HTBL Wien 10

Höhere Technische Bundeslehranstalt für Elektronik & technische Informatik



DIPLOMARBEIT

Autonom Fahrendes Auto

Ausgeführt im Schuljahr 2019/20 von: Betreuer:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Simon Liewald |  | 5AHEL |  |  | Prof. Harald Wörndl-Aichriedler |
| Milan Nikolic |  | 5AHEL |  |  |  |

Jan-Gottfried Schmölzer 5AHEL

Wien, am 17.04.2020

Abgabevermerk:

Datum: Betreuer:

Erklärung über Eigenständigkeit der Diplomarbeit

Wir versichern, dass wir die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht haben.

Wien, am 17.04.2020 Verfasser:

Simon LIEWALD

Milan NIKOLIC

Jan-Gottfried SCHMÖLZER Danksagung

An dieser Stelle möchten wir uns bei all denjenigen bedanken, die uns beim Anfertigen dieser Diplomarbeit unterstützt und motiviert haben.

Insbesondere möchten wir uns bei unseren Diplomarbeitsbetreuer Prof. Harald Wörndl-Aichriedler für die Unterstützung und Betreuung bedanken.

Außerdem danken wir allen Fachlehrern und Professoren, die uns bei der Realisierung der Diplomarbeit unterstützt und uns mit dem nötigen Wissen versorgt haben.

Zum Schluss wollen wir uns auch bei der HTBL Wien 10 bedanken, welche uns die nötigen Räumlichkeiten und Geräte zur Verfügung gestellt haben.

DIPLOMARBEIT

DOKUMENTATION

|  |  |
| --- | --- |
| Name der Verfasser/innen | Simon LIEWALD  Milan NIKOLIC  Jan-Gottfried SCHMÖLZER |
| Jahrgang Schuljahr | 5AHEL 2019/20 |
| Thema der Diplomarbeit | Autonom fahrendes Auto |
| Kooperationspartner | - |

|  |  |
| --- | --- |
| Aufgabenstellung | Aufgabe war es ein Modellauto, welches autonom fahren kann, zu bauen. Mittels  Sensoren wird die Entfernung zu  Hindernissen gemessen. Das  Modellauto lenkt wenn ein Hindernis in einer gewissen Nähe zum Modellauto steht und fahrt weiter. Mit einem Display werden die Entfernungen, welche von den Sensoren gemessen werden, ausgegeben. |

|  |  |
| --- | --- |
| Realisierung | Die Karosserie des Autos wurde zugekauft. Das Modellauto verwendet einen Mikrocontroller, welcher mittels Arduino IDE programmiert wird. Das  Modellauto lenkt mit einen Servo. Die  Prototyp-Schaltung wurde auf einem  Steckbrett aufgebaut. Für die finale Schaltung wurde eine Platine angefertigt. Die Entfernungsmessung wird mit Hilfe von Infrarotsensoren durchgeführt. |

|  |  |
| --- | --- |
| Ergebnisse | Das Modellauto fährt und kann jedem genügend hohen Hindernis ausweichen. Jedoch sind die Sensoren noch nicht richtig montiert und ein ganzer Sensor fehlt. Das Gehäuse ist komplett auf die Sensoren angepasst. Das Auto wurde so programmiert, dass es langsamer wird, wenn ein Hindernis näher kommt und bei einem gewissen Abstand erst lenkt. Um das Auto vor der |
|  | „Gefangenschaft“ in einer Ecke zu bewahren, wurde es so programmiert, dass es nach hinten fährt und gleichzeitig in eine Richtung lenkt wenn es in einer Ecke steckt. |

|  |  |
| --- | --- |
| Typische Grafik, Foto etc. (mit Erläuterung) | Hier sieht man das Auto von oben ohne Gehäuse. Vorne ist ein Schaumstoff angebracht, um das Auto bei Kollisionen zu schützen. Die Sensoren sind über dem Schaumstoff angebracht. Außerdem sieht man hier auch noch den Servo, den Motor und die H-Brücke.    Bei dieser Abbildung sieht man den Rest der Schaltung. |

|  |  |  |  |
| --- | --- | --- | --- |
| Teilnahme an Wettbewerben, Auszeichnungen | | -  - | |
|  | |  | |
| Möglichkeiten der Einsichtnahme in die Arbeit | |  | |
|  | |  | |
| Approbation  (Datum/Unterschrift) | Prüfer/in |  | Direktor/in, Abteilungsvorstand/-ständin |

