

# *Entwicklung mithilfe von MicroPython for EV3*

## *Der HotRod (und sein kleiner Freund)*

*Raphaele Salvatore Licciardo - 60559*

*Projektarbeit lego::lab*
*Wintersemester 2019/2020*

*Hr. Prof. Dr. rer. pol. Uwe Haneke*
*Hr. M. Sc. Matthias Mruzek-Vering*

# I. Einleitung

Ein HotRod, ist ein speziell modifiziertes Fahrzeug aus des 1920er bis 1940er Jahren. Aufgrund der klar definierten Stilrichtung eines HotRod's war der Ford Modell T bzw. der Ford Modell A die Ideale Basis hierfür.
Heutzutage nutzt man verschiedene Karosserie Bauarten, um diesen Look nachzuempfinden, z. B. wird das Dach gechoppt (d. h. die Dachsäulen werden verkürzt, somit liegt das Dach tiefer).
Innerhalb dieser Projektarbeit wird versucht, mithilfe von Lego ein solches Fahrzeug nachzubauen. Doch damit nicht genug. Im 21. Jahrhundert können Fahrzeuge Autonom fahren. So auch der HotRod. Der HotRod ist ein klassisches ferngesteuertes Fahrzeug, welches mit einer kleinen Fernbedienung gesteuert werden kann. Das zweite Fahrzeug dient dazu, moderne autonome Fahrzeuge zu imitieren. Mit einer Follow-Me Funktion oder einem modernen Bremssystem kann dieser selbstständig ein Zielfahrzeug verfolgen  oder sogar vor Hindernisse Abbremsen.

## I. Motivation

Als Voraussetzung für die Hardware (den klassischen HotRod Look) bedient man sich dem EV3 LEGO MINDSTORM EDUCATION Packet. Dieser Baukasten beinhaltet alle notwendigen Bausteine um diesen Look zu erstellen. Die Softwareseitige Lösung wird mittels Python umgesetzt. Lego bietet hierbei eine solide Schnittstelle zwischen Hardwarebausteinen, wie z. B. Sensoren oder auch Motoren, und dem Python Quellcode.

## II. Zielsetzung

Das Ziel dieser Projektarbeit besteht darin, ein Projekt alleine zu analysieren, Planen und zu Implementieren bzw. zu Bauen. Der HotRod selbst, soll mit einer Fernbedienung fahren können oder von selbst einem definierten Ziel hinterherfahren. Um es zu verkürzen, er soll alle Funktionen von einem modernen selbstfahrenden Fahrzeug beinhalten und problemlos umsetzen können.

# II. Planung

Die Planung eines Projektes ist der wichtigste Punkt, ohne diesen ist das Projekt zum Scheitern verurteilt. Während der Analyse werden unter anderem, mögliche Probleme herausgearbeitet und ein Design angestrebt. Eine Problemstelle des HotRod's kann z. B. die Lenkung sein. Ein Fahrzeug das Moderne Fahrzeuge imitieren soll braucht auch eine Moderne Lenkung. Doch wie baut man etwas Dynamisches und Flexibles mit starren Legobausteinen? Doch dazu im folgenden mehr.

## I. Grundsätzlicher Aufbau

Das Fahrgestell des HotRod's dient als Hauptankerpunkt für alle weiteren Bausteine. Somit muss dieses Robust und stabil genug für Gewicht und Aufprall. Dennoch muss dieses auch noch genug Platz für Sensoren, Motoren und Mini - Computer haben. Daher wurde das Fahrgestell zweigeteilt. Der vordere Teil des Fahrgestelles ist für

- Stoßstange bzw. Kühlergrill
- Lenkungsmotor
- Lenkachse
- Sensoren

zuständig. Der hintere Teil des Fahrgestelles ist für

- Lenkungsmotor
- Antriebsmotor
- Antriebsachse
- EV3 Stein

zuständig. Das Lastgewicht ist somit gut verteilt, bei einem Aufprall werden die wichtige Bauteile des HotRod's geschützt. Die Stoßstange fängt den Aufprall ab und leitet diesen nach unten zu den Rädern ab. Die Motoren oder der Mini - Computer bekommen von dem Aufprall nichts mit.

