Aufbau eines Modellfahrzeugs mit Einplatinencomputern, das mit Hilfe unterschiedlicher Funknetztechnologien gesteuert werden kann.

Bachelorarbeit

vorgelegt von

Jonas Rövenich

Angefertigt im Studiengang Bachelor of Arts (B.A.) in Informatik an der Fachhochschule Frankfurt am Main,

Fachbereich 2 Informatik und Ingenieurwissenschaften

Wintersemester 2017/18

Erstprüferin oder Erstprüfer:	Prof. Dr. Christian Baun
Zweitprüferin oder Zweitprüfer:	Prof. Dr. Matthias Deegener

Eidesstaatliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Abschlussarbeit selbstständig und nur unter Zuhilfenahme der ausgewiesenen Hilfsmittel angefertigt habe. Alle Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter der Angabe der Quellen kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Frankfurt, 11. Dezember 2017

_____

Jonas Rövenich

Danksagung

Hiermit möchte ich mich bei allen bedanken die mich beim Absolvieren meines Studiums unterstützt haben. Insbesondere bei meiner Familie sowie meiner Freundin. Außerdem danke ich Herrn Prof. Dr. Baun und Herrn Prof. Dr. Deegener, dass sie mir als Betreuer sowie als Prüfer zur Verfügung standen.

Zusammenfassung

Diese Bachelorthesis beschreibt den Aufbau und die Verwendung von unterschiedlichen Funknetztechnologien zur Steuerung eines Modellautos mithilfe eines Einplatinencomputers. Hierzu wird mithilfe eines Raspberry Pi und eines Motor Steppers, das Modellauto in Bewegung versetzt. Dabei kommen zwei Möglichkeiten zum Einsatz, um das Auto zu steuern. Zum einen ein PlayStation 4 Controller sowie zum anderen die Möglichkeit der Steuerung über eine Webseite mit einem Kamerabild. Für den Einsatz der beiden Steuerungsmöglichkeiten werden Python Programme gezeigt und erläutert sowie der technische Aufbau und die Anwendung dessen.

Abstract

This bachelor thesis describes the construction and use of different radio network technologies for the control of a remote control car using a single board computer. To control the remote car, a Raspberry Pi and a motor stepper is used. There are two ways to control the car, by a PlayStation 4 controller, as well as the possibility of controlling via website with a camera video. Python programs are shown and explained for use, as well as the technical construction and application of this.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

BCM	Broadcom SOC channel
BSD	Berkeley Software Distribution
DC	Direct Current
DHCP	Dynamic Host Configuration Protocol
Engine	Antrieb
GND	Ground ~ negatives Spannungslevel
GPIO	General Purpose Input Output
GUI	General User Interface
HTTP	Hypertext Transfer Protocol
LCD	Liquid Crystal Display
NFC	Near Field Communication
PS4	PlayStation 4
PWM	Pulsweitenmodulation
RFID	Radio Frequency Identification
SSH	Secure Shell
V	Volt
VCC	positive Versorgungsspannung

1. Einleitung

Seit einigen Jahren wächst der technologische Fortschritt in Bezug auf Einplatinencomputer. Diese werden von Version zu Version immer leistungsfähiger. Die Anwendungsgebiete reichen von dem privaten Bereich in dem Experimente realisiert werden können bis hin zu industriellem Standard. Dabei geht es nicht nur um die Größe dieser Einheit oder die Leistung, sondern auch um die sehr sparsame Energieaufnahme und der damit verbundenen Hitzeentwicklung die das Gerät generiert. So können diese Einplatinencomputer ohne Lüfter betrieben werden und agieren auch noch bei wärmeren Temperaturen performant. Das macht den Raspberry Pi zu einem der erfolgreichsten Einplatinencomputer der heutigen Zeit.

2012 kam der erste Raspberry Pi, als Experimentier- und Lernsystem auf den Markt. Die Platine des Computers hat ungefähr die Größe einer Kreditkarte.[1] Der ursprüngliche Verwendungszweck bei der Entwicklung eines Einplatinencomputers war es, Kindern mit geringeren finanziellen Mitteln, das Erlenen des Programmierens sowie den Umgang mit Computern zu ermöglichen durch die Funktionalitäten, die der Raspberry Pi mit sich bringt und dem verhältnismäßig günstigen Preis.[2] Für die aktuelle Generation des Raspberry Pi 3 Model B liegt der Preis momentan zwischen 35 € und 42€ (Vergleich: Amazon 13.10.2017). Weitere Kosten können für Tastatur, Maus, Netzteil, Gehäuse, HDMI - Kabel, Netzwerkkabel sowie durch einen Bildschirm hinzukommen. Auch bei dieser Bachelorarbeit kommt die Experimentierfreudigkeit des Raspberry Pi zum Einsatz und unterstützt somit die unterschiedlichen Funknetztechnologien des DualShock 4 Controllers mit Bluetooth sowie die Verbindung zu einer Webseite via WLAN.

1.1 Motivation

Die Motivation zur Realisierung der unterschiedlichen Steuerungsmöglichkeiten mit Hilfe von Funknetztechnologien und mit Hilfe eines Einplatinencomputers stammt von der Faszination für die Steuerung von Drohnen. Drohnen lassen sich in der heutigen Zeit mittels Virtual Reality

[1] (Maier, 2016)
[2] (Wirminghaus, 2013)

Brille fliegen und kontrollieren. Dabei ermöglicht die Brille den Pilotenblick, der sogenannte First-Person-View, aus der Sicht der Drohne. Das lässt den Piloten aus der Sicht der Drohne interagieren und verstärkt das Erlebnis. Es kommen unterschiedliche Möglichkeiten der Fernbedienung zum Einsatz, wie über Touch-Steuerung, Gestensteuerung oder einfach über analoge Tasten. Für alle Möglichkeiten werden unterschiedliche Funknetztechnologien verwendet.[3] Daraus entstand die Anregung, eine Abwandlung dieser Technologie bei ferngesteuerten Modellautos zum Einsatz zu bringen und diese zu realisieren.

1.2 Ziel

Das Ziel der vorliegenden Arbeit ist die erfolgreiche Steuerung des Modellautos mit einem DualShock PlayStation 4 Controller via Bluetooth sowie eine Steuerung mit der Tastatur über eine Webseite via WLAN. Dies soll mithilfe von einem Raspberry Pi Zero (Einplatinencomputer) als zentrales Steuerungselement neben einem Stepper Motor realisiert werden. Dazu kommen weitere Verbesserungen und Anpassungen, die das Erlebnis verstärken sollen, wie bspw. eine Steuerung für Licht oder auch eine NFC – Schnittstelle (Near Field Communication), die einen schnellen Verbindungsaufbau zur Webseite ermöglicht. Ein LCD-Display wird ebenso verwendet, da dies die einzige Schnittstelle für Systemereignisse und Informationen darstellt und diese ausgeben kann.

[3] (Donath, 2017)

2. Stand der Technik

Im folgenden Kapitel wird der Stand der Technik für die vorliegende Aufgabenstellung genauer erläutert. Abschnitt 2.1 veranschaulicht die einzelnen Raspberry Pi Modelle. Anschließend wird in Abschnitt 2.2 die Motorsteuerung betrachtet. Das darauf folgende Kapitel 2.3 beschreibt unterschiedliche Modellautos, die für die zugrundliegende Aufgabe in Frage kommen. Der letzte Abschnitt befasst sich mit dem NFC- Chip.

2.1 Raspberry Pi Modelle

Der Raspberry Pi ist eine Linux-basierende PC Platine und stellt derzeit den weltweit beliebtesten Singleboard-Computer(SBC) dar. Die aktuelle Version ist der Raspberry Pi 3 Modell B V1.2, wie in Abb. 1 zu sehen ist.[4] Es wurden bereits über acht Millionen Exemplare seit der Präsentation des ersten Raspberry Pi im Jahre 2012 verkauft. Inzwischen sind bereits neun Modelle auf dem Markt erhältlich, die sich in drei Generationen gliedern.

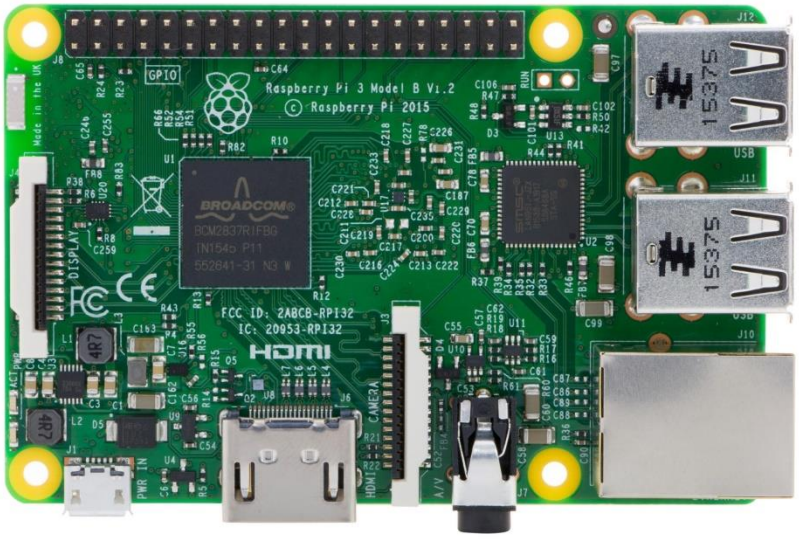

Abbildung 1 - Raspberry Pi 3 Modell B V1.2[5]

Die Raspberry Pi Foundation, eine in England ansässige Wohltätigkeitsorganisation, bedient auch seit 2015 individuelle

[4] (Schmidt, Raspberry Pi 3 ist da: Mini-PC endlich mit WLAN-Funk, 2016)
[5] Quelle: http://www.avc-shop.de/Raspberry-Pi-3-Modell-B; 06.11.2017

Kundenwünsche aus der Industrie bei der Herstellung der Einplatinencomputern.[6]

1. Generation
    - Model A (2013)
    - Model B (2012)
    - Model A+ (2014)
    - Model B+ (2014)
2. Generation
    - Model B (2015)
    - Model B 1.2 (2016)
3. Generation
    - Model B (2016)

Zu den bestehenden Generationen der Raspberry Pi Familie bestehen zusätzlich weitere Modellreihen. Diese werden als Pi Zero bezeichnet. Derzeit gibt es zwei Pi Zero-Modelle, die sich einzig und alleine darin unterscheiden, dass die Version Pi Zero W, aus dem Jahre 2017 kommt, ein integriertes WLAN, eine Kameraschnittstelle sowie ein Bluetooth 4.1 besitzt, wie in Abb. 3 zu sehen ist. Die Abbildung 2 zeigt die Pi Zero-Version aus 2015 auf.

- Zero (2015)
- Zero W (2017)

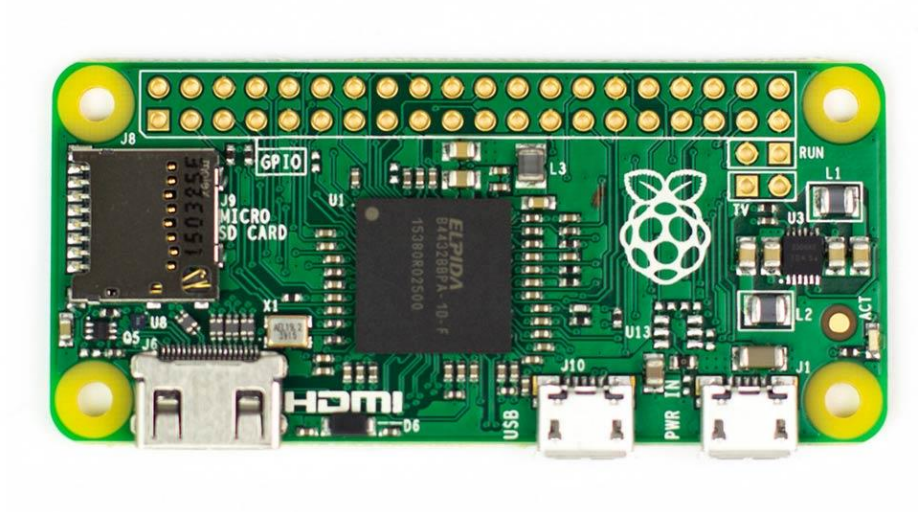

Abbildung 2 - Raspberry Pi Zero (2015)[7]

[6] (Kuther, 2016)

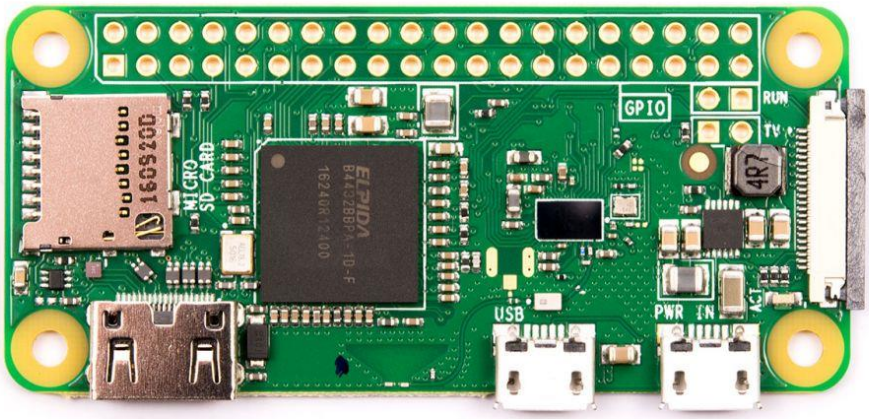

Abbildung 3 - Raspberry Pi Zero W (2017)[8]

2.2 Passende Modellautos für den Betrieb

Bei der Entscheidung für ein passendes Modellauto für die zugrunde liegende Arbeit gibt es einige Faktoren, die zu betrachten sind. Darunter fallen nicht nur Größe und Gewicht des Modellautos, sondern auch die Anordnung der verbauten technischen Teile und die Leistungsfähigkeit der Motoren. In Anbetracht dessen, dass weitere Anbauteile folgen werden und dass das Auto in seine Einzelteile zerlegt werden muss, ist es ratsam ein großes, stabiles Auto zu nehmen, dass ausreichend Antrieb bietet. Dadurch dass das Auto schwerer wird und damit langsamer, empfiehlt es sich, dass im Inneren ausreichend Platz zur Verfügung steht, um alle essentiellen Anbauteile wartungsgerecht installieren zu können. Nicht alle Autos erfüllen die Anforderungen, um weitere Projekte damit fortzuführen. Eine weitere Möglichkeit besteht darin, mit Hilfe eines Bausatzes, ein Auto nach seinen eigenen Vorstellungen zu kreieren und dieses auf die Anforderungen anzupassen und zu gestalten.

[7] Quelle: http://www.gizmodo.co.uk/2015/11/raspberry-pi-zero-is-a-4-computer-that-comes-free-with-a-magazine/; 06.11.2017
[8] Quelle: https://www.kiwi-electronics.nl/raspberry-pi-zero-w; 06.11.2017

Abbildung 4 – Beispiel für Ferngesteuerte Autos[9]

2.3 Motorsteuerung für Elektromotoren im Modellbetrieb

Eine Motorsteuerung, auch Motorsteuergerät genannt, ist für eine bestimmte Art von Elektromotoren konzipiert, welche die Steuerung, Regelung und Überwachung von den Motorfunktionen übernimmt. Es gibt zwei Arten von Elektromotoren, die im Modellbetrieb genutzt werden. Zum einen Motoren, die mit Gleichstrom betrieben werden und zum anderen Motoren, die mit Wechselstrom betrieben werden können.[10] Bei der Auswahl der passenden Motoren kommt es auf den Anwendungsfall an. Ausschlaggebende Kriterien, die dabei eine entscheidende Rolle spielen, sind z.B. die Stromaufnahme, das Gewicht, die Drehzahl, das Drehmoment sowie die Belastbarkeit, die Spannung, die Stromquelle, die Bedienbarkeit, die Lautstärke und der Preis.[11] Nach der Auswahl der Elektromotoren kann eine Motorsteuerung gewählt werden, die den Spezifikationen der Elektromoren entspricht.[12] Eine

[9] Quelle: https://www.rcferngesteuerteautos.de/index.php?cPath=25&gclid=Cj0KCQiArYDQBRDoARIsAMR8s_TvgvCRd3ub4sCCeRdt6hKmPmvkEArv8Idlh7hEZ5cJnFqsqd4XMbwaAnguEALw_wcB; 06.11.2017

[10] (Kirchberger, 2013)

[11] (D-Edition, 2014)

[12] (Raspberry Pi Schrittmotor ansteuern, 2014)

H-Brücke (L298N), wie in Abb. 4, wird gerne in Zusammenhang mit einem Raspberry Pi eingesetzt, da diese viele Möglichkeiten der Ansteuerung bietet.

Abbildung 5 - H-Brücke (L298N)[13]

2.4 NFC- Chip

Der Near Field Communication – Chip (NFC-Chip) ist ein Nahfeldkommunikationschip. Dieser stellt einen neuen Funkstandard zur drahtlosen Datenübertragung dar. Wie der Name schon andeutet, geht es dabei um zwei Elemente, die sich nahe beieinander befinden. Die Datenübertragung kann nur stattfinden, wenn diese sich innerhalb einer Reichweite von wenigen Zentimetern befinden. Ein Beispiel für den Einsatz eines NFC-Chips stellt die Ec-Karte dar. Jedoch gibt es noch weitaus mehr Einsatzmöglichkeiten von NFC Chips. Die Übertragungsgeschwindigkeit von 424 KByte/s ist geringer als die von Bluetooth, aber immer noch ausreichend um z.B. Internetlinks innerhalb eines Bruchteils einer Sekunde zu übertragen. Somit stellt NFC eine neue und sehr innovative Möglichkeit des Datenaustauschs dar, ohne größere Sicherheitsrisiken eingehen zu müssen.[14] Die Möglichkeit des Datenaustausches wird im Laufe des

[13] Quelle: https://www.roboter-bausatz.de/143/l298n-motortreiber-mit-doppelter-h-bruecke; 06.11.2017

[14] (Bauer, 2017)

Projektes genutzt, um den Aufruf der Internetseite mit allen Geräten, die NFC fähig sind, zu vereinfachen. Zuerst erhält der NFC-Chip eine statische IP, da die Funktionsweise in dieser Ausbaustufe auf das interne Netzwerk beschränkt bleibt. In einer weiteren Stufe, lässt sich das auch über eine dynamische IP- Adresse realisieren.

3. Technische Grundlagen

Die Kapitel beinhaltet alle technischen Komponenten, die für die Realisierung des Projektes genutzt worden sind sowie eine umfassende Erläuterung und Nutzung der Anbauteile. Ferner wird dargestellt, wie die Kommunikation und Steuerung mit anderen Bauteilen realisiert werden kann.

3.1 Raspberry Pi Zero W (2017)

Für die nachfolgende Arbeit wurde der Raspberry Pi Zero W verwendet. Vor diesem Hintergrund werden daher zuerst die technischen Grundlagen des Raspberry Pi Zero aufgezeigt. Der Raspberry Pi Zero W ist im Februar 2017 auf dem Markt erschienen und weist im Gegensatz zu seinem Vorgänger, dem Raspberry Pi Zero aus dem Jahre 2015, 2.4 GHz WLAN b/g/n und Bluetooth 4.1 mit einem geringen Stromverbrauch auf. Aufgrund seiner Abmessungen von 65 mm in der Länge, 31.2 mm in der Breite und 5 mm in der Höhe, ist damit der Pi Zero W der kleinste Einplatinencomputer, der in der Raspberry Familie hergestellt wurde. Er weist ein Gewicht von 9 Gramm auf und zählt damit zu den Leichtgewichten der Raspberry Pi Familie. Trotz seiner geringen Abmessungen wurde eine CPU mit 1000 MHz mit einer ARMv6 (32-bit) Architektur, stammend aus der Familie der ARM11, verbaut. Die GPU ist eine vom Typ Broadcom Dual Core VideoCore IV, wie sie bei allen Raspberry Pi zum Einsatz kommt. Ausschließlich das neuste Modell der Pi 3 Model B läuft mit einem höheren Takt von 300/400 MHz. Im Gegensatz zu den anderen Modellen, die mit 250 MHz getaktet sind. Dies ermöglicht es ein Full HD Video mit den Spezifikationen von 1080p und einer Wiederholungsrate von 30 Bildern pro Sekunde flüssig abzuspielen[15]. Die GPIO PINS (General Purpose Input Output), programmierbare Ein- und Ausgänge für allgemeine Zwecke, sind nicht wie bei den anderen Produkten der Raspberry Familie bereits vorinstalliert, sondern müssen bei Bedarf angelötet werden. Über diese ist es möglich verschieden Module und Anbauteile über elektrische Impulse zu steuern. Der Raspberry Pi ist zudem mit folgenden Hardwareteilen ausgestattet, wie in Abbildung 6 dargestellt.

