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

# Vorwort

# Inhaltsverzeichnis

[Erklärung ii](#_Toc388976318)

[Aufgabenstellung iii](#_Toc388976319)

[Vorwort iv](#_Toc388976320)

[Inhaltsverzeichnis v](#_Toc388976321)

[Abkürzungsverzeichnis vi](#_Toc388976322)

[Abbildungsverzeichnis vii](#_Toc388976323)

[1 Grundlagen 1](#_Toc388976324)

[1.1 Raspberry Pi 1](#_Toc388976325)

[1.2 Unix-Derivate 2](#_Toc388976326)

[1.3 Tools 4](#_Toc388976327)

[1.4 Netztechniken 7](#_Toc388976328)

[2 Problemstellung 9](#_Toc388976329)

[3 Zielsetzung 10](#_Toc388976330)

[4 Realisierung 11](#_Toc388976331)

[4.1 Ansteuerung der Servo-Motoren 11](#_Toc388976332)

[4.2 Abstandsmessung mittels Ultraschall 13](#_Toc388976333)

[4.3 Web Server 14](#_Toc388976334)

[4.4 Backupmanagement 17](#_Toc388976335)

[4.5 Ausblick 18](#_Toc388976336)

[5 Fazit 19](#_Toc388976337)

[Quellenverzeichnis 20](#_Toc388976338)

[Literaturverzeichnis 21](#_Toc388976339)

[Anhang 22](#_Toc388976340)

# Abkürzungsverzeichnis

AMD64: 64Bit Mikroprozessor-Architektur

GUI: Graphical User Interface

RPi: Raspberry Pi

SSH: Secure Shell

UNIX: Mehrbenutzer-Betriebssystem

x86: 32Bit Mikroprozessor-Architektur

# Abbildungsverzeichnis

[Abbildung 1: Grobstruktur des Linux-Kernels 3](#_Toc388975846)

[Abbildung 2: Vim-7-Funktion: Rechtschreibprüfung 6](#_Toc388975847)

[Abbildung 3: RPi GPIO 11](#_Toc388975848)

[Abbildung 4: Win32DiskImager 17](#_Toc388975849)

# Grundlagen

## Raspberry Pi

### RPi

Der RPi ist ein kostengünstiger, kreditkartengroßer Einplatinencomputer der britischen Raspberry Pi Fundation, der Menschen in allen Altersklassen ermöglichen soll grundlegende Computer- und Programmierkenntnisse mit Sprachen wie Scratch und Python zu erlernen.

Dafür bietet der RPi USB-Anschlüsse für Tastatur und Maus, eine Netzwerkschnittstelle, Audioklinke, sowie analogen Videoausgang und HDMI. Das Betriebssystem wird direkt von einer SD-Karte im integrierten Kartenleser gebootet und benötigt keine herkömmliche Installation. Es muss lediglich einmal das gewünschte System auf die SD-Karte kopiert werden.

Das Ein-Chip-System verwendet einen Broadcom mit einem 700-[MHz](http://de.wikipedia.org/wiki/Megahertz)-ARM11-Prozessor und je nach Modell 256 oder 512 MB Arbeitsspeicher. Damit ist es auch möglich den RPi als Desktopcomputer zu nutzen, im Internet zu browsen, Musik und Filme abzuspielen, Officearbeiten und kleine Spiele zu spielen.

Durch den geringen Preis und seine Vielfältigkeit hat der RPi längst seinen Platz in der „DIY-Szene“ gefunden. Es gibt unzählige Erweiterungen und Bastelanleitungen rund um den RPi. Dazu gehören z.B. das Kamera-Modul und die Möglichkeit Servo-Motoren über die GPIO-Pins des RPi anzusteuern.

Inzwischen hat sich der RPi über 2,5 Millionen mal verkauft[[1]](#footnote-2) und es befindet sich einige Konkurrenzprodukte mit potenterer Hardware anderer Hersteller auf dem Markt.