DIPLOMA THESIS

DOCUMENTATION

|  |  |
| --- | --- |
| Name der Verfasser/innen | Simon LIEWALD  Milan NIKOLIC  Jan-Gottfried SCHMÖLZER |
| Jahrgang Schuljahr | 5AHEL 2019/20 |
| Thema der Diplomarbeit | autonomous driving car |
| Kooperationspartner | - |

|  |  |
| --- | --- |
| Assignment of Task | Our task was to build a self-driving car.  The distance within the car and obstructions is measured. The car steers when the distance is beneath a set value. On a display the distance to an obstacle is shown. |

|  |  |
| --- | --- |
| Realisation | The body of the car was bought. The automobile uses a microcontroller, which is programmed with Arduino IDE. It steers with a servo. The prototype circuit was built on a breadboard. For the final circuit a circuit board was made. The sensors are infrared. |

|  |  |
| --- | --- |
| Results | The car drives and can handle every obstacle high enough. However, the sensors are not placed right, and one sensor is missing. The case is fully adjusted to the sensors. The car is programmed, that it slows down when it drives near to an obstacle. It steers when the distance is a certain value. To maneuver the car out of a corner, the car drives back and steers to a certain direction. |

|  |  |
| --- | --- |
| Illustrative graph, photo etc. (incl. Explanation) | The following picture shows the top view of the car without the case. In the front is a foam, which protects the car from collisions. The Sensors are above the foam. Furthermore, you see the servo, the engine and the h-bridge    This picture shows the other part of the circuit. |

|  |  |  |  |
| --- | --- | --- | --- |
| Participation in competitions,  Awards | | -  - | |
|  | |  | |
| Accessibility of Diploma Thesis | |  | |
|  | |  | |
| Approval  (Date/Signature) | Examiner |  | Head of College / Department |

Inhalt

[1. Einleitung (Schmölzer) 10](#_Toc71586)

[2. Idee und Vorarbeit (Schmölzer) 10](#_Toc71587)

[3. Stand der Technik (Schmölzer) 11](#_Toc71588)

[4. Hardware 12](#_Toc71589)

[4.1. Distanzmessung (Schmölzer) 12](#_Toc71590)

[4.2. LC-Display (Schmölzer) 16](#_Toc71591)

[4.3. Servo (Schmölzer) 19](#_Toc71592)

[4.4. Spannungsregler (Schmölzer) 20](#_Toc71593)

[4.5. Motoransteuerung 22](#_Toc71594)

[4.5.1. Motor (Nikolic) 22](#_Toc71595)

[4.5.1.1. Allgemeines 22](#_Toc71596)

[4.5.1.2. Aufbau 22](#_Toc71597)

[4.5.1.3. Funktionsprinzip 23](#_Toc71598)

[4.5.2. Akku (Nikolic) 24](#_Toc71599)

[4.5.3. H-Brücke (Schmölzer) 25](#_Toc71600)

[4.6. Blockschaltbild (Schmölzer) 31](#_Toc71601)

[5. Software (Liewald) 32](#_Toc71602)

[5.1. Arduino 32](#_Toc71603)

[5.2. Arduino Eingang- und Ausgangpins 33](#_Toc71604)

[5.2.1. Digitale Pins 33](#_Toc71605)

[5.2.2. Analoge Pins 34](#_Toc71606)

[5.2.3. Pull-Up und Pull-Down Widerstände 34](#_Toc71607)

[5.3. Arduino DIE 35](#_Toc71608)

[5.3.1. Serial Monitor 36](#_Toc71609)

[5.3.2. Arduino Sprache 37](#_Toc71610)

[5.3.3. Datentypen 38](#_Toc71611)

[5.4. I2C Bus 38](#_Toc71612)

[5.4.1. I2C Adressierung 39](#_Toc71613)

[5.4.2. Read-/Write Bit 39](#_Toc71614)

[5.4.3. Single Master Modus 40](#_Toc71615)

[5.5. Arduino PWM 40](#_Toc71616)

[5.6. Bibliotheken 41](#_Toc71617)

[5.7. Servo Lenkung 42](#_Toc71618)

[5.8. Motor ansteuern 43](#_Toc71619)

[5.9. LC Display 46](#_Toc71620)

[5.10. Der ganze Code 48](#_Toc71621)

[6. Mechanik und Gehäuse (Nikolic) 55](#_Toc71622)

[6.1. Auswahl 55](#_Toc71623)

[6.2. Einbau des Stirnrads 57](#_Toc71624)

[6.3. Einbau der Antriebswelle 57](#_Toc71625)

[6.4. Befestigung der Motor – Anbauteile 58](#_Toc71626)

[6.5. Motor Einbau 59](#_Toc71627)

[6.6. Zusammenbau des hinteren Differenzialgetriebes 60](#_Toc71628)