## II. Lenkung (-motor)

Normale Roboter haben eine einfache Lenkung. Es gibt zwei Motoren, einer ist für den linken Antriebsstrang, der andere für den rechten Antriebsstrang zuständig. Möchte man eine Linkskurve fahren, muss der rechte Motor schneller als der linke Motor arbeiten. Bei einer Rechtskurve analog. Doch Moderne Fahrzeuge arbeiten so nicht, diese haben einen gemeinsamen Motor für einzigen Antriebsstrang. Vorderräder können sich unabhängig von dem Fahrgestell bewegen, somit kann Rechts- oder Links eingelenkt werden. Mit Legobauteilen ist diese Funktion nicht trivial umsetzbar. Deswegen bedient man sich einem Lenkmotor, welcher mit Zahnräder und Kreuzachsen mit der Lenkachse verbunden ist. Dreht sich der Motor nach Links, so drehen sich die Reifen bzw. die Lenkachse nach Links. Dies gilt Analog für eine Rechtskurve. Wie bereits in beschrieben, ist bei dem HotRod eine andere Art von Lenkung gefragt. Deswegen wird ein extra Motor herangezogen. Dieser dient dazu, die Lenkachse in die gewünschte Richtung zu drehen. Dies wird mit einem relativ komplexen Zahnrad Konstrukt erreicht. Somit ist die Drehrichtung des Lenkmotors, identisch mit der Einschlag Richtung der Vorderräder.

## III. Antrieb (-smotor)

Ein Rennauto wie der HotRod muss in der Lage sein, schnell hohe Geschwindigkeiten aufzubauen und schnelle Kurven durchführen zu können. Hohe Geschwindigkeiten werden mit einem entsprechenden Motor umgesetzt, welcher in der Software gedrosselt oder entdrosselt werden kann.
Die Schnellen Kurven werden ebenso bei dem Antriebsmotor und nicht bei der Lenkungspatie erstellt. Um diesen Effekt zu erreichen, wird der Motor nicht direkt mit der Lenkachse verbunden, sondern über ein Differenzialgetriebe. Dieser Effekt sorgt für die gewünschten schnellen Kurven. Jedoch verzichtet man hierbei auf einen kompletten Hinterradantrieb, da nur ein Rad mit dem Motor verbunden ist. Um etwas schneller zu werden kann man auf den Effekt des Differenzialgetriebes verzichtet, indem man ein Zahnrad herausnimmt.

# III. Umsetzung

Entsprechen der Funktionen die der HotRod erfüllen soll, sind diverse Meilensteine notwendig. Im Folgenden werden diese aufgeführt und erklärt.

## I. Lenkung

Um ein voll funktionsfähiges und modernes Fahrzeug zu simulieren, sind nicht nur 4 Räder und ein Motor notwendig. Die Lenkung ist wie folgt aufgebaut. Der Medium Motor ist über vier Zahnräder mit der Lenkachse verbunden. Hierbei dienen zwei Zahnräder dazu, dass der Motor die Zahnräder nicht kaputt machen kann. Hat der Code sich zum Beispiel aufgehangen o. ä. Und dreht dauerhaft durch, so verhindern diese Zahnräder größeren schaden. Ein weiters Zahnrad ist mit einer Kreuzachse mit der Lenkachse verbunden. Dreht sich der Medium Motor nach Links, so lenkt auch die Lenkachse nach Links ein. Da sich die „Anti - Blockier“ Zahnräder direkt in front des HotRod’s befinden, wurde ein Kühlergrill angebracht. Bei einem Aufprall, verhindert dieser größeren Schaden und verstärkt den HotRod Look.