### ARM-Architektur

Die ARM-Architektur wurde 1983 vom britischen Unternehmen Acorn entwickelt und seit 1990 von ARM Limited weitergeführt. Inzwischen gibt es die 32-Bit-Chips in der 7. Version und bieten neben einer Multiprozessorstruktur eine Leistung bis zu 2Ghz. Für 2014 sind ARMv8 Chips angekündigt mit einer 64-Bit-Architektur und einer Leistung bis 3Ghz. Aufgrund des geringen Energiebedarfs findet ARM besonders im Embedded Bereich große Anwendungsmöglichkeiten und ist heute bereits in vielen mobilen Geräten namhafter Hersteller integriert. Die ARM-CPU ist eine RISC-Architektur und verfügt über einen 3-Register-Befehlssatz.

Auf dem RPi kommt ein 32-Bit Mikroprozessor der ARM11-Familie mit ARMv6-Architekur zum Einsatz. Der Prozessor erschien erstmals im Jahr 2002 un bietet bis zu 700Mhz Leistung. Bei Bedarf kann dieser aber bis zu 1Ghz ohne große Risiken übertaktet werden. Damit bietet er eine perfekte Balance für den RPi zwischen Stromverbrauch und möglicher Rechenleistung.

## Unix-Derivate

### Linux

Linux ist ein Kern (engl. kernel) eines Betriebssystems welcher 1991 von dem Finnen Linus Torvalds ursprünglich für die x86-Architetktur entwickelt wurde. Linux als Begriff setzt sich aus dem Namen Linus und dem x aus Unix zusammen. Der Kern wurde unter der freien GNU General Public License (GPL) veröffentlicht und findet deshalb mit großer Beliebtheit in vielen Betriebssystemen und Distributionen Anwendung. Der modulare, monolithische Betriebssystemkern ist zuständig für Prozessverwaltung, Multitasking, Speicherverwaltung, Lastverteilung, Sicherheit und E/A-Operationen auf verschiedenen Geräten und ist fast ausschließlich in C geschrieben.

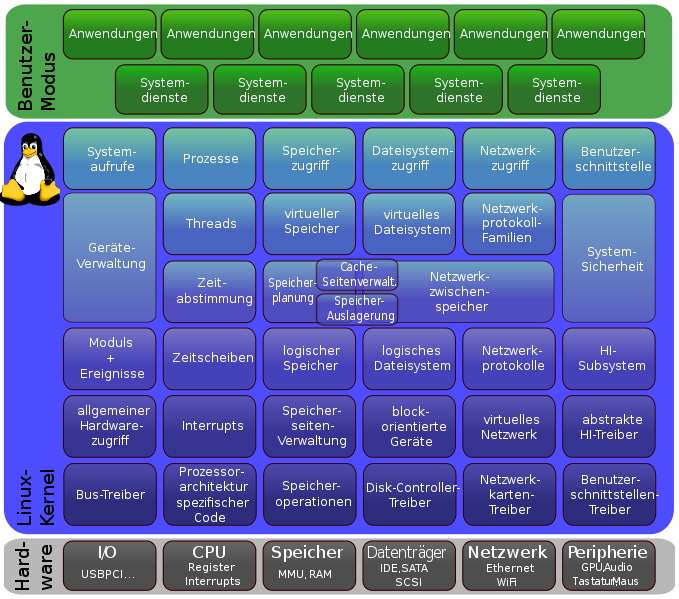


Abbildung 1: Grobstruktur des Linux-Kernels

### Raspbian “Wheezy“

Raspbian ist ein freies Betriebssystem auf Basis von Debian Linux optimiert für die RPi Hardware. Ein Betriebssystem ist eine Menge von grundlegenden Programmen um den RPi laufen zu lassen. Darüber hinaus bietet Raspbian mehr als das pure BS: es kommt mit über 35.000 vorkompilierten Paketen, die sich einfach installieren und sofort einsetzen lassen. Der erste Build von Juni 2012 wurde für den RPI angepasst und optimiert um die beste Leistung der begrenzten Hardware herauszuholen. Raspbian befindet sich noch weiterhin in Entwicklung und wird stetig erweitert.

Die aktuelle Version von Januar 2014 basiert auf Debian 7.3 und trägt den Beinamen „Wheezy“

## Tools

### Python

Als universelle und dynamische Sprache gehört Python zu den höheren Programmiersprachen. Aufgrund der guten Lesbarkeit und Dynamik wird Python meist als Skriptsprache genutzt. Größere Softwareprojekte sind aber ebenso möglich. Entwickelt wurde die Sprache Anfang der 90er Jahre von Guido van Rossum in Amsterdam mit dem Ziel mit möglichst einfacher Syntax schnell einsetzbar und übersichtlich zu sein. Im Gegensatz zu vielen anderen höheren Sprachen (wie C oder C++) gibt es keine Typprüfung. Dateitypen werden dynamisch verteilt und freier Speicher vom „Garbage Collector“ selbstständig bereinigt.

