertragung zu ermöglichen.

In diesem Projekt werden Ajax Aufrufe mithilfe der Bibliothek JQuery genutzt. Dieser vereinfacht die Ajax Logik und bietet Funktionen an die auf jedem Browser funktionieren.

In Abbildung XYX ist ein einfacher Code Abschnitt zur Demonstration von Ajax. In diesem Beispiel wird PHP als serverseitige Skriptsprache genutzt um die Aufrufe zu verarbeiten. Es wird per Post Methode, der Skript „Script.php“ aufgerufen die in diesem Fall nichts zurückliefert. Der Aufruf sendet die zuletzt gedrückte Taste an das Skript, der diese Taste interpretiert und letztendlich den Roboter fahren lässt. Näheres zur Serverseite ist im Kapitel Versuch 1.

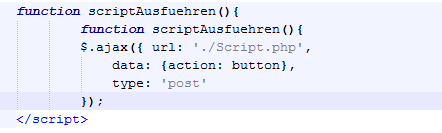


Abbildung 37 : Aufruf der Steuerung per Ajax

Ajax wird in unserem Projekt genutzt um die Steuerung per Weboberfläche zu ermöglichen. Dabei ist es essentiell, dass die Seite nicht neugeladen werden muss. Ansonsten hat dies hohe Lasten an den Raspberry zur Folge, der bei jedem Tastendruck die Seite senden und verarbeiten muss. Clientseitig ist das auch nicht sehr sinnvoll, da die Videoübertragung der Kamera bei jeder Steuerungsmaßnahme unterbrochen wird und somit das Nutzerlebnis beeinträchtigt.

Der letzte Aspekt in Verbindung mit JavaScript ist die Same Origin Policy (kurz SOP). Diese untersagt, dass JavaScript oder CSS[[24]](#footnote-24) auf Objekte zugreifen, die von einer anderen Webseite stammen. Eine Webseite ist schon „fremd“ falls diese auf einem anderen Port läuft als die Webseite von der aus der Zugriff erfolgen soll. Webseiten auf dem gleichen Webserver und Port sind davon nicht betroffen.

# WebSockets

WebSockets sind bidirektionale Verbindungen die einem Webserver und einer Webanwendung ermöglicht eine offene Verbindung herzustellen. WebSockets basieren auf TCP, im Gegensatz dazu bleibt die Verbindung nach dem ersten Mal offen. Die beiden Teilnehmer können nun die Verbindung nutzen ohne auf den anderen Teilnehmer zu warten, diese Technik ermöglicht Real-Time Anwendungen[[25]](#footnote-25).

Außerdem werden nach dem ersten Handshake keine Header Dateien mehr weitergesendet um unnötigen Overhead zu vermeiden. In diesem Projekt wird der Webserver Tornado genutzt. Dieser ermöglicht Verbindungen per WebSockets.

In Abbildung XXX sieht man die grundlegende Funktion und wie sie aufgebaut ist.

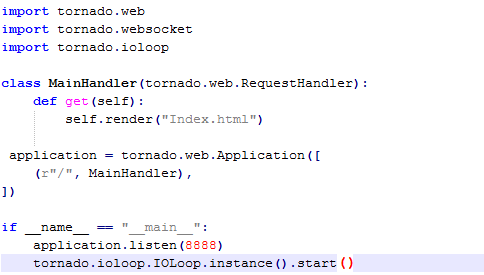


Abbildung 38: Websocket mit Tornado

Die grundlegenden Tools für Webanwendungen und WebSockets können mit import verwendet werden. Ein Handler für die Funktionen muss definiert werden und die Aktionen die es durchführen soll. In diesem Beispiel wird die „Index.html“ angezeigt. Dieser Handler muss nun registriert werden und beschrieben werden wann sie ausgelöst werden soll. In diesem Beispiel wird der „MainHandler“ aufgerufen, wenn nur die IP-Adresse und der Port, der im nächsten Schritt mit 8888 definiert wird, adressiert werden. Dies würde für unseren Raspberry Pi von einem anderen Client aus so aussehen:

Adresse Websockets

Abbildung 39 : Adresse um „Index.html“ zu sehen

Die letzte Zeile dient nur zum dauerhaften Start des Webservers.

