



# Research Lab for Deep Learning Mars Science Laboratory Curiosity Rover

Studienarbeit (T3\_3101)  
für die Prüfung zum  
Bachelor of Science

des Studiengangs Angewandte Informatik an der  
Dualen Hochschule Baden-Württemberg Mosbach

von

Niklas Koopmann

Abgabedatum	1. Mai 2020
Bearbeitungszeitraum	19 Wochen
Matrikelnummer, Kurs	9742503, MOS-TINF17B
Ausbildungsunternehmen	Deutsche Bundesbank
Gutachter der Dualen Hochschule	Prof. Dr. Carsten Müller

# Ehrenwörtliche Erklärung

## Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit (bzw. Studien- und Projektarbeit) mit dem Thema „Mars Science Laboratory Curiosity Rover“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt. \*

\* falls beide Fassungen gefordert sind

---

Ort, Datum

---

Niklas Koopmann

**Hinweis:** Aus Gründen der besseren Lesbarkeit wird in dieser Arbeit auf die gleichzeitige Verwendung männlicher und weiblicher Sprachform verzichtet (*generisches Maskulinum*). Sämtliche Personenbezeichnungen gelten gleichwohl für beiderlei Geschlecht.

# **Abstract**

# **Zusammenfassung**

# Inhaltsverzeichnis

<b>Abstract</b>	<b>III</b>
<b>Abkürzungsverzeichnis</b>	<b>V</b>
<b>Abbildungsverzeichnis</b>	<b>VI</b>
<b>Quelltextverzeichnis</b>	<b>VII</b>
<b>1 Spezifikation</b>	<b>1</b>
1.1 Motivation . . . . .	3
1.2 Aufgabenstellung . . . . .	3
<b>2 Modell</b>	<b>5</b>
2.1 Vorderansicht . . . . .	5
2.2 Seitenansicht . . . . .	5
2.3 Ansicht von oben . . . . .	10
<b>3 Implementierung</b>	<b>14</b>
3.1 Installation und Konfiguration Raspberry Pi . . . . .	16
3.2 Installation und Konfiguration Kamera . . . . .	20
<b>4 Nachweis der Leistungsfähigkeit</b>	<b>22</b>
<b>5 Fazit</b>	<b>24</b>
5.1 Ausblick . . . . .	24
<b>Literaturverzeichnis</b>	<b>26</b>
<b>Anhang</b>	<b>29</b>

# **Abkürzungsverzeichnis**

<b>ALSA</b>	Advanced Linux Sound Architecture
<b>CSI</b>	Camera Serial Interface
<b>DHBW</b>	Duale Hochschule Baden-Württemberg
<b>GPIO</b>	General Purpose Input/Output
<b>HAT</b>	Hardware Attached on Top
<b>HDMI</b>	High Definition Multimedia Interface
<b>HSV</b>	Hue, Saturation, Value
<b>NASA</b>	National Aeronautics and Space Administration
<b>RGB</b>	Rot, Grün, Blau
<b>SSH</b>	Secure Shell
<b>VNC</b>	Virtual Network Computing

# Abbildungsverzeichnis

2.1	Gerenderte Frontansicht des digitalen Rover-Modells . . . . .	6
2.2	Fotografie des realen Rover-Modells aus Frontansicht . . . . .	7
2.3	Gerenderte Seitenansicht des digitalen Rover-Modells . . . . .	8
2.4	Fotografie des realen Rover-Modells aus Seitenansicht . . . . .	9
2.5	Gerenderte Draufsicht des digitalen Rover-Modells . . . . .	11
2.6	Fotografie des realen Rover-Modells aus Draufsicht . . . . .	12
3.1	Varianten einer Linkskurve beziehungsweise -rotation beim genutzten Antriebskonzept . . . . .	15
3.2	Belegung der Steuerungsausgänge zu den einzelnen Motoren an den drei BrickPis des Steuergerätes . . . . .	17
3.4	Powerbank der Firma XT Power, die zur Stromversorgung genutzt wird .	17
3.3	Belegung der Ein- und Ausgänge eines einzelnen BrickPi 3 . . . . .	18
3.5	Alternative zur Powerbank: Batteriepack aus dem Lieferumfang des BrickPi	19
3.6	Für die Sprachausgabe genutzter Lautsprecher (Inspirion GmbH) . . . .	19
3.7	Blauer LEGO-Stein 3005 . . . . .	20

# **Quelltextverzeichnis**

3.1 Auszug aus der Konfiguration mit Standardwerten für die farbbasierte Objekterkennung . . . . .	21
--	----

# 1 Spezifikation

## Studienarbeit

### Spezifikation

<b>Titel</b>	<b>Research Lab for Deep Learning (Prio 1)</b>
	Mars Science Laboratory Curiosity Rover
<b>Bearbeiter</b>	Niklas Koopmann
<b>Zielsetzung</b>	Konzeptionierung und Implementierung eines Rover für die Erkundung einer (Mars-)Oberfläche.
<b>Wichtige Hinweise</b>	<b>Präsentation ausgewählter Ergebnisse auf der renommierten Konferenz „ICSI 2020“, der „AI Night“ sowie der „Hannover Messe 2020“.</b>

#### Beschreibung:

Mars Science Laboratory ist eine NASA-Mission im Rahmen des Flagship-Programms, die den Mars hinsichtlich seiner aktuellen und vergangenen Eignung als Biosphäre erforscht.

Hierzu wurde auf der Oberfläche ein weitgehend autonomer Rover mit dem Namen Curiosity abgesetzt, der mit Instrumenten zur Untersuchung von Gestein, Atmosphäre und Strahlung ausgerüstet ist.

#### Animation

<https://www.youtube.com/watch?v=P4boyXQuUIw>



Eine LKW-Plane wird mit Mustern – zwecks Klassifizierung von Gesteinstypen – bedruckt. Unter der LKW-Plane wird Sand aufgeschüttet, um ein Gelände mit Höhen und Tiefen zu schaffen.

Der Rover ist durch eine geeignete Fahrwerkskonstruktion weitestgehend gegen Kippen zu sichern.

Über die Sprachkommandos *Start*, *Move Left*, *Move Right* und *Stop* wird der Rover gesteuert.

Zusätzlich werden rechteckige kleine Objekte (Wasser) an beliebigen Stellen auf der LKW-Plane platziert. Der Rover bewegt sich innerhalb definierter Grenzen auf der LKW-Plane. Hat der Rover ein (trainiertes) bekanntes Objekt gefunden, wird die Klassifizierung zu Simulationszwecken als Sprachausgabe kommuniziert, wie z.B. Water found. Bei dem Fund von Wasser wird zusätzlich das Objekt als Probe aufgenommen und in einem Behälter gesammelt.

<b>Organisation</b>	5. Semester: Konzeptionierung, Prototypische Implementierung; 6. Semester: Produktive Implementierung, Test, Dokumentation.
<b>Rahmenbed.</b>	Raspberry Pi, Pixy-Kamera, Programmiersprache: Python; ggf. zusätzliche Teile sind im 3D-Druck zu erstellen  Das Lego-Modell ist formschön für die Messe-Präsentation zu realisieren.  <b>Raspberry Pi und Lego-Teile für die prototypische Implementierung werden leihweise von der DHBW (Labor Digitale Fabrik) zur Verfügung gestellt.</b>
Status	Final

Dr. Carsten Müller // 12.10.2019

## Projektorganisation

### Meilensteine 5. Semester (1. Abschnitt)

Zielsetzung ist eine qualitativ hochwertige Studienarbeit mit der Bewertung 1.0.

5. Semester: Konzeptionierung und Prototypische Implementierung kritischer Komponenten		
#	Meilenstein	Termin
M01	Basis-Installation und Konfiguration Raspberry Pi	20.10.2019
M02	Steuerung Rover über Sprachkommandos	03.11.2019
M03	Basis-Installation und Konfiguration Pixy-Kamera	10.11.2019
M04	Erkennung von Wasserobjekten und Sprachausgabe	24.11.2019
M05	Sechsrädriges Antriebssystem für unebenes Gelände	01.12.2019
M06	Dokumentation mit BrickLink Studio (Version 2.0.10) <a href="https://studio.bricklink.com/v3/studio/download.page">https://studio.bricklink.com/v3/studio/download.page</a>	08.12.2019

Die Termine zu den Meilensteinen M01-M06 sind verbindlich.

Zu jedem Meilenstein findet eine Skype Session zwecks Qualitätssicherung und Abnahme statt.  
Bitte vor der Skype Session die Dropbox mit qualitativ hochwertigen Arbeitsergebnissen aktualisieren.