[6.7. Einbau des hinteren Differenzialgetriebes 61](#_Toc71629)

[6.8. Anbringen der hinteren Querlenker 62](#_Toc71630)

[6.9. Hinterachsen 62](#_Toc71631)

[6.10. Einbau des hinteren Stoßfängers 63](#_Toc71632)

[6.11. Hinterachsen Einbau 64](#_Toc71633)

[6.12. Vorderer Querlenker 65](#_Toc71634)

[6.13. Zusammenbau des vorderen Differenzialgetriebes 65](#_Toc71635)

[6.14. Einbau des vorderen Differenzialgetriebes 65](#_Toc71636)

[6.15. Anbringen der vorderen Querlenker 66](#_Toc71637)

[6.16. Einbau des vorderen Stoßfängers 66](#_Toc71638)

[6.17. Vordere Achsenschenkel 66](#_Toc71639)

[6.18. Einbau der vorderen Achsenschenkel 66](#_Toc71640)

[6.19. Zusammenbau des Stoßdämpfers 67](#_Toc71641)

[6.20. Einbau der Stoßdämpfer 68](#_Toc71642)

[6.21. Lenkgestänge 69](#_Toc71643)

[6.22. Befestigung des Urethan–Stoßfängers 69](#_Toc71644)

[6.23. Einbau des Lenkgestänges 70](#_Toc71645)

[6.24. Zusammenbau der Lenkanlage 71](#_Toc71646)

[6.25. Einbau der Lenkanlage 72](#_Toc71647)

[6.26. Einbau der Räder 72](#_Toc71648)

[6.27. Karosserie 73](#_Toc71649)

[7. Platinenanfertigung (Nikolic) 74](#_Toc71650)

[7.1. Eagle 74](#_Toc71651)

[7.2. Platine 75](#_Toc71652)

[8. Komplettes Auto (Nikolic) 76](#_Toc71653)

[9. Ausblick auf die Zukunft 77](#_Toc71654)

[9.1. Hardware (Schmölzer) 77](#_Toc71655)

[10. Quellennachweis 77](#_Toc71656)

[11. Abbildungsverzeichnis 78](#_Toc71657)

# Einleitung (Schmölzer)

Im Rahmen der Diplomarbeit beschäftigten wir uns mit der Programmierung und Entwicklung eines autonom fahrenden Modellautos. Die Aufgabe war es, ein Auto mittels Sensoren autonom fahren zu lassen. Das gesamte Projekt wurde von uns geplant und aufgebaut. Hierbei wurde die Diplomarbeit in 3 Prozesse aufgeteilt: die

Programmierung oder die Software; der Schaltungsaufbau oder die Hardware; die Mechanik und das Gehäuse.

In der Diplomarbeit experimentierten wir mit einigen elektrischen Bauelementen und passten einige an, damit sie in das Projekt reinpassten. Dafür wurden auch die benötigten Bauteile beschafft.

Die Software wird von Simon Liewald programmiert. Das Ziel hierbei war die richtige

Lenkung und das Fahrzeug funktional fahren zu lassen. Die Mechanik und das Gehäuse übernahm Milan Nikolic. Hierbei wurde ein Modellauto ausgewählt, gekauft, zusammengebaut und angepasst. Nikolic übernahm auch die Platinen Anfertigung und einen Teil der Motoransteuerung. Beauftragter für die Hardware war JanGottfried Schmölzer. Das Aufbauen der Schaltungen und das Zusammenführen von Hardware und Software waren seine Ziele. Für das Platzmanagement im Inneren des Autos waren Nikolic und Schmölzer zuständig.

# Idee und Vorarbeit (Schmölzer)

Wir haben von den sogenannten „CrazyCars“ (durch einen Parkour fahrende Autos) gehört und uns war klar, dass wir so etwas konstruieren und realisieren wollten. Hiermit war das Diplomarbeitsthema fixiert. Um uns auf die Diplomarbeit beziehungsweise auf ein selbstfahrendes Auto vorzubereiten, haben wir uns erstmal ein ferngesteuertes Auto zugelegt, damit experimentiert und uns die Frage gestellt, wie so etwas überhaupt funktioniert. Anschließend haben wir einen sogenannten „Linienfolger“-Modus dazu entwickelt, da dies leichter zu realisieren war als ein vollkommen selbstmanövrierbares Auto. Ein Linienfolger ist ein Gefährt, welches mit Sensoren, die auf den Boden gerichtet sind, ausgestattet ist. Es folgt einer auf den

Boden aufgemalter oder ausgelegter Linie und korrigiert sich selbst, wenn es die Linie verlässt. Als wir damit fertig waren, haben wir im Rahmen des Projektunterrichts der vierten Klasse dieses Modell zu einem selbstfahrenden Modell umgebaut. Hierbei wurden Ultraschallsensoren benutzt. Als wir mit diesem Projekt fertig waren, begannen zu Ende des 4. Jahrgangs die Recherche und Besorgungen, welche wir für unser Selbstfahrendes Auto benötigen. Hiermit begann die Arbeit an der Diplomarbeit und unserem Projekt.