In Abbildung XXX ist die Clientseitige Ansteuerung des WebSockets. Im Gegensatz zum vorherigen Beispiel wird hier ein anderer Handler angesprochen, uns zwar einer der auf „/websocket“ reagiert. Anschließend wird ihm mit dem sende Befehl der String „left“ geschickt. Für die Steuerung des Roboters nach links. Die genaue Steuerung des Roboters wird im nächsten Abschnitt Basics Steuerung eklärt

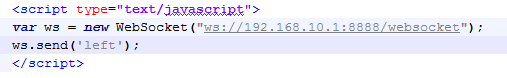


Abbildung 40 : Clientseitige Öffnung eines Websockets

# Steuerung Basics

Die Steuerung des Raspberry Pi soll über eine Weboberfläche ermöglicht werden die auch als Kameraausgabe dient. Der Roboter soll per Buttons oder auch per Tastatur aus gesteuert werden können. Dazu werden zunächst wie in Abbildung XXX zu sehen Buttons erstellt.

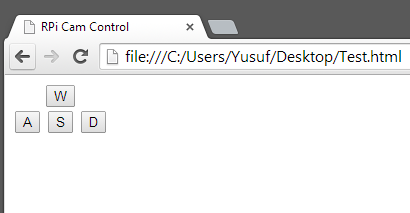


Abbildung 41 : Buttons für die Steueurng

Anschließend sind in JavaScript die gleichen Tasten definiert um die Fahranweisung an den Webserver bzw. Raspberry Pi weiterzusenden. Dies ist in Abbildung XXX zu sehen. Hier ist ein Ausschnitt zu sehen, bei dem die Tasten W und A dargestellt werden. Im Gegensatz zum normalen „event.keyCode“ wird hier „event.which“ von JQuery eingesetzt um bei jedem Browser den gleichen Code für die Tasten zu erhalten. Für die Taste W ist dies „87“ und für A die „65“. Dies nutzen wir bei dem Event „keydown“, welches bei einem Tastendruck ausgelöst wird. Mit dem switch case wird abgefragt ob einer dieser Tasten die von uns genutzten Tasten („W, A, S, D“) sind. Falls dies der Fall ist, wird überprüft ob die letzte gedrückte Taste die dritte aktive Taste ist. Dazu wird in dem Array „arrButton“ gezählt wie viele Felder schon auf true, also aktiv gedrückt, sind. Ist es mehr als eins, können keine weiteren Tasten mehr aktiviert werden, d.h. das drücken der Taste bleibt wirkungslos. Diese Maßnahme wird benötigt um nicht gleichzeitig nach vorne, links und rechts Fahren auszulösen, welches physikalisch unmöglich zu bewerkstelligen ist. Es wird außerdem überprüft ob die Taste für rückwärtsfahren aktiv ist, da es nicht möglich sein soll schräg rückwärtsfahren zu können. Falls diese Bedingungen erfüllt sind, werden im Array „arrButton“, dem jeweiligen Feld der Taste auf „true“ gesetzt.

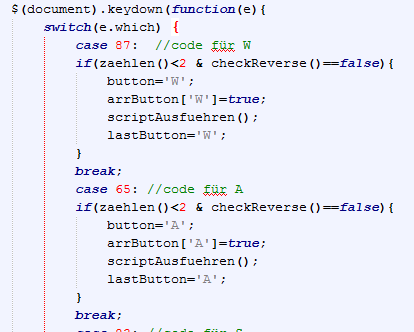


Abbildung 42 : Zuweisung der Tasten W und A

Eine ähnliche Funktion gibt es beim Loslassen einer Taste, diese heißt „keyup“ und unterscheidet sich nur davon dass die letzte losgelassene Taste aus dem Array „arrButton“ gelöscht wird. „scriptAusfuehren“ wird anschließend auch in diesem Fall ausgeführt.

Nach dem ausführen des Fahrvorgangs wird die letzte Taste in einer Variablen abgespeichert. Das wird benötigt um die Fahraktion nicht mehrmals auszulösen. Beim gedrückt halten einer Taste wird diese mehrmals ausgelöst, so würde die Anweisung mit diesem Code immer wieder ausgelöst werden, mehrere Dutzendmal pro Sekunde.

Aus diesem Grund wird die letzte Taste vor dem Senden der Aktion für das Fahren überprüft, um unnötige Aktionen zu vermeiden. Diese Abfrage zusammen mit dem differenzieren der Fahrtrichtung ist in Abbildung XXX zu sehen.