Das Strukturieren von Blöcken wird mit Einrückungen realisiert und ist zwingend notwendig. Das Einklammern des Blockes (z.B. mit {}) ist nicht möglich.

Als einfaches Beispiel eine Funktion zur Berechnung der Fakultät.

Beispiel[[2]](#footnote-3):

**def** fakultaet**(**x**):**

**if** x **>** 1**:**

**return** x **\*** fakultaet**(**x **-** 1**)**

**else:**

**return** 1

Python wird in vielen großen Firmen eingesetzt, wie z.B. Facebook, Google oder EveOnline (CCP Games).

"Python has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language."[[3]](#footnote-4)

Mit Cython enstand 2007 aus einer Abspaltung von Pyrex eine performante und weitgehend Python kompatible Programmiersprache mit der Übersetzung in C. Ebenso steht seit 1997 mit Jython (früher JPython) eine Java Implementierung von Python zur Verfügung. Jython eignet sich wie Python als Skriptsprache und ist auf jeder Java-Plattform lauffähig.

### Vim

Vi (Vi IMproved) wurde 1991 von Bram Moolenaar als freie Open-Source Weiterentwicklung des Texteditors vi veröffentlicht. Damals Ende der 1980er Jahre noch unter dem Namen vi Imitation war für Bram Moolenaar mit Version 1.0 nur das Ziel die vi Funktionalität auf einem Amiga nachzubilden. Der Editor entwickelte sich schnell weiter und wurde 1992 unter dem Namen Vi IMproved nach MS-DOS und Unix portiert. Seit August 2013 ist Vim in Version 7.4 verfügbar und umfasst weitere Funktion wie Rechtschreibüberprüfung, Mehrsprachigkeit, Tabs uvm. Außerdem ist es möglich für Vim eigene Makros zu schreiben und dem Editor weitere Befehle hinzuzufügen. Diese werden in der Datei *.vimrc* definiert.

Die Bedienung erfolgt weiterhin ausschließlich über die Tastatur und so ist Vim wie vi auf jedem Terminal ausführbar. Dies hat den Vorteil, dass nicht immer eine GUI auf dem System vorhanden sein muss. Gerade bei ferngesteuerten Terminals bringt das nicht vernachlässigbare Geschwindigkeitsvorteile.

Wenn trotzdem eine GUI gewünscht ist, kann man eine der zahlreichen Weiterentwicklungen wie GVim mit einer modernen grafischen Benutzerschnittstelle nutzen.

Der größte Vorteil von Vim ist zugleich auch sein größter Nachteil. Die zahlreichen Funktionen und Kommandos, die nur per Tastaturbefehle nutzbar sind, sind für Anfänger zunächst schwer zu finden und zu merken. Nach einer gewissen Zeit kann man dann aber mit wenigen Tastatureingaben zügig ganze Texte formatieren und bearbeiten, meist viel schneller als mit gewöhnlichen Texteditoren.

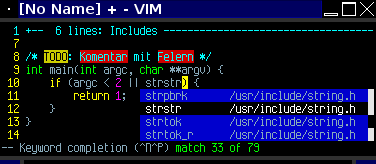


Abbildung 2: Vim-7-Funktion: Rechtschreibprüfung

### Kommandozeile

Die Kommandozeile, oft auch Befehlszeile, Konsole oder Terminal genannt, ist ein Eingabebereich zur typischerweise textbasierten Eingabe von Befehlen an eine Software. Die Kommandos werden über die Tastatur als Zeichenketten eingeben und mit der der Enter (bzw. Return) Taste bestätigt.

Für eine Kommandozeile gibt es eine Vielzahl von Befehlen, die fast ausschließlich aus dem Englischen stammen und der Einfachheit wegen in gekürzter Form eingetippt werden.

Die eigentliche Kommandozeile wird durch einen Kommandozeilen-Interpreter bereitgestellt und ausgewertet. Dieser kann viele Erscheinungsformen haben, von einer einfachen schwarz-weiß bis hin zu einer farbigen Darstellung. Einige bieten sogar eine Maussteuerung und die Implementierung von Formatierungsbefehlen.