# Stand der Technik (Schmölzer)

Die Anfänge für ein autonom fahrendes Auto, kamen von Chrysler mit dem

Tempomat im Jahre 1958. Mit dem Abstandregeltempomat kam die nächste Stufe in

Autonomie im Fahren mit Autos ins Spiel. Diese Technik regelt nicht nur die

Längsbeschleunigung, sondern auch den Abstand zum vorderen Auto. Mit dieser Technik ist ein Auto ein teilweise autonomes Fahrzeug. Vollständig autonom bezeichnet man ein Fahrzeug, welches keinen Fahrer benötigt und beschleunigen, bremsen und lenken vollkommen von sich aus macht. Jedoch ist dies nur bei echten Fahrzeugen der Fall. Bei der Diplomarbeit sahen wir uns erst einmal ferngesteuerte Autos an. Hierbei wird das Fahrzeug mit einem Controller gesteuert. Dabei spricht man meistens auch von Funkferngesteuerten Autos (RC-Car). Wie der Name schon verrät, kommuniziert der Controller mittels Funk mit dem Auto und steuert es somit. RC-Cars werden meist in 2 Gruppen unterteilt: in langsame Spielzeugautos und in teure und schnelle Modellautos. Modellautos werden meistens in Bausätzen verkauft, wobei man oft ziemlich viele Teile einzeln kaufen muss. Autos dieser Art verwenden auch oft einen Verbrennungsmotor und können teilweise so schnell wie ein normaler PKW fahren. 1970 wurden die ersten funkferngesteuerten Autos hergestellt. Die Neuheit des Projekts ist nicht unbedingt gegeben, jedoch ist dieses Projekt besonders, da alles selbst zusammengefügt und kein vollständiger Bausatz verwendet wurde. Inspiriert wurde diese Diplomarbeit von dem „CrazyCar“

Wettbewerb der FH Joanneum. Bei diesem Wettbewerb fahren viele selbst kreierte autonom fahrende Autos durch ein Labyrinth und das Auto, welches den Parkour am schnellsten bestreitet, gewinnt das Turnier. Unsere Diplomarbeit hebt sich insofern von den meisten CrazyCars ab, dass wir als Grundgerüst ein Modellauto benutzen und der Code, die Bauteile und die Schaltung nicht vorgegeben waren, sondern selbst geschrieben, gesucht und gebaut wurden.

# Hardware

## Distanzmessung (Schmölzer)

Damit das Fahrzeug selbstständig fahren kann, war klar, dass man Sensoren benötigt, welche die Entfernung zu einer Wand oder einem Hindernis messen. Als erstes Beispiel wurde der Ultraschallsensor genauer begutachtet. Ein Ultraschallsensor misst die Entfernung, wie der Name schon besagt mittels einem

Ultraschallsignals. Vorteil eines Ultraschallsensors ist die geringe Stromaufnahme. Jedoch ist die Auflösungs- und Aktualisierungsrate gering. Die Auflösung ist die minimale Distanzänderung, welche von dem Sensor wahrgenommen werden kann.

Die Aktualisierungsrate hat die Einheit Hertz (Hz) und beschreibt die Anzahl an

Messwerten pro Sekunde. Das Modell „PING)))“ ist beispielsweise ein Ultraschallsensor der Firma Parallax.

Als optoelektronische Distanzmessung werden alle Distanzmessungen, welche mit optoelektronischen Bauteilen, also Sensoren oder Dioden funktionieren. In der Praxis finden Infrarot- und Lasersensoren ihre Anwendung.

Eine Möglichkeit die Entfernung zu messen ist die Laser-Methode, welche zum

Beispiel bei Polizeiarbeiten zum Einsatz kommt. Die Genauigkeit der LaserEntfernungsmesser liegt bei mehreren Millimetern und der Messbereich liegt bei bis zu 1011m. Ein Beispiel ist das Modell „GP2Y0A710K0F“ der Firma Sharp. Die Infrarotsensoren Distanzmessung ist weniger fehleranfällig, jedoch ist der Messbereich auch kleiner. Die meisten Infrarotsensoren reichen bis ca. 5m. Außerdem hat ein Infrarotsensor eine gute Aktualisierungsrate. Ein weiterer Nachteil wäre die relativ hohe Stromaufnahme (im Vergleich zum Ultraschallsensor). Ein

Beispiel für einen Infrarotsensor ist der GP2Y0A60SZLF. Wir haben uns nach langer Überlegung und Recherche auch für den GP2Y0A60SZLF, also einen Infrarotsensor entschieden.