[15] (Baykara, 2017)

- 1 GHz 32-bit Single-Core ARMv6 CPU
- Bluetooth 4.0 und Bluetooth Low Energy (BLE)
- 2.4 GHz 802.11n Wireless LAN
- 5 v via Micro-USB-Anschluss
- 1080 P HD Video und Stereo-Audio via mini-HDMI-Anschluss
- Individuelle Speicherkapazität via Micro SD-Karte
- Micro USB Ausgänge
- 40 GPIO Pin

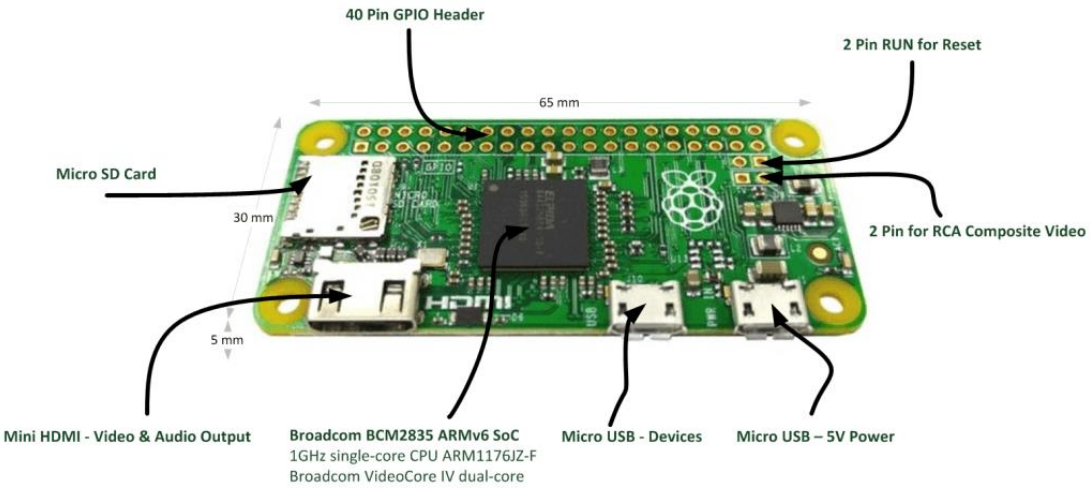

Abbildung 6 – Bauteile Pi Zero W[16]

Der Fokus des Raspberry Pi Zero, wie in Abb. 6 zu sehen ist, liegt bei einem Low-Cost Computer. Dieser birgt einige Besonderheiten, welche zu beachten sind, bevor dieser für die Realisierung eines Projektes verwendet wird. Durch die ARMv6 Architektur ist die Auswahl der Betriebssysteme beschränkt auf alle kompatiblen Systeme, die auf der ersten Generation des Pi funktioniert haben sowie weitere Betriebssysteme, wie bspw. Raspbian, Noobs, OpenElec und OSMC. Neuere Betriebssysteme, wie Ubuntu MATE oder Windows 10 IoT Core, sind im Moment nicht lauffähig. Für komplizierte und rechenintensive Anwendungen ist der Raspberry Pi Zero nicht konzipiert, da er nur mit einer Single-Core CPU und 512 MB Arbeitsspeicher ausgestattet worden ist. Durch seine geringen Abmessungen wird aus Kosten- und Platzgründen auch auf einen Netzwerkanschluss verzichtet. Für das Benutzen von Geräten, die einen USB- Anschluss haben, wird ein Mirco-

[16] Quelle: https://raspberry.tips/faq/raspberry-pi-zero-oft-gestellte-fragen; 06.11.2017

USB-Adapter benötigt. Für die Verwendung mehrerer Geräte wird ein USB-Hub benötigt, da standardmäßig nur ein Mirco-USB-Gerät Platz findet.

3.2 Motorsteuerung

Für die Steuerung der Motoren bei dem Modellfahrzeug wird ein L298N Mirco Controller Modul verwendet. Mit Hilfe dessen besteht die Möglichkeit sowohl DC (Direct Current) Motoren (Gleichstrom) als auch Schrittmotoren(oft auch Stepper genannt), die mit Wechselstrom betrieben werden, zu verwenden. DC Motoren habe nur zwei Leiter (Strom und Masse). Wenn Strom auf den Motor gelegt wird, fängt dieser an sich in eine Richtung zu drehen. Die Geschwindigkeit eines DC Motors wird durch Pulsweitenmodulation (PWM) reguliert. Dies stellt eine Technik dar, die durch schnelles Pulsen der Stromzufuhr erreicht wird, wie in Abbildung 7 aufgezeigt.[17]

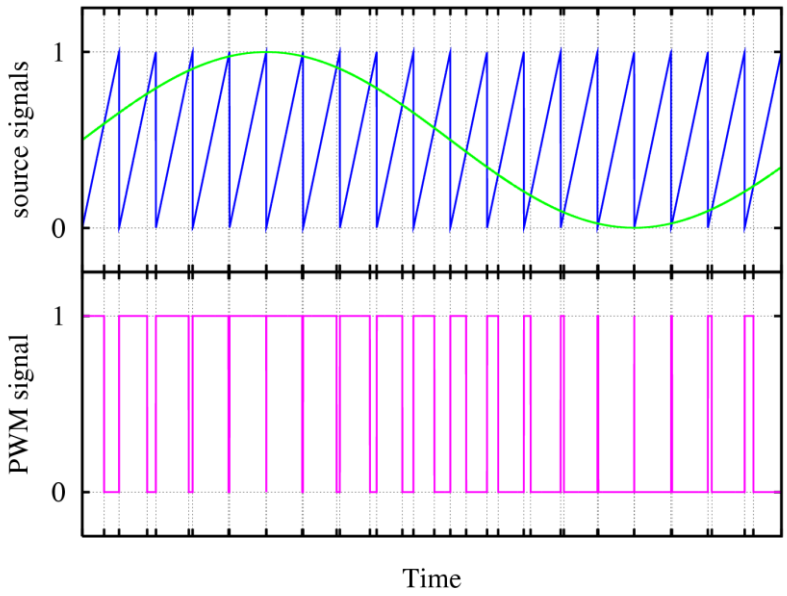

Abbildung 7 – Pulsweitenmodulation[18]

Der Prozentsatz der Zeit, die für das Durchlaufen des Ein- und Aus-Verhältnisses verwendet wird, bestimmt die Geschwindigkeit des Motors. Wenn die Leistung mit 50 % (halb an, halb aus) gewechselt wird, dreht sich der Motor mit der halben Geschwindigkeit von 100 %. Jeder Impuls ist so

[17] (MODMYPI, 2015)
[18] Quelle: https://de.wikipedia.org/wiki/Pulsweitenmodulation; 06.11.2017

schnell, dass der Motor ununterbrochen rotiert und ohne Stottern läuft. Eine vereinfachte Darstellung eines DC Motors wird in Abb. 8 betrachtet.

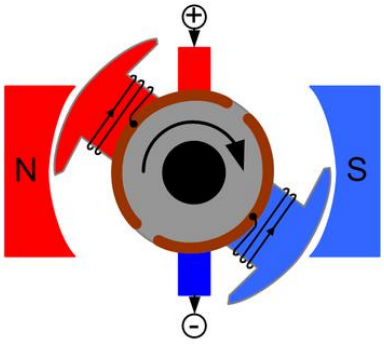

Abbildung 8 - Funktionsweise DC Motor[19]

Ein Schrittmotor verwendet eine andere Methode der Motorisierung. Schrittmotoren verwenden mehrere gezahnte Elektromagneten, die um einen zentralen Gang angeordnet sind, um die Position zu definieren. Diese benötigen einen externen Steuerkreis oder Mikrocontroller (z.B. einen Raspberry Pi), um jeden Elektromagneten individuell anregen zu können und die Motorwelle zu drehen. Wenn der Elektromagnet „A“ angetrieben wird, zieht er die Zähne des Zahnrads an und richtet sie aus, leicht versetzt zum nächsten Elektromagneten „B“. Wenn „A“ ausgeschaltet ist und „B“ eingeschaltet ist, dreht sich der Zahnkranz leicht, um sich mit „B“ auszurichten und s1 weiter um den Kreis herum, wobei jeder Elektromagnet um das Zahnrad herum angeregt und entregt wird, um eine Drehung zu erzeugen. Jede Drehung von einem Magnet zum nächsten wird als „Schritt“ bezeichnet. Somit kann der Motor durch präzise vordefinierte Schrittwinkel eine volle 360-Grad Drehung vollziehen.[20] Die Funktionsweise eines Schrittmotors wird in Abb. 9 dargestellt und erläutert. Eine derart präzise Steuerung von Motoren bei ferngesteuerten Modelautos wird nicht so häufig verwendet, wie die Nutzung von DC Motoren in diesem Bereich.

[19] Quelle: http://letsgoingwiki.reutlingen-university.de/mediawiki/index.php/Aktor:_DC_Motor; 06.11.2017

[20] (Zielosko, 2006)

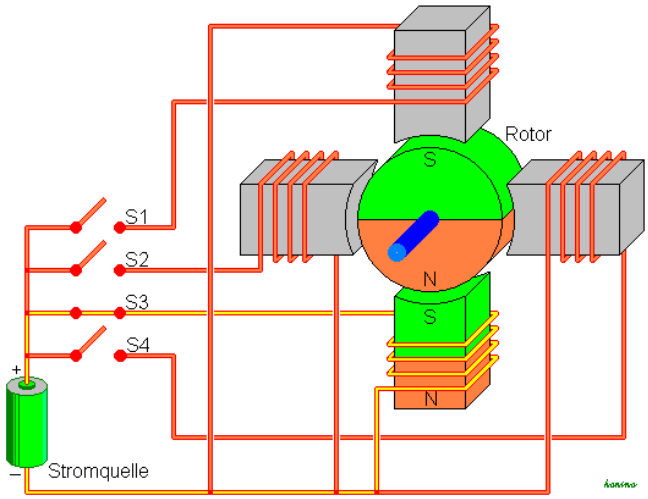

Abbildung 9 - Funktionsweise eines Schrittmotors[21]

Eine H-Brücke wird in der Regel für die Geschwindigkeit- und Richtungssteuerung von Motoren verwendet, kann aber auch für andere Projekte verwendet werden. Ein Beispiel ist die Helligkeitssteuerung bestimmter Beleuchtungsprojekte, wie leistungsstarker LED-Arrays. Eine H-Brücke ist eine Schaltung, die Strom in jeder Polarität antreiben kann und durch Impulsweitenmodulation(PWM) gesteuert werden kann. Motoren sind auf bestimmte Spannungen ausgelegt und können beschädigt werden, wenn die Spannung zu stark anliegt oder zu schnell herunterfällt, um den Motor herunterzufahren. Das Modul kann bis zu 46 Volt aufnehmen und einer Spannung von bis zu 3 A maximal standhalten. Es ist in der Lage einen 2-Phasen Schrittmotor, einen 4-Phasen Schrittmotor oder 2 DC Motoren zu betreiben. Zudem bietet es einen eigenen Stromanschluss für 5 Volt, wenn die Stromzuvor über 12 Volt hinausgeht. Die Spezifikationen sind wie folgt:

- Logische Spannung: 5 V
- Schaltung Spannung: 5 V-35 V
- Logischer Strom: 0 mA-36 mA
- Schaltung Strom: 2 A (Max. single Bridge)
- Max Power: 25 W
- Abmessungen: 43 x 43x 26 mm
- Gewicht: 26 g

Die Anschlüsse werden in Abb. 10 erläutert und sehen folgendermaßen aus.

[21] Quelle: https://de.wikipedia.org/wiki/Schrittmotor; 06.11.2017

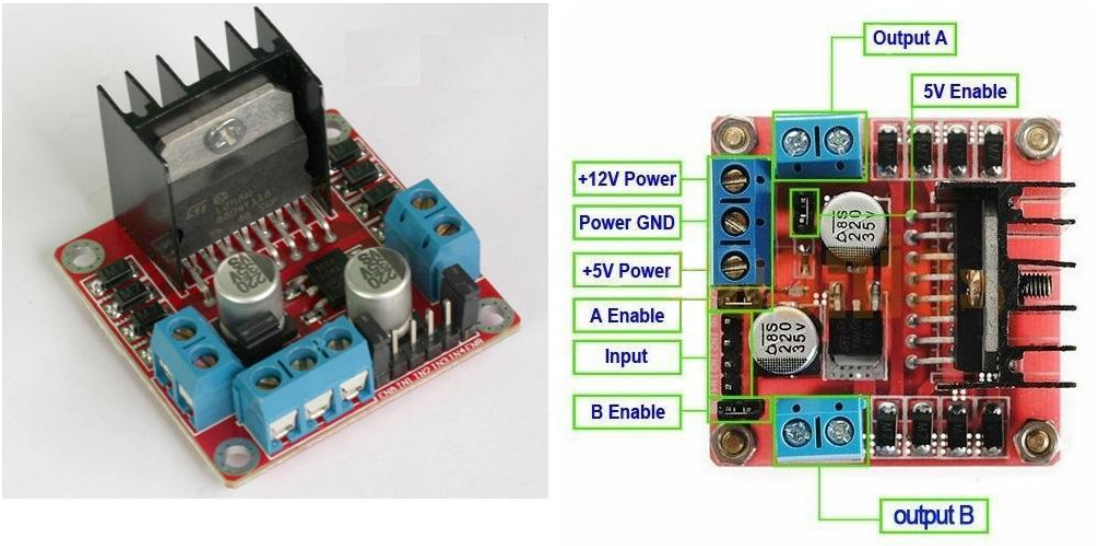

Abbildung 10 - Anschlüsse L298N Modul[22]

3.3 Ferngesteuertes Modelauto

Bei der Wahl des ferngesteuerten Modelautos sind einige Faktoren zu beachten. Dabei geht es bei dem vorliegenden Fall darum, genügend Platz auf dem Fahrzeug zu erhalten, um alle zusätzlich zu installierenden Teile befestigen zu können. Es eignet sich ein Fahrzeug, das sich auseinander bauen lässt, um an alle benötigten Teile, die darin verbaut worden sind, gut zu erreichen und damit arbeiten zu können. Durch die verbauten Teile lastet mehr Gewicht auf der gesamten Konstruktion. Dabei sollte die Bodenfreiheit immer noch gewährleistet sein sowie ausreichend Vortrieb durch die verbauten Motoren am Antrieb vorhanden sein. Da es sich bei dem Modelaufbau um ein ferngesteuertes Elektroauto handelt, werden in der Regel DC Motoren zur Verwirklichung des Antriebes genutzt. Diese sind im Gegensatz zu anderen Motoren leicht anzusteuern, da sie nur über die Stromzufuhr gesteuert werden und nicht noch ein weiteres Signal voraussetzen, wie bspw. Schrittmotoren.

[22] Quelle: http://lampatronics.com/product/l298n-motor-driver-board-module-for-arduino/; 06.11.2017

Abbildung 11 - Ferngesteuertes Modelauto[23]

Dieses Modelauto in Abb. 11 erfüllt alle benötigten Faktoren, um die Umsetzung der vorliegenden Aufgaben im Projekt zu realisieren. Die Größe beträgt 38 cm Länge, 18.5 cm Breite und eine Höhe von 17.5 cm und steht in einem Verhältnis von 1:18 zum Original. Das Fahrzeug besitzt eine Geschwindigkeit von 10 km/H, hat eine Reichweite von 70-80 Metern unter Verwendung der mitgelieferten 2.4 GHz Funkfernbedienung und einen effektiven Aktionszeitraum von 10-12 Minuten bei voller Ladung, die 180 Minuten dauert. Betrieben wird das Modelauto mit einem 6.4 V starken Lithium - Ionen Akku, der eine Kapazität von 13400/320 mAh aufweist, um die Lenkung sowie den Antrieb realisieren zu können. Insgesamt besitzt das Fahrzeug vier Motoren und drei gefederte Achsen, wobei die Vorderachse von zwei Motoren betrieben wird. Der eine Motor ist für den Antrieb verantwortlich, wohingegen der andere die Lenkung übernimmt. Des Weiteren besitzt das Fahrzeug zwei Lichtleisten, die eine ist an der Vorderstoßstange mit zwei Lichtern montiert und eine weitere befindet sich auf dem Dach mit vier Lichtern. Durch das Demontieren der beiden Reifen auf dem Auto bietet sich ausreichend viel Fläche, um alle verwendeten Bauteile anbringen zu können. Durch die eben beschriebenen technischen Grundlagen eignet sich dieses Modellauto für die Realisierung des Projektes. Durch die Verschraubung der Einzelteile mit Kreuzschlitzschrauben lässt sich dieses zudem in Einzelteile zerlegen.

[23] Quelle: https://www.alibaba.com/product-detail/Wholesale-wltoys-18628-2-4G-6WD_60596459457.html; 06.11.2017

3.4 DualShock 4 Wireless Controller

Der Wireless Controller ist aus der 4.ten Generation, wie in Abb. 12 dargestellt, und wurde zusammen mit der Playstation 4 am 20.Februar 2013 in New York vorgestellt. Das Design unterscheidet sich nicht großartig von den vorhergehenden Generationen. Die Optik und das Handling hingegen haben sich dahingehend stark verändert, dass dieser mehr Griffigkeit, neue Elemente für mehr Spielerlebnis, technische Anpassungen sowie ein integriertes Touchpad, einen Mono-Lautsprecher und einen Audioausgang bietet. Die aufgezählten Punkte heben die vierte Generation stark von den vorhergehenden Modellen ab und ermöglichen einen weiteren Einsatz für die nächsten Jahre.[24] Der integrierte Akku weist eine Kapazität von 1000 mAh auf und liefert in der Praxis ungefähr sieben Stunden Akkulaufzeit. Die Akkulaufzeit ist sowohl abhängig von Audiologs, die über den integrierten Lautsprecher wiedergegeben werden und der Lautstärke, die vom PS4 Controller wiedergegeben wird als auch der Helligkeit der Lichtleiste an der Vorderseite sowie der Intensivität der Vibrationen der beiden Motoren.[25]

Abbildung 12 - DualShock4 Wireless Controller[26]

Auf dem Controller befinden sich insgesamt 17 Knöpfe und zwei Sticks.