## 1.1 Motivation

Am 6. August 2012 landete der Rover *Curiosity* als Teil der Mission *Mars Science Laboratory* der US-amerikanischen Raumfahrtbehörde National Aeronautics and Space Administration (NASA) im Gale-Krater auf dem Mars [19]. Seither hat er rund 21,93 Kilometer zurückgelegt (Stand: Sol 2695) [15] und die Landschaft entlang der Fahrtstrecke auf vielfältige Weise im Hinblick auf ihre frühere und aktuelle Eignung als Lebensraum für Organismen untersucht. Ein wichtiger Aspekt für den Nachweis der Existenz von Lebewesen ist das Vorhandensein flüssigen Wassers [16], dessen Rückstände der Rover mithilfe eines aktiven Neutronenspektrometers in Mineralien nachzuweisen versucht. [19]

Um weite Strecken im schroffen, unebenen Gelände der Marsoberfläche zurücklegen zu können, ist *Curiosity* mit einem komplexen, sechsrädrigen Antriebs- und Federungssystem ausgestattet. Das Fahrwerk ist nach dem Rocker-Bogie-System der NASA aufgebaut, um Hindernisse bis zu einer Höhe des Raddurchmessers von 0,5 m überwinden zu können [1]. Eine solche Aufhängung ermöglicht dem Rover eine hohe Geländegängigkeit ohne den Einsatz von Achsen und erlaubt der Verbindung der hinteren Räder beider Seiten eine freie Drehung um den Verbindungspunkt zur Hauptstrebe, an der auch das jeweilige Vorderrad montiert ist [3].

Dieses Projekt (Mars Science Laboratory Curiosity Rover) wird im Rahmen des Research Lab for Deep Learning der Dualen Hochschule Baden-Württemberg (DHBW) zeitgleich mit der Erforschung schwarmbasierter Logistikabläufe durchgeführt. Langfristig soll das Ergebnis unter anderem zum Zwecke einer schwarmbasierten Erkundung marsähnlicher Landschaften weiterentwickelt werden können.

## 1.2 Aufgabenstellung

Ziel dieser Arbeit ist die „Konzeptionierung und Implementierung eines Rover für die Erkundung einer (Mars-)Oberfläche“ [13] in Form einer Machbarkeitsstudie (Proof of Concept). Die Suche nach Wasser wird dabei ebenfalls abgebildet und soll mithilfe der visuellen Erkennung blauer LEGO-Steine simuliert werden. Zusätzlich ist das Antriebskonzept des *Curiosity*-Rovers originalgetreu nachzubilden: Es sollen sechs angetriebene Räder installiert sein, von denen vier gelenkt werden. Eine grundlegende mechanische Federung soll die Geländetauglichkeit steigern.

## 1.2 Aufgabenstellung

In einzelnen Punkten ist nachträglich einvernehmlich von der Spezifikation abgewichen worden: Das Aufsammeln und Transportieren der Wasserobjekte ist obsolet. Wenn ein Wasserobjekt erkannt wird, soll dies lediglich über die Sprachausgabe verkündet werden. Der Fokus der Entwicklung ist von der Geländetauglichkeit hin zur Stabilität und Formschönheit gerückt worden. Das klassische Rocker-Bogie-Fahrwerk wird im Sinne der Originaltreue in leicht abgewandelter Form verbaut (siehe dazu Abschnitt 2.2).

# 2 Modell

Im ersten Schritt soll die digitale Modellierung des zu erschaffenden Mars Rovers erfolgen. Dazu wird gemäß der Spezifikation die Anwendung BrickLink-Studio (in der Version 2.0.10) genutzt. Diese ermöglicht die umfangreiche und präzise Modellierung mit LEGO-Elementen sowie beispielsweise Funktionen zur Stabilitätsanalyse und Generierung schrittweiser Bauanleitungen. In den folgenden drei Abschnitten sind jeweils eine gerenderte Front- und Seitenansicht und eine Draufsicht des Mars-Rover-Modells dargestellt. Zusätzlich ist zum Vergleich in jedem Abschnitt eine Fotografie des realen Modells aus der entsprechenden Perspektive abgebildet.

## 2.1 Vorderansicht

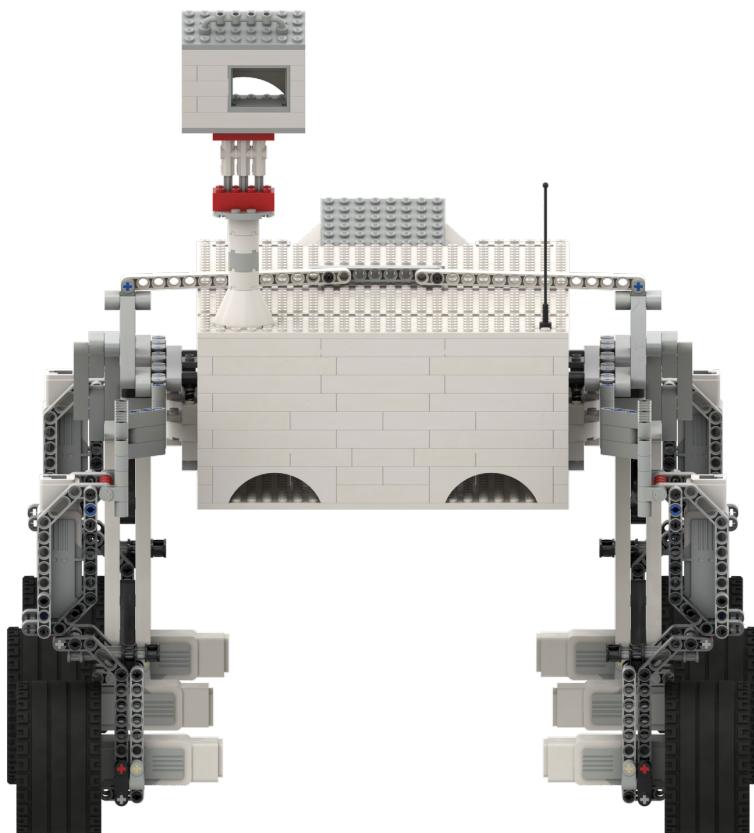
In der Frontalansicht erkennt man die zwei breiten Vorderräder, die über jeweils einen Motor zum Antrieb und einen für die Lenkung des Rades verfügen. Über eine Aufhängung sind sie mit dem Korpus des Rovers verbunden und etwas von diesem abgesetzt, um die Stabilität zu erhöhen. Weiterhin ist in der Realaufnahme die genutzte Kamera (siehe dazu Abschnitt 3.2), platziert im charakteristischen Kopf des Rovers, erkennbar. Der Kopf ist um 22,5° nach unten geneigt, sodass die Erkennung von Wasserobjekten auch auf kurze Distanz möglich bleibt. Durch die beiden Öffnungen an der Unterkante des Korpus sind die Verbindungskabel zu den vorderen Motoren verlegt.

## 2.2 Seitenansicht

Der im vorherigen Abschnitt beschriebene Neigungswinkel des Kopfes wird in der Seitenansicht deutlich. Weiterhin sind die mittleren und die Hinterräder des Rovers zu sehen. Auf beiden Seiten sind diese starr miteinander verbunden und gemeinsam an einem einzigen Punkt mit einer vom Korpus ausgehenden, gefederten Strebe verbunden. Dies entspricht dem Aufbau des Rocker-Bogie-Systems, welches bei *Curiosity* zum Einsatz gekommen ist. So ist sichergestellt, dass die mittleren Räder, die den Rover seitlich stabilisieren und antreiben, sowie die hinteren Räder, die ihn antreiben und gegebenen-