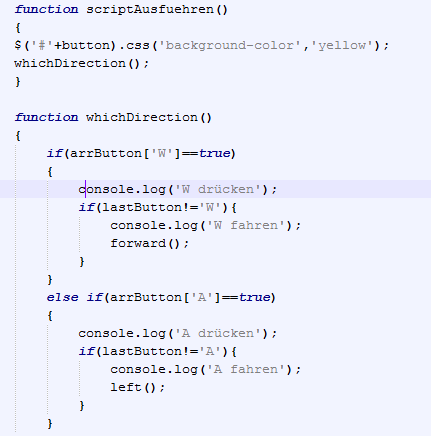


Abbildung 43 : Entscheidung der Fahrtrichtung

Nach dem drücken einer Taste und dem speichern welche Tasten gedrückt sind, wird der Button der diese Taste repräsentiert gelb markiert, um anzuzeigen welche Tasten gerade aktiv sind. „whichDirection“ ermittelt in welche Richtung der Raspberry Pi fahren soll. Wenn keine aktive Taste vorhanden ist, wird der Befehl zum Halten gesendet. In diesem Code sind zunächst nur die einfachen Fahrtrichtungen möglich. Schräg Fahren muss lediglich als zusätzliche Abfrage nach der Taste „W“ hinzugefügt werden, indem abgefragt wird ob die Tasten „A“ oder „D“ gedrückt sind.

Um verschiedene Methoden zum Senden der nächsten Aktionen zu ermöglichen wurden hier für jede Fahrtrichtung eine Methode deklariert um sie austauschen zu können. Die genauen Sendemethoden werden in den verschiedenen Versuchen beschrieben. Alle Möglichkeiten haben jedoch gemeinsam, dass sie zuletzt den Python Script zum Fahren ansprechen müssen.

# Versuch 1

# Versuch 2

Beim zweiten Versuch konzentrierte sich das Augenmerk auf die Kamera. Das Ziel war es die Kamera so zu streamen, dass diese in eine Webseite eingebunden werden kann. Statt einer Webseite für die Kameraanwendung, soll eine Webseite für die Steuerung erstellt werden, um den Kamerastream einzubinden. So wird SOP vermieden und eine Kompatibilität zu allen Browsern gesichert.

Dieser Versuch besteht im Grundsatz daraus, dass die Raspberry Kamera offizielle Funktion zur Videoaufnahme als Stream im Netz gesendet wird. Dazu ist es wichtig das Programm „ffmpeg“ zu installieren. Dieses Programm ermöglicht es Video- und Audioaufnahmen zu konvertieren. Außerdem kann es, wie in diesem Fall in Echtzeit aufnehmen. In Abbildung XXX ist zu sehen wie diese Funktion umgesetzt wird.

C:\Users\Yusuf\AppData\Local\Microsoft\Windows\INetCache\Content.Word\raspivid zu ffmpeg.png

Abbildung 44 : Aufnahme des Videos

Mit Raspivid wird ein Video in dem Format 960x540 und 25 Frames pro Sekunde. Dieses Video wird von ffmpeg aufgenommen und mit dem gleichen Format zu einem Flash Video konvertiert.

Um das Videomaterial zu streamen, bedienen wir uns dem RTMP[[26]](#footnote-26) Server „crtmpserver“. Dieser überträgt die von „ffmpeg“ verarbeitenden Videoaufnahmen im Netzwerk.

Auf der Client Seite wird ein Player benötigt, der diesen Flash Video Stream anzeigen kann. Für diesen Zweck wird der auf JavaScript basierende Player „JW Player“ genutzt. Dieser ist mit allen gängigen Browsern kompatibel und muss nur wie in Abbildung XXX zu sehen in der Webseite eingebunden werden.

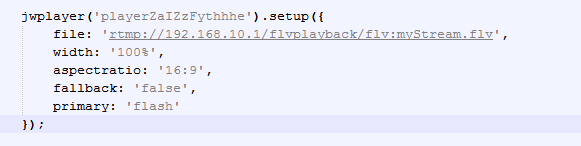


Abbildung 45 : JavaScript Player „JW Player“

Diese Variante des Videostreamings hat den Nachteil, dass die Verzögerungen vom Video bis zu 5 Sekunden lang sein können. Dies ist für die gewollte Anwendung nicht zweckmäßig und kann nicht eingesetzt werden.