Ein RPi lässt sich vollständig über die Kommandozeile und damit wesentlich ressourcenschonender als über eine X11-Anwedung steuern.

Über SSH kann man sich durch Eingabe der RPi-Netzwerkadresse mittels Benutzer und Passwort anmelden und direkt Befehle auf dem RPi ausführen.

Mit „ls“ lassen sich die Dateien im aktuellen Verzeichnis auflisten, mit „mkdir“ lässt sich ein Ordner erstellen und mit „apt-get install“ lassen sich weitere Pakete installieren. Mit „reboot“ oder „halt“ lässt sich der RPi neustarten bzw. runterfahren.

Es gibt noch unzählige weitere Befehle um den RPi steuern und mit ihm arbeiten zu können.

## Netztechniken

### Secure Socket Shell

Secure Socket Shell (SSH), meist einfach nur Secure Shell genannt, ist ein UNIX-Befehlsschnittstelle und Protokoll für sicheren Zugriff auf einen Remote-Computer. Es wird allgemein von Netzwerk-Administratoren verwendet, um Server aus der Ferne steuern zu können. Dabei wird meist die Verbindung auf eine ressourcenschonende Kommandozeile reduziert. Aber auch X11 kann über SSH verschlüsselt übertragen werden.

SSH ist eigentlich eine Suite von drei Utilities - slogin, ssh, scp, welches die sichereren Versionen der früheren UNIX-Dienstprogramme rlogin, rsh und rcp sind. SSH-Befehle sind auf verschiedene Weise verschlüsselt und sicher. Beide Enden der Client / Server-Verbindung werden unter Verwendung eines digitalen Zertifikats authentifiziert, und Passwörter sind verschlüsselt geschützt.

SSH benutzt RSA-Public-Key-Kryptographie sowohl für Verbindung und Authentifizierung. Zur Verschlüsselung werden die Algorithmen Blowfish, DES und IDEA angeboten. Wobei IDEA die Standardeinstellung ist.

SSH2, die neueste Version, beinhaltet eine Sammlung von Standards, vorgeschlagen von der Internet Engineering Task Force (IETF).

Die neuere Protokoll-Version SSH-2 bietet weitere Funktionen wie Datenübertragung per [SFTP](http://de.wikipedia.org/wiki/SSH_File_Transfer_Protocol).

### Secure Copy

Secure Copy (SCP), ist ein Protokoll zum sicheren Kopieren von Daten zwischen zwei Computern über ein Netzwerk. SCP nutzt das SSH-Protokoll zum Verbindungsaufbau und implementiert über diesen Kanal die Datenübertragung. Als Alternativen bieten sich FTP oder SFTP an.

Der Vorteil bei SCP liegt in der sparsamen Implantierung und der simplen Syntax.

Der meist gebräuchlichste Client ist unter Unixsystemen das kommandozeilenbasierte scp.

Beispiel:

Eine Datei zum Host kopieren:

**scp** source.foo [user@host:directory/target.foo](mailto:user@host:directory/target.foo)

Eine Datei vom Host kopieren:

**scp** user@host:directory/source.foo target.foo

# Problemstellung

# Zielsetzung

# Realisierung

## Ansteuerung der Servo-Motoren

### GPIO

Zusätzlich zu den direkten Anschlüssen, wie USB oder HDMI, bietet der RPi 26 GPIOs (General Purpose Input/Output), die jeweils über eine 2,54mm Stiftleiste zugänglich sind. Alle GPIOs haben einen maximalen Pegel von 3,3V und sind nicht kompatibel zu 5V. Überschreitet man diesen Pegel, kann es zu unvorhersehbaren Ergebnissen bis hin zur vollständigen Beschädigung des RPi führen.

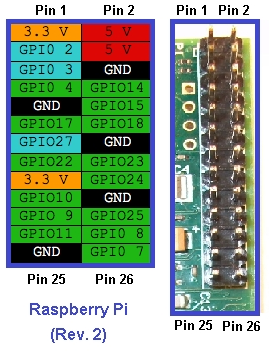


Abbildung 3: RPi GPIO

Da zur Ansteuerung der Motoren Python verwendet werden soll, muss die Python GPIO Library installiert werden. Dies geschieht am Einfachsten durch die automatische Installation über die Kommandozeile mit apt. Falls noch nicht geschehen müssen zuerst die Python-Entwicklungstools installiert werden.