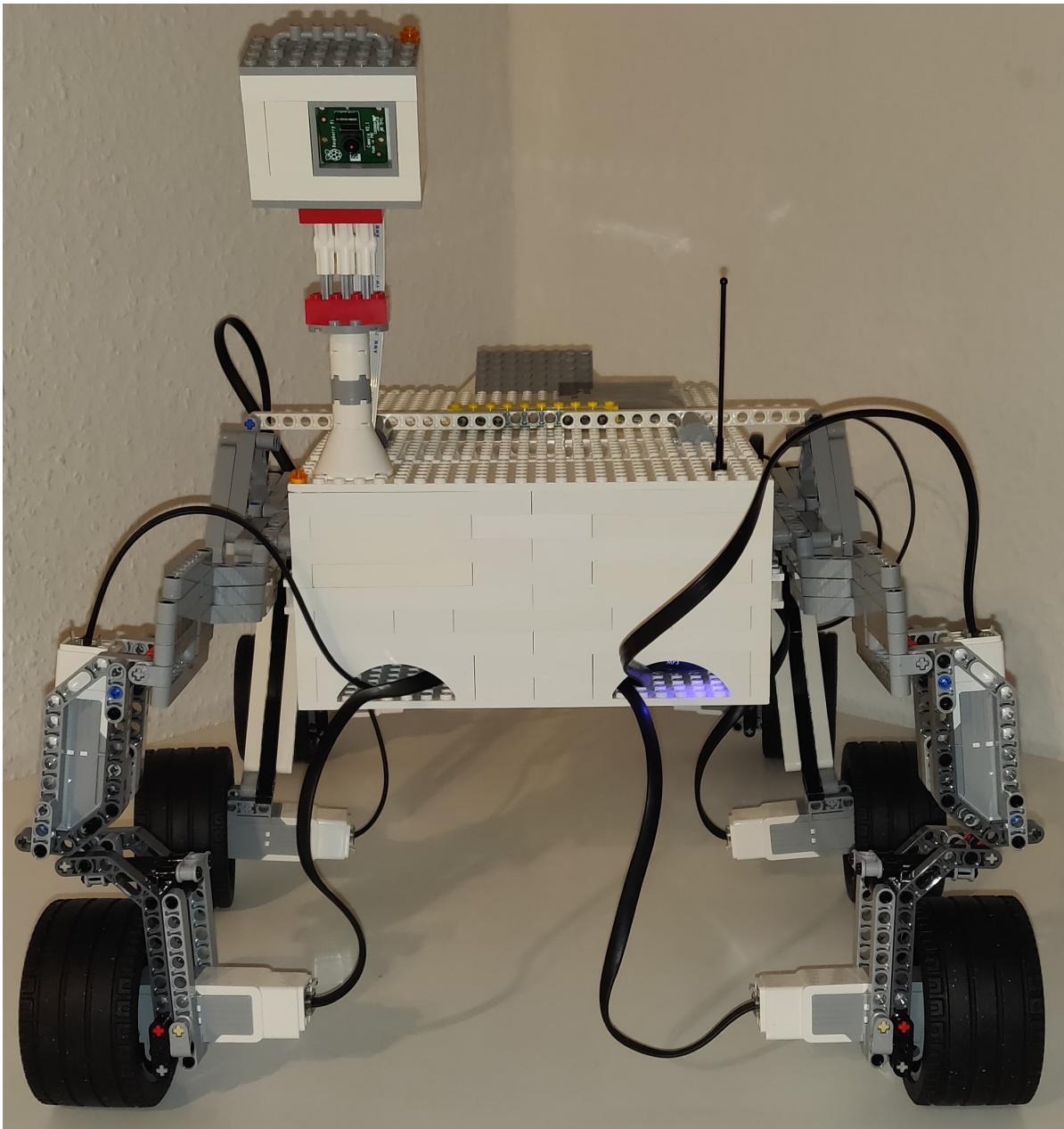
## 2.2 Seitenansicht

---



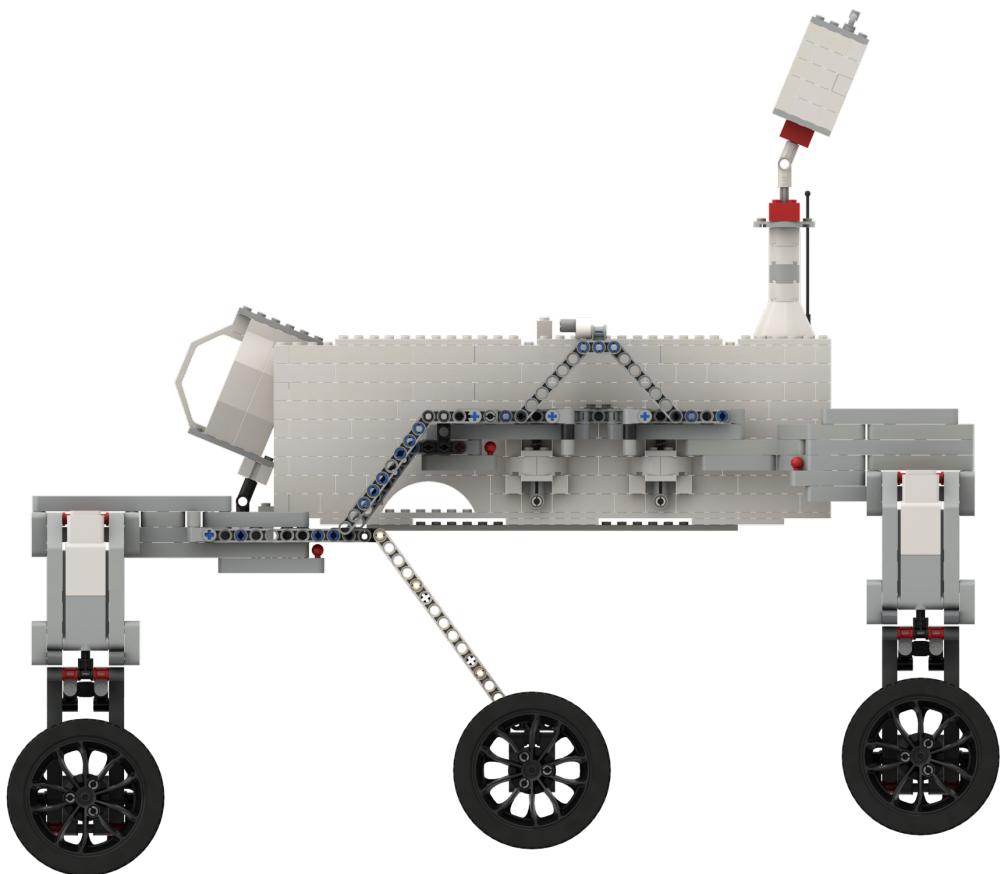
Quelle: eigene Darstellung (Render-Export aus BrickLink Studio 2.0.10)

Abbildung 2.1: Gerenderte Frontansicht des digitalen Rover-Modells



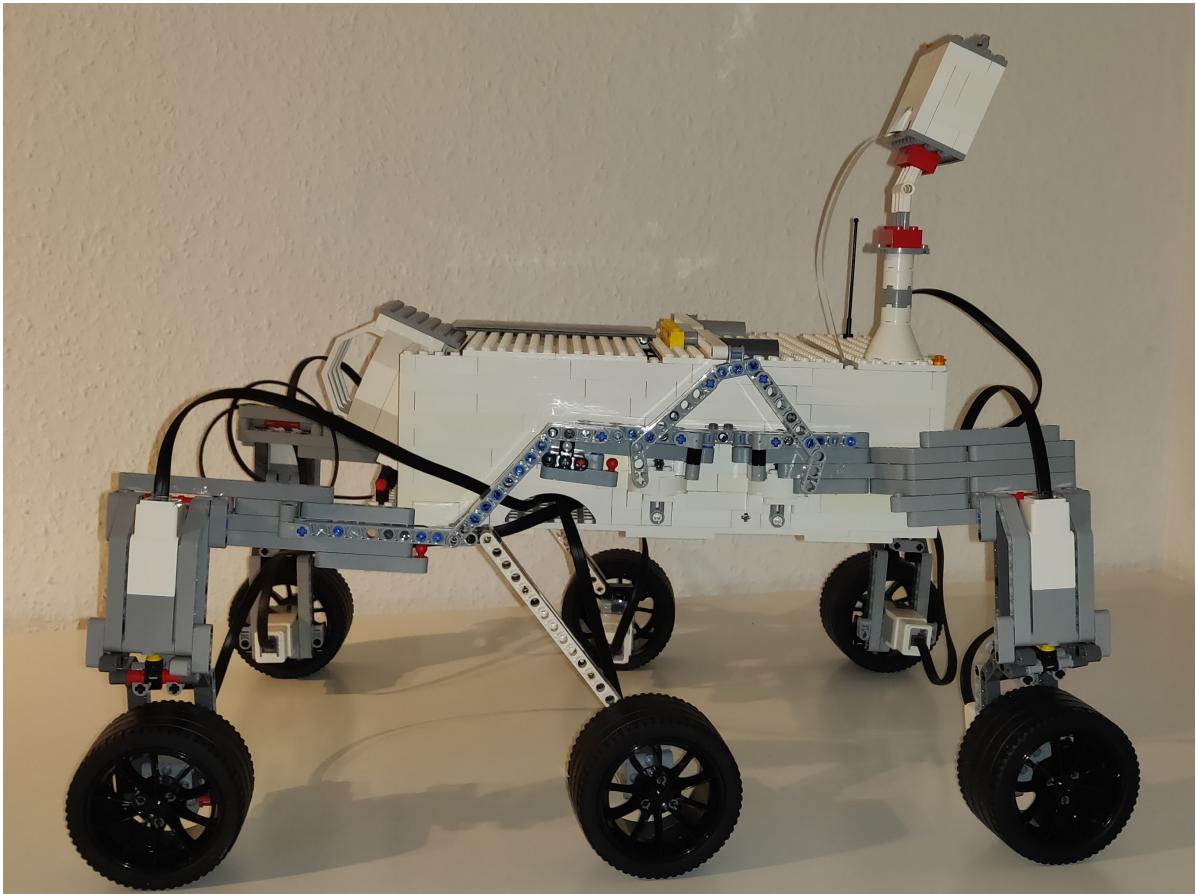
Quelle: eigene Aufnahme

Abbildung 2.2: Fotografie des realen Rover-Modells aus Frontansicht



Quelle: eigene Darstellung (Render-Export aus BrickLink Studio 2.0.10)