## II. Antrieb

Da die Lenkung unabhängig von dem Antrieb umgesetzt worden ist, konnte auf unabhängige Antriebe der Hinterräder verzichtet werden. Der Large Motor, treibt über (insgesamt) vier Zahnräder, zwei pro Seite, die Hinterachse an. Um scharfe und schnelle Kurven fahren zu können, besitzt der HotRod ein Differenzial Getriebe. Dieses besitzt drei weitere Zahnräder und ein spezielles Bauteil. Damit werden die Räder gleichmäßig angetrieben. In Kurven dreht somit das äußere Rad schneller als das innere Rad, der HotRod hat mehr Stabilität und fährt deutlich schneller wie im Vergleich zu anderen Robotern.

## III. Fernsteuerung

Die Fernsteuerung sendet per Infrarot Strahlen, Integer Zahlen. Ein spezieller Infrarot Sucher empfängt diese und verarbeitet diese Werte im Python Quellcode. Hierbei stehen die folgenden Werte für die folgende Fahraktion:

- 2 = Bremsen oder Rückwärtsfahren
- 8 = Rechts einlenken
- 128 = Vorwärtsfahren
- 512 = Links einlenken

## IV. Bremsassistent

Um den HotRod vor möglichen Schäden zu bewahren und um erneut eine Funktion von modernen Fahrzeugen zu simulieren, besitzt der HotRod ein Bremsassistent. Der Infrarot Sucher, der mit der Fernsteuerung (einseitig) kommuniziert, scannt den Abstand um sich herum. Diese Python Funktion liefert ebenso einen Integer zurück der sich zwischen 0 und 100 befindet. 0 steht hierbei für Crash und 100 für freie Fahrt. Sollte der Hot Rot einen Wert von 50 messen, bremst dieser Automatisch ab und liefert dem Benutzer visuelle und akustische Warnsymbole.

## V. Zahlen und Fakten

Nun ein paar Zahlen und Fakten des HotRod’s. Je nach Position des Senders und Empfängers kann es hierbei zu unterschiedlichen Latenzzeiten kommen. Ist der Sender direkt vor dem Empfänger ist die Latenzzeit auf ein Optimum reduziert. Befindet sich dieser jedoch nicht in dem direkten Suchfeld, kann es zu kleineren Verzögerungen kommen. Dies kann auftreten wenn, der Sender unmittelbar über dem Empfänger positioniert ist oder der Sender sich hinter bzw. neben dem Empfänger aufhält. Sind Gegenstände zwischen Sender und Empfänger, kann dies ebenso einen erheblichen Zeitverlust ergeben. Doch auch die Software ist hierfür nicht optimal ausgelegt. Der HotRod befindet sich in einer Endlosschleife. Direkt zu anfangen werden alle notwendigen Werte

ausgemessen und erst am Ende der Schleife wird darauf reagiert. Folgendes Szenario, Der HotRod misst die Signale und erkennt „Rechts einlenken“. Darauf hin reagiert dieser mit einem entsprechenden Kommando an den Lenkmotor. Doch was, wenn das Kommando an den Empfänger, kurz nach dem Messen und Empfangen ankam und was würde passieren, wenn das Kommando nach dem Einlenken ankommt. Alles in allem ist somit eine optimale Messung der Latenz bzw. Reaktionszeit nicht möglich. Zudem ist anzumerken, dass dieser Effekt eine starke Auswirkung auf den Bremsassistenten hat. Ist die Position des HotRods zu einem Gegenstand optimal, so wird weiter gefahren. In dem Szenario das der Abstand optimal ist, aber dieser kurz nach dem Messer es nicht mehr ist, kann der HotRod nicht mehr reagieren und es entsteht ein Crash. Dies könnte jedoch mit diversen Softwareseitigen Verfahren optimieren, welche in dem HotRod jedoch nicht verbaut worden sind.