**apt-get install** python-dev

**apt-get install** python-rpi.gpio

Nach der Installation ist es möglich die Library in Python zu importieren.

**import** RPi.GPIO **as** GPIO

Jeder Servo-Motor wird über den Schrittmotor mittels zwei Jumper-Kabeln an jeweils zwei GPIOs angesteckt. Um also gleich zwei GPIO für einen Motor anzusteuern wird eine Funktion „ControlAPairOfPins“ implementiert.

**def** ControlAPairOfPins(FirstPin,FirstState,SecondPin,SecondState):

**if** FirstState == "1":

GPIO.output(int(FirstPin),True)

**else**:

GPIO.output(int(FirstPin),False)

**if** SecondState == "1":

GPIO.output(int(SecondPin),True)

**else:**

GPIO.output(int(SecondPin),False)

**return**

Mit dem folgenden Aufruf wird zum Beispiel Pin 19 auf „1“ und Pin 21 auf „0“ gesetzt, sodass der linke Motor vorwärts dreht.

ControlAPairOfPins("19","1","21","0")

Setzt man dazu noch Pin 24, an dem der zweite Motor angeschlossen ist, auf 1, so bewegen sich beide Motoren vorwärts und der Robby fährt geradeaus. Die Steuerung des Robby funktioniert aufgrund der Gummiketten ohne Lenkung vergleichbar mit einem Kettenfahrzeug, wie z.B. einem Panzer.

### PiZyPwm-Modul

Das PiZyPwm-Modul, welches öffentlich unter github zu finden ist[[4]](#footnote-5), ermöglicht eine PWM (Pulse Width Modulation) Ausgabe der GPIOs unter Python. Der Vorteil des PWM-Signals besteht darin, dass es durch zwei Spannungsebenen (Low- und High-Pegel) gebildet wird. Bei der schaltungstechnischen Realisierung eines PWM-Generators können diese im verlustarmen Schaltbetrieb arbeiten. Dadurch werden eine kontrollierte Ansteuerung der Motoren und damit eine definierbare Geschwindigkeit ermöglicht. Anstatt den festen Werten „0“ und „1“ sind jetzt Interger-Werte von „0“ bis „100“ möglich.

In der Motorensteuerung wird PiZyPwm importiert:

**from** pizypwm **import** \*

Da das PiZyPwm-Modul nur als fertige .py-Datei aus Github bezogen wird und nicht über einen Paketmanager installiert, muss die Datei entweder manuell den Python Librarys hinzugefügt oder lokal im selben Verzeichnis abgelegt warden.

Die Ansteuerung von Pin 19 mit halber Geschwindigkeit sieht nun folgendermaßen aus:

first = PiZyPwm(20000, 19, GPIO.BOARD)

first.start(50)

## Abstandsmessung mittels Ultraschall

### Ansteuerung des SRF04

Das verwendete Ultraschallmodul SRF04 wird mit GND und VCC des RPi verbunden und an der Front des Robby befestigt. Außerdem wird zur Auswertung Trigger an Pin 8 (GPIO14) und Echo an Pin 10 (GPIO15) gesteckt.

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO\_TRIGGER = 14

GPIO\_ECHO = 15

Ausgelegt ist das Modul für eine Spannungsversorgung von 5 Volt, hier ist allerdings zu beachten dass die GPIO Pins am Eingang nur 3,3 Volt aushalten. Aufgrund der geringen Spannung misst das SRF04 nur Entfernungen nur bis ca. 50 cm. Versuche mit einem Widerstand und Steckbrett brachten zwar eine höhere Spannungen und die Möglichkeit größere Entfernungen zu messen, aber bedeuteten zusätzliche Bauteile und mehr Gewicht am Robby. Da aber die Werte unter 50cm in verschiedenen Tests mit einer höheren Präzision und zuverlässiger gemessen wurden, wurde entschieden das höhere Gewicht in Kauf zu nehmen und das SRF04 Modul mit 5V zu betreiben.

In einer Dauerschleife wird jede Sekunde die Entfernung immer wieder gemessen und auf der Kommandozeile ausgegeben. Dazu wird die Zeit gemessen, die der Ultraschall für den Weg vom Robby bis zum Hindernis und wieder zurück benötigt. Diese wird mit 34300 multipliziert und durch zwei dividiert, da wir nur einen Weg wissen möchten.