[24] (DUALSHOCK 4 Wireless-Controller, 2013)
[25] (Kämpfer, 2015)
[26] Quelle: https://www.bestbuy.com/site/sony-dualshock-4-wireless-controller-for-playstation-4-jet-black/5580915.p?skuId=5580915; 06.11.2017

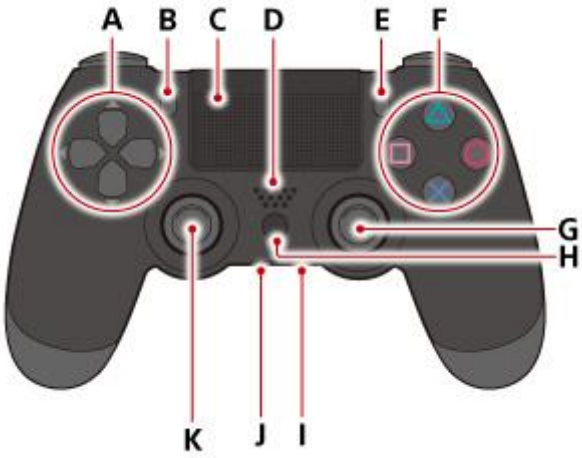

Abbildung 13 - DualShock4 Vorderseite[27]

A) Richtungstasten
B) SHARE-Taste
C) Touchpad/Touchpad-Taste
D) Lautsprecher
E) OPTIONS-Taste
F) △-Taste/ ○-Taste/ ✕-Taste/ □-Taste
G) Rechter Stick/R3-Taste
H) PS-Taste
I) Stereo Headset-Anschluss
J) Erweiterungsport
K) Linker Stick / L3-Taste

[27] Quelle: http://manuals.playstation.net/document/de/ps4/basic/pn_controller.html; 06.11.2017

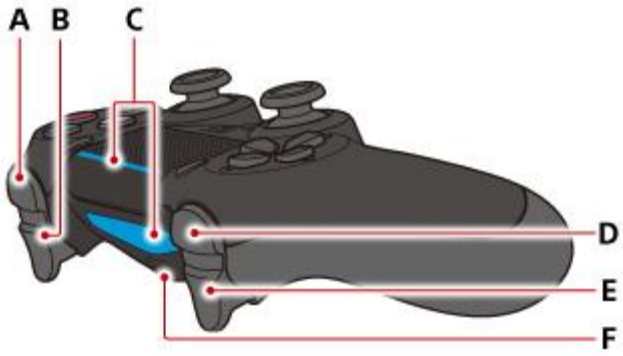

Abbildung 14 - DualShock4 Oberansicht[28]

A) R1-Taste
B) R2-Taste
C) Leuchtleiste
D) L1-Taste
E) L2-Taste
F) USB-Port

Neben den Tasten und Eigenschaften des Controllers, wie in Abb. 13 und 14 erläutert, weist der DualShock 4 Wireless Controller neben hochempfindlichen Beschleunigungsmessern ein eingebautes Gyroskop auf, der das Messen aller Bewegungen übernimmt und ausgelesen werden kann.[29]

3.5 Raspberry Pi Kamera

Der Raspberry Pi Zero W besitzt einen Kameraanschluss mit dem es möglich ist unterschiedliche Kameras anzuschließen und einzusetzen. Aufgrund der geringeren Abmessungen des Pi Zero, im Gegensatz zu den anderen Modellen seiner Familie, wird ein Kameramodul für den Pi Zero benötigt und dieses ist nicht kompatibel zu anderen Raspberry Pi, sondern ausschließlich für die Raspberry Pi Zero Generation verwendbar.[30] Bei der Auswahl der Kameras gibt es Unterschiede bezüglich ihrer Auflösung, der Bauart, der

[28] Quelle: http://manuals.playstation.net/document/de/ps4/basic/pn_controller.html; 06.11.2017
[29] (DualShock 4 , 2017)
[30] (König, 2016)

Länge des vorinstallierten Kabels, der Anzahl der Megapixel sowie ihrer Tages- und Nachttauglichkeit. Für den vorliegenden Verwendungszweck, der First-Person View aus dem Cockpit des Autos, wird eine Kamera mit sehr kleinen Abmessungen benötigt und einer guten Auflösung, die das Fahrerlebnis flüssig wieder gibt. Dabei ist die Bildwechselfrequenz ausschlaggebend. Diese bezeichnet die Anzahl der Einzelbilder für eine gewisse Zeitspanne, i.d.R. wird diese pro Sekunde angegeben. Für eine flüssige Darstellung liegt die Bildwechselfrequenz bei 24 Bildern pro Sekunde, welche zudem den aktuellen Standard für Filme darstellt[31]

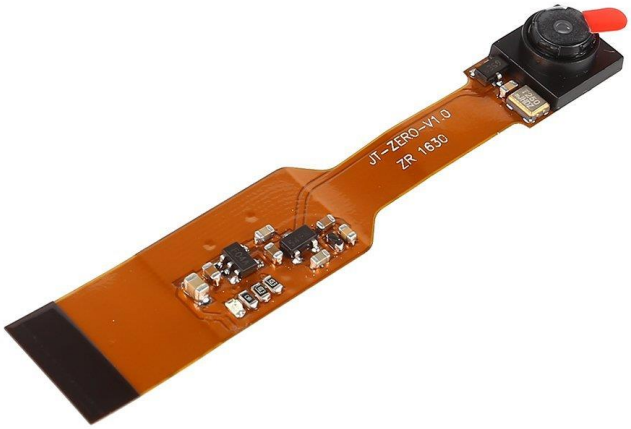

Abbildung 15 - Kamera Modul 5MP[32]

Das Kameramodul in Abb.15 besitzt eine Kamera mit 5 Megapixel Auflösung, einem Gewicht von 18 Gramm sowie den Maßen von 10.9 cm Länge, 5.1 cm in der Breite und 2.3 cm in der Höhe. Zudem schafft die Kamera bei einer Auflösung von 1080p eine Bildwiederholungsrate von 30 Bildern. Die Besonderheit dieses Moduls liegt darin, dass der Chip der Kamera im Datenübertragungsband integriert worden ist, ohne eine weitere Platine zu benötigen auf der die Kamera angebracht ist, die zusätzlich weiteren Platz in Anspruch nimmt. Dies ermöglicht den Einbau im Cockpit des Modelautos mit

[31] (Wegner, 2011)
[32] Quelle: http://www.dx.com/p/5mp-720p-1080p-mini-camera-module-for-raspberry-pi-zero-447589#.WgBn3Wj9Tcs; 06.11.2017

einer ausreichenden Bildqualität und noch ausreichend Platz für die Montage.

3.6 LCD Display-Modul HD 44780

Bei einem LCD Display-Modul gibt es unterschiedliche Ausstattungsmerkmale, wie die Größe der Spalten, der Anzahl der anzuzeigenden Symbole, die Helligkeit der Hintergrundbeleuchtung, die Betriebsspannung und Leistungsaufnahmen sowie die unterschiedlichen Möglichkeiten der Ansteuerung des Display-Moduls. Nachdem alle nötigen Eigenschaften auf dem Raspberry Pi Zero programmiert und konfiguriert worden sind, wird das Display die einzige Kommunikationsschnittstelle im laufenden Betrieb darstellen. Alle anderen Möglichkeiten der Interaktion oder das Abrufen des aktuellen Status sind nicht mehr vorgesehen und lassen sich ohne einen Bildschirm oder einen PC nicht realisieren. Somit lassen sich die wichtigsten Funktionen über das Display ausgeben sowie stellt es in einer Erweiterung seiner Primärfunktion eine Rückleuchte für das Modelauto dar. Diese sollte in der Lage sein, das Blinken zu simulieren und das Rückwärtsfahren darzustellen.

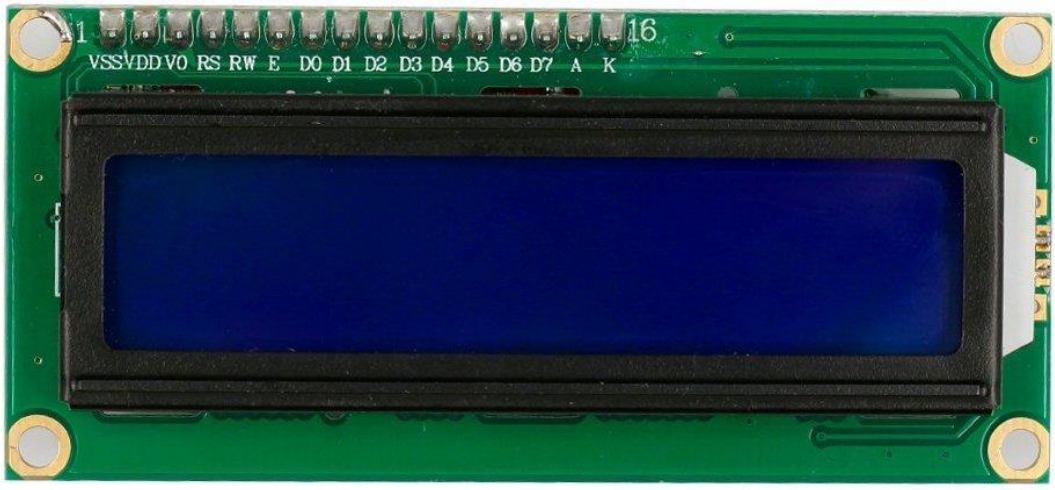

Abbildung 16 - Display Modul Vorderseite[33]

[33] Quelle: https://www.thingbits.net/products/standard-lcd-16x2-display; 06.11.2017

Ein Standard LCD Display-Modul besitzt 16 Pins, die für den Datenaustausch sowie für die Stromzuvor zuständig sind. Die Pins müssen bei den meisten LCD-Displays eigenständig angelötet werden, wie in Abb. 16 bereits geschehen ist. Nachfolgend wird die Beschreibung der Pins für die weitere Verwendung von links nach rechts dargestellt. Hierzu wird bei Pin 1 begonnen:

- VSS: Erdung
- VDD: +5 Volt
- VEE: Kontrast
- RS: Daten / Anweisungen
- R/W: lesen / schreiben
- E: Aktivieren
- D0: Daten Bus
- D1: Daten Bus
- D2: Daten Bus
- D3: Daten Bus
- D4: Daten Bus
- D5: Daten Bus
- D6: Daten Bus
- D7: Daten Bus
- A: Anode (+)
- K: Kathode (-)

In diesem Versuchsaufbau wird ein I^2C Anschluss genutzt, welcher die Anzahl der zu belegenden Pins von 16 auf 4 verringert. Damit werden Fehler verhindert und 12 Pins eingespart, die ansonsten an den Raspberry Pi Zero angeschlossen werden müssen. Bei diesem Aufbau wird ein Modul der Firma „SunFounder“ verwendet, das mit der Rückseite des LCD-Displays verlötet wird (rote Platine), wie in Abb. 17 aufgezeigt. Der GND Anschluss der Platine wird mit dem Masseanschluss des Raspberry Pi verbunden. Der VCC wird mit einem 5 Volt Anschluss und den beiden Anschlüssen SDA sowie SCL auf dem Raspberry Pi verbunden.

Abbildung 17 - I²C Platine verlötet am LCD-Display[34]

3.7 Near Field Communication- Chip

Ein Nahfeldkommunikationschip ist ein Übertragungsstandard zum kontaktlosen Austausch von Daten mittels RFID – Technik (RadioFrequency Identification). Damit bietet sich die Möglichkeit Daten über kurze Distanzen von wenigen Zentimetern mit einer maximalen Übertragungsrate von 242 KBit/s zu übertragen. Bisher kam die Technologie bei Micropayments – die Bezahlung kleiner Beträge- zum Einsatz.[35] Insgesamt gibt es vier unterschiedliche Arten von NFC-Chips, die Inhalt der Abbildung 18 sind.

NFC – Typen

- Type 1 Tag
- Type 2 Tag
- Type 3 Tag
- Type 4 Tag

Abbildung 18 - NFC Tag

Dabei unterscheiden sich die genannten Typen in mehrere Kategorien. Die typische Speicherkapazität für die unterschiedlichen Typen variiert und erstreckt sich von 96 Bytes bis 512 Bytes für Type 1, über 48 Bytes bis 888

[34] Quelle: https://funduino.de/nr-19-i%C2%B2c-display; 06.11.2017
[35] (DATACOM, 2014)

Bytes bei Type 2, sowie 1 KB/ 4 KB/ 9 KB bei Type 3 bis hin zu 2 KB/ 4 KB/ 8 KB/ 32 KB bei Type 4. Zudem gibt es bei allen Typen die Option zwischen lesenden und schreibenden Zugriff oder aufgrund des Schreibschutzes nur gelesen zu werden. Zudem gibt eine weitere Funktion, die ausschließlich der Type 4 Tag vorbehalten ist. Hierbei handelt es sich um die Funktion „Aktive Inhalte“, welche es dem NFC Tag ermöglicht seinen Inhalt selbst zu ändern, wie zum Beispiel die Anzahl der Auslesungen über einen Zähler.[36] Somit lassen sich verschiedene Aufgaben mit Hilfe von NFC Tags automatisieren, die man sonst manuell durchführen müsste. Trigger ist eine App, die es ermöglicht bestimmte Aktionen bei Berührung des Smartphones mit dem NFC Chip auszuführen. Das Stellen von Weckern, das Aufrufen von Internetseiten sowie das Einstellen der Lautstärke des Mobiltelefons erfordern somit keiner weiteren Handlungen bis auf den kurzen Kontakt mit dem NFC-Tag, wie in Abb. 18 dargestellt oder einem NFC- Chip.

3.8 Funknetztechnologien

Bei dem vorliegenden Projekt kommen zwei unterschiedliche Arten von Funknetztechnologien zum Einsatz. Bei der einen Methode, bei der die Steuerung via PlayStation 4 Controller zustande kommt, wird als Funknetztechnologie Bluetooth verwendet, um das Modelauto zu steuern. Bei der anderen Methode, bei der die Steuerung des Autos über eine Webseite realisiert wird, kommt die Funknetztechnologie WLAN zum Einsatz. Im Gegensatz zu WLAN ist Bluetooth darauf spezialisiert eine Verbindung über wenige Meter herzustellen sowie eine einfache Verbindung zwischen den Geräten aufzubauen. Das sogenannte Pairing funktioniert über die Eingabe eines Zugangscodes beim ersten Verbinden. Bluetooth benötigt nur zwei Voraussetzungen für ein erfolgreiches Pairing. Die Geräte müssen nahe beieinander sein und die Empfangsbereitschaft muss bei beiden Geräten aktiviert sein. Dabei ist die Anzahl der verbundenen Geräte nicht auf zwei beschränkt. Je nach Bluetooth-Klasse kann eine höhere Reichweite erzielt werden, insofern die verbundenen Geräte in der gleichen Bluetooth-Klasse sind. Insgesamt werden die Bluetooth-Klassen in drei Klassen unterteilt. Die

[36] (NFCtags)

Klasse I besitzt eine Reichweite von bis zu 100 Metern, währen die Klasse II eine Reichweite von bis zu 20 Metern aufweist und die Klasse III eine Reichweite von 10 Metern besitzt.[37] Sowohl das integrierte Bluetooth Modul auf dem Pi Zero als auch der PlayStation 4 Controller gehören zu der Klasse III und besitzen eine Reichweite von 10 Metern, abhängig von der Umgebung.[38]

Die zweite Methode der Steuerung des ferngesteuerten Autos wird mit den Pfeiltasten auf einer Tastatur über eine Webseite mithilfe von WLAN realisiert. Sowohl die Reichweite als auch die Datenübertragungsrate sind höher als bei einer Bluetooth Verbindung. Dagegen ist bei WLAN der Verbindungsaufbau wesentlich komplexer. Im Ergebnis decken diese beiden Technologien unterschiedliche Einsatzbereiche ab.[39] Die unterschiedlichen Einsatzbereiche in der Praxisanwendung bei diesem Projekt werden in Kapitel 4 aufgezeigt.

3.9 Zubehör für den Aufbau

Eine weitere Komponente, die für den Aufbau benötigt wird, ist die Verkabelung der einzelnen Bestandteile. Dabei muss unterschieden werden, ob es sich um eine Stromzufuhr oder eine Datenübertragungsleitung handelt. Hierbei ist der Durchschnitt der verwendeten Kabel von Bedeutung, da ein dickerer Durchschnitt einen höheren Stromdurchsatz aufweisen kann. Für die Sicherstellung der Datenübertragung sind insbesondere Jumperkabel zu verwenden, da diese sich aufgrund ihres Durchmessers gut biegen lassen. Ein Druckknopf wird benötigt, um die Anzeige auf dem LCD-Display mittels Interaktion zu verändern. Hierbei bietet dieser die Möglichkeit drei Zustände zu verwalten. Im Gegensatz zu einem Schalter, der nur die zwei Zustände „An“ und „Aus“ aufweist. Dies wird in Abbildung 19 aufgezeigt:

[37] (Ek(CF), 2016)
[38] (Löwenstein, 2013)
[39] (Metzger, 2003)

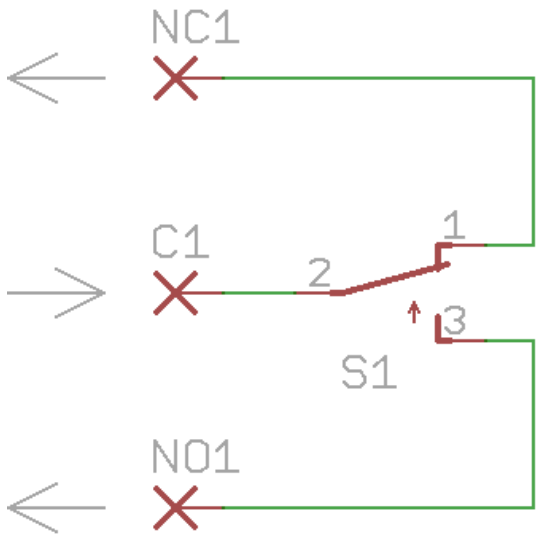

Abbildung 19 - Schaltung[40]

NC1 in Abbildung 19 steht für Normally Closed und NO1 steht für Normally Open. Zur Überprüfung, ob an C1 ein Signal ankommt, wenn der Taster gedrückt wurde, wird der Drucktastenschalter verwendet. Durch das Verwenden des Drucktastenschalters wird der Stromkreislauf zwischen NO1 und C1 geschlossen bis dieser wieder durch das Verwenden des Drucktastenschalters gelöst wird.

Um nun die Schaltung richtig für den vorliegenden Verwendungszweck anzuschließen, wird dies in der nachfolgenden Abbildung verdeutlicht. In Abbildung 20 sind fünf Stecker des Drucktastenschalters dargestellt. Zwei der fünf Stecker sind für die Stromzufuhr und die Masse zuständig. Um den Lichtring auf der Oberseite durchgehend leuchtend zu lassen, ohne dass etwas gedrückt sein muss, wird die Stromzufuhr an C1überbrückt.

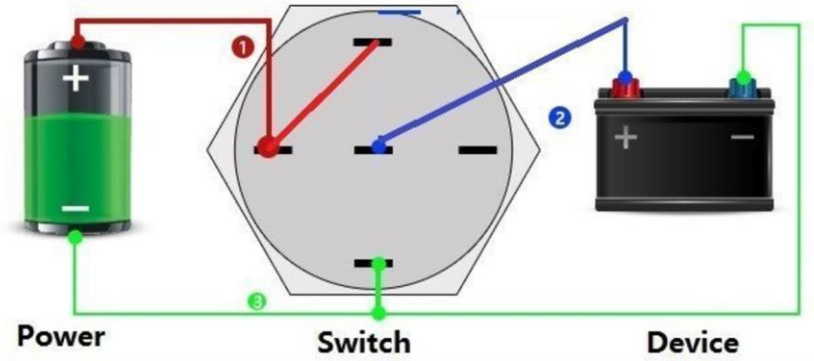

Abbildung 20 - Verkabelung Schalterdruckknopf[41]

[40] Quelle: http://www.ledhilfe.de/viewtopic.php?f=35&t=18282; 06.11.2017

Zusätzlich zu dem bereits vorhandenen Akku für die Motoren benötigen alle genannten Komponenten eine weitere externe Stromquelle. Die Verwendung von nur einer externen Stromquelle mindert jedoch drastisch die Laufzeit des Fahrzeugs. Zudem ist der Akku mit einer Spannung von 6.7 Volt zu hoch für die meisten Komponenten und kann Schäden verursachen bei der nicht korrekten Wahl eines Vorwiederstandes. Eine handelsübliche Powerbank, wie in Abb. 21 abgebildet, reicht für den Verwendungszweck aus.[42] Diese muss nicht besonders groß sein oder eine große Kapazität haben, da die Leistungsaufnahme der Komponenten im Verhältnis gering ist. Die externe Stromquelle kann so gewählt werden, dass diese sich gut verbauen lässt. Alle Platinen und elektronischen Gegenstände, die auf dem Auto verbaut wurden, sollten durch Schrauben an einer Halterung befestigt werden, um das Leichtmodellauto vor Schäden durch das Befestigen von Einzelteilen zu schützen. Die hier verwendeten Platinen haben bereits vorgebohrte Löcher, die isoliert sind, um die Platine daran befestigen zu können.

Abbildung 21 - Beispiel einer Powerbank[43]

[41] Quelle: https://www.amazon.com/KNACRO-Self-locking-Protection-Explosionproof-Compression/dp/B06ZZGC1C4; 06.11.2017
[42] (Ben, 2016)
[43] Quelle: https://www.anker.com/products/variant/PowerCore-5000mAh/A1109011; 06.11.2017

4. Entwicklung eines ferngesteuerten Modellfahrzeug mit Hilfe eines Einplatinencomputers

Wie in Kapitel 3 aufgezeigt, werden für die Entwicklung eines ferngesteuerten Modellfahrzeugs noch weitere Komponenten neben dem Raspberry Pi Zero W benötigt. Im folgenden Kapitel 4 wird daher aufgezeigt, wie diese in Verbindung miteinander eingesetzt und programmiert werden müssen, um als Resultat ein ferngesteuertes Modellfahrzeug mit Hilfe eines Einplatinencomputers zu entwickeln.

Zu Beginn wird für die Stromversorgung während der Einrichtung und Installation aller Komponenten ein Netzteil benötigt. In der Regel ist ein solches im Lieferumfang des Raspberry Pi enthalten oder alternativ eines zu verwenden, das über den Mirco-USB-Stecker 5 Volt und 2 Ampere liefert.[44] Des Weiteren wird eine performante Micro SD-Karte benötigt, um das Optimale aus dem Raspberry Pi herausholen zu können.

Aufgrund der vorliegenden Aufgabenstellung die Steuerung über eine Webseite mit einer Kamera-Schnittstelle zu verwirklichen, wird eine SD-Karte benötigt, die schnelle Lese – und Schreibzyklen aufweist. Die Auswahl fällt daher auf eine Samsung Pro 32 GB Micro SD-Karte für den Raspberry Pi Zero.[45]

Im nachfolgenden Kapitel wird die Vorgehensweise der Entwicklung des ferngesteuerten Modellfahrzeugs aufgezeigt und erläutert.