# IV. Veränderungen

Während der Umsetzung eines Projektes fallen dem Entwickler meistens neue Ideen und Änderungsmöglichkeiten ein. So wurde auch der HotRod um einige Features erweitert. Schnell wurde aus einem Ferngesteuerten Fahrzeug, ein hochmodernes (selbstfahrendes) Fahrzeug. Mit diesem kann man nun moderne Fahrzeuge wie der Tesla Model X oder gar den Cybertruck verstehen und nachvollziehen.

## I. Fernsteuerung

Die Infrarot Fernsteuerung von dem Lego Mindstorm Set hat zwar alle notwendigen Funktionen ist jedoch etwas klein und nicht gerade Benutzerfreundlich. Eine hochmoderne Fernsteuerung von aktuellen Spielekonsolen wäre deutlich angenehmer, z. B. der von einer X-Box. Um einen solchen mit dem Python Quellcode zu verbinden, muss eine stabile Bluetooth Verbindung aufgebaut werden. Python ist hierbei ziemlich benutzerfreundlich, denn diese Programmiersprache unterstütz t die Entwicklung von Library von dritten, diese können dann in den Python Packetmanager (z. B. pip) hochgeladen werden. Jedoch werden diese meistens für die drei großen Betriebssysteme hergestellt, Microsoft Windows, MacOS und Linux. Der EV3 Stein hat jedoch keinen dieser drei Betriebssysteme und kann somit keine Python Library verwenden. Lediglich die von Lego vorgesehene Library *pybricks* kann problemlos verwendet werden. Damit ist die Idee von einem X-Box Controller leider beendet.

## II. Ampelerkennung

Moderne Fahrzeuge können selbstständig Ampeln erkennen, diese aktuelle Phase scannen und auf diese dementsprechend reagieren. So auch der HotRod. Mit einem Color Scanner können Farben eingelesen werden, je nach Phase reagiert der HotRod entsprechend. Bei Rot kann dieser zum Beispiel stehen bleiben.

## III. Follow-me

Was stellt man sich heutzutage unter modernen Fahrzeugen vor? Schnell kommen Begriffe wie Tesla und selbstfahrende Fahrzeuge, so auch der HotRod. Ein zweites Fahrzeug, kann mit diversen Sensoren, die im Folgenden erklärt werden, dem HotRod folgen. Ganz autonom fährt das zweite Fahrzeug dem ersten hinterher. Der Pseudo Code könnte wie folgt aussehen:

```
Scanne den Abstand nach vorne
abstand > 0:
        Gebe Gas
Scanne den Abstand nach links
abstand > 0:
        Lenke links ein
Scanne den Abstand nach rechts
abstand > 0:
        Lenke rechts ein
```

Die Idee hierbei war folgende, der HotRod ist mit drei Infrarot Sensoren ausgestattet. Nach vorne, links und rechts, weitere waren aufgrund der begrenzten Anzahl von Steckplätzen nicht möglich. Diese Scannen deren Umgebung nach Objekten. Sollte einer der Sensoren ein Objekt entdecken, so fährt das vorausfahrende Fahrzeug wohl in diese Richtung. Der Sensor hat idealerweise den HotRod gescannt. Doch was bei anderen Objekten, wie einem Stuhlbein o. ä. verliert dieser schnell sein eigentliches Ziel. Weitere Funktionen, Sensoren oder Möglichkeiten sind mit Python leider nicht möglich, daher verwendet das zweite Fahrzeug Java als Programmiersprache. Java arbeitet mit einer Virtuellen Maschine, somit ist das eigentliche Betriebssystem nicht wichtig. Library können problemlos verwendet werden. Daher befindet sich auf dem HotRod nun ein Infrarot Ball, dieser Sendet wie eine Bombe in alle möglichen Richtungen Infrarot strahlen. Ein Sensor von Hi Technik, der sich auf dem Autonomen Fahrzeug befindet, scannt ebenso Infrarot strahlen, doch

dieser merkt sich den Eintrittswinkel. Mit Hilfe dem Eintrittswinkel kann das Fahrzeug optimal einlenken. Der Pseudo Code hiervon sieht wie folgt aus

```
Scanne die Umgebung nach Infrarot Strahlen
eintrittswinkel == 0:
        Gebe Gas
eintrittswinkel > 0:
        Lenke links ein
eintrittswinkel < 0:
        Lenke rechts ein
```