**while** True:

GPIO.setup(GPIO\_TRIGGER,GPIO.OUT) # Trigger

GPIO.setup(GPIO\_ECHO,GPIO.IN) # Echo

start = time.time()

GPIO.output(GPIO\_TRIGGER, False)

**while** GPIO.input(GPIO\_ECHO)==0:

stop = time.time()

**while** GPIO.input(GPIO\_ECHO)==1:

stop = time.time()

elapsed = stop-start

distance = elapsed \* 34300

distance = distance / 2

**print** "Distance : %.1f cm" % distance

time.sleep(1)

GPIO.cleanup()

## Web Server

### Tornado

Tornado ist ein Python Mikro-Webframework und Webserver, welcher von FriendFeed entwickelt und unter einer Open-Source-Lizenz veröffentlicht wurde. Durch das Nutzen von nichtblockierenden Netzwerk-E/A wird es ermöglicht tausende Verbindungen gleichzeitig zu öffnen und abzufragen. Das macht es ideal für WebSockets, die viele Anfragen und/oder langlebendende Verbindungen zum Nutzer benötigen.

Tornado ist in PyPI (Python Package Index) gelistet und kann mit pip oder easy\_install installiert werden.

**easy\_install** tornado

### Python Steuerung

Neben den benötigten Modulen RPI.GPIO und PiZyPwm wird für die Steuerung über den Web Server ebenso tornado in Python importiert.

**import** tornado.web

**import** tornado.websocket

**import** tornado.ioloop

Mit dem tornado-Mainhandler wird die „index.html“,welche im selben Verzeichnis liegt, für den Webserver geladen. Sobald der Nutzer „raspberryip:8888“ aufruft, wird im diese HTML zurückgegeben und im betreffenden Browser dargestellt.

**class** **MainHandler**(tornado.web.RequestHandler):

**def** get(self):

self.render("index.html")

Simultan wird mit der jQuery Datei verfahren, die für die Anzeige der Steuerungsbefehle und die Darstellung des Kamera-Streams zuständig ist.

**class** **JQueryHandler**(tornado.web.RequestHandler):

**def** get(self):

self.render("jquery-2.1.0.min.js")

Am Ende der Python Websteuerung wird in der „main“ der Tornado Webserver mit dem Port 8888 in einer Dauerschleife, die auf Ein-/Ausgabe wartet, gestartet.

**if** \_\_name\_\_ == "\_\_main\_\_":

application.listen(8888)

tornado.ioloop.IOLoop.instance().start()

Läuft der Server auf dem RPi und jemand ruft über einen Webbrowser die HTML auf, kann er mit den Steuerungstasten Nachrichten an den Webserver schicken. Wird z.B. die Taste „w“ gedrückt, bedeutet das „vorwärts“ und an den Webserver wird die Nachricht mit dem Inhalt „forwards“ gesendet.

Da es unter Python keine direkte Switch-Case Implementierung, wird die Nachricht durch mehrere If-Abfragen ausgewertet. Sobald die Nachricht bei einem If oder Elif zutrifft, wird der Befehl ausgeführt und die anderen If-Abfragen werden ignoriert. So wird im besten Fall nur eine If-Abfrage durchgeführt und im schlechtesten Fall die maximal Anzahl der verfügbaren Richtungen. In dem Fall sind es vier Stück.

**class** **WebSocketHandler**(tornado.websocket.WebSocketHandler):

**def** on\_message(self, message):

self.message = message

**if** self.message == 'forwards':

action = forwards()

**elif** self.message == 'reverse':

action = reverse()

**elif** self.message == 'left':

action = left()

**elif** self.message == 'right':

action = right()

**elif** self.message == 'stop':

action = stop()

Indem vorhin genannten Beispiel der “w”-Taste und daraus resultierenden Nachricht “forwards”, wird hier die erste If-Abfrage zutreffen und die Funktion “forwards()” wird ausgeführt warden.