4.1 Installation des Images

Zur Installation des Images (Betriebssystem) wird zunächst die MicroSD-Karte formatiert, wodurch alle Daten, die sich zuvor auf der SD-Karte befunden haben, gelöscht werden. Die Formatierung der SD-Karte ist im Format FAT32 vorzunehmen. Hierzu kann zum einen das Programm „SD Memory Card Formatter“ benutzt werden oder zum anderen kann die Formatierung über einen Mac Computer im Terminal vorgenommen werden. Der Aufruf des Terminals unter macOS Sierra funktioniert mithilfe der

[44] (Merz, 2016)
[45] (Paul, 2015)

Tastenkombination Command ⌘ + Leertaste. Dadurch öffnet sich die Suchleiste, in die „Terminal“ eingegeben werden muss. Der erste Treffer in der untenstehenden Liste öffnet das Terminalfenster.

```
1   diskutil list
```

Das Terminal zeigt alle Informationen über die verfügbaren Festplatten sowie ihre Partitionierung an. Die SD-Karte kann somit identifiziert werden und der BSD (Berkeley Software Distribution) Name (/dev/disk3) kann für die weiteren Schritte verwendet werden. Die Partition heißt dann /dev/disk3s1. Aufgrund dessen, dass die gesamte SD-Karte formatiert werden soll, wird auch der BSD-Name /dev/disk3 verwendet, welcher die gesamte Festplatte anspricht. Um die Festplatte auszuwerfen und diese für das Image vorzubereiten, wird der Befehl verwendet:

```
1  diskutil unmountDisk /dev/disk3
```

Um das Image auf die SD-Karte zu kopieren, muss es zuvor heruntergeladen werden. Hierfür sind mehrere Auswahlmöglichkeiten auf der offiziellen Raspberry Pi Webseite[46], unter dem Reiter Downloads, zur Verfügung gestellt. Für die weitere Verarbeitung eignet sich eine graphische Oberfläche und erleichtert die Bearbeitung, wenn nicht ausschließlich mit SSH und einer Kommandozeile gearbeitet werden soll. Auf der Webseite im Bereich Downloads verbirgt sich das Image „Debian Jessie“ mit der Kernel Version 4.9, dass bei der vorliegenden Arbeit zum Einsatz kommt.

Version Image:

- RASPBIAN JESSIE
- Veröffentlichungsdatum: 2017-06-22
- Kernel Version: 4.9

Über den Knopf „Download ZIP“ wird es nun automatisch heruntergeladen. Diese muss nach dem erfolgreichem Download entpackt werden und auf dem Desktop abgelegt werden. Andernfalls wird noch der Pfad des Images benötigt, in dem er abgelegt wird. Der Name der Downloaddatei beinhaltet das Veröffentlichungsdatum, die vorliegende Architektur sowie den Name der aktuellen Version (2017-06-22-rpd-x86-jessie.img).

[46] https://www.raspbian.org

```
1   sudo dd bs=1m if=2017-06-22rpd-x86-jessie.img
    of=/dev/rdisk4
```

Dieser Vorgang wird abhängig von der Größe des Images und der Schreibgeschwindigkeit der SD-Karte einige Zeit in Anspruch nehmen. Um sich den Fortschritt anzeigen zu lassen, kann das Signal mittels der Tastenkombination Command ⌘ + T gesendet werden. Dadurch wird der Fortschritt mittels einer prozentualen Zahl dargestellt. Nachdem der Vorgang erfolgreich war, kann die Festplatte ausgeworfen werden.

```
2   sudo diskutil eject /dev/rdisk3
```

Alternativ kann auch über Disk Utility die SD-Karte entfernt werden.

4.2 Inbetriebnahme des Raspberry Pi

Vor der ersten Inbetriebnahme muss die SD-Karte mit dem Image in den Raspberry Pi eingesteckt werden. Dafür ist die Pinleiste für die GPIO Pins zu verlöten, ein passendes HDMI Kabel mit Monitor vorhanden sein sowie eine Tastatur mit integrierter Maus oder ein USB-Hub an den zwei Geräten angeschlossen werden können. Wenn alle erforderlichen Komponenten verfügbar sind, kann das Netzteil an den Raspberry Pi angeschlossen werden. Durch das Anstecken des Netzteils wird der Raspberry Pi in den laufenden Zustand versetzt. Nachdem der Raspberry Pi gestartet wurde, erscheint zunächst das „Raspberry Pi Software Configuration Tool (raspi-config)“. Dabei handelt es sich um eine Konfigurationsunterstützung für das Raspbian Betriebssystem.

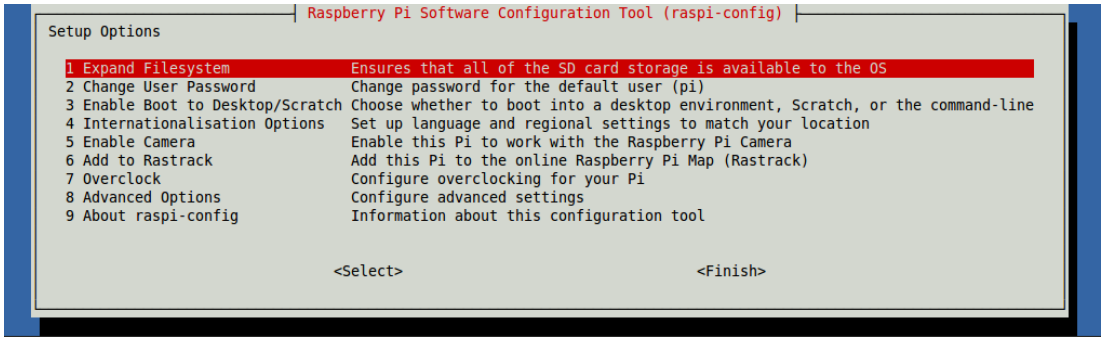

Abbildung 22 - Raspbian Configuration Tool

Dieses Tool wird nur beim ersten Start angezeigt. Es kann jedoch jederzeit mit dem Befehl:

```
1  sudo raspi-config
```

in der Kommandozeile wieder aufgerufen werden.

Als erstes sollte der erste Punkt in den Setup Options „Expand Filesystem“ ausgewählt werden, damit das Dateisystem sowie das Betriebssystem auf der gesamten SD-Karte ausgebreitet wird.[47] Des Weiteren ist es empfehlenswert bei den Setup Options das Standardpasswort zu ändern. Dies kann auch zu einem späteren Zeitpunkt erfolgen, da es vorerst im eigenen Netzwerk zur Anwendung kommt und ein Zugriff von Dritten dort nicht gewährleistet ist. Als nächstes wird eingestellt, wie der Raspberry nach einem Neustart bootet. Standardmäßig bootet dieser mit einer Desktopumgebung, welche mehr Ressourcen benötigt, aber die Bearbeitung vereinfacht.

Die Option Nummer 4 „Internationalisation Options“ sollte konfiguriert werden. Zum einen greifen unterschiedliche Module auf die Daten zu. Zum anderen wird dadurch auch das Layout der Tastatur bestimmt und vereinfacht die Eingaben oder die Vermeidung der Darstellung falscher Symbole. Im nächsten Schritt ist „enable Camera“ auszuwählen, sodass die Einstellung auf „enable“ steht. Dies erleichtert die Bedienung, wenn die Kamera angeschlossen wird. Von einer Übertaktung des Raspberry Pi sollte abgesehen werden, da diese sowohl zu Lasten der Stabilität des Systems geht als auch die Energieaufnahme und die Hitzeentwicklung erhöht. Eine letzte Einstellung muss unter dem Reiter „Advanced Options“ vorgenommen werden, welcher mehrere Unterpunkte aufweist. Um einen einfacheren und schnelleren Zugriff zu ermöglichen, wird die SSH Verbindung freigeschaltet, sodass ein Fernzugang per Kommandozeile auf dem Raspberry Pi möglich ist. Das bedeutet, dass die SSH-Verbindung sowie I2C müssen auf „enable“ stehen. I2C Interfaces ermöglichen die Kommunikation mit Modulen, die einen I2C Kernel aufweisen, wie das verwendete LED-Display.

4.3 Vergabe einer statischer IP-Adresse

Zuerst muss der Raspberry Pi mit einem Netzwerk verbunden werden. Da dieser keinen LAN Anschluss aufweist, wird für die Verbindung das

[47] (Burgess, 2016)

integrierte WLAN benutzt. Über die grafische Oberfläche lässt sich in der oberen rechten Ecke auf dem WLAN- Symbol das Heimnetzwerk auswählen und ein Passwort eingeben, wie in Abb. 23 zu sehen ist.

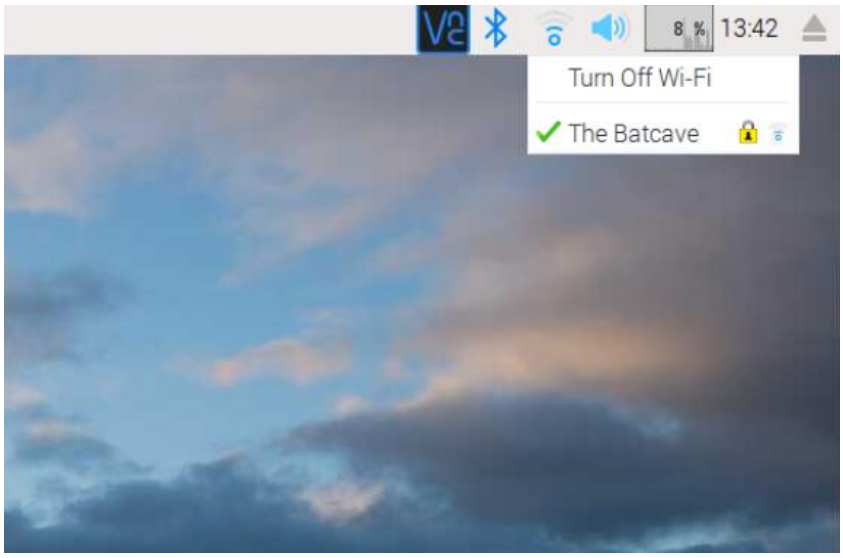

Abbildung 23 - WLAN verbinden

Solange sich der Raspberry Pi im selben Netzwerk befindet und verwendet wird, empfiehlt es sich, eine statische IP-Adresse für den Pi zu vergeben, um den Konfigurationsaufwand möglichst gering zu halten. Andernfalls besteht die Möglichkeit, dass sich nach jedem Neustart die IP-Adresse ändert und diese über eine grafische Oberfläche (Monitor) herausgefunden werden muss. Zudem müssen alle IP-Adressen geändert werden, die zuvor verwendet worden sind, um den Raspberry Pi über das Netzwerk zu erreichen. Sofern ein Bildschirm und eine Tastatur vorhanden und angeschlossen sind, lässt sich über die Eingabe des Befehls die IP Adresse herausfinden, wie in Abb. 24 dargestellt.

```
1  ifconfig
```

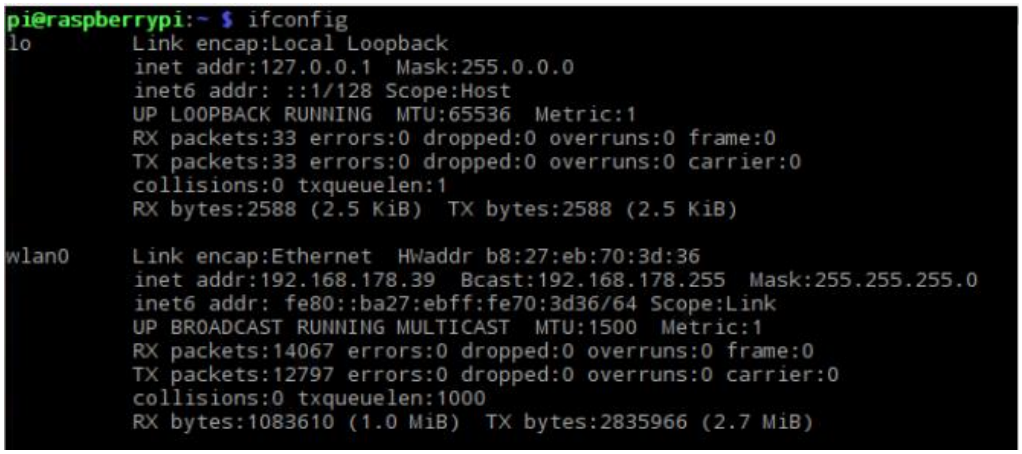

Abbildung 24 – Netzwerkadressen

Wie auf Abbildung 24 zu sehen ist, hat der Raspberry Pi die Netzwerkadresse 192.168.178.39. Dieses Beispiel zeigt ein Klasse C Netzwerk, dass nicht in weitere Subnetze unterteilt wurde. Dies ist erkennbar anhand der Subnetzmaske: 255.255.255.0. Die Vergabe von IP-Adressen in einem solchen Netzwerk erfolgt üblicherweise über einen DHCP-Server, sobald sich ein neues Netzwerkgerät im Netzwerk angemeldet hat. Aktuelle Router bieten die Möglichkeit angeschlossenen Geräten im Netzwerk bei wiederholtem Verbinden feste IP-Adressen zuzuweisen. In dieser Konstellation kommt eine „FritzBox 7490“ zum Einsatz, in deren Interface eine feste IP-Adresse festgelegt wird. Unter Aufruf eines Browsers und der Eingabe in die Suchleiste von „fritz.box“ erfolgt der Zugriff auf den Router. Durch Eingabe des Passwortes für den Router erfolgt die Weiterleitung in das Hauptmenü. Auf der linken Seite unter folgendem Pfad „Heimnetz → Heimnetzübersicht → Details von 192.168.178.39“ besteht die Möglichkeit den Haken zu setzen, dass immer die gleiche IPv4-Adresse zugewiesen wird, wie in Abb. 25 zu sehen ist.

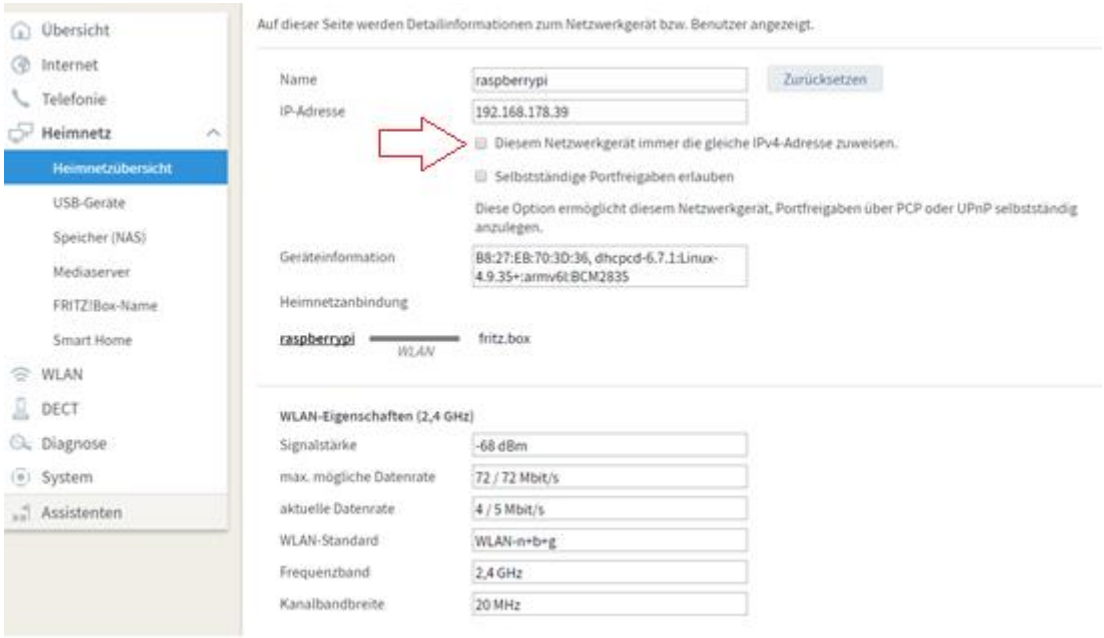

Abbildung 25 - FritzBox 7490 OS: 06.83

Die Vergabe einer statischen IP-Adresse vereinfacht das weitere Vorgehen, da diese für den gesamten Versuchsaufbau nicht mehr geändert werden muss und der Raspberry Pi immer über die gleiche IP-Adresse erreichbar ist.

4.4 Desktop-Sharing per VNC mit RealVNC

Seit September 2016 befindet sich im offiziellen Raspbian Image mit dem Namen Jessie standardmäßig der VNC-Server von RealVNC vorinstalliert. Die Aktivierung ist z.B. über die „raspi-configuration“ möglich. Während der Aktivierung des RealVNC-Server Dienstes über den Desktop unter dem Raspberry-Symbol → Preferences → Raspberry Pi Configurations zu finden ist. Im Reiter Interfaces ist danach der Haken bei VNC auf „enabled“ zu setzen. Durch einen Neustart werden alle Änderungen übernommen. Falls dieser Dienst nicht vorinstalliert ist, kann dieser manuell über die Kommandozeile mit dem nachfolgend aufgezeigten Befehl installiert werden.[48]

```
1  sudo apt-get install realvnc-vnc-server
```

Anschließend ist das RealVNC-Server Symbol rechts oben in der Panel-Leiste wiederzufinden. Dort befindet sich das Optionsmenu für die Angaben der Konfiguration im entfernten VNC-Client, wie in Abb. 26 dargestellt. Um

[48] (Raspberry Pi: Erste Schritte bei der Konfiguration, 2016)

eine Verbindung mit dem RealVNC-Client aufzubauen, wird die für alle gängigen Betriebssysteme passende Software von RealVNC benötigt.

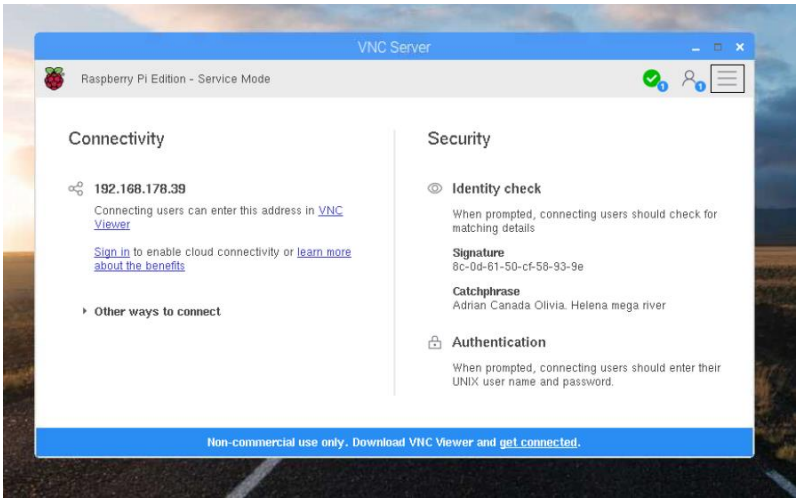

Abbildung 26 - RealVNC Server

Nach der Installation erhält der Client aus dem Netzwerk die IP-Adresse des RealVNC-Servers, der auf dem Raspberry Pi automatisch läuft. Vor diesem Hintergrund muss eine manuelle Eingabe der IP-Adresse nicht erfolgen. Im nächsten Schritt wird der Benutzername und das Passwort des Raspberry PI erfragt, welche standartmäßig „Pi“ und „raspberry“ sind.[49] Um den Server wieder auszuschalten, reicht es in den Einstellungen des Raspberry Pi den Knopf auf „disable“ zu verändern. Die Vorteile bei der Benutzung sind zum einen der einfache Zugriff auf die Raspberry Pi Konfiguration sowie das automatische Aktivieren des RealVNC-Server nach dem Starten und zum anderen die Steuerung per Remote-Host unter Verwendung von einem Bildschirm, ohne dass hin und her gewechselt werden muss oder ein weiterer benötigt wird. Durch eine funktionierende Zwischenablage zwischen dem VNC-Client und dem Remote-Host ermöglicht dies einen einfachen Datenaustausch in der Zwischenablage. Eine stabile Version für den Raspberry Pi rundet das Gesamtpaket ab.[50]

4.5 Verbindung per SSH aufbauen

Um eine Verbindung zu dem Raspberry Pi per SSH aufbauen zu können wird ein SSH-Client sowie die IPv4-Adresse des Raspberry Pi benötigt, die

[49] (Raspberry Pi: Fernwartung und Remote-Desktop mit VNC,RDP und SSH, 2016)
[50] (Kofler, 2016)

automatisch beim Verbinden in ein Netzwerk generiert wird. Bei Computersystemen, wie Mac OS oder Linux, ist der SSH-Client bereits unter der Kommandozeile integriert. Es wird lediglich noch ein Terminal-Programm benötigt. Im Gegensatz hierzu wird für Windows zusätzlich noch ein SSH-Client benötigt, wie z.B. PuTTY. SSH wurde bereits zur Installation unter den Konfigurationsmöglichkeiten beim Raspberry Pi eingestellt und lässt sich nun verwenden. VNC ist für den allgemeinen Gebrauch gut geeignet, da es sich einfach steuern lässt. Benötigt man jedoch weitere Konfigurations-möglichkeiten, oder den Datentransfer von mehreren Dateien, eignet sich ein SSH-Client besser, da er die Möglichkeit des Drag & Drop mit sich bringt sowie eine grafische Oberfläche zur besseren Verwaltung aufweist. Der Datenaustausch mehrerer Daten wird somit einfach und übersichtlich dargestellt und ist weniger fehleranfällig.