Abbildung 1 GP2Y0A60SZLF

Um zu verstehen wie ein Infrarotsensor überhaupt funktioniert, wurde ein Infrarotsensor selbst aufgebaut. Hierfür wurde zuerst im Internet recherchiert. Für einen selbstgebauten Infrarotsensor benötigt man eine Infrarot-LED (IR-LED) um das Infrarot-signal zu schicken und eine Photodiode um das reflektierte Signal zu empfangen.

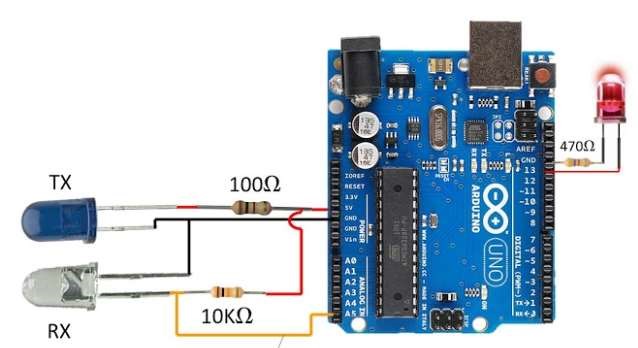


Abbildung 2 Schaltung selbstgebauter IR-Sensor

(Quelle: https://drive.google.com/drive/folders/0Bw4yk\_bZeea6cmRUY3dseUJFc2c)

Bei Abbildung 2 sieht man wie ein selbstgebauter Infrarotsensor aussieht und versorgt wird. Die rote LED rechts am Bild ist zur Überprüfung der Funktionalität. RX (RECEIVE) beschreibt hier im Bild die Photodiode und TX (TRANSMIT) beschreibt die IR-LED. Beide Dioden benötigen einen Vorwiderstand, dieser ist bei RX 10k und bei TX 100. Da eine IR-LED in alle Richtungen strahlt benötigt man etwas zur

Eingrenzung in eine Richtung, da es sonst sein könnte, dass die IR-LED direkt in die Photodiode strahlt und somit nur die Distanz zwischen den beiden Dioden gemessen wird. Hierbei kann man zum Beispiel Gebrauch von einen Schrumpfschlauch machen.

GP2Y0A60SZLF ist ein analoger Infrarot-Distanzmesser der Firma Sharp. Diesen gibt es in der 3,3V und 5V Version.

Die Distanz wird zwischen 10 und 150cm richtig gemessen. Das bedeutet, Werte unter 10cm und 150cm werden nicht richtig erfasst. Unter den Infrarotsensoren ist ein IC, welches die Größe berechtigt. Um den Infrarotsensor zu benutzen benötigt man eine extra Platine. Diese wird von der Firma Pololu hergestellt. Für diese Platine, oder Träger des Infrarotsensors gibt es ebenfalls eine 3,3V und eine 5V Version am Markt. Auf diese Platine wird der Infrarotsensor und vier Stiftleisten, welche die Pins zugänglicher machen, gelötet.