Da die Follow - me Funktion schnell Ausmaße einer eigenen Projektarbeit annahm, baut die Follow - me nicht auf Scannen der Umwelt, sondern auf eine Car to Car Kommunikation auf. Das bedeutet, dass ein modernes autonomes Fahrzeug zwar nicht umsetzbar war, jedoch nur aufgrund fehlender Steckplätze. Mit weiteren Steckplätze wäre die erste Methode erfolgreich gewesen. Ein Port Multiplexer war jedoch ebenso nicht Erfolgreich. Die Option das der HotRod dem hinterfahrenden Fahrzeug seine Geschwindigkeit und Einschlagwinkel übermittel war mit Python aufgrund den bereits genannten Problem ebenso nicht machbar.

# V. Programmiersprachen

Um ein optimales Ergebnis zu erreichen, laufen die Fahrzeuge nicht nur mit Python (Aufgabenstellung), sondern ebenso mit Java. Im zweiten Fahrzeug wurde wegen bereits erwähnten Gründen auf Python verzichtet. Hierbei wurde, um Arbeitszeit zu sparen, sich an dem Soccer Bot 2 orientiert.

## I. Python

Um mit Python ein EV3 Stein erfolgreich zu programmieren, sind diverse Schritte notwendig. Diese sowie die Vor- und Nachteile werden im folgenden Abschnitt beschrieben.

- Visual Studio Code installieren
- EV3 MicroPython Extension installieren
- Balena Etcher installieren
- Micro SD Karte (maximal 32 GB)
- EV3 MicroPython Image herunterladen
- EV3 MicroPython Image mithilfe von Balena Etcher auf die Micro SD Karte flashen.

Waren diese Schritte erfolgreich, kann man mit der Programmierung beginnen. Dazu muss die nun präparierte Micro SD-Karte in den EV3 Stein gesteckt werden, dieser sollte gestartet werden. Um ein Programm zu startet, sollte den *File System* zu der Python Datei navigiert werden, selektiert und bestätigt. Nun startet das Programm. Damit genau dieses beendet wird, wird die *zurück* Taste (oben Rechts - unabhängig von dem Steuerkreuz) betätigt werden. Die MicroPython Dokumentation und Beispiel Programme befinden sich in der dafür vorgesehenen MarkDown Datei.

## II. Java

Aus den oben genannten gründen musste das zweite Fahrzeug auf eine Python Programmierung verzichten. Somit wurde es mit Java (Lejos) programmiert. Die Vorbereitung der IDE und des SD Karte erfolgt ähnlich zu Python. Eine SD Karte muss mit dem Java bzw. Lejos SD-Karten Image beschrieben werden, auch hier kann Balena Etcher verwendet werden. In Python wird der EV3 Stein mittels einem Datenkabel mit der IDE verbunden. Java bzw. Lejos setzt hierbei auf eine Drahtlose Verbindungsmöglichkeit, angenehmer da es Kabelfrei ist, jedoch ist der Bluetooth Verbindungsaufbau etwas komplizierter.

# VI. Meeting

Während einer Projektarbeit werden Meetings abgehalten. Diese dienen dazu den aktuellen Fortschritt zu kontrollieren und ggf. Verbesserungen o. ä. Zu besprechen. Im folgenden werden diese Meetings festgehalten. Im Folgenden Abschnitt werden diese Meetings zusammengefasst und aufgeführt.