Als Sendemethode für die Steuerungsmaßnahme ist in diesem Versuch die Wahl auf das Python Framework „Flask“ gekommen. Es werden per Ajax die einzelnen Webseiten aufgerufen, die für die einzelnen Richtungen zuständig sind. D.h. wenn der Raspberry Pi nach rechts fahren sollte wurde beim Auslösen der Taste „D“ per Ajax ein Seitenaufruf gesendet. Dieser Seitenaufruf war auf die Seite „right“ gerichtet die „Flask“ interpretiert hat und die Fahranweisung ausgeführt hat.

Diese Sendemethode hatte die Probleme, dass die Anweisungen sehr spät ausgelöst worden sind und ein Steuerung in Echtzeit nicht möglich war.

# Versuch 3

In diesem Versuch wird per „MJPEG Streamer“ ein MJPEG[[27]](#footnote-27) Videocodec gestreamt. Für diesen Zweck wird ein Fork speziell für den Raspberry, der die Kamera Funktionen enthält genutzt. Dies ermöglicht es wie in Abbildung XX zu sehen eine einfache Art die Videos zu streamen.

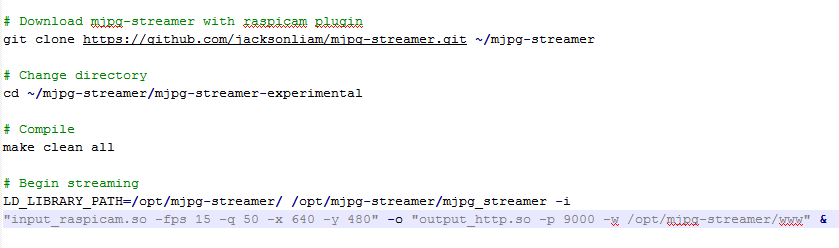


Abbildung 46 : Installieren und Starten von MJPEG Streamer

Nach dem Start des Streams muss dieser nur noch in die Webseite eingebunden werden. Dabei werden die einzelnen Bilder in die Webseite eingebunden und per JavaScript mit den nächsten ersetzt. So entsteht eine Videoaufnahme die in Echtzeit angezeigt wird. Der einzige Nachteil dieser Variante ist eine Verzögerung die selten auftritt.

In diesem Versuch ist der Stream in eine Webseite eingebunden, bei der die Steuerungsanweisungen per WebSockets gesendet werden.

# Fazit und Ausblick

# Appendix

# Glossar

# Literaturverzeichnis

A. Spillner, T. L. (2005). *Basiswissen Softwaretest.*

A. Spillner, T. R. (2008). *Praxiswissen Softwaretest - Testmanagement: Aus- und Weiterbildung zum Certified Tester - Advanced Level nach ISTQB-Standard.* Dpunkt Verlag.

Department, K. (kein Datum). ISTQB Schulung. In ISTQB.

DICKIE-TAMIYA Modellbau GmbH & Co. KG. (16. April 2014). *Tamiya*. Abgerufen am 26. Mai 2014 von Tamiya: http://www.tamiya.de/de/produkte/rcmodelltrucks/zugmaschinen114/produktdetails.htm?sArtNr=300056338

Hewlett-Packard Development Company, L. (5. 21 2013). *hp.com*. Abgerufen am 5. 21 2013 von United States HP.com: http://www8.hp.com/de/de/software-solutions/software.html?compURI=1172957#.US32Vpjtkrp

HP. (2009). Using Quicktest Professional 10 Student Guide. In HP.

ISTQB. (2010). ISTQB / GTB Glossar der Fehlerbegriffe.

Kreditservice, S. H. (2010). Leitfaden Testautomatisierung. In M. Lehmann.

o.V. (7. 6 2010). *Heise.de*. Abgerufen am 3. März 2013 von http://www.heise.de/developer/meldung/Mangelhafte-Testprozesse-Hauptgrund-fuer-Softwarefehler-1016334.html

Sebastian Hammer, M. D. (2010). Kosten-Nutzen-Analyse und Die Nutzwertanalyse. In M. D. Sebastian Hammer. Grin.