Abbildung 2.3: Gerenderte Seitenansicht des digitalen Rover-Modells



Quelle: eigene Aufnahme

Abbildung 2.4: Fotografie des realen Rover-Modells aus Seitenansicht

falls lenken, in jedem Fall den Bodenkontakt beibehalten. Bezuglich der Verbindung zum Korpus wurde vom Rocker-Bogie-System insofern abgewichen, dass dieser nicht frei um einen Punkt rotieren kann, da eine kontrollierte Verlagerung seines Schwerpunkts entlang der Fahrtrichtungssachse aktuell nicht ermöglicht werden kann. Entsprechend bestünde zum Beispiel durch Reibung an der Verbindungsstelle zur Aufhängung oder Windeinwirkung von vorn oder hinten die Gefahr, dass der Korpus kippt und nicht selbsttätig zurückgedreht werden kann. Es ist jedoch eine minimale Drehung um den zentralen Ankerpunkt zwischen Korpus und Fahrwerk möglich. Durch zwei Federn, die an beiden Seiten auf gleicher Höhe zu diesem Punkt angebracht sind, wird der Korpus wenn nötig zurück in die Waagerechte gefedert.

Die Öffnungen links der Ankerpunkte dienen hier auch der Kabelführung, in diesem Fall für die Kabel zu den mittleren und hinteren Motoren.

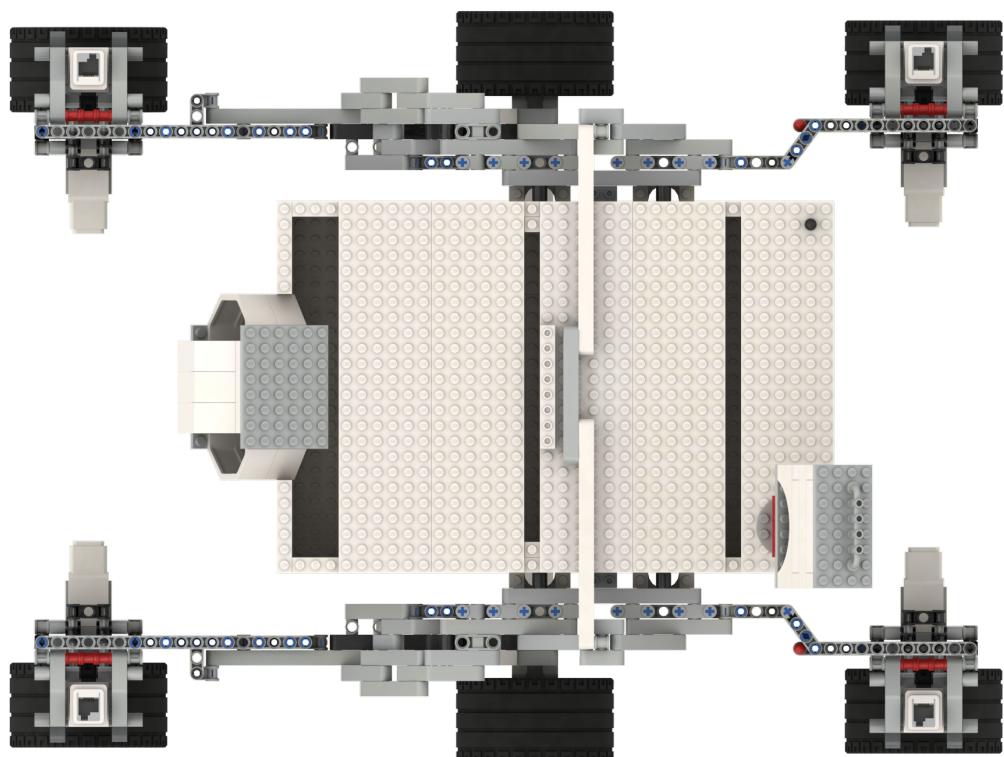
## 2.3 Ansicht von oben

In der Draufsicht ist ein aus einigen Liftarm-Streben bestehender, brückenartiger Überbau auf dem Rover zu erkennen. Dieser dient einerseits der gleichmäßigeren Lastverteilung und unterstützt andererseits die Ankerpunkte zwischen Korpus und Fahrwerk, da die Teile dauerhaft unter Spannung stehen und dadurch seitlich von außen Druck auf die Ankerpunkte ausüben. Im digitalen Modell ist die Strebe in der Mitte dieses Überbaus asymmetrisch zum Rest, da BrickLink Studio es nicht ermöglicht, die Teile unter Spannung anzubringen. Die Fotografie des realen Modells zeigt, wie die Anbringung konzipiert ist, und dass die diagonal angeordneten Streben an den Ankerpunkten (in Abbildung 2.6 oben und unten mittig) zur Mitte gezogen werden.

Bei Betrachtung des realen Modells von oben erkennt man das Solarpaneel auf dem Korpus des Rovers. Dieses soll die Energiegewinnung aus Sonnenlicht von Curiosity abbilden, wenngleich die Nennleistung des Moduls mit 750 mW auf das entwickelte Gesamtsystem keine großen Auswirkungen hätte. Aus einer Messung geht hervor, dass in das Gesamtsystem bereits im Leerlauf (alle BrickPis und Raspberry Pi eingeschaltet, keine Motoren aktiv) eine Leistungsaufnahme von rund 6,8 W hat. Unter Vollast (alle BrickPis und Raspberry Pi eingeschaltet, alle Motoren laufen auf 100 % Leistung) nimmt das Gesamtsystem knapp 21 W auf. Das Modul wird daher in der aktuellen

### 2.3 Ansicht von oben

---

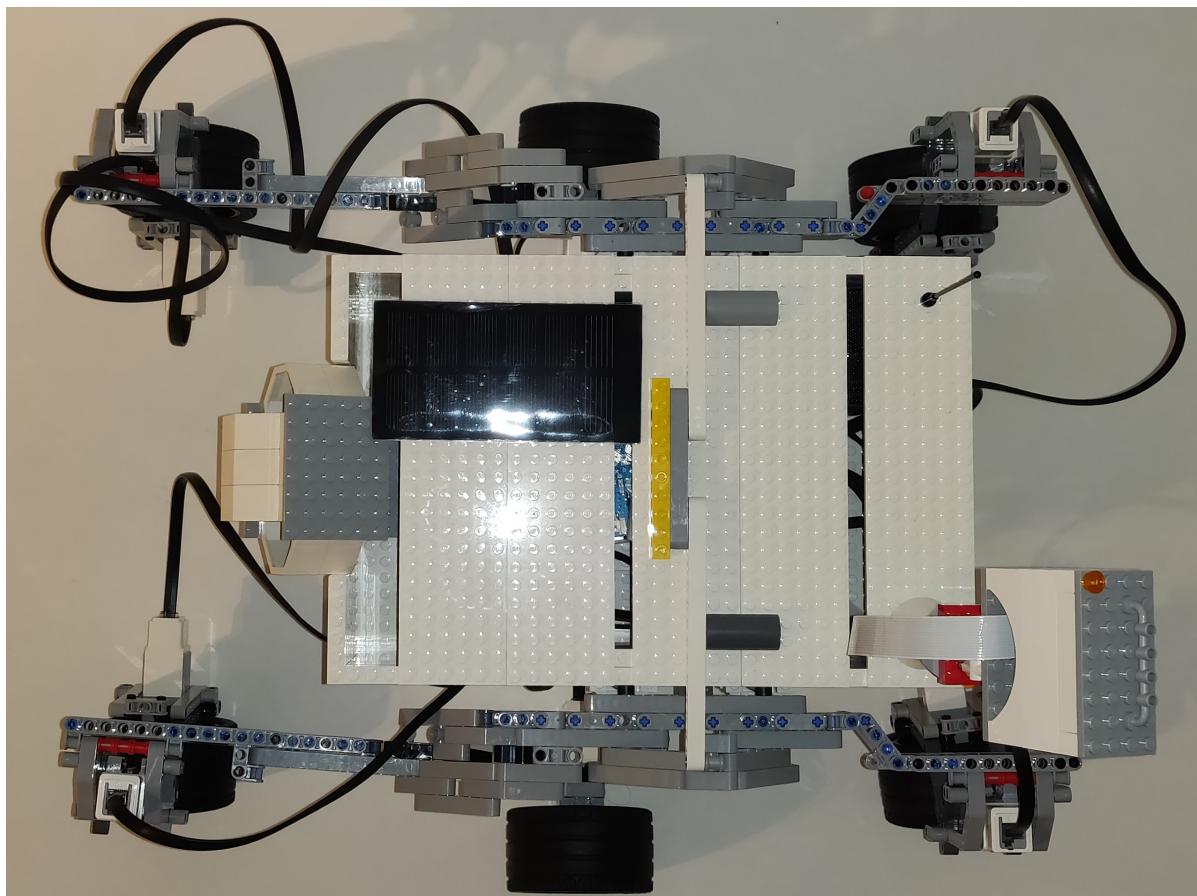


Quelle: eigene Darstellung (Render-Export aus BrickLink Studio 2.0.10)

Abbildung 2.5: Gerenderte Draufsicht des digitalen Rover-Modells

### 2.3 Ansicht von oben

---



Quelle: eigene Aufnahme

Abbildung 2.6: Fotografie des realen Rover-Modells aus Draufsicht

Entwicklungsstufe nicht zur Stromerzeugung genutzt. Weiterhin unterscheidet sich das reale Modell durch kleinere Dekorationselemente auf dem Korpus vom digitalen Modell. Diese haben allerdings keinen Einfluss auf die Funktionalität.