**def** forwards():

first.start(0)

second.start(50)

third.start(50)

fourth.start(0)

**return** "forwards"

Am Beginn der Datei wird der auszuführende Interpreter “#!/usr/bin/python2” angegeben und die Datei mit „u+x foo.py“ ausführbar gemacht, sodass der Webserver einfach mit „./foo.py“ ausführbar ist. So ist es möglich den Webserver in den Autostart des RPi zu integrieren und eine direkte Steuerung inklusive Kamera-Stream über einen Webbrowser zu ermöglichen, ohne dass noch Nutzereingaben per Kommandozeile notwendig sind.

### Erweiterung: „selbständiges Fahren“

Auf der Weboberfläsche im Browser werden zwei Buttons integriert, die es erlauben zwischen der gewohnten Steuerung mit den „WASD“-Tasten und dem sogenannten „autonomy mode“ zu wechseln. In diesem „autonomy mode“ fährt der Robby immer geradeaus und misst dabei die Entfernung zum nächsten Objekt vor sich. Fällt der Abstand unter einen einstellbaren Wert (im Beispiel auf 20cm eingestellt), stoppt der Robby, dreht sich für ein festes Intervall nach rechts und prüft erneut den Abstand. Findet er kein Objekt mehr vor sich, fährter wieder geradeaus. Andernfalls führt er wieder einer Drehung nach rechts durch. Währenddessen ist es dem Benutzer möglich das aktuelle Bild der Kamera zu überwachen und weiß immer wo sich der Robby aktuell befindet.

## Backupmanagement

### Win32DiskImager

Es gibt verschiedene Möglichkeiten von einem System eine bootfähige Wiederherstellungskopie zu erstellen. Da die Ressourcen des RPi begrenzt sind wird bewusst auf ein integriertes Backupsystem verzichtet. Auch eine Kopie aus dem laufenden System heraus erweist sich aufgrund der aktuell geöffneten Dateien als schwierig.

Deswegen wird das Backup über ein externes System erstellt. Dafür wird ein Windows Rechner mit einem SD-Kartenlesegerät verwendet.

Die SD-Karte wird nach dem herunterfahren des RPi aus dessen Lesegerät entfernt und in den Windows Rechner gesteckt. Das Tool „Win32DiskImager“ liest die Daten aus einem gemounteten Wechseldatenträger wie SD-Karten, USB-Speicher oder CD/DVD-Laufwerken aus und erstellt aus diesen ein komplettes Image im Format „.img“.

Dieses lässt sich genauso simpel wieder mit dem Tool einlesen und auf einen Wechseldatenträger zurückspielen.

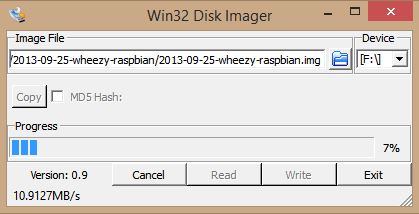


Abbildung 4: Win32DiskImager

## Ausblick

# Fazit

# Quellenverzeichnis

[1] Wikipedia: Python

http://de.wikipedia.org/wiki/Python

Einsichtnahme 24.02.2014

# Literaturverzeichnis

[a] ERNESTI JOHANNES/KAISER, PETER (2008):

Python – Das umfassende Handbuch

ISBN 978-3-8362-1110-9

[b] MONK, SIMON (2013):

Programming the Raspberry Pi: Getting Started with Python (2013):

ISBN 978-0071807838

[c] BIONDI, PHILLIPE (2009):

Scapy Documentation (Release 2.1)

http://www.secdev.org/projects/scapy/doc/

# Anhang

### Quellen der Abbildungen

Abbildung 1: http://de.wikipedia.org/wiki/Linux\_(Kernel) (01.04.2014)

Abbildung 2: http://de.wikipedia.org/wiki/Vim (01.04.2014)

Abbildung 3: https://pypi.python.org/pypi/RPi.GPIO (12.05.2014)

Abbildung 4: Screenshot Windows 7 – Simon Buttke (27.05.2014)

1. RPi Fundation: http://www.raspberrypi.org/a-birthday-present-from-broadcom/ [↑](#footnote-ref-2)
2. Wikipedia: Python [2] [↑](#footnote-ref-3)
3. Peter Norvig, [Google, Inc.](http://google.com/) / http://www.python.org/about/quotes/ (09.08.2013) [↑](#footnote-ref-4)
4. Github PiZyPwm: https://github.com/aboudou/pizypwm/tree/master/rpi.gpio [↑](#footnote-ref-5)