## I. Eröffnungsmeeting

| | |
|---|---|
| Anwesend: | Matthias Mruzek-Vering<br>Raphaele Salvatore Licciardo |
| Örtlichkeit: | lego::lab |
| Datum: | Dienstag den 08.10.2019 um 14:00 Uhr |
| Thema: | Einführung in das Lego Labor |
| Inhalt: | Ideenfindung, was für ein Lego Roboter werde ich bauen. Zur Auswahl Stande ein Kran und ein Rennauto. Entschieden für Rennauto. Dieser Roboter soll mit Python Programmiert werden, daher soll ich auch die *MicroPython for EV3* Dokumentation selbst noch einmal als Markdown Dokument formulieren. |

## II. Meeting 1

| | |
|---|---|
| Anwesend: | Hr. M. Sc. Matthias Mruzek-Vering,<br>Hr. Prof. Dr. rer. pol. Uwe Haneke,<br>Raphaele Salvatore Licciardo |
| Örtlichkeit: | lego::lab |
| Datum: | Donnerstag den 31.10.2019 um 14:00 Uhr |
| Thema: | Meilenstein Meeting Nummer 1 |
| Inhalt: | Das Fahrzeug ist ziemlich instabil und muss verbessert werden. Da die Lenkung ins unendliche einlenkt ist eine kleine Korrektur der Fahrtrichtung nicht möglich, entweder ganz einlenken oder garnicht. Auch dies ist zu verbessern. Neue Funktionen des Fahrzeuges könnte eine Follow-me Funktion sein, d. h. mehrere Fahrzeuge fahren hintereinander her, nur das erste wird mit der Fernsteuerung gesteuert. Ebenso soll das Fahrzeug mit einem X-Box Controller gesteuert werden sowie eine optionale Ampelerkennung. |
| Neue Ziele: | Lenkung verbessern, Stabilität verbessern, Follow-me Funktion, X-Box Controller als Fernbedienung, Ampelerkennung. |

## III. Meeting 2

| | |
|---|---|
| Anwesend: | Hr. M. Sc. Matthias Mruzek-Vering,<br>Hr. Prof. Dr. rer. pol. Uwe Haneke,<br>Raphaele Salvatore Licciardo |
| Örtlichkeit: | lego::lab |
| Datum: | Dienstag den 03.12.2019 um 11:30 Uhr |
| Thema: | Meilenstein Meeting Nummer 2 |
| Inhalt: | Das Hauptaugenmerk war deutlich die Follow-me Funktion da der Rest Problemlos funktioniert hat. Die Follow-me hat in scharfen Kurven jedoch Aussetzer, da die Sensoren das Fahrzeug verlieren. Daher hat man sich auf eine Car to Car Kommunikation geeinigt. Da Python keine zusätzliche Library auf dem EV3 Stein unterstütz hat man sich ebenso auf die Programmiersprache Java geeignet, da diese eine VM verwendet, somit ist das darunter liegende BS zweitrangig. Da die Follow-me Funktion Ausmaße einer zweiten Projektarbeit hat, durfte diese auf dem Soccer Bot 2 aufbauen. Somit folgt das Fahrzeug dem Ball, welcher sich bei dem Fahrzeug befindet. Deswegen wird ein drittes Meeting stattfindet, um sich über diesen Fortschritt zu bereden bevor abgegeben wird. |
| Neue Ziele: | Follow-me Funktion mittels Car to Car Kommunikation simulieren. |