Darüber hinaus ist in dieser Ansicht ein guter Überblick über die Platzierung der einzelnen Räder und Motoren in Relation zum Korpus gegeben. Um den Rover seitlich zu stabilisieren, sind die beiden mittleren Räder etwas weiter außen platziert. Alle Räder liegen jeweils mehrere Zentimeter vom Korpus entfernt am Boden auf.

Im „Bienenstock“ am Heck des Rovers (in den Abbildungen 2.3, 2.4, 2.5 und 2.6 links zu sehen) befindet sich beim realen Vorbild ein Atomreaktor, auf den in dieser Entwicklung allerdings verzichtet wurde. Stattdessen lässt sich in diesem sehr gut der verwendete Lautsprecher (siehe dazu Abschnitt 3.1 und Abbildung 3.6) unterbringen.

# 3 Implementierung

In diesem Kapitel wird die gesamte Systemumgebung um das Steuerelement in Form eines Raspberry Pi beschrieben und erläutert. Die gesamte Implementierung der Steuerungsskripts erfolgt in Python und wird sinnvoll in Module unterteilt. Der Quelltext zum gesamten Projekt wird auf GitHub veröffentlicht.

Insgesamt werden vier Module implementiert, die alle diskrete Funktionalitäten des Modells abdecken: Die Funktionen zur Motoransteuerung sind im Modul `Actuator` implementiert. Für die Bilderkennung der Wasserobjekte auf einem Kamera-Feed der verwendeten Kamera (Details siehe Abschnitt 3.2) ist das Modul `Camera` entwickelt worden. Die programmierten Skripte für Sprachein- und -ausgabe befinden sich respektive in den Modulen `Voice_Recognition` und `Voice_Output`. Koordiniert wird die auf diesen vier Modulen basierende Roversteuerung durch das Skript `rover.py`. In der Konfigurationsdatei `config.py` werden Variablen zur globalen Nutzung zentral deklariert und mit passenden Werten initialisiert.

Um die gleichzeitige Sprach- und Objekterkennung zu ermöglichen, erzeugt das Hauptskript zwei Threads, die eine quasi-zeitgleiche Verarbeitung der entsprechenden Funktionen ermöglichen [7, 20]. Die Funktion, welche im Thread für die Wassererkennung aufgerufen wird, nimmt als Argument eine Callback-Funktion entgegen. Hier wird die Funktion `water_found` des Moduls `Voice_Output` übergeben, sodass bei jeder Erkennung eines Wasser-Objektes der Text „Water found“ ausgegeben wird. Die Ausgabefunktionalitäten der Software beruhen auf der `pyttsx3`-Bibliothek, welche umfangreiche Text-to-speech-Funktionen und Unterstützung für verschiedene Synthesizer bereitstellt. Als Sprach-Synthesizer wird hier wegen seiner Leichtgewichtigkeit und Kompatibilität zu Raspbian *eSpeak* [14] genutzt. Die Spracherkennung ruft entsprechende Funktionen zur Motorsteuerung nach der Erkennung von Eingabebefehlen direkt auf und benötigt demnach keine Callback-Funktion. Zur Erkennung der Sprachbefehle kann *PocketSphinx* genutzt werden. Mit einigem Training könne dieses gute Resultate produzieren [vgl. 14, S. 644], also eine zuverlässige Erkennung der Sprachbefehle auch offline ermöglichen. Für die Dauer der Entwicklung wird die Spracherkennung mithilfe eines frei zugänglichen Online-Dienstes von Google durchgeführt, da hier der Trainingsaufwand entfällt und die Erkennung schnell und weitestgehend präzise geschieht. Allerdings ist dieser entsprechend

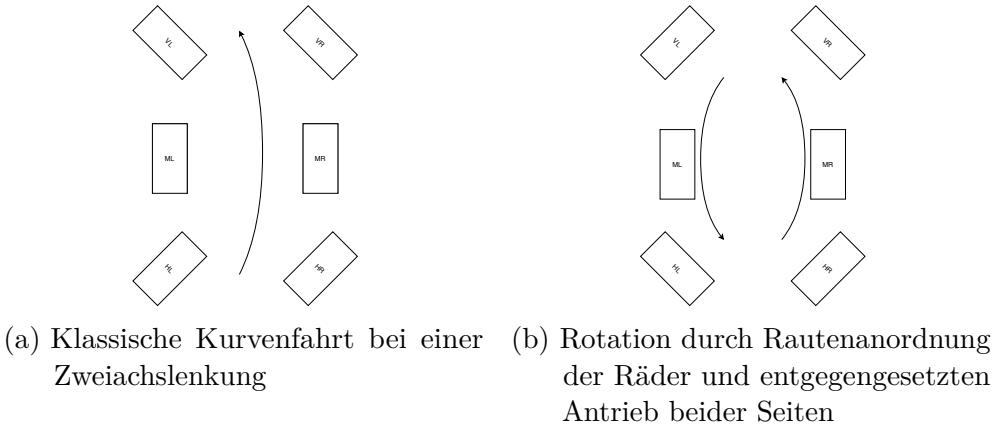


Abbildung 3.1: Varianten einer Linkskurve beziehungsweise -rotation beim genutzten Antriebskonzept

nur erreichbar, wenn eine aktive Internetverbindung mit ausreichend geringer Latenz besteht. Darüber hinaus kann für *PocketSphinx* ein selbst-definierter Satz zu erkennender Worte (zum Beispiel die Steuerbefehle laut Spezifikation) festgelegt werden, wodurch viele Resultate kategorisch ausgeschlossen und die Erkennung beschleunigt werden kann.

Die gesamte Python-Entwicklung ist mithilfe der `setup.py` als zusammenhängendes Paket aufgesetzt worden. So kann innerhalb einer virtuellen Umgebung (hier `venv` genannt) von jedem Modul auf jedes andere zugegriffen werden. Hinweise und Anweisungen zum Setup und der Aktivierung einer virtuellen Python-Umgebung sind in [10] festgehalten.

Bezüglich der Motoransteuerung hat sich gezeigt, dass mindestens 25 % der von der Hardware erlaubten Motorleistung abgerufen werden sollten. Bei Testfahrten sind teilweise Steuersignale bei niedrigerer Leistungsangabe nicht vollständig übertragen worden, vermutlich aufgrund einer zu niedrig angelegten Spannung am BrickPi. Darauf hinaus führt ein zu hohes Gewicht des Rover-Körpers dazu, dass die Motoren mit höherer Leistungsstufe laufen müssen. Ansonsten können sie unter Umständen kein ausreichendes Drehmoment aufbauen, um den Rollwiderstand der Räder zu überwinden.

Mangels Achsendifferentialen wird statt einer üblichen Kurvenfahrt wie in Abbildung 3.1a dargestellt eine Rotation des Rovers auf der Stelle in beide Richtungen implementiert. Abbildung 3.1b zeigt die dafür angestrebte Anordnung der Räder in Winkeln von jeweils  $\pm 45^\circ$  zur Fahrtrichtung und die jeweilige Antriebsrichtung der Räder – beispielsweise werden für eine Linksrotation die drei linken Räder rückwärts und die drei rechten

vorwärts angetrieben. So lässt sich ein Auseinanderdriften der einzelnen Räderpaare (vorn, Mitte, hinten) weitestgehend vermeiden. Trotzdem werden zum Abschluss einer Rotationsbewegung die lenkbaren Räder für eine kurze Strecke um 15° zur Fahrtrichtungsachse gerichtet, da sich dieses Auseinanderdriften auch bei der Rotation nicht vollständig vermeiden lässt.