4.6 Installation und Montage der Motorsteuerung

Um die Motorsteuerung anbringen zu können und über diese mittels Impulsen das Fahrzeug in Bewegung versetzen zu können, wird das komplette Modellauto in seine Einzelteile zerlegt. Zuvor jedoch ist es wichtig, dass der Akku vom Fahrzeug entfernt worden ist, um eventuelle Schäden vorzubeugen. Das Modelauto wird bis auf die Grundplatte zerlegt, um an die Steuereinheit des Fahrzeuges zu gelangen, wie in Abb. 27 zu sehen ist. Dabei ist der Aufbau im Gegensatz zu der genauen Zuordnung aller Kabel nicht so wichtig. Vor diesem Hintergrund ist zwischen den Motoren für den Antrieb, die Lampen, die Akkueinheit und der Lenkung zu unterscheiden. Sofern alle Kabel von der Steuereinheit entfernt wurden, empfiehlt es sich, für die leichtere Identifikation der zugehörigen Kabel beim Zusammensetzen alle zusammengehörigen Kabel zu beschriften.

Abbildung 27 - Einbau Motorsteuerung

Um die Funktionsweise zu testen, ist es ausreichen die drei Motoren, die für eine Bewegung in Vorwärts- und Rückwärtsrichtung verantwortlich sind, miteinander zu verdrillen. Dabei werden alle Kabel (rot bedeutet stromführend; schwarz bedeutet Erdung) einen Zentimeter weit ab isoliert, um Kontakt miteinander herstellen zu können. Die drei roten Kabel und die drei schwarzen Kabel werden miteinander verdrillt. Der Motor für die Lenkung wird ebenfalls ab isoliert sowie die Kabel die an den Akku angeschlossen werden. Die Kabel werden folgendermaßen an die Motorsteuerung angeschlossen:

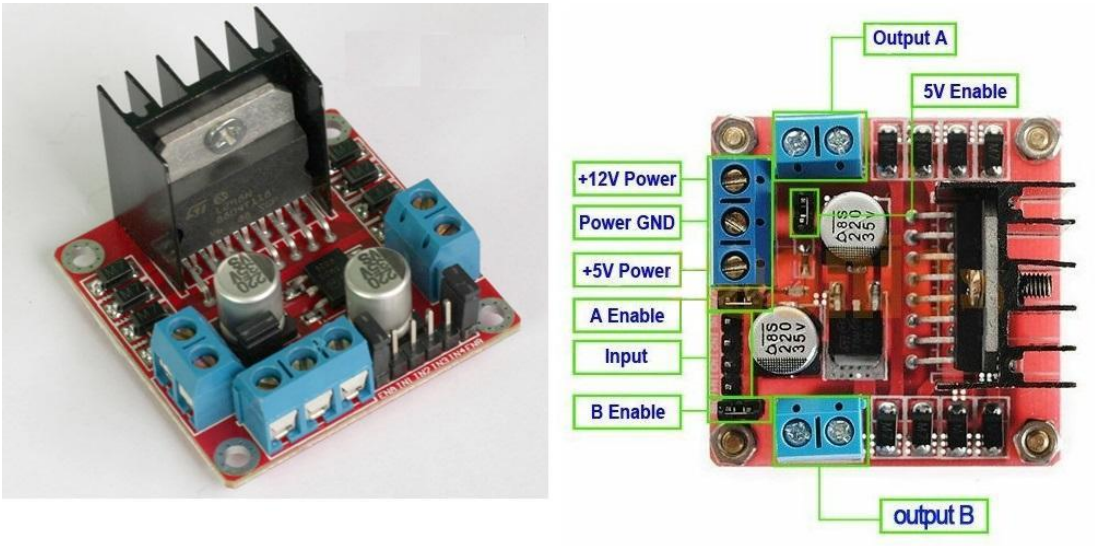

Abbildung 28 - Motorsteuerung Bezeichnung[51]

Die verdrillten Kabel für die drei Motoren, die für die Bewegung in Vorwärts- und Rückwärtsrichtung verantwortlich sind, werden an den Output A angeschlossen. Dabei ist die Anordnung der unterschiedlichen Kabel von untergeordneter Relevanz, da es mehr darauf ankommt, dass die Kabel auf der anderen Seite bei Output B in der gleichen Weise angeordnet werden. Dies erleichtert im Nachhinein die Ansteuerung und die Programmierung. Da die Motoren mehr als 5 Volt benötigen, um ihre volle Leistung zu entfalten, wird über den 12 V Strom Anschluss das rote Kabel des Akkus und über Power GND das schwarze Kabel (Erdung) angeschlossen. Für den Input werden Jumperkabel benutzt, um eine Interaktion zwischen dem Raspberry Pi und der Motorsteuerung herzustellen. Dabei ist zwingend erforderlich, dass eine Erdung zwischen Raspberry Pi und der Motorsteuerung vorhanden ist. Hierbei ist es allerdings irrelevant, welche Erdung am Raspberry Pi genutzt wird. Es wird eine der acht möglichen Pins verwendet sowie der Power GND Anschluss auf der Motorsteuerung, indem bereits das schwarze Kabel des Akkus angeschlossen worden ist. Im Folgenden wird eine Übersicht über alle GPIO Pins des Raspberry Pi Zero W vorgestellt sowie die Belegung für die Motorsteuerung.

[51] Quelle: https://www.amazon.de/L298N-Bridge-Stepper-Controller-Arduino-Red/dp/B013QTC18K/ref=sr_1_4_mimg_1_ce_display_on_website?ie=UTF8&qid=1510063675&sr=8-4&keywords=l298n&dpID=517NpuTjRsL&preST=_SY300_QL70_&dpSrc=srch; 07.11.2017

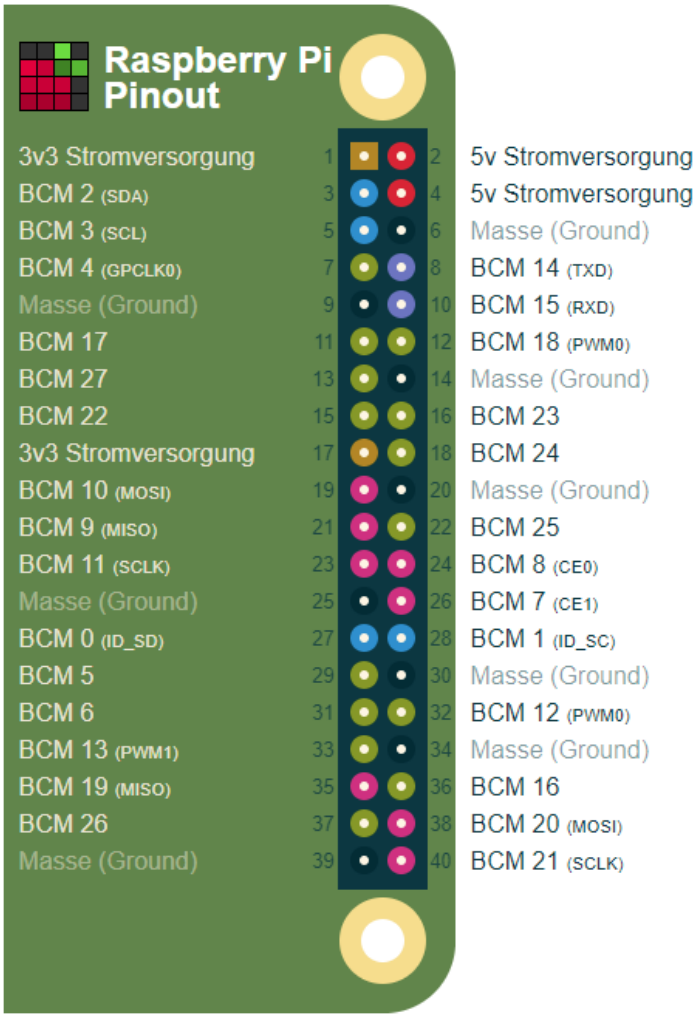

Abbildung 29 - GPIO Pinout Belegung[52]

Die benötigte Masse vom Raspberry Pi an die Motorsteuerung kommt von Pin 9, wie in Abb. 29 zu sehen ist. Die vier Inputkabel wurden mit den Pins 11, 13, 16, 18 in aufsteigender Reihenfolge und auf der Motorsteuerung von oben nach unten verbunden. Hierbei ist es wichtig, dass keinerlei Stromzufuhr erfolgt, bevor nicht alles korrekt verkabelt worden ist, da es sonst zu Schäden an den Platinen und den Motoren kommen kann. Um nun die korrekte Belegung und Ansteuerung testen zu können, wird ein Programm in der Programmiersprache Python verwendet, um die grundlegenden Funktionalitäten testen zu können. In diesem Beispiel wird ausschließlich Python 3 verwendet. Die bereits vorinstallierte Entwicklungsumgebung ist in der oberen linken Ecke, unter dem Raspberry Pi Symbol → Programming → Python 3 (IDLE) zu erreichen. Dadurch öffnet

[52] Quelle: https://de.pinout.xyz/; 07.11.2017

sich ein neues Fenster mit einem leeren Dokument. Falls sich kein neues Fenster öffnet, kann dieses unter File → New File erstellt werden.

```
1  import RPi.GPIO as motor
2  import time
3  from time import*
4
5  def initDrive():
6      motor.setmode(motor.BCM)
7      motor.setup(17, motor.OUT)
8      motor.setup(22, motor.OUT)
9
10 def init():
11     motor.setmode(motor.BCM)
12     motor.setup(23, motor.OUT)
13     motor.setup(24, motor.OUT)
14
15 def forward():
16     initDrive()
17     motor.output(17, False)
18     motor.output(22, True)
19
20 def reverse():
21     initDrive()
22     motor.output(17, True)
23     motor.output(22, False)
24
25 def left():
26     init()
27     motor.output(23, False)
28     motor.output(24, True)
29
30 def right():
31     init()
32     motor.output(23, True)
33     motor.output(24, False)
34
35 def stopDrive():
36     initDrive()
37     motor.output(17, motor.LOW)
38     motor.output(22, motor.LOW)
39
```

```
40 def stopDirection():
41     init()
42     motor.output(23, motor.LOW)
43     motor.output(24, motor.LOW)
44
45 #------MAIN---------
46 forward()
47 sleep(2)
48 reverse()
49 sleep(2)
50 stopDrive()
51 left()
52 sleep(2)
53 right()
54 stopDirection()
```

Dieses Programm beinhaltet die grundlegenden Funktionen des Motors, wie die Fortbewegung und die Lenkung. Die ersten drei Zeilen beschreiben den Import der GPIO Pins, verwendbar unter dem Namen „motor“ und die Bibliothek „time“, um im späteren Verlauf die Zeit anhalten zu können. Andernfalls würden die einzelnen Befehle so schnell hintereinander ausgeführt werden, sodass keine Bewegung wahrgenommen werden würde. In der ersten Definition initDrive() wird eine Klasse der Funktion motor angewendet, die motor.setmode heißt. Der Parameter, der dort übergeben wird, bestimmt die Nummerierung der GPIO Pins für die weitere Verwendung. Insgesamt können nur 2 Parameter (BCM und Board) verwendet werden. Der Parameter BOARD verwendet die Nummerierung, die auf die Platine gedruckt ist, während der Parameter BCM die Pins nach „Broadcom SOC Channel“ referenziert. Beide Optionen bringen weder Vor- noch Nachteile mit sich, allerdings ist eine einheitliche Verwendung im weiteren Programm unerlässlich. Alle Definitionen übernehmen eine Aufgabe und sind klar voneinander getrennt. Zudem weisen sie ein klares Design auf und eine gute Struktur erleichtert das experimentelle Arbeiten. Zur Entgegennahme von Kommandos definiert die Anweisung motor.setup (17, motor.OUT) den Pin 17 (BCM) als Output. Die Zeile 22 des „motor.output“ (17, True) sowie „motor.output“ (22, False) sorgt für die Stromzuführung an die Motoren. Dabei muss die Rotationsrichtung durch Versuchen bestimmt werden, da die Motoren bereits fest verbaut worden und

die Einbaurichtung sich nicht ablesen lässt. In der Definition stopDrive() sowie stopDirection() wird die Stromzufuhr auf allen verwendeten Pins unterbrochen. Dieses Konzept sorgt für einen genauen Überblick aller benutzten Pins und deren Status. In Zeile 45 beginnt das Hauptprogramm mit dem Raute Zeichen, das einen einzeiligen Kommentar einleitet. Es besteht die Möglichkeit die Funktionen mit einer Variablen aufzurufen, wie z.B. mit einer Zeitangabe. Der Aufruf der Funktion würde dann entsprechend angepasst werden forward(int time), sodass das Auto sich in Vorwärtsrichtung um die Zeitangabe „time bewegt“, bevor es zum Stillstand kommt. In der Praxis zeigt es sich allerdings, dass nicht im Vorhinein bestimmt werden kann, wie lange die Taste gedrückt wird. Deshalb müssen die einzelnen Funktionen dann ausgeführt werden, sobald diese im Zusammenhang mit einem Kontroller ausgelöst werden.

4.7 Steuerung mit einem PS4 Controller

Für die Steuerung des Modelautos wird ein handelsüblicher PlayStation 4 DualShock Controller der Marke Sony verwendet. Um diesen einzuschalten, ist der Einschaltknopf in der Mitte des Controllers zu betätigen. Zudem ist die Bluetooth-Funktion bei dem Raspberry Pi zu aktivieren, die sich oben rechts auf dem Desktop befindet sowie die in den Standardeinstellung deaktiviert ist und damit ausgegraut erscheint. Durch die Auswahl des ersten Auswahlpunktes „Turn On Bluetooth“ ist der Raspberry Pi auch für andere Geräte sichtbar. Zur Verbindung der beiden Geräte müssen sich diese in Reichweite voneinander befinden und die Bluetooth-Funktion muss eingeschaltet sein. Aufgrund dessen, dass sich auf der Seite des PlayStation 4 Controllers kein Interface befindet, muss dies auf der Seite des Raspberry Pi erfolgen. Der dritte Punkt unter dem oben beschriebenen Bluetooth Symbol des Raspberry Pi enthält die Möglichkeit ein neues Bluetooth-fähiges Gerät mit dem Raspberry zu verbinden, wie in Abb. 30 zu sehen ist.

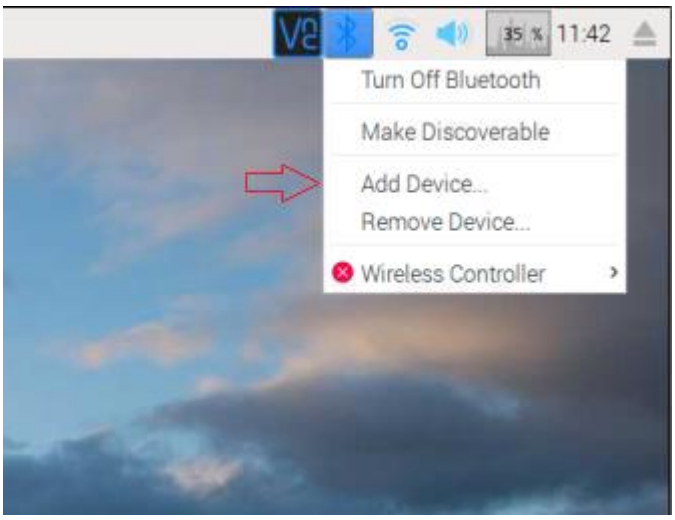

Abbildung 30 - DualShock Controller Pairen mit dem Pi

Das Gerät, das verbunden werden soll muss bei erstmaliger Benutzung manuell ausgewählt werden. In diesem Fall handelt es sich um den PS4 Controller. Bei den darauf folgenden Verbindungsversuchen funktioniert dies automatisch und bedarf keiner weiteren Konfiguration durch das Interface des Raspberry Pi, wie in Abb. 31 abgebildet.

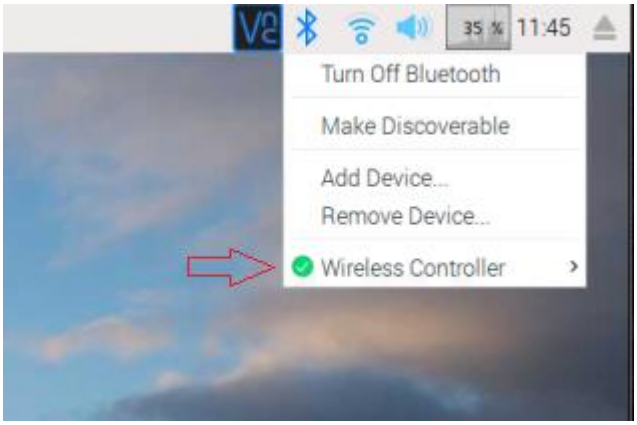

Abbildung 31 - PS4 Controller via Bluetooth verbinden

Um nun alle Knöpfe und Funktionalitäten des Dual Shock 4 Controllers vollständig verwenden zu können, müssen diese ausgelesen werden. Dafür wird ein Grundgerüst von dem PyGame[53] benutzt, das sich bereits mit unterschiedlichen Ebenen der Spieleprogrammierung auseinandersetzt. Für den hier vorliegenden Fall wird nur die Controller „engine“ verwendet. Dabei hat diese den Vorteil, dass sie mit jedem angeschlossenem JoyStick, Gamepad oder Controller interagieren und diesen auslesen kann. Durch das

[53] https://www.pygame.org/docs/

Herunterladen des Beispielcodes[54] für das JoyStick Module und das Einbinden in Python wird die Möglichkeit geboten bereits durch eine General User Interface (kurz: GUI) alle Funktionalitäten des angeschlossenen Gerätes zu verwenden und zu testen. Nicht nur die Knöpfe, sondern auch die Gyrosensoren im DualShock Controller, können auf diese Weise ausgelesen werden. Bevor das PyGame ausgeführt wird, muss der Controller bereits mit dem Raspberry Pi verbunden sein, da ansonsten die Initialisierung zu Beginn des Programmes nicht funktioniert und kein Controller gefunden wird. Wurde der Controller zuvor verbunden und danach das Spiel über die Entwicklungsumgebung gestartet, werden alle Knöpfe, Steuerkreuze, Potentiometer und Gyrosensoren angezeigt und übermitteln eine Echtzeitaufnahme ihres aktuellen Status.

```
1  import pygame
2
3  # Define some colors
4  BLACK    = (   0,   0,   0)
5  WHITE    = ( 255, 255, 255)
6
7  # This is a simple class that will help us print to the screen
8  # It has nothing to do with the joysticks, just outputting the
9  # information.
10 class TextPrint:
11     def __init__(self):
12         self.reset()
13         self.font = pygame.font.Font(None, 20)
14
15     def print(self, screen, textString):
16         textBitmap = self.font.render(textString, True, BLACK)
17         screen.blit(textBitmap, [self.x, self.y])
18         self.y += self.line_height
19
20     def reset(self):
21         self.x = 10
22         self.y = 10
23         self.line_height = 15
24
25     def indent(self):
```