## IV. Meeting 3

| | |
|---|---|
| Anwesend: | Hr. M. Sc. Matthias Mruzek-Vering,<br>Hr. Prof. Dr. rer. pol. Uwe Haneke,<br>Raphaele Salvatore Licciardo |
| Datum: | Dienstag den 07.01.2020 11:30 Uhr |
| Thema: | Meilenstein Meeting Nummer 3 |
| Inhalt: | Die Follow-me Funktion ist gut gelungen jedoch fährt der kleine Freund noch dem HotRod auf. Dies muss verbessert werden, idealerweise ohne einen extra Sensor. der Rückfederungseffekt der Lenkung wurde angesprochen, doch dieser ist Hardware- und Softwareseitig (mit der aktuellen Bauart) nicht umsetzbar. Da der HotRod nun wieder freie Steckplätze hat, soll dieser wieder die Ampelerkennung verbaut bekommen. Sieht dieser ein Hindernis in bestimmter Farbe, soll er einfach stehen bleiben. |
| Neue Ziele: | Ampelerkennung einbauen, Follow-me Funktion darf nicht auffahren. |

## V. Abschlussmeeting

| | |
|---|---|
| Anwesend: | Hr. M. Sc. Matthias Mruzek-Vering,<br>Hr. Prof. Dr. rer. pol. Uwe Haneke,<br>Raphaele Salvatore Licciardo |
| Datum: | Todo |
| Thema: | Curriculum |
| Inhalt: | Todo |

# VII. Bauteile

Farben werden bei der Angabe der Teile nicht beachtet. Ebenso kann es zu Abweichungen kommen, da erst nach des Bauens gezählt worden ist. Dies ist zu entschuldigen.

## I. HotRod - Leader

| Name | Nummer | Anzahl |
|---|---|---|
| Tire Low Wide | 6035364 | 6 |
| Rim Wide | 4634091 | 6 |
| Technic 3M Beam | 4211655 | 13 |
| Technic 5M Beam | 4210686 | 5 |
| Technic 7M Beam | 4494931 | 8 |
| Technic 9M Beam | 4655730 | 6 |
| Technic 11M Beam | 4603472 | 4 |
| Technic 13M Beam | 4522933 | 4 |
| Technic 15M Beam | 4542576 | 8 |
| Technic Ang. Beam 4x2 90 Deg. | 4141270 | 2 |
| Double Cross Block | 4121667 | 2 |
| Cross Axle 3M | 4211815 | 1 |
| Cross Axle 5M | 4211639 | 3 |
| Cross Axle 9M | 4535768 | 2 |
| Conical Wheel Z12 | 4565452 | 4 |
| Gear Wheel Z24 | 4514558 | 3 |
| 24 Tooth Clutch Gears | | 2 |
| Double Conical Wheel Z36 | 4255563 | 1 |
| Beam Frame | 4539880 | 1 |
| Bream H. Frame | 4540797 | 1 |
| * Connector Peg W. Friction | 4122715 | 44 |
| * Conn. Bush W. Fric / Crossable | 4206482 | 2 |
| * Connector Peg. W. Friction 3M | 4514553 | 20 |
| Cross Block 90 Deg. | 4188298 | 2 |
| Double Angular Beam 3x7 45 Deg | 4210668 | 2 |
| Bushing, 1/2M | 4239601 | 4 |

| Name | Nummer | Anzahl |
|---|---|---|
| Bushing, 1M | 4211622 | 5 |
| Differenzial Getriebe - Gehäuse | | 1 |
| Axle with stop, 3M | 6135494 | 2 |
| T-Beam, 3x2 | 4552347 | 1 |
| P-Brick | 6009996 | 1 |
| Cable | 6024581 | 3 |
| Infrared Sensor | 6132629 | 1 |
| Medium Motor | 6008577 | 1 |
| Large Motor | 6148278 | 1 |
| IR Beacon / Remote | 6127283 | 1 |
| Technic Beam 1x11.5 DoubleBent | 4120668 | 2 |
| Technic Pin Connector Round | 6245501 | 2 |
| **Total** | | **173** |