## 3.1 Installation und Konfiguration Raspberry Pi

Die Steuerung des Mars-Rover-Modells erfolgt zentral über einen Raspberry Pi 3 Model B Plus (Revision 1.3). Auf diesem ist die Debian-basierte Distribution Raspbian for Robots der US-amerikanischen Firma Dexter Industries installiert worden. Dieses Betriebssystem enthält standardmäßig viele vorinstallierte Anwendungen und Treiber für den Einsatz des Raspberry Pi zur Steuerung von Robotern. Zusätzlich sind hier standardmäßig Secure Shell (SSH) und Virtual Network Computing (VNC) Server installiert und aktiviert, sodass über das Netzwerk auch auf einen Pi, der in einem Roboter verbaut ist, zugegriffen werden kann [7, 12]. Insbesondere sind alle benötigten Bibliotheken zur Nutzung der zusätzlichen Platine BrickPi 3 vorinstalliert und -konfiguriert. Der BrickPi ist ebenfalls ein Produkt von Dexter Industries und stellt eine Schnittstelle zwischen den verschiedenen LEGO-Mindstorms-Sensoren und -Motoren zum Raspberry Pi bereit. Er ging 2013 aus einer enorm erfolgreichen Kickstarter-Kampagne hervor [2] und ist seither weiterentwickelt worden, sodass in diesem Projekt der BrickPi 3 (2017) genutzt werden kann. Die Platine fungiert dabei auch als Stromquelle für diese sowie für den Raspberry Pi selbst und muss dazu, ähnlich einer Hardware Attached on Top (HAT), lediglich auf dessen Pins für General Purpose Input/Output (GPIO) aufgesteckt werden. Da jeder BrickPi nur über vier Ausgänge zum Betrieb von EV3-Motoren verfügt, ist der gleichzeitige Einsatz von drei BrickPis nötig, um die zehn Motoren des Modells anzutreiben. Dabei kann jedes der sechs Räder separat durch einen Motor angetrieben sowie die äußeren vier Räder einzeln gelenkt werden. Eine Lenkung der mittleren Räder ist zum aktuellen Zeitpunkt nicht notwendig, kann aber problemlos über die verbleibenden beiden Ports der BrickPis ergänzt werden, beispielsweise um seitliche Fahrten zu ermöglichen.

In Abbildung 3.2 ist die standardmäßige Belegung der drei BrickPis sowie deren Seriennummern dargestellt. Die Spalten A bis D entsprechen den Motorsteuerausgängen

### 3.1 Installation und Konfiguration Raspberry Pi

---

Betrachtungsrichtung ≈ Fahrtrichtung

	A	B	C	D
Antrieb Mitte SN: 45C31FAF514D3937304B2020FF15122B	Mitte links	kein Motor	kein Motor	Mitte rechts
Lenkung SN: 84880DFD514D3937304B2020FF0A172A	hinten links	vorn links	vorn rechts	hinten rechts
Antrieb vorn/hinten SN: 07976FB6515035524E202020FF101B0C				
Raspberry Pi				

Quelle: eigene Darstellung

Abbildung 3.2: Belegung der Steuerungsausgänge zu den einzelnen Motoren an den drei BrickPis des Steuergerätes

MA bis MD des BrickPi 3. Eine Übersicht über dessen verschiedene Anschlüsse im Einzelnen ist der Vollständigkeit halber in Abbildung 3.3 zu sehen. Die Bezeichnungen der Motoren anhand ihrer Richtung (vorn, hinten, links, rechts) sind dabei unter der Voraussetzung, dass der Rover in Fahrtrichtung betrachtet wird, zutreffend.

Um die Stromversorgung sicherzustellen, werden drei Powerbanks (siehe Abbildung 3.4) mit einer Gesamtkapazität von 60,3 Ah und einem Gesamtgewicht von 1251 g im Korpus des Rovers untergebracht. Der Raspberry Pi selbst wird durch die BrickPis über seine GPIO-Pins mit Energie versorgt [4] und benötigt entsprechend keine eigene Stromquelle.

Zu beachten ist, dass beim Einschalten der drei BrickPis über die jeweiligen Schalter einzelne Platinen unter Umständen die Präsenz eines Eingangsstroms nicht feststellen (können). Es hat sich jedoch gezeigt, dass durch wiederholtes Aus- und Wiedereinschalten der BrickPis ein Zustand erreicht werden kann, in dem alle mit Strom versorgt sind. Zu erkennen ist dies an blinkenden, orangefarbenen Status-Leuchtdioden neben den Schaltern.

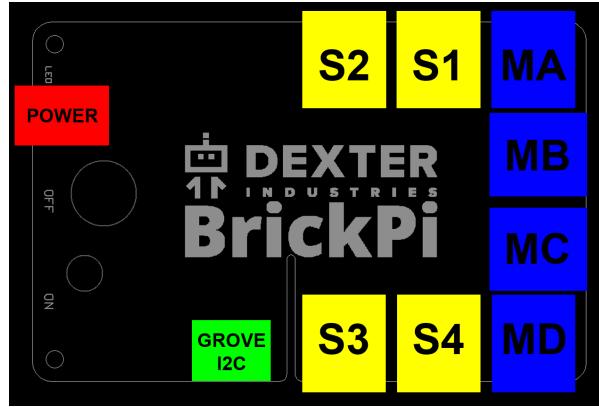
Weiterhin führt ein Leerlauf des Systems gelegentlich dazu, dass sich die bereitgestellten Powerbanks bei vermeintlicher Nicht-



Quelle: eigene

### 3.1 Installation und Konfiguration Raspberry Pi

---



Quelle: [6]

Abbildung 3.3: Belegung der Ein- und Ausgänge eines einzelnen BrickPi 3

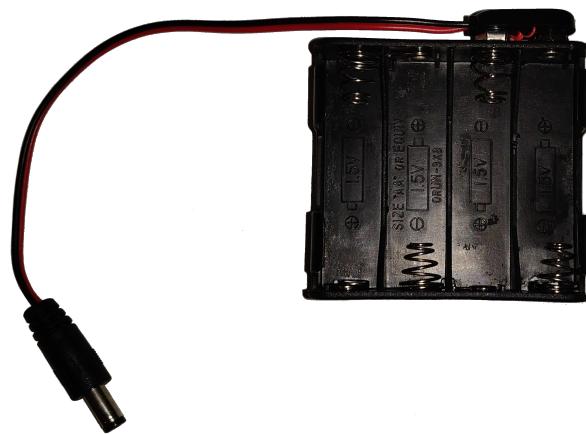
benutzung selbst abschalten. Dadurch ist ein BrickPi nicht mehr mit ausreichender Spannung versorgt, um die Motoren anzutreiben. Das Gesamtsystem bleibt aber durch die anderen beiden Powerbanks mit Strom versorgt. Wird eine der Powerbanks durch eines der Batteriepacks aus dem Lieferumfang des BrickPi (dargestellt in Abbildung 3.5) mit acht AA-Batterien ersetzt, steht jederzeit an allen Platinen ausreichend Spannung und Strom bereit, um die Motoren zu betreiben.

Für die Sprachein- und -ausgabe wird eine USB-Soundkarte angeschlossen, die jeweils eine 3,5-mm-Klinkenbuchse für Lautsprecher und Mikrofon bereitstellt. Die Konfiguration geschieht über die Advanced Linux Sound Architecture (ALSA) Utilities – die unter Raspbian vorinstallierten Soundtreiber für den Raspberry Pi. Neben dem Onboard-Sound über den Klinkenausgang des Raspberry Pi unterstützen diese auch die digitale Ausgabe über High Definition Multimedia Interface (HDMI) sowie die genutzte Soundkarte über USB. [14]

Die Ausgabe geschieht dabei mithilfe eines Bluetooth-Lautsprechers (vertrieben durch die Inspiration GmbH), welcher über einen eigenen Akku verfügt und mittels Klinke an die oben genannte Soundkarte angeschlossen wird. Der genutzte Lautsprecher ist in

### 3.1 Installation und Konfiguration Raspberry Pi

---



Quelle: eigene Aufnahme

Abbildung 3.5: Alternative zur Powerbank: Batteriepack aus dem Lieferumfang des BrickPi



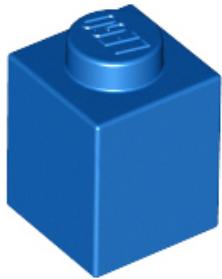
Quelle: eigene Aufnahme

Abbildung 3.6: Für die Sprachausgabe genutzter Lautsprecher (Inspirion GmbH)

Abbildung 3.6 dargestellt.

## 3.2 Installation und Konfiguration Kamera

Für die Videoeingabe wird die originale Raspberry Pi Camera V2 genutzt. Diese nutzt den CMOS-Sensor Sony IMX219 mit einer Auflösung von bis zu 8 Megapixeln [17] und kann mithilfe eines mitgelieferten Flachbandkabels an das Camera Serial Interface (CSI) des Raspberry Pi angeschlossen werden [8]. Die Aufnahmen erfolgen mit einer reduzierten Auflösung von  $1280 \times 720$  Pixeln, um eine Frequenz von  $60\text{ Hz}$  zu ermöglichen. So ist sichergestellt, dass Wasserobjekte auch während der Fahrt gut erkannt werden können. Alternativ kann eine Frequenz von  $90\text{ Hz}$  bei geringerer Auflösung ( $640 \times 480$  oder niedriger) erreicht werden [8]. Allerdings zeigte sich während der Entwicklung, dass bei den oben genannten Parametern die Erkennung am präzisesten funktioniert.