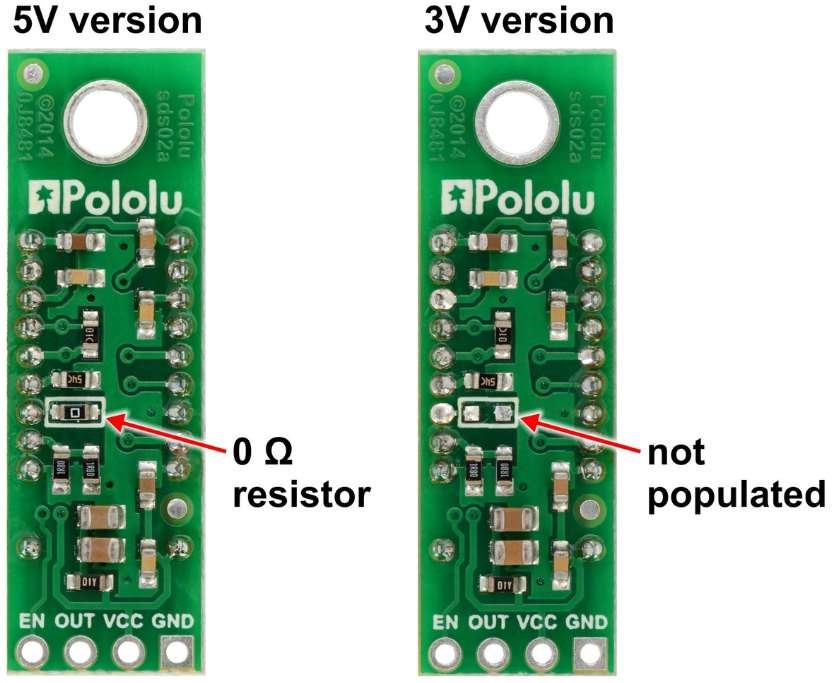


Abbildung 3 Träger des Sensors

(Quelle: https://www.pololu.com/product/2474)

Der 0 Widerstand ist da, um einen Kurzschluss darzustellen, sodass der Sensor mit 5V betrieben wird. In der 3V3 Version der Platine ist der 0 nicht vorhanden, also ist es ein Leerlauf.

Die Pins sind wie folgt angereiht:

|  |  |  |  |
| --- | --- | --- | --- |
| GND | VCC | OUT | EN |
| Ground | Versorgungsspannung  5V | Output | Enable |

Um die Richtigkeit der Distanzmessung zu überprüfen, wurde ein Sensor aufgestellt und angesteuert. Gegenüber wurde ein Holzbrett von 150cm näher geschoben und die Messung war korrekt. Dies wurde mit 4 Sensoren durchgeführt, um fehlerhafte Sensoren zu erkennen. Versorgt werden die Sensoren mit dem „ARDUINO UNO“.

Dafür wurden die 5V, GND und 4 Analoge Pins (für jeden Sensor einen) verwendet. Um die Distanz zu messen wird ein Infrarotsignal ausgesendet. Dieses Signal wird dann von einem Körper oder Hindernis reflektiert und der Infrarotsensor empfängt dieses. Der IC im Infrarotsensor unterteilt die Distanz bzw. die Outputspannung mittels Analog-Digital-wandler in Stufen um und gibt dann eine bestimmte Output Spannung am „OUT“ Pin aus.

Aus dem Datenblatt der Infrarotsensoren lässt sich der zeitliche Ablauf bildlich darstellen.

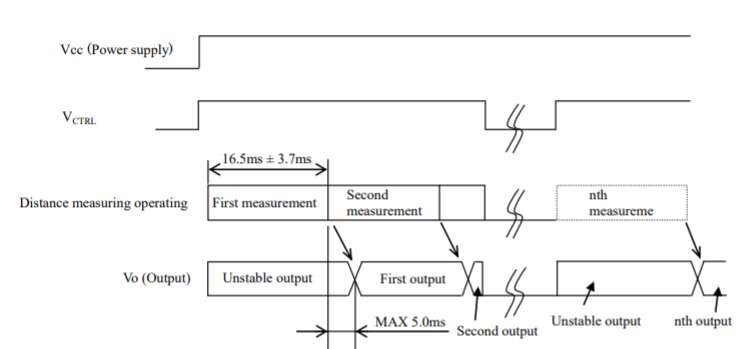


Abbildung 4 zeitliche Ablauf IR-Sensor

Die Versorgungsspannung (VCC) ist immer auf high geschalten, wenn sie eingeschalten wurde. Mit einer „high“ Kontrollspannung (VCTRL) werden die

Messungen durchgeführt. Die Kontrollspannung wird auf den Enable-Pin angelegt. Der Enable-Pin wird nicht benutzt, da die Sensoren immer messen sollen. Klar zu erkennen ist, dass der Erste Output (First output) maximal 0,5ms nachdem die erste Messung (First Measurement) abgeschlossen ist, angezeigt wird. Zu Beginn, also wenn der Sensor eingeschalten wurde, ist die Ausgangsspannung instabil.

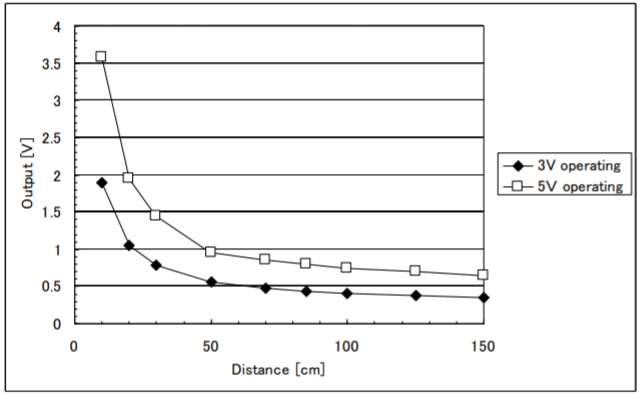


Abbildung 5 Verlauf der Distanzmessung

Bei dieser Darstellung sieht man, dass vor 10cm keine Messung möglich ist.