[54] https://www.pygame.org/docs/ref/joystick.html

```
26          self.x += 10
27
28      def unindent(self):
29          self.x -= 10
30
31
32  pygame.init()
33
34  # Set the width and height of the screen [width,height]
35  size = [500, 700]
36  screen = pygame.display.set_mode(size)
37
38  pygame.display.set_caption("My Game")
39
40      #Loop until the user clicks the close button.
41      done = False
42
43  # Used to manage how fast the screen updates
44  clock = pygame.time.Clock()
45
46  # Initialize the joysticks
47  pygame.joystick.init()
48
49  # Get ready to print
50  textPrint = TextPrint()
51
52  # -------- Main Program Loop -----------
53  while done==False:
54      # EVENT PROCESSING STEP
55      for event in pygame.event.get(): # User did something
56          if event.type == pygame.QUIT: # If user clicked close
57              done=True
58       # Flag that we are done so we exit this loop
59
60        # Possible joystick actions: JOYAXISMOTION JOYBALLMOTION
61        JOYBUTTONDOWN JOYBUTTONUP JOYHATMOTION
62          if event.type == pygame.JOYBUTTONDOWN:
63              print("Joystick button pressed.")
64          if event.type == pygame.JOYBUTTONUP:
65              print("Joystick button released.")
66      # DRAWING STEP
```

```
67    # First, clear the screen to white. Don't put other drawing
68      commands
69    # above this, or they will be erased with this command.
70     screen.fill(WHITE)
71     textPrint.reset()
72
73    # Get count of joysticks
74     joystick_count = pygame.joystick.get_count()
75
76     textPrint.print
77   (screen, "Number of joysticks: {}".format(joystick_count) )
78     textPrint.indent()
79
80    # For each joystick:
81     for i in range(joystick_count):
82         joystick = pygame.joystick.Joystick(i)
83         joystick.init()
84
85         textPrint.print(screen, "Joystick {}".format(i) )
86         textPrint.indent()
87
88         # Get the name from the OS for the controller/joystick
89         name = joystick.get_name()
90         textPrint.print
91      (screen, "Joystick name: {}".format(name) )
92
93         # Usually axis run in pairs, up/down for one, and
94                                      left/right for
95         # the other.
96         axes = joystick.get_numaxes()
97         textPrint.print
98     (screen, "Number of axes: {}".format(axes) )
99         textPrint.indent()
100
101            for i in range( axes ):
102                axis = joystick.get_axis( i )
103                textPrint.print
104         (screen, "Axis {} value: {:>6.3f}".format(i, axis) )
105            textPrint.unindent()
106
107            buttons = joystick.get_numbuttons()
```

```
108             textPrint.print
109             (screen, "Number of buttons: {}".format(buttons) )
110             textPrint.indent()
111
112             for i in range( buttons ):
113                 button = joystick.get_button( i )
114                 textPrint.print
115             (screen, "Button {:>2} value: {}".format(i,button) )
116             textPrint.unindent()
117
118             # Hat switch. All or nothing for direction,
119             not like joysticks.
120             # Value comes back in an array.
121             hats = joystick.get_numhats()
122             textPrint.print
123             (screen, "Number of hats: {}".format(hats) )
124             textPrint.indent()
125
126             for i in range( hats ):
127                 hat = joystick.get_hat( i )
128                 textPrint.print
129             (screen, "Hat {} value: {}".format(i, str(hat)) )
130             textPrint.unindent()
131
132             textPrint.unindent()
133
134
135         # ALL CODE TO DRAW SHOULD GO ABOVE THIS COMMENT
136
137         # Go ahead and update the screen with what we've drawn.
138         pygame.display.flip()
139
140         # Limit to 20 frames per second
141         clock.tick(20)
142
143     # Close the window and quit.
144     # If you forget this line, the program will 'hang'
145     # on exit if running from IDLE.
146                                 pygame.quit ()
```

Von Zeile 10 bis Zeile 32 befindet sich die Klasse TextPrint, die sich ausschließlich um die Ausgabe in der GUI kümmert. Von Zeile 34 bis 50 wird

das sichtbare Fenster, dass sich bei der Initialisierung öffnet, konfiguriert und initialisiert sowie die Überschriften und die Aktualisierungsrate festgelegt. Das Hauptprogramm beginnt ab Zeile 52 und beschreibt die möglichen Joystick- Aktionen, die ausführbar sind. Diese werden in sogenannte Eventtypen kategorisiert. Für jede Einheit auf dem Controller gibt es einen Eventtypen, der durch das Drücken oder das Loslassen der Knöpfe angesprochen wird. Alle Funktionalitäten auf dem Controller interagieren und lassen eine Veränderung in der grafischen Oberfläche des PyGame deutlich werden, wie in Abb. 32 deutlich wird. Für die 14 Knöpfe auf dem Controller wird als Wert der Knöpfe eine 1 dargestellt für den Fall, dass der Knopf gedrückt worden ist. Ansonsten wird eine 0 dargestellt. Außerdem wird registriert, ob der Knopf gehalten wird oder nur kurz gedrückt worden ist. Dieser Event-Type im PyGame nennt sich

```
1  pygame.JOYBUTTONDOWN
```

und

```
1  pygame.JOYBUTTONUP
```

Um den Status einer der 14 Knöpfe auf dem Controller zu erfahren wird der folgende Befehl verwendet.

```
1  joystick.get_button()
```

Dabei muss der Funktion die Nummer des auszulesenden Knopfes als Parameter übergeben werden. Im Gegensatz zu den Achsen auf dem Controller hat der Status der Knöpfe entweder den Wert 0 oder 1. Die beiden Joysticks auf dem Controller haben jeweils zwei Achsen. Dabei ist der Ausgangswert aller Achsen bei 0. Der Event-Type dazu ist wie folgt.

```
1  pygame.JOYAXISMOTION
```

Die Achsen sind unterteilt in horizontaler und vertikaler Ausrichtung und haben eine Genauigkeit auf ein Tausendstel. Wird der linke Stick nach oben geschoben, weist der Wert für die horizontale Achse -1.000 auf. Wird der Stick nach unten geschoben, verändert sich der Wert auf 1.000. So verhält es sich ebenfalls mit der vertikalen Achse und dem anderen Joystick.

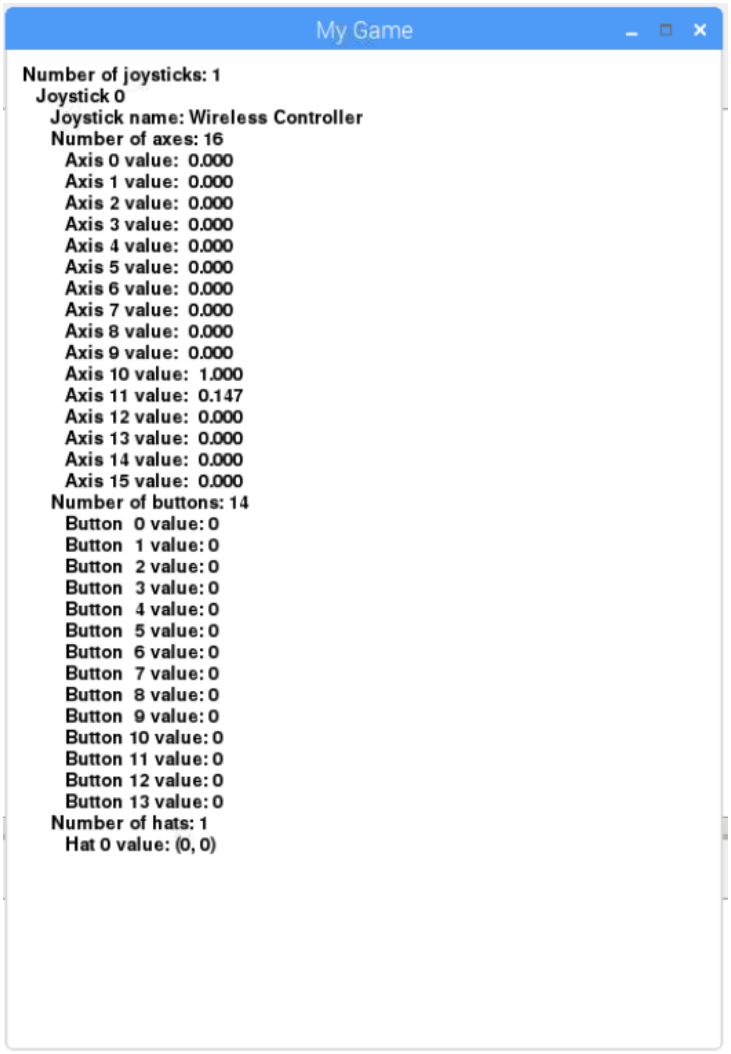

Abbildung 32 – PyGame & PS4 Controller

Insgesamt befinden sich drei Gyrosensoren innerhalb des PlayStation 4 Controllers, die abhängig von der aktuellen Lage des Controllers seine Richtung angeben. Um den Wert einer Achse auszugeben, wird der Befehl verwendet.

```
Joystick.get_axis()
```

Zusammen mit dem Übergabeparameter der Achse dessen Wert man erfahren möchte. Als Rückgabewert erhält man eine Kommazahl, mit drei Nachkommastellen, die zwischen den Werten +1 und -1 liegt. Um den Wert des Steuerkreuzes zu erfahren wird der folgende Befehl benutzt.

```
Joystick.get_hat()
```

Der Rückgabewert setzt sich aus zwei Zahlen zusammen. Die erste Zahl beschreibt die Richtung in der Horizontalebene, während die zweite Zahl die

Richtung in der Vertikalebene beschreibt. Die Werte haben standardmäßig einen Wert von 0 und können entweder -1 oder +1 annehmen bei Betätigung dieser Knöpfe. Es gibt noch eine weitere Funktion, die aber bei diesem Beispiel aufgrund des Controllers nicht zum Einsatz kommt.

```
1  Joystick.get_ball()
```

Diese Funktion liefert die relative Bewegung seit dem letzten Aufruf. Die Darstellung erfolgt über ein Wertepaar mit ganzzahligen Werten. Ähnlich zu anderen Autorennspielen wird auch bei dieser Version die Belegung der Tasten erfolgen. Um vorwärts zu fahren, wird die Taste R2 gedrückt, während dessen für das Rückwärtsfahren die Taste L2 zu drücken ist und für die Änderung der Richtung kann das Steuerkreuz sowie der linke Joystick verwendet werden. Bei dieser Konfiguration kommen unterschiedliche Event-Typen zum Einsatz, wie bspw. Knöpfe, Achsen, Joystick und ein Steuerkreuz. Zu Beginn der Konfiguration ist es hilfreich, wenn alle Knöpfe auf dem Controller einmal ausprobiert und die Funktionalität der jeweiligen Knöpfe notiert werden, um später einfacher die gewünschten Knöpfe und Aktionen ansprechen zu können. Das PyGame wird um den Teil der Motorsteuerung, das im vorherigen Kapitel bereits angesprochen wurde, erweitert. Dabei hilft die Modularität der einzelnen Definitionen, die einfach in das PyGame kopiert werden können. Definitionen sind gekennzeichnet mit dem Anfangswort „def“ und können oberhalb der „Main“ hinein kopiert werden, um den Rumpf des Programmes übersichtlich zu gestalten und die Wartbarkeit für Veränderungen zu erhalten.

```
1  if  event.type == pygame.JOYAXISMOTION or
2  event.type == pygame.JOYHATMOTION:
3
4      direction = joystick.get_axis(0)
5      hatPosition = joystick.get_hat(0)
6
7      if direction < 0 or hatPosition == (-1, 0):
8          left()
9      if direction > 0 or hatPosition == (1, 0):
10         right()
11     if direction == 0 and hatPosition == (0, 0):
12         stopDirection()
```

Diese Funktion ermöglicht es die Steuerung sowohl über das Steuerkreuz als auch über den linken Joystick durchzuführen. Das Event kann nun über die Pfeiltasten oder den Stick ausgelöst werden, während der Status von beiden überprüft wird und die entsprechende Funktion für das Ändern der Richtung aufgerufen wird. Für die Fortbewegung werden die Tasten L2 und R2 auf dem Controller verwendet, die zum Event-Type ButtonDown gehören.

```
1  if  event.type == pygame.JOYBUTTONDOWN:
2
3       rev = joystick.get_button(6)
4       throttle = joystick.get_button(7)
5
6      #Throttle
7      if throttle == True:
8          forward()
9       if rev == True:
10          reverse()
```

4.8 LCD Display steuern via Druckknopf

Um das HD44780 LCD Display per I2C mit dem Raspberry Pi anzusteuern, wird eine zusätzliche Software benötigt. Diese lässt sich auf der Internetseite[55], welche eine Hilfestellung im Umgang mit dem Display darstellt oder über den Raspberry Pi in der Kommandozeile mit dem Kommandozeilenbefehl herunterladen.

```
1  mkdir hd44780 && cd hd44780
2  wget http://tutorials-raspberry.de/wp/content/
       uploads/scripts/hd44780_
3  unzip hd44780_i2c.zip
```

Bevor der Code verwendet werden kann, wird das I2C Tool benötigt, welches vorher installiert werden muss. Über die Kommandozeile lautet der Befehl wie folgt.

```
1  Sudo apt-get install python-smbus i2c-tools
```

[55] https://tutorials-raspberrypi.de/hd44780-lcd-display-per-i2c-mit-dem-raspberry-pi-ansteuern/

Danach muss in den Einstellungen des Raspberry Pi unter dem nachfolgenden Befehl das I2C freigeschaltet werden, insofern es noch nicht vorher freigeschaltet wurde.

```
1  Sudo raspi-config
```

Unter dem achten Punkt „Advanced Options“ wird es aktiviert. Am Ende der „modules“ Datei wird noch der entsprechende Eintrag hinzugefügt.

```
1  Sudo nano /ect/modules
```

Durch den dargestellten Befehl können Änderungen an der Datei „modules“ durchgeführt und im Editor geöffnet werden. An das Ende der Datei kommen die folgenden zwei Anweisungen.

```
1  I2c-bcm2708
2  i2c-dev
```

Anschließend muss der Raspberry Pi neugestartet werden, damit alle Änderungen in Kraft treten. Dies kann mit dem folgenden Befehl ausgeführt werden.

```
1  Sudo reboot
```

Um das Display zu testen, muss es zuvor angeschlossen werden. Aufgrund des I2C-Controllers lässt sich das Display nun über nur 4 Pins ansteuern, von denen zwei für die Stromversorgung zuständig sind. VCC auf der Platine wird an einen 5 V Anschluss des Raspberry Pi angeschlossen. In diesem Aufbau wurde der Pin mit der Platinen Nummer 2 ausgewählt, der eine Spannung von 5 Volt aufweist. GND von der Platine wird an den Pin des Raspberry Pi 6 angeschlossen, der Masse aufweist. SDA und SCL sind die beiden Pins über die die Datenverbindung zustande kommt. Diese sind in dem Aufbau die Pinnummern 3 für SDA und 5 für SLC, da diese die Datenleitung des I2C-Bus darstellen.[56] Nachdem das LCD-Display angeschlossen ist, kann das Modul nun getestet werden mit dem folgenden Befehl.

```
1  Sudo i2cdetect –y 1
```

Die Ausgabe sollte folgende sein, wie in Abb.33 dargestellt.

[56] https://de.pinout.xyz/

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- 27 -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Abbildung 33 - i2cdetect

Sollte eine andere Zahl als die 27 angezeigt werden, muss dies in der Datei lcddriver.py in Zeile 8 auf die angezeigte Adresse geändert werden. Der Beispielcode ist für ein 20x4 Zeichen Modul ausgelegt. Dieser kann aber auch, wie hier aufgezeigt, von einem 16x2 LCD-Display genutzt werden.

```
1  import lcddriver
2  from time import *
3  lcd = lcddriver.lcd()
4  lcd.lcd_clear()
5  lcd.lcd_display_string("Tutorials-", 1)
6  lcd.lcd_display_string("RaspberryPi.de", 2)
```

Dabei muss das Python Programm im selben Verzeichnis liegen, wie die beiden Dateien vom LCD-Modul. Ansonsten muss beim Import noch der Pfad dieser Dateien angegeben werden. Um Zeichen an das Modul schicken zu können, wird der Befehl lcd.lcd_display_string() verwendet, Dieser nimmt 2 Parameter entgegen (eine Zeichenkette und einen ganzzahligen Wert). Die beiden Parameter stehen für die Ausgabe und die Zeile, in der der Text nach der Ausführung angezeigt werden soll. Um einen oder mehrere Zeichen zu verändern, muss der gesamte Text übermittelt werden. Es ist nicht möglich einzelne Buchstaben zu ersetzen ohne den Rest zu aktualisieren.[57] Für die Einstellung und Veränderung des Kontrastes vom Display befindet sich an dem I2C Modul auf der Rückseite des LCD-Display ein Rädchen mit dem dieser eingestellt werden kann. Ein guter Kontrast kann erzielt werden, indem die dargestellte Schrift auch bei einem schrägen Winkel gut ablesbar ist. In der lcddriver.py Datei befinden sich noch weitere Funktionalitäten, um

[57] https://tutorials-raspberrypi.de/hd44780-lcd-display-per-i2c-mit-dem-raspberry-pi-ansteuern/

das Display unterschiedlich einzustellen, wie bspw. das Einschalten der Hintergrundbeleuchtung oder das Löschen des gesamten Displays. Darüber hinausgehende Funktionalitäten lassen sich in einem Python Programm definieren. Da es die einzige Schnittstelle im späteren Gebrauch repräsentiert, sind Informationen über das System und den Status nützlich. Sowohl Systeminformationen als auch die das Modellfahrzeug betreffende Funktionalitäten lassen sich wie folgt implementieren. Damit wird auf dem System die IP-Adresse, das Gateway und der Hostname des Raspberry Pi mit einer Unterbrechung von zwei Sekunden angezeigt.

```
1  def showNetwork():
2      gw = os.popen("ip -4 route show default").read().split()
3      s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
4      s.connect((gw[2],0))
5      ipaddr = s.getsockname()[0]
6      gateway = gw[2]
7      host = socket.gethostname()
8      lcd.lcd_display_string("IP Adresse:", 1)
9      lcd.lcd_display_string(ipaddr, 2)
10     sleep(2)
11     lcd.lcd_clear()
12     lcd.lcd_display_string("Gateway:", 1)
13     lcd.lcd_display_string(gateway, 2)
14     sleep(2)
15     lcd.lcd_clear()
16     lcd.lcd_display_string("Hostname:", 1)
17     lcd.lcd_display_string(host, 2)
18     sleep(2)
```

Die CPU Temperatur lässt durch folgenden Code darstellen.

```
1  def getCPUtemp():
2      lcd.lcd_clear()
3      res = os.popen('vcgencmd measure_temp').readline()
4      lcd.lcd_display_string("Temperatur CPU:",1)
5      lcd.lcd_display_string(res.replace("temp=","").
6      replace("'\n'",""),2)
7      sleep(2)
```

Die Auslastung der CPU ist insbesondere auch bei Fehlfunktionen und längeren Ausführungszeiten von Nutzen.

```
1  def getCPUload():
```

```
2       lcd.lcd_clear()
3       load = "{:2.0f}%".format(os.getloadavg()[0]*100)
4       lcd.lcd_display_string("CPU Load",1)
5       lcd.lcd_display_string(load ,2)
6       sleep(2)
```

Eine Datums- und Uhrzeitfunktion wird von der Systemzeit übernommen und kann so in unterschiedlichen Zeilen angezeigt werden.

```
1  def showDateTime():
2       lcd.lcd_clear()
3       datum = datetime.datetime.now().strftime("%d.%m.%Y")
4       uhrzeit = datetime.datetime.now().strftime("%H:%M:%S")
5       lcd.lcd_display_string(datum,1)
6       lcd.lcd_display_string(uhrzeit,2)
7       sleep(2)
```

Des Weiteren gibt es drei Funktionen die das Auto betreffen. Zwei Funktionen simulieren ein Blinken in die gewünschte Richtung. Dabei wird diese nicht automatisch ausgeführt, sondern über den Controller vom Benutzer angesprochen.

```
1  def leftBlink():
2       lcd.lcd_clear()
3       for i in range (0,5):
4           lcd.lcd_display_string(chr(255)+chr(255)+chr(255),1)
5           lcd.lcd_display_string(chr(255)+chr(255)+chr(255),2)
6           sleep(0.2)
7           lcd.lcd_clear()
8           sleep(0.2)
9
10 def rightBlink():
11      lcd.lcd_clear()
12      for i in range (0,5):
13          lcd.lcd_display_string
14         ("                    "+chr(255)+chr(255)+chr(255),1)
15          lcd.lcd_display_string
16         ("                    "+chr(255)+chr(255)+chr(255),2)
17          sleep(0.2)
18          lcd.lcd_clear()
19          sleep(0.2)
```

Eine weitere Funktion, die automatisch ausgeführt wird, simuliert ein Blinken in einem Intervall von 0,2 Sekunden, um andere zu warnen, wenn das Auto rückwärtsfährt.

```
1  def driveBackward():
2      lcd.lcd_clear()
3      for i in range (0,5):
4          lcd.lcd_backlight('ON')
5          sleep(0.2)
6          lcd.lcd_backlight('OFF')
7          sleep(0.2)
```

Um die Informationen, den Raspberry Pi betreffend anzuzeigen, wird ein Drucktastenschalter auf dem Fahrzeug verwendet, der durch Drücken die einzelnen Informationen auf dem LCD-Display anzeigt. Um nun den Schalter nutzen zu können, der einen Eingang C1 und zwei Ausgänge besitzt (NC1 und NO1), wird dieser mit dem Raspberry Pi verbunden. Da in dieser Konfiguration nur die Taster Funktion genutzt wird, werden nur C1 und NC1 mit dem Raspberry Pi verbunden. Der Stromkreis zwischen C1 und NC1 wird dann geschlossen, wenn der Taster gedrückt wurde und kann somit interpretiert werden als eingehendes Signal.

```
1  pot.setmode(pot.BCM)
2  pot.setup(26, pot.IN)
```

Der vorhergehende Code setzt den Pin 26 im BCM- Modus als Input. Dieser Input kann dadurch abgefragt werden, dass dieser als Boolean gesetzt wird, wie im nachfolgenden Codeabschnitt zu sehen ist.

```
1  pressed = pot.input(26)
```

Eine Umsetzung dessen ist es, dies in eine Funktion einzubauen, die die Informationen über den Raspberry Pi beim Drücken zurückgibt.

```
1  If (pressed):
2  getCPUtemp()
       counter++
```

Der Counter wird benötigt, um nach Abfrage aller Funktionen, diese von vorne als Endlosschleife beginnen zu lassen. Somit besteht die Möglichkeit alle Funktionen hintereinander mit der Betätigung der Taste ausgeben zu lassen.