Tinkerforge GmbH. (kein Datum). *Bricklets / Distance US Bricklets*. Abgerufen am 16. April 2014 von Tinkerforge: http://www.tinkerforge.com/de/doc/Hardware/Bricklets/Distance\_US.html#distance-us-bricklet

Tinkerforge GmbH. (kein Datum). *Bricks / Master Brick*. Abgerufen am 16. April 2014 von Tinkerforge: http://www.tinkerforge.com/de/doc/Hardware/Bricks/Master\_Brick.html#master-brick

Tinkerforge GmbH. (kein Datum). *Bricks / Servo Brick*. Abgerufen am 16. April 2014 von Tinkerforge: http://www.tinkerforge.com/de/doc/Hardware/Bricks/servo\_Brick.html#servo-brick

Tinkerforge GmbH. (kein Datum). *Master Extensions / WIFI Extension*. Abgerufen am 16. April 2014 von Tinkerforge: http://www.tinkerforge.com/de/doc/Hardware/Master\_Extensions/WIFI\_Extension.html#wifi-extension

Tinkerforge GmbH. (kein Datum). *Software / Brick Daemon*. Abgerufen am 16. April 2014 von Tinkerforge: http://www.tinkerforge.com/de/doc/Software/Brickd.html#brickd

Tinkerforge GmbH. (kein Datum). *Software / Brick Viewer*. Abgerufen am 16. April 2014 von Tinkerforge: www.tinkerforge.com/de/doc/Software/Brickv.html#brickv

Tinkerforge GmbH. (kein Datum). *Tutorials und FAQ / Erste Schritte - Tutorial*. Abgerufen am 16. April 2014 von Tinkerforge: http://www.tinkerforge.com/de/doc/Tutorials/Tutorial\_Extending/Tutorial.html#tutorial-first-steps

Tinkerforge GmbH. (kein Datum). *Was ist Tinkerforge*. Abgerufen am 2014. Mai 27 von Tinkerforge: http://www.tinkerforge.com/de/home/what\_is\_tinkerforge/

1. (Tinkerforge GmbH (a)) [↑](#footnote-ref-1)
2. (Tinkerforge GmbH (a)) [↑](#footnote-ref-2)
3. (Tinkerforge GmbH (a)) [↑](#footnote-ref-3)
4. (Tinkerforge GmbH (i)) [↑](#footnote-ref-4)
5. Vgl. (Tinkerforge GmbH (a)) [↑](#footnote-ref-5)
6. (Tinkerforge GmbH (a)) [↑](#footnote-ref-6)
7. (Tinkerforge GmbH (a)) [↑](#footnote-ref-7)
8. Vgl. (Tinkerforge GmbH (a)) [↑](#footnote-ref-8)
9. Vgl. (Tinkerforge GmbH (e)) [↑](#footnote-ref-9)
10. Vgl. (Tinkerforge GmbH (d)) [↑](#footnote-ref-10)
11. (Tinkerforge GmbH (f)) [↑](#footnote-ref-11)
12. (Tinkerforge GmbH (g)) [↑](#footnote-ref-12)
13. Vgl. (Tinkerforge GmbH (g)) [↑](#footnote-ref-13)
14. Vgl. (DICKIE-TAMIYA Modellbau GmbH & Co. KG, 2014) [↑](#footnote-ref-14)
15. (DICKIE-TAMIYA Modellbau GmbH & Co. KG, 2014) [↑](#footnote-ref-15)
16. (Tinkerforge GmbH (a)) [↑](#footnote-ref-16)
17. (Tinkerforge GmbH (a)) [↑](#footnote-ref-17)
18. Vgl. (Tinkerforge GmbH (h)) [↑](#footnote-ref-18)
19. http://www.tinkerforge.com/de/doc/Software/API\_Bindings.html#api-bindings [↑](#footnote-ref-19)
20. http://www.tinkerforge.com/de/doc/Software/Bricklets/DistanceUS\_Bricklet\_Java.html [↑](#footnote-ref-20)
21. http://www.tinkerforge.com/de/doc/Software/IPConnection\_Java.html [↑](#footnote-ref-21)
22. S. bytecreation seite [↑](#footnote-ref-22)
23. Das Schreiben von JavaScript oder CSS in der gleichen Datei die es auch beeinflussen soll [↑](#footnote-ref-23)
24. Cascading Style Sheets ist eine deklarative Sprache für den Design einer Seite [↑](#footnote-ref-24)
25. Real-Time Anwendungen sind äußerst fokussiert auf schnellen Datenaustausch in Echtzeit [↑](#footnote-ref-25)
26. Real Time Messaging Protocol ist ein proprietäres Protokoll von Adobe System um Audio und Video über das Netzwerk zu übertragen [↑](#footnote-ref-26)
27. Motion JPEG ist ein Videocodec bei dem die Einzelbilder JPEG Bilder sind [↑](#footnote-ref-27)