Dies gilt auch für Distanzen über 150cm.

## LC-Display (Schmölzer)

Ein LC-Display (auch LCD (zu Deutsch Flüssigkristallanzeige) genannt) ist ein Bildschirm welcher die Polarisationsrichtung von Licht mit Flüssigkristallen beeinflusst, wenn eine bestimmte elektrische Spannung angelegt ist.

Ein LCD ist in mehrere Segmente unterteilt. Jedes Segment ändert die Transparenz unabhängig voneinander. Die Ausrichtung der Flüssigkristalle beeinflusst die

Transparenz der Flüssigkristalle in einem Segment. Jedoch wird die Ausrichtung der

Flüssigkristalle mit der angelegten Spannung bestimmt. Durch die Ausrichtung der

LCs ändert sich auch die Durchlässigkeit von polarisiertem Licht, welches dank

Polarisationsfilter erzeugt werden kann. Polarisiertes Licht wird durch einfallendes Umgebungslicht oder durch Hintergrundbeleuchtung gefiltert. Sind die Segmente in regelmäßige Raster angeordnet, werden sie Pixel genannt. Bei einer SiebenSegment-Anzeige haben die Segmente alle eine bestimmte Form. Dies gilt auch für Anzeigen die nur ausgewählte Zeichen darstellen sollen. LCDs Anwendungen sind zum Beispiel Unterhaltungselektronik, Digitaluhren oder Messgeräte.

 Abbildung 6 Flüssigkristallbildschirm

Vorteile:

Ein klarer Vorteil bei LC-Displays ist die geringe Leistungsaufnahme. Der anzuzeigende Bildinhalt wird über den ganzen Bildschirm immer konstant beleuchtet. Die Lichtausbeute ist deshalb nur 2 bis 4 lm/W. Lichtausbeute (η) beschreibt die

Effizienz einer Lichtquelle und hat die Einheit Lumen pro Watt (lm/W). Ein weiterer Vorteil ist, dass Flüssigkristallanzeigen nur ein schwaches Magnetfeld erzeugen. Weitere Vorteile wären das geringe Gewicht, die geringe Bautiefe und ein flimmerfreies, verzerrungsfreies scharfes Bild bei guter Auflösung.

Nachteile:

Nachteile wurden mit der Entwicklung ausgebessert, somit sind der geringe Kontrast und die langen Schaltzeiten kein Problem mehr.

Funktionsweise:

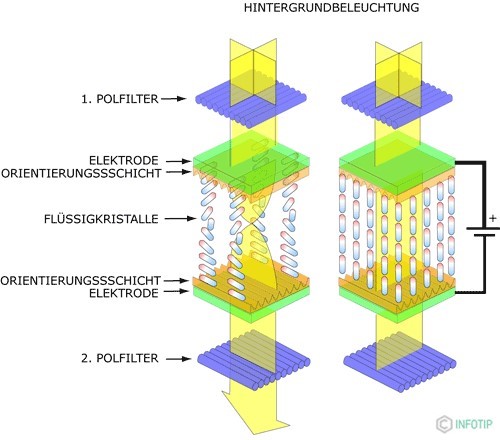


Abbildung 7 Funktionsweise LCD

(Quelle: https://www.leifiphysik.de/optik/polarisation/ausblick/funktion-von-lcd-displays)

In einer TN LCD (Twisted Nematic Liquid Crystal Display) Zelle stehen sich 2 um 90° verdrehte Polarisationsfilter gegenüber. Zwischen den Filtern befinden sich 2 durchsichtige Elektroden (Bsp.: Glasplatten mit dünner Metallschicht). Die

Flüssigkristalle befinden sich zwischen den 2 Elektroden. Die Ausrichtung der Flüssigkristalle ist von den beiden Orientierungsschichten, welche auf der Elektrode sind, abhängig. Zu beachten ist, dass die Orientierungsschichten, wie die Polfilter um

90° zu einander verdreht sind. Das heißt, die Flüssigkristalle ordnen sich auch um 90° verdreht zueinander an.

Die Hintergrundbeleuchtung, also ein unpolarisiertes Licht fällt auf den ersten Polfilter und wird linear polarisiert. Das Licht geht durch die durchsichtige Elektrode und trifft somit die Flüssigkristalle. Da die Flüssigkristalle verdreht angeordnet sind, sind sie dafür verantwortlich, dass die Polarisationsebene des durchgelaufenen Lichts um 90° gedreht wird. Danach passiert das Licht den zweiten Polfilter, da die

Polarisationsrichtung gerade auf die Durchlassrichtung des Filters liegt. Die LCDZelle ist somit lichtdurchlässig.