Quelle: <https://www.bricklink.com/v2/catalog/catalogitem.page?P=3005&C=7>

Abbildung 3.7: Blauer LEGO-Stein 3005

Die Objekterkennung für Wasserobjekte erfolgt mithilfe der Programmbibliotheken picamera und OpenCV. Erstere stellt eine Schnittstelle zwischen der Raspberry-Pi-Kamera und Python zur Verfügung [5]. Letztere Bibliothek ermöglicht die Echtzeitauswertung der Kamerabilder [18]: Die Objekte werden auf Basis ihrer blauen Farbe erkannt, da sich diese deutlich von der charakteristischen Farbe der Vorbildlandschaften des Mars unterscheidet. Marslandschaften sind von einer „predominantly yellowish brown color with only subtle variation“ [11, S. 1] und entspricht damit in etwa der Komplementärfarbe zu blau. Die LEGO-Steine, welche als Wasserobjekte fungieren, haben die LEGO-Farb-ID 7. Diese korrespondiert mit dem Farocode #0057A6 in Hexadezimaldarstellung. Ein solcher LEGO-Stein ist in Abbildung 3.7 dargestellt.

Da eine farbwertbasierte Erkennung der Wasserobjekte genutzt wird, entfällt das Trainieren eines Erkennungsalgorithmus, wie es bei einem neuronalen Netz nötig wäre. Die Farbeigenschaften des LEGO-Steins werden in Komponenten für rote, grüne und blaue Farbanteile aufgeteilt und in der Konfiguration hinterlegt. Das Python-Skript berechnet für die RGB-Farbangaben die entsprechenden HSV-Werte (Hue,

```
# colour for object recognition
# blue lego brick: RGBA(0, 87, 166, 1)
TARGET_RED = 0
TARGET_GREEN = 87
TARGET_BLUE = 166
RECOGNITION_TOLERANCE = 10 # plus/minus on hue
MIN_S_V = 100 # min saturation and value
```

Quelltext 3.1: Auszug aus der Konfiguration mit Standardwerten für die farbbasierte Objekterkennung

Saturation, Value). In letzterer Darstellung ist die Information über tatsächliche Farbe nur noch im Hue-Wert (0 bis 255) hinterlegt, sodass für diesen eine bestimmte Abweichung (Standard:  $\pm 10$ ) definiert werden kann. Die Sättigung (Saturation) und der Hellwert (Value) (beide ebenfalls von 0 bis 255 definiert) werden in einem breiten Spektrum toleriert: Standardmäßig lässt das Skript hier Werte von mindestens 100 und höchstens 255 zu. Mithilfe dieser Parameter wird eine solide Erkennung der Präsenz der LEGO-Steine gewährleistet. In Quelltext 3.1 ist ein Auszug aus dem Konfigurationsskript, in dem die oben beschriebenen Werte festgelegt werden, beispielhaft dargestellt.

Auf Basis dieser Farbwerte wird zunächst das Eingangsbild gefiltert, sodass ein Binärbild (Maske) entsteht: Alle mit dem definierten Blauwert erkannten Pixel sind daraufhin weiß (1), die anderen schwarz (0). Anschließend werden alle vorhandenen Konturen auf der Maske bestimmt. So wird sichergestellt, dass keine einzelnen Pixel oder Kleingruppen zu einer Erkennung als Wasserobjekt führen. Einen sinnvollen Rückschluss auf die Anzahl der Wasserobjekte im Bild bietet die Anzahl erkannter Konturen in der aktuellen Implementierung noch nicht. Die angewandten und weiterführende Bildverarbeitungsfunktionen von OpenCV erläutert Pajankar ausführlich in [18].

Die implementierten Skripts ermöglichen eine Live-Anzeige des durch die Raspberry-Pi-Kamera aufgezeichneten Videostreams auf der Raspbian-Benutzeroberfläche. Erkennt der Rover ein Wasserobjekt, so wird dies neben der Sprachausgabe auch in Form eines blauen Schriftzugs auf dem Videobild kommuniziert.

# 4 Nachweis der Leistungsfähigkeit

Der Nachweis der Leistungsfähigkeit entsprechend der angepassten Anforderungen wird durch die folgenden Dokumentationsteile erbracht:

Das BrickLink-Modell wird möglichst originalgetreu gehalten und dokumentiert die Struktur des Rovers. Durch verschiedene Teile, die unter bestimmten Bedingungen (aufgebockt, stehend, fahrend et cetera) unter Spannung stehen und bewegliche Teile kann das digitale Modell nicht exakt die reale LEGO-Struktur abbilden. An einzelnen Stellen sind derartige Teile (häufig beschränkt auf einzelne Pins) ausgelassen worden, um Überschneidungen im digitalen Modell zu vermeiden. Die Teile sind aber in den verschiedenen Fotografien des Rovers (siehe Kapitel 2) zu erkennen.

Durch Videoaufnahmen verschiedener Einsatzszenarien ist die Leistungsfähigkeit der einzelnen Anforderungen nachgewiesen. In einer ersten Aufnahme klassifiziert der Rover stehend Wasserobjekte, die vor ihm auf dem Boden verteilt werden, und es erfolgt eine Sprachausgabe des spezifizierten Textes „Water found“. Eine weitere Aufnahme zeigt den Rover stehend, während er wiederholt Wasserobjekte in einer Hand erkennt, die abwechselnd verdeckt und präsentiert werden. Hierbei wird besonders die Sprachausgabe hörbar, da der Rover aus der Nähe gefilmt wird. Die Fahrtauglichkeit wird durch eine Aufnahme dokumentiert, in welcher dem Rover ein Sprachkommando „Start“ gegeben wird, woraufhin er aus dem Stand losfährt, bis er nach drei Sekunden wieder zum Stehen kommt. Für die Spracheingabe muss zum aktuellen Zeitpunkt ein kabelgebundenes Mikrofon genutzt werden, da die Bestellung eines neuen Mikrofons aufgrund äußerer Einflüsse nicht möglich war. Zuletzt wird die Rotationsfahrt des Rovers in Form zweier Videoaufnahmen dokumentiert. Die Sprachbefehle „Move left“ und „Move right“ initiieren respektive die Links- und Rechtsrotation, die entsprechend der Beschreibung in Kapitel 3 ausgeführt wird. Am Ende seiner Rotation entdeckt der Rover vor ihm liegende Wasserobjekte um kommuniziert die Erkennung per Sprachausgabe. Es wird jeweils einmal die Links- und die Rechtsrotation nach diesem Vorgehen dokumentiert.

Neben dem digitalen Modell und den Videoaufnahmen dient auch der Inhalt dieser Arbeit der ausführlichen Dokumentation des entwickelten Rovers. Die Python-Quelltexte der Steuerungsskripts werden ebenfalls zur Bewertung eingereicht und im Nachgang auf GitHub veröffentlicht. Dabei ist zu beachten, dass bei Änderungen an der Standardkonfigu-

ration die einwandfreie Funktionsfähigkeit der einzelnen Komponenten nicht gewährleistet werden kann. Auch eine ausführliche Bauanleitung für das LEGO-Modell wird mithilfe von BrickLink Studio 2.0.10 erstellt und als Teil der Dokumentation eingereicht.

# 5 Fazit

Im Rahmen dieser Studienarbeit ist ein weitgehend funktionsfähiges und nutzbares Mars-Rover-Modell nach dem realen Vorbild, dem *Curiosity*-Rover der NASA, in Form einer Machbarkeitsstudie (Proof of Concept) von Grund auf entwickelt worden. Das Ergebnis entspricht gänzlich den in Kapitel 1 definierten und im Verlauf der Entwicklung angepassten Anforderungen. Besonderes Augenmerk wurde bereits bei der Konzeption des Rovers auf die originalgetreue und formschöne Abbildung wichtiger Komponenten wie dem Rocker-Bogie-Fahrwerk und der Kamerapositionierung im *Kopf* gelegt. Zudem sind charakteristische Strukturelemente von *Curiosity*, zum Beispiel der stets leicht geneigte Kopf oder der „Bienenstock“ am Heck, in das Design eingeflossen. Dies ist in Kapitel 2 ausführlich in Form von Fotografien des realen und Render-Exporten des digitalen Modells aus verschiedenen Perspektiven dokumentiert. Die Verarbeitung der Eingangsdaten und die Steuerung des Antriebs geschehen zentral auf dem Raspberry Pi unter Nutzung der drei BrickPi-Platinen. Hierfür sind verschiedene Python-Module von Grund auf entwickelt worden. Diese Entwicklung sowie die Zusammenarbeit von Software und Hardware sind gemeinsam in Kapitel 3 erläutert worden. In den Skripts kommen verschiedene Bibliotheken zum Einsatz; sie sind jeweils in den entsprechenden Abschnitten dieses Dokumentes benannt und inklusive der genutzten Versionen in einer separaten Readme-Datei dokumentiert. Abschließend ist die gesamte Funktionsfähigkeit des Rovers nachgewiesen worden. In Kapitel 4 ist festgehalten, wie diese Nachweise erbracht wurden.