4.9 Steuerung über eine Webseite

Für die Steuerung über eine Webseite wird sowohl ein Webserver benötigt als auch eine Webseite, die das Kamerabild als Stream implementiert, um das Fahrzeug steuern zu können. Dafür wird bereits ein bestehendes Raspberry Pi Kamera Web Interface benutzt. Aufgrund der Vielzahl von Applikationen in dem Programm und der hohen Konfigurierbarkeit mit Makro Scripts sowie der Möglichkeit diese auf den gängigsten Browsern zu benutzen, eignet sich dieses Programm für die vorliegende Aufgabenstellung. Dieses wird den Anforderungen des Projektes entsprechend bearbeitet und verändert. Die Installation wurde für die Betriebssysteme Wheezy und Jessie konzipiert und kann hier angewendet werden. Bevor das Programm installiert werden kann, wird eines der beiden Betriebssysteme auf dem Raspberry Pi benötigt. Danach muss in den Konfigurationen des Raspberry Pi die Kameraunterstützung eingeschaltet werden. Es ist hierbei sinnvoll den Raspberry Pi upzudaten, um die neuste Version der installierten Module zu erhalten.

```
1  sudo apt-get update
```

Hiermit wird das Neueinlesen der Paketliste durchgeführt.

```
1  sudo apt-get upgrade
```

Im zweiten Schritt werden Pakete auf verbesserte und aktualisierte Versionen installiert, insofern das möglich ist. Um nun das Programm auf dem Raspberry Pi zu installieren, wird die Kommandozeile mit folgendem Befehl genutzt.

```
1  git clone https://github.com/silvanmelchior/
2  RPI_Cam_Web_Interface.git
```

Damit werden die Dateien, die auf der Internetseite vorhanden sind, geklont und in ein neues Verzeichnis auf dem Raspberry Pi mit dem Namen „RPi_Cam_Web_Interface“ kopiert. Mit diesem Befehl lässt sich das aktuelle Verzeichnis wechseln zu dem die neu kopierten Dateien liegen.

```
1  cd RPI_Cam_Web_Interface
```

Um nun die heruntergeladenen Dateien zu installieren, wird der Befehl in dem aktuellen Verzeichnis ausgeführt.

```
1  ./install.sh
```

Danach öffnet sich in der Konsole das Konfigurationsfenster, um den Webserver auf dem Raspberry Pi aufzusetzen, wie in Abb. 34 zu sehen ist. Dabei müssen einige Einstellungen vorgenommen werden, um die richtige Ausführung für den vorliegenden Anwendungsfall zu gewährleisten.

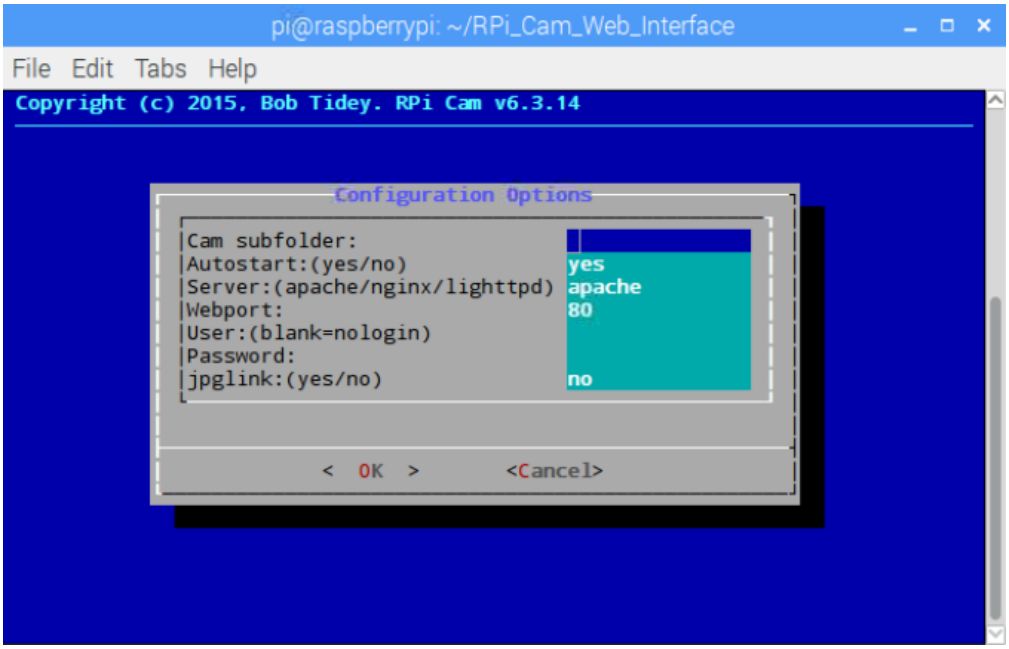

Abbildung 34 - Installationsmenü Webserver

Ein „Cam subfolder“ dient der Vergabe eines Namens für einen Ordner in dem die heruntergeladenen Dateien später installiert werden sollen. In diesem Fall wird unter diesem Punkt nichts eingetragen, da es die Startseite darstellen soll und in keinem Unterverzeichnis liegen darf. Der „Autostart“ wird auf „yes“ gestellt und sorgt dafür, dass beim Starten des Raspberry Pi der Webserver automatisch mitgestartet wird, sobald das Betriebssystem hochgefahren ist. Beim Server stehen drei Alternativen zur Verfügung (apache/nginx/lighttpd). Ein signifikanter Unterschied zwischen den unterschiedlichen Webservern lässt sich bei einer statischen Webseite nicht erkennen.[58]

In diesem Aufbau wurde ein Apache-Webserver zur Realisierung eingesetzt. Der „Webport“ ist standardmäßig auf 80 eingestellt, welches auch den offiziellen Port für HTTP darstellt.[59] Ein Benutzer und ein Passwort sind für den Gebrauch im eigenen Netzwerk nicht nötig. Wird der Raspberry Pi mit einem unbekannten Netzwerk verbunden, ist es jedem möglich auf die

[58] (Ziegelwanger, 2015)
[59] (Adams, 2017)

Internetseite des Raspberry Pi unter Verwendung seiner IP-Adresse zu gelangen. Der letzte Punkt „jpglink“ bietet die Möglichkeit beim Schießen von Fotos, diese mit einem Link weiterzuleiten und einzeln aufzurufen. Diese Option kann ausgeschaltet werden. Nach der Installation ist ein Neustart empfehlenswert, um alle Einstellungen zu übernehmen. Nach dem Einschalten des Raspberry Pi sollte das Kameramodul aktiv sein und der Webserver laufen. Dafür wird ein Browser auf dem Raspberry Pi aufgerufen und in die Suchleiste „localhost“ eingegeben. Alternativ kann auch die IP-Adresse des Raspberry Pi verwendet werden.

Abbildung 35 - Auslastung CPU

Diese Option führt allerdings zu einer 100 % Auslastung der CPU und lässt weitere Handlungen stark verzögern, wie in Abb. 35 gezeigt wird. Ob der Webserver im Netzwerk erreichbar ist, lässt sich von einem anderen netzwerkfähigen Gerät mit einem Browser testen, indem man die IP-Adresse des Raspberry Pi eingibt. Hierbei sollte die Startseite des RPi_Web_Cam_Interface zu sehen sein. Mit der Installation in das Verzeichnis /var/www, welches das Verzeichnis für den Webserver darstellt, wurde darüber hinaus auch ein Verzeichnis /home/pi/RPi_Cam_Web_Interface angelegt. In diesem Verzeichnis befinden sich fünf ausführbare Skripte.

- Install.sh – startet die Installation auf dem Raspberry Pi
- Update.sh – überprüft ob es Updates gab und startet dann die Install.sh
- Start.sh – startet die Software, die auf dem Server läuft
- Stop.sh – stoppt die Software und den Webserver
- Remove.sh – löscht die Software vollständig vom Raspberry Pi

Diese Skripte können über die Kommandozeile ausgeführt werden und ermöglichen den Zugriff auf einzelne Funktionalitäten.

```
./"Skript".sh
```

Um nun Änderungen an dem Interface vornehmen zu können, werden Schreibrechte für das Stammverzeichnis benötigt, welche einem als Pi Benutzer nicht zur Verfügung stehen. Es empfiehlt sich die Rechte des Benutzer Pi unverändert zu lassen, da dieser den Standardbenutzer darstellt und stattdessen lieber zu „root“ zu wechseln. Auf diese Weise besteht dauerhaft oder temporär die Möglichkeit systemweite Änderungen vorzunehmen, ohne „sudo“ vor jede Aktion schreiben zu müssen.

```
1  sudo su
```

Nach der Eingabe des Passwortes wird die normale Shell zur Root-Shell. Um diesen Modus wieder zu verlassen, reicht es den Befehl „exit“ in die Kommandozeile zu schreiben und diesen mit Enter zu bestätigen. In dem Verzeichnis /var/www befindet sich eine Datei mit dem Namen index.php. Diese Datei wird beim Seitenaufruf im Browser geöffnet. Diese beinhaltet alle wichtigen Funktionen sowie die optischen Details und Einstellungen. Um nun das Auto sowohl mit Button auf der Webseite als auch mit den Pfeiltasten auf der Tastatur steuern zu können, muss die Datei index.php angepasst werden. Diese Änderungen können auf dem Raspberry direkt per SSH Zugriff oder über einen FTP-Client und der Bearbeitung auf einem anderen Computer geschehen. Im Body-Teil der HTML-Datei wird ein neuer Bereich definiert mit den folgenden Worten.

```
1  <div>
2  „Inhalt“
3  </div>
```

Um nun vier Buttons zu erstellen, die für die Fortbewegung des Modelautos zuständig sind, wird folgender Code benötigt. Dies dient ausschließlich der optischen Darstellung, damit diese im späteren Verlauf genutzt werden können.

```
1  <div>
2      <center>
3       <br>
4  <button id="myP" onmousedown="mouseDown(02)"
5      onmouseup="mouseUp(02)">Backward</button>
6  <button id="myP" onmousedown="mouseDown(03)"
7      onmouseup="mouseUp(03)">Forward</button>
8  <button id="myP" onmousedown="mouseDown(34)"
9      onmouseup="mouseUp(34)">Turn right</button>
```

```
10 <button id="myP" onmousedown="mouseDown(24)"
11     onmouseup="mouseUp(24)">Turn left</button>
12     <br>
13     </center>
14 </div>
```

Um diese Buttons nun auch drücken und darauf reagieren zu können, wird ein Event-Listener verwendet. Dieser registriert, wann ein Knopf gedrückt worden ist und wann er wieder losgelassen wurde. Der Event-Listener wird mit JavaScript umgesetzt und wird damit in den Head-Teil der Datei geschrieben.

```
1  <script type="text/javascript">
2      window.addEventListener("keydown", keydown,false);
3     window.addEventListener("keyup", ќeyup,false);
4  function keydown(key)
5  {
6      if(key.keyCode =='37'){mouseDown(24);}
7      if(key.keyCode =='38'){mouseDown(03);}
8      if(key.keyCode =='39'){mouseDown(34);}
9      if(key.keyCode =='40'){mouseDown(02);}
10 }
11 function ќeyup(key)
12 {
13     if(key.keyCode =='37'){mouseUp(24);}
14     if(key.keyCode =='38'){mouseUp(03);}
15     if(key.keyCode =='39'){mouseUp(34);}
16     if(key.keyCode =='40'){mouseUp(02);}
17 }
18 </script>
```

Dazu werden zwei Funktionen benötigt, die jeweils den „key“ übergeben bekommen. Der „key“ stellt eine Nummer dar, je nachdem welche Taste auf der Tastatur gedrückt wurde. Dabei hängt der KeyCode für Keyboard Events von dem verwendeten Browser ab. Bei einer Vielzahl von Zeichen und Symbolen stimmen die gängigsten Browser überein.[60] Die KeyCodes für die Pfeiltasten stimmen bei den meisten Browsern überein und eignen sich damit für die Verwendung. Falls der KeyCode mit einem der Events übereinstimmt, kann die Funktion für die Fortbewegung aufgerufen werden. Als Vorlage für

[60] (Wolter, 2012)

die Ansteuerung der Buttons dient ein anderes Projekt.[61] Der Code kann über ein Google Share[62] heruntergeladen werden. Die Datei html.zip enthält fünf Dateien nach dem Entpacken. Der Ordner „html“ kann dem Hauptverzeichnis der Internetseite hinzugefügt werden. In der Datei index.php muss nun noch eine weitere Zeile hinzugefügt werden, um auf die neuen Dateien zugreifen zu können. Diese wird in den Head-Teil der Datei geschrieben und bezieht auf eine JavaScript-Datei, um diese verwenden zu können.

```
1  <script src="camera.js"></script>
```

Somit lässt sich auf alle Funktionen in dem html-Ordner zugreifen. Durch Drücken der Pfeiltasten oder der Buttons wird ein Key-Event ausgelöst, dass die Datei camera.js in dem html-Ordner aufruft. Diese Funktion bekommt den „pin“ und den „Status“ übergeben, der für die jeweilige Richtung steht bzw. ob die Taste gedrückt ist oder nicht. Diese Informationen werden an die Datei „camera_rotate.php“ übergeben, in der die Ausführung des Befehls umgesetzt wird. Hier kommen vier Funktionen zum Einsatz, um das Modelauto in Bewegung zu versetzen. Dabei gibt es pro Richtung zwei Funktionen, um diese einzuleiten und diese wieder zu stoppen, falls der Knopf oder die Taste losgelassen wurde.

```
1  if (($rotate == 1)&&($pin == 03)){
2                                  // run forward
3                                  shell_exec("echo 0 >temp");
4                                  $rotate = 0;
5                                  system("gpio -g write 17 0");
6                                  system("gpio -g write 22 1");
7                                  system("gpio -g write 23 0");
8                                  system("gpio -g write 24 0");
9      }
10 else{
11 if (($rotate == 0)&&($pin == 03)){
12                                 // stop forward
13                                 shell_exec("echo 1 >temp");
14                                 $rotate = 1;
15                                 system("gpio -g write 17 0");
16                                 system("gpio -g write 22 0");
17                                 system("gpio -g write 23 0");
```

[61] (whitebank, 2017)
[62] (whitebank, Raspberry Pi Remote Control Car Camera, 2017)

```
18                                  system("gpio -g write 24 0");
19                                  }
20      }
```

Die Belegung der Pins ist die gleiche wie bei dem PlayStation 4 Controller. Die Nummerierung gleicht der des BCM_GPIO Modus. Dies wird deutlich mit dem Flag – g. Ist dieser nicht gesetzt, wird die Standardnummerierung genutzt.[63] Wird der Pin auf 1 gesetzt, fließt Strom. Wird er jedoch auf 0 gesetzt, wird die Stromzufuhr unterbrochen. Dabei ist es ratsam alle benutzten Pins innerhalb der Funktionen zu definieren, um Seiteneffekte zu vermeiden. Als Ergebnis funktioniert die Steuerung auf der Webseite via Pfeiltasten sowie dem Anklicken der integrierten Buttons.

4.10 Ausführbarkeit aller Komponenten

Um nun alle Funktionalitäten miteinander zu verknüpfen und diese ohne weitere Handlungen, per Tastatur oder Monitor, in den laufenden Betrieb zu integrieren, bedarf es einiger Änderungen. Nach einem Start des Raspberry Pi oder nach einem Neustart soll die Möglichkeit bestehen, dass der Raspberry Pi vollständig hochfährt, damit alle Funktionalitäten verfügbar sind. Des Weiteren soll eine gewisse Zeit eingeräumt werden, in der der Benutzer die Möglichkeit hat seinen Dual Shock 4 Controller mit dem Raspberry Pi zu verbinden. Danach soll der PlayStation Controller im vollen Umfang einsatzbereit sein sowie der Webserver funktionsfähig sein, um auch darüber das Fahrzeug steuern zu können und das übertragene Kamerabild empfangen zu können. Unter anderem wird in diesem Schritt auch das LCD-Display zur weiteren Verwendung in Verbindung mit dem Druckschalttaster initialisiert. Während der gesamten Zeit nach dem Hochfahren wird dem Benutzer auf dem LCD-Display dargestellt, in welcher Phase des Setups dieser sich gerade befindet. Dafür wird ein weiteres Python Programm geschrieben, dass nach dem Hochfahren des Raspberry Pi automatisch gestartet wird und nach der Verbindung mit einem PlayStation 4 Controller die beiden Programme für die Steuerung des Modelautos mit dem Controller sowie für die Informationsausgabe über das LCD-Display startet. Das Programm hat den Namen CountdownToConnectPS4.py und wird in dem

[63] (Wiring Pi, 2017)

Ordner „RPi_testArea“ abgelegt, der sich im Verzeichnis /home/pi/ befindet. Dieses Programm besteht aus zwei Definitionen, die eine ist waiting(), die sofort nach dem Hochfahren ausgeführt wird und einen Countdown von 30 Sekunden beinhaltet, um ausreichend Zeit zum Starten aller Dienste zu gewährleisten. Die zweite Definition ist countdown() und bietet dem Anwender 20 Sekunden Zeit, um den Playstation Controller mit dem Raspberry Pi zu verbinden, bevor das Programm gestartet wird. Nachfolgend wird die Funktion waiting() nach dem Hochfahren aufgezeigt, die 30 Sekunden läuft.

```
1  def waiting():
2      lcd.lcd_clear()
3      lcd.lcd_display_string("booting...", 1)
4      lcd.lcd_display_string("Please wait..", 2)
5      timer = 30;
6      for timer in range(30,0,-1):
7     lcd.lcd_clear()
8     lcd.lcd_display_string("booting...", 1)
9          lcd.lcd_display_string("Please wait " + str(timer)+"s",
   2)
10         sleep(1)
11     lcd.lcd_clear()
```

Die Funktion countdown(), ausgeführt nach waiting().

```
1  def countdown():
2      timer = 20;
3      lcd.lcd_display_string("Time to connect:", 1)
4      for timer in range(20,0,-1):
5          lcd.lcd_display_string(str(timer) + " Sekunden", 2)
6          sleep(1)
7      lcd.lcd_clear()
```

In der „main“ des Programmes wird zuerst das LCD-Display initialisiert bevor die beiden Funktionen aufgerufen werden.

```
1  lcd = lcddriver.lcd()
2  lcd.lcd_clear()
3  waiting()
4  countdown()
```

Um das Programm automatisch nach dem Hochfahren starten zu lassen, wird ein Cronjob benötigt. Dieser ermöglicht es zu speziellen Zeiten

automatisierte Aufgaben (jobs) auszuführen. Sobald ein Cronjob gespeichert wurde, führt das System diesen zu einer festgelegten Zeit auch mehrfach oder in Intervallen aus.[64] Um einen Cronjob festzulegen, wird die Kommandozeile und der folgende Befehl benötigt.

```
1  sudo crontab -e
```

Am Ende des Textes besteht die Möglichkeit einen eigenen Dienst hinzuzufügen sowie eine Startzeit, wann dieser ausgeführt werden soll, wie in Abb.36 in der letzten Zeile abgebildet.