* Kann zu Abweichungen kommen

## II. Kleiner Freund - Follower

| Name | Nummer | Anzahl |
|---|---|---|
| Large Motor | 6148278 | 2 |
| Hi Technic IR Seeker | | 1 |
| Hi Technic IR Ball | IRB1005 | 1 |
| P-Brick | 6009996 | 1 |
| Cable | 6024581 | 5 |
| Technic 3M Beam | 4211655 | 2 |
| Technic 5M Beam | 4210686 | 4 |
| Technic 7M Beam | 4494931 | 6 |
| Technic 9M Beam | 4655730 | 2 |
| Technic 11M Beam | 4296265 | 2 |
| Technic 13M Beam | 4522933 | 5 |
| Technic Ang. Beam 5x3 90 Deg. | 4211713 | 2 |
| Beam Frame | 4539880 | 1 |
| Bream H. Frame | 4540797 | 1 |

| Name | Nummer | Anzahl |
|---|---|---|
| Ball bearing | 4610380 | 1 |
| Steel Ball | 6023956 | 1 |
| * Connector Peg W. Friction | 4122715 | 22 |
| * Conn. Bush W. Fric / Crossable | 4206482 | 40 |
| * Connector Peg. W. Friction 3M | 4514553 | 11 |
| Gear Wheel Z24 | 4514558 | 4 |
| Cross Axle 9M | 4535768 | 2 |
| Cross Axle 11M | 6130012 | 2 |
| Bushing, 1M | 4211622 | 6 |
| **Total** | | **124** |

* Kann zu Abweichungen kommen

## III. Anhänger

| Name | Nummer | Anzahl |
|---|---|---|
| Technic 3M Beam | 4211655 | 1 |
| Technic 5M Beam | 4210686 | 2 |
| Technic 7M Beam | 4494931 | 2 |
| Bushing, 1M | 4211622 | 2 |
| Technic Axle and Pin Connector | 6536199 | 1 |
| Cross Axle 6M | 600125 | 1 |
| Technic 9M Beam | 4655730 | 10 |
| Technic 15M Beam | 4542576 | 22 |
| Double Angular Beam 3x7 45 Deg | 4210668 | 2 |
| * Conn. Bush W. Fric / Crossable | 4206482 | 25 |
| Connector Peg. W. Friction 3M | 4514553 | 4 |
| Connector Peg W. Friction | 4122715 | 12 |
| Bushing, 1M | 4211622 | 3 |
| Große Reifen von Hr. Gniot | | 2 |
| Cross Axle 9M | 4535768 | 2 |
| Tube, 2M | 4526985 | 2 |
| **Total** | | **84** |

* Kann zu Abweichungen kommen

# VIII.Fazit

Der HotRod hat die Zielsetzung eines ferngesteuerten Fahrzeuges zu 90% erfüllt. Aufgrund der Fernsteuerung Problematik kann dieser entweder Rückwärts fahren oder Bremsen. Wäre eine andere Fernsteuerung verbaut, die mind. fünf verschiedene Modi bzw. Knöpfe unterstütz, würde der HotRod 100% erfüllen. Zum aktuellen Zeitpunkt kann dieser Hindernisse erkennen und dementsprechend reagieren. Eine Identische Funktion erfolgt bei Farben. Der kleine Freund erfüllt die Zielsetzung nur zu 70%, weil dieser nicht dem HotRod folgt, sondern nur dem Ball, der sich auf dem HotRod befindet. Hierbei gibt es einen idealen Lösungsansatz der problemlos arbeiten würde, jedoch reichen hierfür die Steckplätze nicht aus. Hierfür würde man zwei Farbsensoren verbauen, welche die Strecke vermessen, der kleine Freund darf erst eine Kurve fahren, wenn es die Streckenmarkierungen auch zulassen (Verweis auf den Linienfolger - baubar). Nun würde der kleine Freund 3 Infrarot Sensoren bekommen, für den Abstand nach vorne, links und rechts. Ist der gemessene Abstand größer als der Abstand, des letzten Messzeitpunktes, wird in diese Richtung eingemessen (Verweis auf die erste Version des Kleinen Freunds - baubar). Somit wäre eine ideale Follow-me Funktion, wie diese bei Mercedes, Tesla und co. verbaut wäre.