Im folgenden Abschnitt sollen abschließend noch einige Möglichkeiten der Weiterentwicklung des Modells in verschiedenen Bereichen aufgezeigt werden.

## 5.1 Ausblick

In diesem Proof of Concept ist bestätigt worden, dass sich ein realitätsnahes Abbild des Mars Rovers *Curiosity* der NASA aus LEGO erstellen und mithilfe von EV3-Motoren betreiben lässt. Das Ergebnis bildet eine solide Grundlage für weitere Entwicklungen beispielsweise bezüglich Schwarmintelligenz, künstlicher Intelligenz und Objekterkennung. Nichtsdestotrotz bietet es einigen Punkten Potential für Verbesserungen.

Insbesondere bei der Kurven- und Geländefahrt ist die Stabilität des LEGO-Modells nicht uneingeschränkt sichergestellt. Hierbei zeigte sich während der Entwicklung unter bestimmten Bedingungen ein Auseinanderdriften der jeweiligen Räderpaare (wie in Kapitel 3 angesprochen) und seltener ein seitliches Kippen einzelner Räder. Es ist zu vermuten, dass dies in weiten Teilen auf die ursprüngliche Unterdimensionierung des Fahrwerks (Entwurf für einen deutlich leichteren Korpus) zurückzuführen ist. Werden die entsprechenden Schwachstellen an den zwei Rotationspunkten jeder Seite des Rocker-Bogie-Fahrwerks sowie den Verbindungen zwischen Lenkmotoren und angetriebenen Rädern verstärkt, so ist eine deutliche Steigerung der seitlichen Stabilität des Fahrwerks zu erwarten. Durch geschickt angebrachte Federung, die Nutzung von Abstandssensoren zur Überwachung des Radstandes oder das Ersetzen einzelner Fahrwerksteile durch stabilere Streben, die sich mithilfe des 3D-Drucks herstellen lassen, kann die Stabilität weiter erhöht werden.

Unter Umständen lässt sich durch die Reduzierung der mitgeführten Powerbanks auf eine einzelne zusätzlich das Gewicht des Korpus deutlich verringern. Zuvor sollte jedoch die Nutzbarkeit eines entsprechenden 1-zu-3-Splitters auch unter länger andauernder Systemlast geprüft werden. Die produktive Nutzung des angebrachten Solarmoduls (siehe Abbildung 2.6) könnte in diesem Fall bei der Nutzung unter freiem Himmel zu einer messbaren Reduzierung der Ladestopps führen. Für eine längere voll-autonome Nutzung ist die Selbstversorgung mit Energie ohnehin unabdingbar.

Bezüglich der Weiterentwicklung im Studiengang Angewandte Informatik sollte der Fokus allerdings auf einen Ausbau der Softwarefunktionalität gelegt werden. Insbesondere im Bereich der Objekterkennung bietet das Modell viel Potential. So kann beispielsweise in nächsten Schritten eine Klassifikation auf Basis künstlicher Intelligenz erfolgen. Der Rover wäre durch diese Technologie in der Lage, neben blauen LEGO-Steinen zum Beispiel auch echte Wasserpfützen und Hindernisse wie Steine und Löcher zu erkennen und zu unterscheiden.

Auch im Bereich der Schwarmintelligenz bieten sich Möglichkeiten der Weiterentwicklung. So lässt sich die Erkundung einer Marslandschaft deutlich verkürzen, wenn diese zeitgleich durch mehrere Rover geschieht, die untereinander kommunizieren und ihre Betrachtungsgebiete aufteilen.

# Literaturverzeichnis

- [1] R. Arvidson et al. „Mars Science Laboratory Curiosity Rover Terramechanics Initial Results“. In: *Proceedings of the 44th Lunar and Planetary Science Conference*. The Woodlands, Texas, 03/2013 (siehe S. 3).
- [2] R. Barnes, Hrsg. *The Official Raspberry Pi Projects Book*. London, UK: Upton, Liz, 2015 (siehe S. 16).
- [3] D. Bickler. „Roving on Mars“. In: *Mechanical Engineering* (04/1998), S. 74–77 (siehe S. 3).
- [4] J. Cole. „Building bots with LEGO“. In: *The MagPi* 17 (10/2013), S. 4–7 (siehe S. 17).
- [5] T. Cox. *Raspberry Pi Cookbook for Python Programmers*. Birmingham, UK: Packt Publishing, 04/2014. ISBN: 978-1-84969-662-3 (siehe S. 20).
- [6] Dexter Industries. *BrickPi3 Getting Started Step 3 – Build your BrickPi3 Robot*. 2017. URL: <https://www.dexterindustries.com/BrickPi/brickpi3-getting-started-step-3-build-robot-attach-lego-mindstorms-nxt-ev3-sensors-motors-technic-parts/> (besucht am 25.04.2020) (siehe S. 18).
- [7] W. Donat. *Learn Raspberry Pi Programming with Python. Learn to Program on the World’s Most Popular Tiny Computer*. 2. Aufl. Palmdale, CA: Apress Media, 2018. ISBN: 978-1-4842-3769-4 (siehe S. 14, 16).
- [8] G. Halfacree. *The Official Raspberry Pi Beginner’s Guide. How to use your new computer*. 2. Aufl. Cambridge, UK: Raspberry Pi Press, 2019. ISBN: 978-1-912047-62-8 (siehe S. 20).
- [9] B. Horan. *Practical Raspberry Pi*. Kalifornien: Apress Media, 2013. ISBN: 978-1-4302-4972-6.

- [10] H. P. Lantangen und A. Johansen. *Creating Virtual Python Software Environments with Virtualenv*. 11.04.2016 (siehe S. 15).
- [11] J. N. Maki, J. J. Lorre, P. H. Smith, R. D. Brandt und D. J. Steinwand. „The color of Mars: Spectrophotometric measurements at the Pathfinder landing site“. In: *Journal of Geophysical Research* 104.E4 (25.04.1999), S. 8781–8794 (siehe S. 20).
- [12] S. McManus und M. Cook. *Raspberry Pi for Dummies*. 3. Aufl. Hoboken, NJ: John Wiley & Sons, Inc., 2017. ISBN: 978-1-119-41200-7 (siehe S. 16).
- [13] C. Müller. *Studienarbeit: Spezifikation*. Version final. 12.10.2019 (siehe S. 3).
- [14] D. Molloy. *Exploring Raspberry Pi. Interfacing to the Real World with Embedded Linux*. Indianapolis, IN: John Wiley & Sons, Inc., 2016. ISBN: 978-1-119-18868-1 (siehe S. 14, 18).
- [15] NASA. *Where is Curiosity?* 2020. URL: [https://mars.nasa.gov/msl/mission/where-is-the-rover/?page=0&per\\_page=25&order=sol+desc&search=&category=176%3A295&url\\_suffix=%3Fsite%3Dmsl](https://mars.nasa.gov/msl/mission/where-is-the-rover/?page=0&per_page=25&order=sol+desc&search=&category=176%3A295&url_suffix=%3Fsite%3Dmsl) (besucht am 25.03.2020) (siehe S. 3).
- [16] NASA JPL. *Mars Science Laboratory: Curiosity Rover*. Präsentation. NASA, 2013 (siehe S. 3).
- [17] M. Pagnutti et al. „Laying the foundation to use Raspberry Pi 3 V2 camera module imagery for scientific and engineering purposes“. In: *Journal of Electronic Imaging* 26.1 (11.02.2017). DOI: 10.1117/1.JEI.26.1.013014 (siehe S. 20).
- [18] A. Pajankar. *Raspberry Pi Computer Vision Programming*. Birmingham, UK: Packt Publishing, 05/2015. ISBN: 978-1-78439-828-6 (siehe S. 20 f.).
- [19] A. R. Vasavada et al. „Overview of the Mars Science Laboratory mission: Bradbury Landing to Yellowknife Bay and beyond“. In: *Journal of Geophysical Research: Planets* 119 (05.06.2014), S. 1134–1161. DOI: 10.1002/2014JE004622 (siehe S. 3).

- [20] S. Yamanoor und S. Yamanoor. *Python Programming with Raspberry Pi*. Birmingham, UK: Packt Publishing, 04/2017. ISBN: 978-1-78646-757-7 (siehe S. 14).

# **Anhang**