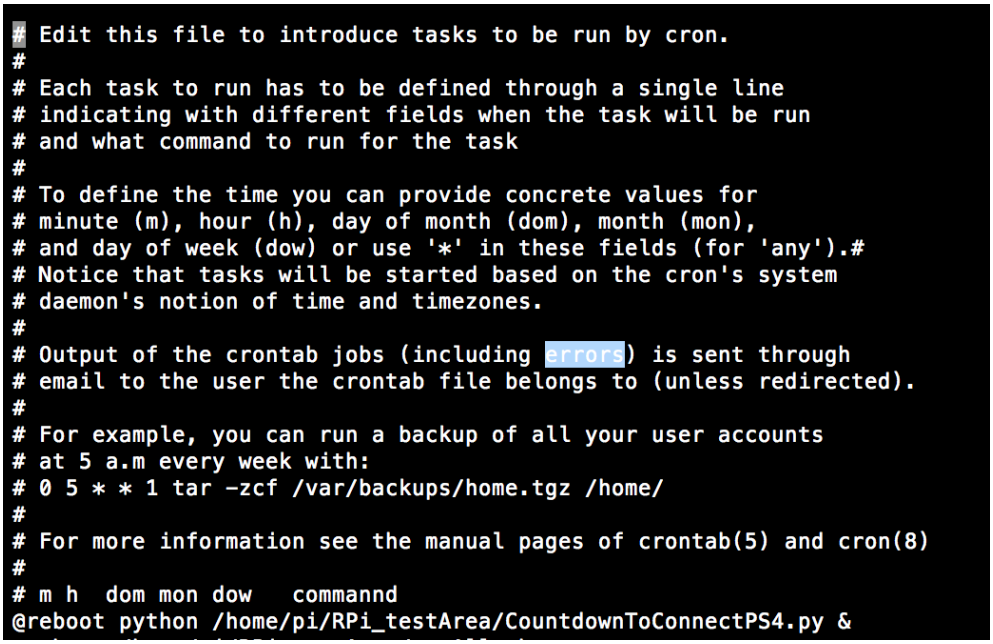

Abbildung 36 - Cronjob CountdownToConnectPS4

Das „@“ zu Beginn der Zeile leitet den Befehl ein sowie das „reboot“ den Zeitpunkt. Dabei ist dieser bereits vordefiniert, wie weitere andere Ausführungszeitpunkte. Hierbei muss Python vor die Datei geschrieben werden, um diese ausführen zu können. Der Pfad in dem die Datei liegt, muss vollständig angegeben werden, da andernfalls die Datei nicht gefunden werden kann. Die beiden Programme pottiTry.py, das für die Steuerung des LCD-Display in Verbindung mit dem Tastendruckschalter verantwortlich ist, und das Programm pyGameAllFunctions.py, das die Steuerung mit dem PlayStation Controller realisiert, müssen parallel ausgeführt werden. Um diese Aufgabe zu realisieren, wurde ein „Bash Script“ geschrieben, das die beiden Dateien ausführt. Bash Skripte eignen sich hervorragend für

[64] (Radtke)

Routineaufgaben. Um ein Skript zu erstellen, reicht ein Editor. In die erste Zeile des Skriptes kommt der sogenannte „Shebang“, ein symbolischer Link zu „sh“, der zur „dash“ weist, die Standardshell für Shellskripte.

```
1  #!/bin/sh
```

Zuvor läuft das Programm CountdownToConnectPS4.py, das insgesamt 50 Sekunden nach dem Ausführen sowie beim Hochfahren in Anspruch nimmt. Zeitgleich wird das Shellskript gestartet, das um 50 Sekunden pausiert und dann die beiden Programme für die weiteren Funktionalitäten ausführt.

```
1  #!/bin/sh
2  sleep 50
3  python3 /home/pi/RPi_testArea/pyGameAllFunctions.py &
4  python3 /home/pi/RPi_testArea/pottiTry.py &
```

Wie auch zuvor wird das Programm benötigt mit dem es ausgeführt werden soll. In diesem Fall wird python3 verwendet, damit es bei der Ausführung zu keinen Problemen aufgrund der Compilerwahl sowie des vollständigen Pfads zu den beiden Dateien kommt. Dieses Bashskript wird ebenfalls zu der Liste der Cronjob hinzugefügt, so wie in Abb.37 beschrieben.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   commannd
@reboot python /home/pi/RPi_testArea/CountdownToConnectPS4.py &
@reboot /home/pi/RPi_testArea/runAll.sh
```

Abbildung 37 - Cronjob runAll.sh

Der Befehl in die Kommandozeile, löst einen Neustart des Raspberry Pi aus und lässt die zuvor definierte Abfolge von Cronjobs überprüfen.

```
1  sudo reboot
```

Der Webserver wurde bereits im Installationsmenu bei der Funktion „Autostart“ auf „yes“ eingestellt. Somit muss keine weitere Handlung nach dem Einschalten des Raspberry Pi erfolgen, da der Webserver automatisch gestartet wird. Um den NFC-Chip auf dem Modelauto zu beschreiben, wird ein NFC-Fähiges Gerät und ein Programm benötigt, mit dem es möglich ist den Chip zu beschreiben. Auf einem Android Handy kann die App „Trigger Writer“ verwendet werden. Die NFC Funktion muss auf dem Handy eingeschaltet sein und das Programm „Trigger Writer“ gestartet werden, wie in Abb. 38 dargestellt.

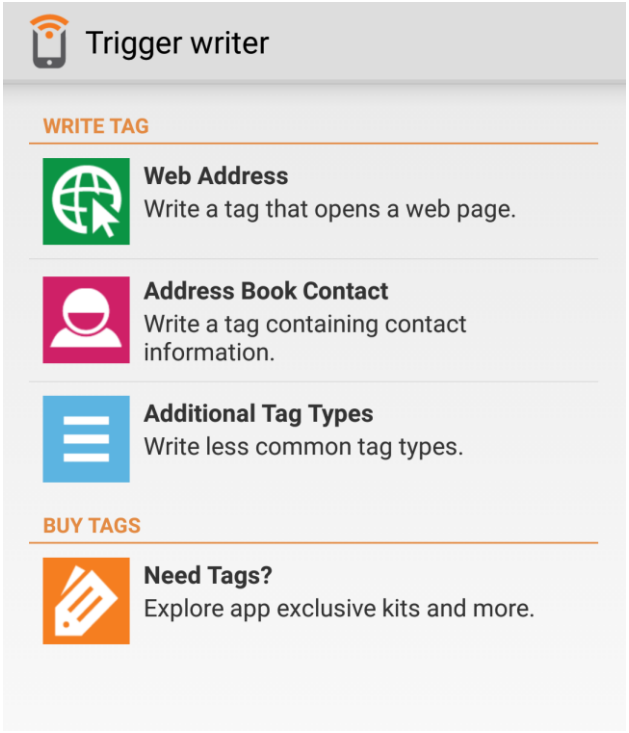

Abbildung 38 - Trigger Writer Startmenü

Unter dem Reiter „Write Tag“, lässt sich ein Web Adressen Tag einfügen.

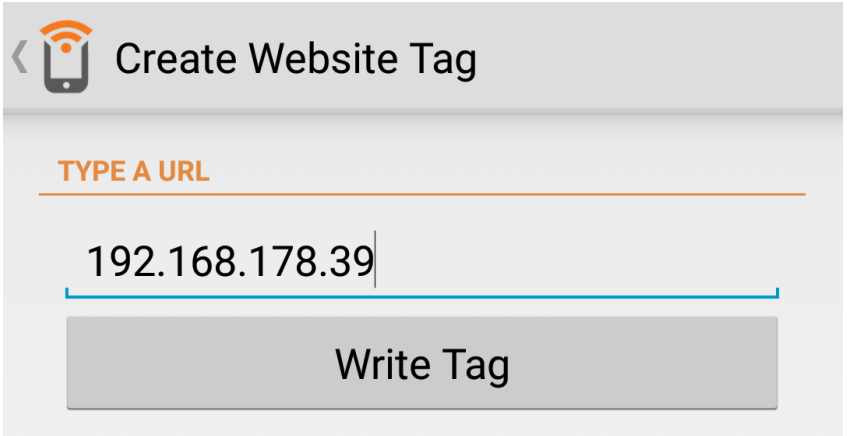

Abbildung 39 - Write Tag mit der IP- Adresse

Durch Drücken des Knopfes „Write Tag“ sowie durch die Herstellung des Kontaktes mit dem NCF-Chip wird das Schreiben des Website Tag auf den Chip ermöglicht, wie in Abb. 39. Durch den Kontakt mit einem NFC-fähigen Gerät auf dem Chip wird das Event ausgelöst und es öffnet sich in dem Standardbrowser die eingegebene Webseite auf der die Möglichkeit besteht das Modelauto zu steuern.

5. Fazit und Ausblick

Ziel der vorliegenden Arbeit ist es, den Aufbau eines ferngesteuerten Modellfahrzeugs mit Einplatinencomputern darzustellen, das mit unterschiedlichen Funknetztechnologien gesteuert werden kann.

In den Kapiteln 3 und 4 wurde daher neben den jeweiligen technischen Grundlagen die Verwendung des Raspberry Pi als Möglichkeit des Einplatinencomputers für die vorliegende Aufgabenstellung sowie die Möglichkeiten der Steuerung des Modellfahrzeugs anhand verschiedener Funknetztechnologien aufgezeigt.

Im Ergebnis zeigte sich, dass die Installation und die Konfiguration eines Raspberry Pi zeiteffizient möglich sind, insofern einem die Handhabung der Raspberry Pi Familie (bspw. in Bezug auf die Software) nicht vollkommen fremd ist. In diesem Zusammenhang ist es erforderlich über ein gewisses Basiswissen im Umgang mit dem Betriebssystem Linux zu besitzen. Allerdings besteht eine Vielzahl an Hilfestellungen im Internet, da sich die Raspberry Familie immer größerer Beliebtheit erfreut und bereits eine große Community gefunden hat.

Nichtsdestotrotz besteht eine vergleichbare Art der Ausführung des vorliegenden Projekts in dieser Konzeption noch nicht im Internet (Stand November 2017), sodass alle Module, die verbaut worden sind, in einzelnen Schritten installiert und konfiguriert werden mussten, um das vorgestellte Modellfahrzeug als gemeinsames Resultat zu erhalten.

Hierbei lag die Schwierigkeit des Projekts in der Ausgestaltung der einzelnen Details bzw. Einzelschritte. Von der Fingerfertigkeit beim Zerlegen des Autos, über das Hintergrundwissen bzgl. der Zusammensetzung elektronischer Bauteile sowie das Programmieren unterschiedlicher Module und Anwendungen in diversen Programmier- und Skriptsprachen und das Einarbeiten in bereits bestehende Frameworks, bis hin zur Verwirklichung aller verbauten Komponenten und deren Zusammenspiel.

Eine weitere Zielsetzung im Rahmen dieses Projekts wären sowohl der weitere Ausbau der Steuerung des Modellfahrzeugs über die Webseite als auch die Einbindung weiterer Features gewesen, welche aufgrund der zeitlichen Beschränkung der Bearbeitungszeit der Aufgabenstellung nicht

darstellbar waren. Als Konsequenz hieraus ist zu nennen, dass u.a. die Steuerungsmotorik bzw. -fähigkeit des Modellfahrzeugs noch weiter ausbaufähig gewesen ist, um zu gewährleisten, dass das Modellfahrzeug ad-hoc auf die geforderten Fahrtbewegungen bei der Steuerung über die Webseite reagiert.

Im Ergebnis eignet sich der Raspberry Pi Zero hervorragend aufgrund seiner vielfältigen Einsatzmöglichkeiten, insbesondere durch seine geringen Abmessungen, für die Zielsetzung des vorliegenden Projekts und stellt damit eine empfehlenswerte Plattform für die Steuerung eines Modellfahrzeugs dar. Jedoch ist auch im Rahmen der vorliegenden Arbeit aufzuzeigen, dass sich der Raspberry Pi trotz des verhältnismäßig günstigen Preises im Segment der Einplatinencomputer nicht als Media-Center aufgrund der geringen Rechenleistung und dem Single-Core Prozessor eignet. Auch die fehlende Peripherie, Adapter und zusätzlich benötigte Komponenten sind als Negativaspekt im Rahmen der Verwendung des Raspberry Pi zu nennen.

Das Thema der Einplatinencomputern in Kombination mit der Nutzung unterschiedlicher Funknetztechnologien wird auch in naher Zukunft immer mehr an Bedeutung gewinnen. Denn die heutige Gesellschaft weist immer stärkere Nutzungsmöglichkeiten von effizienten und preisgünstigen Technologien im Alltag auf, wie sich z.B. anhand des cloudbasierte Alexa-Sprachservices von Amazon zeigt.[65]

5.1 Weitere Ausbaumöglichkeiten

Weitere Ausbaustufen des Modellfahrzeugs mit Hilfe des Einplatinencomputers sind vorstellbar. Zum einen könnte eine weitere Ausbaustufe des Modellfahrzeugs weitere Komponenten und Ansichten beinhalten, wie das automatische Erkennen von Objekten und deren Interaktion. Zum anderen könnte eine weitere Funktion des Aufbaus sein, dass der Raspberry Pi sich mit einem WLAN-Hotspot oder einer internetfähigen SIM- Karte mit dem Internet verbinden könnte und damit von

[65] (Gondorf, 2016)

jedem internetfähigen Gerät aus steuerbar wäre. Des Weiteren stellt die Möglichkeit der Nutzung von Split Screens einen Ausblick auf zusätzliche Ausbaustufen dar, um das Kamerabild über eine Videobrille anzuzeigen und daraus resultierend das Fahrerlebnis weiter zu verbessern und zu intensivieren. Die nachfolgende Abbildung mit der Nummer 40 zeigt das Resultat des Modellautos.

Abbildung 40 - Letzter Stand des Modellautos

Literaturverzeichnis

Adams, R. (27. 10 2017). *Service Name and Transport Protocoll Port Number Registry*. Abgerufen am 01. 11 2017 von https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml

Bauer, C. (11. 05 2017). *NFC- Was ist das?* Abgerufen am 25. 10 2017 von http://praxistipps.chip.de/nfc-was-ist-das_12294

Baykara, S. (28. 02 2017). *Raspberry Pi Zero W*. Abgerufen am 14. 10 2017 von http://www.giga.de/hardware/raspberry-pi-zero-w/

Ben. (10 2016). *7 Wege wie du dein Raspberry Pi mit Strom versorgen kannst*. Abgerufen am 21. 10 2017 von https://maker-tutorials.com/7-wege-wie-du-dein-raspberry-pi-mit-strom-versorgen-kannst/

Burgess, P. (27. 05 2016). *Edit Partitions*. Abgerufen am 23. 10 2017 von https://learn.adafruit.com/resizing-raspberry-pi-boot-partition/edit-partitions

DATACOM. (01. 11 2014). *Nahfeldkommunikation*. Abgerufen am 14. 10 2017 von http://www.itwissen.info/Nahfeldkommunikation-near-field-communication-NFC.html

D-Edition. (17. 7 2014). Abgerufen am 14. 10 2017 von http://www.d-edition.de/blog/ratgeber/alles-was-man-ueber-rc-motoren-wissen-sollte/

Donath, A. (25. 04 2017). *Brille ermöglicht Pilotenblick in Drohnen*. Abgerufen am 13. 10 2017 von https://www.golem.de/news/dji-goggles-brille-ermoeglicht-pilotenblick-in-drohnen-1704-127468.html

DualShock 4 . (2017). Abgerufen am 18. 10 2017 von https://www.playstation.com/de-de/explore/accessories/dualshock-4-wireless-controller/

DUALSHOCK 4 Wireless-Controller. (20. 02 2013). Abgerufen am 18. 10 2017 von https://www.playstation.com/de-de/explore/accessories/dualshock-4-wireless-controller/

Ek(CF). (22. 09 2016). *Was ist Bluetooth und wie funktioniert es?* Abgerufen am 21. 10 2017 von http://www.t-online.de/digital/handy/id_49223112/was-ist-bluetooth-und-wie-funktioniert-es-.html

Kämpfer, M. (29. 09 2015). *Playstation 4 - 7 Tipps für mehr Controller-Akkulaufzeit*. Abgerufen am 18. 10 2017 von http://www.gamepro.de/artikel/playstation-4-7-tipps-fuer-mehr-controller-akkulaufzeit,3236993.html

Kirchberger, S. (07. 05 2013). *RC-Modellbau-Fun*. Abgerufen am 14. 10 2017 von http://www.rc-modellbau-fun.de/rc-modellbau-wiki-lexikon/rc-modellbau-elektromotoren/

Kofler, M. (02. 10 2016). *Raspberry Pi: Pixel-Desktio und RealVNC-Server*. Abgerufen am 23. 10 2017 von https://kofler.info/raspberry-pi-pixel-desktop-und-realvnc-server/

König, P. (26. 04 2016). *Neue Raspberry-Pi-Kamera: acht Megapixel, der Preis bleibt gleich*. Abgerufen am 18. 10 2017 von https://www.heise.de/make/meldung/Neue-Raspberry-Pi-Kamera-acht-Megapixel-der-Preis-bleibt-gleich-3186972.html

Kuther, M. (02. 09 2016). *Raspberry Pi A bis Zero, alle 9 Modelle im Überblick*. Abgerufen am 13. 10 2017 von https://www.elektronikpraxis.vogel.de/raspberry-pi-a-bis-zero-alle-9-modelle-im-ueberblick-a-548795/

Löwenstein, R. (13. 09 2013). *Das kann der PS4 Controller*. Abgerufen am 21. 10 2017 von http://www.t-online.de/spiele/id_65478468/das-kann-der-ps4-controller.html

Maier, F. (03. 03 2016). *Die Raspberry Pi-Story*. Abgerufen am 13. 10 2017 von https://www.computerwoche.de/a/die-raspberry-pi-story,3224455

Merz, A. (19. 01 2016). *Der Bastelrechner für stille. dunkle Ecken*. Abgerufen am 23. 10 2017 von https://www.golem.de/news/raspberry-pi-zero-angetestet-der-bastelrechner-fuer-stille-dunkle-ecken-1601-118584.html

Metzger, C. (03. 28 2003). *WLAN und Bluetooth: Gemeinsames und Unterschiede*. Abgerufen am 21. 10 2017 von https://www.pcwelt.de/ratgeber/WLAN-und-Bluetooth-Gemeinsames-und-Unterschiede-200723.html

MODMYPI. (10. 9 2015). Abgerufen am 14. 10 2017 von https://www.modmypi.com/blog/whats-the-difference-between-dc-servo-stepper-motors

NFCtags. (kein Datum). *SMARTRAC NFC Tags and Inlays*. Abgerufen am 14. 10 2017 von http://www.nfctags.com/nfc-tags-inlays.html

Paul. (06. 09 2015). *mircoSd-Karten für den Raspberry Pi*. Abgerufen am 23. 10 2017 von https://willy-tech.de/microsd-karten-fuer-raspberry-pi/

Radtke, T. (kein Datum). *Was Cronjobs sind und wofür sie gut sind.* Abgerufen am 03. 11 2017 von https://cronjob-tipps.de/was-cronjobs-sind-und-wofuer-sie-gut-sind/

Raspberry Pi Schrittmotor ansteuern. (09. 09 2014). Abgerufen am 14. 10 2017 von https://tutorials-raspberrypi.de/raspberry-pi-schrittmotor-steuerung-l293d-uln2003a/

Raspberry Pi: Erste Schritte bei der Konfiguration. (02. 09 2016). Abgerufen am 23. 10 2017 von https://www.elektronik-kompendium.de/sites/raspberry-pi/1906291.htm

Raspberry Pi: Fernwartung und Remote-Desktop mit VNC,RDP und SSH. (02. 09 2016). Abgerufen am 23. 10 2017 von https://www.elektronik-kompendium.de/sites/raspberry-pi/2011101.htm

Schmidt, F. (27. 05 2016). *Raspberry Pi 3 ist da: Mini-PC endlich mit WLAN-Funk.* Abgerufen am 06. 11 2017 von http://www.computerbild.de/artikel/cb-News-PC-Hardware-Raspberry-Pi-3-15161117.html

Schmidt, F. (09. 03 2017). *Raspberry Pi im Wandel: Alle Modelle, Unterschiede und Kauftipps*. Abgerufen am 13. 10 2017 von http://www.computerbild.de/artikel/cb-Tipps-PC-Hardware-Raspberry-Pi-Modelle-Vergleich-17632421.html

Wegner, G. (17. 04 2011). *Verwirrung um die Frameraten*. Abgerufen am 18. 10 2017 von https://gwegner.de/know-how/verwirrung-um-die-frameraten-24-fps-25-fps-30-fps-pal-ntsc-wann-nimmt-man-was/

whitebank. (11. 03 2017). *Raspberry Pi Remote Control Car Camera*. Abgerufen am 01. 11 2017 von https://www.hackster.io/whitebank/raspberry-pi-remote-control-car-camera-a7c7bf

whitebank. (11. 03 2017). *Raspberry Pi Remote Control Car Camera.* Abgerufen am 01. 11 2017 von https://drive.google.com/file/d/0B1UeJfx4pZ1GdEFYbl9BX1FnMDQ/view

Wiring Pi. (2017). Abgerufen am 01. 11 2017 von http://wiringpi.com/the-gpio-utility/

Wirminghaus, N. (20. 11 2013). *Die Erfolgsgeschichte des Scheckkarten-Computers Raspberry Pi*. Abgerufen am 13. 10 2017 von https://www.gruenderszene.de/allgemein/raspberry-pi-eben-upton

Wolter, J. (04. 11 2012). *JavaScript Madness: Keyboard Events*. Abgerufen am 01. 11 2017 von http://unixpapa.com/js/key.html

Ziegelwanger, W. (10. 04 2015). *Raspberry Pi - Webserver Vergleich*. Abgerufen am 01. 11 2017 von https://developer-blog.net/raspberry-pi-webserver-vergleich/

Zielosko, G. (2006). *Ansteuerung von Schrittmotoren.* Aachen: BASIC-Tiger.