Wenn man an die Elektroden eine Spannung anlegt, entsteht ein elektrisches Feld zwischen den beiden Elektroden.

Das elektrische Feld bewirkt, dass sich die Flüssigkristalle nicht mehr den

Orientierungsschichten entlang, sondern mit den Feldlinien ausrichten. Wenn nun ein linear polarisiertes Licht auf die Flüssigkristalle trifft, so ändert sich die

Polarisationsebene des Lichtes nicht. Da sich die Polarisationsebene des Lichtes nicht ändert, wird dieses Licht von den zweiten Polfilter absorbiert. Das heißt, dass die LCD-Zelle bei angelegter Spannung kein Licht durchlässt. Mit der angelegten Spannung kann man den Lichtdurchgang sozusagen stufenlos mit der Größe der angelegten Spannung regeln. Es gilt: je kleiner die angelegte Spannung, desto mehr Licht wird von der LCD-Zelle durchgelassen. Mit dieser beschriebenen LCD-Zelle können auf einen LC-Display nur Grautonstufen dargestellt werden. Solche LCDs nennt man auch monochrome LCDs.

Wir verwenden ein LC-Display, welches mittels I2C mit den Arduino kommuniziert.

Das Display ist das 16x2 QAPASS. Die 16 steht für die Anzahl an möglichen

Symbolen in einer Zeile und die 2 steht für die Anzahl der verfügbaren Zeilen. Die

Ansteuerung erfolgt über den Arduino. Die Ansteuerung wird ebenfalls mittels I2C durchgeführt.

## Servo (Schmölzer)

Ein Servo ist in der Elektrotechnik ein Verbund aus Regelungs- und Antriebseinheit. Servos können eine 180° Drehung durchführen. Sie werden direkt an einen Controller angeschlossen. Jeder Servo hat zum Montieren 2 Befestigungsvorrichtungen, mit denen er ohne viel Mühe festgeschraubt werden kann. Eine Servoeinheit hat 3 Anschlüsse, den VCC Anschluss für die Versorgung, den GND Anschluss für Ground und eine Signalleitung für die Ansteuerung. Der Controller liefert dem Servo eine Versorgungsspannung und ein Signal.

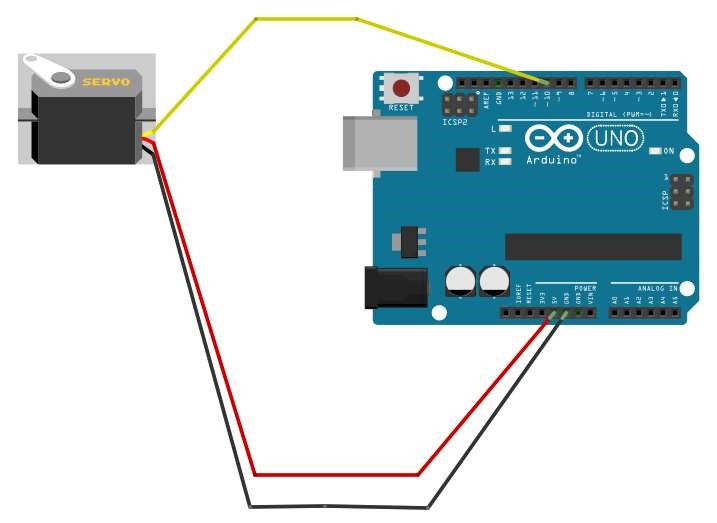


Abbildung 8 Servoansteurerung

In Abbildung 10 sieht man die Ansteuerung eines Servos mittels ARDUINO UNO.

Rot ist die Versorgung, Schwarz ist Ground und Gelb ist die Signalleitung.

Ein Servo hat mehrere Steuerkreise und ein Potentiometer. Dieses Potentiometer ist mit der Ausgangswelle verbunden und mit dem Poti kann die Steuerschaltung den aktuellen Winkel des Servos überwachen. Der Motor im Servo schaltet sich ab, wenn die Welle im richtigen Winkel steht. Wenn jedoch die Steuerschaltung feststellt, dass die Welle nicht im korrekten Winkel steht, so schaltet sich der Motor ein und dreht in die richtige Richtung, bis der Winkel wieder der richtige ist.

Die Ansteuerung eines Servos erfolgt über eine Pulsweitenmodulation (PWM). Die Breite des Pulses bestimmt auf welchen Winkel der Servoarm gestellt werden soll. Laut Datenblatt unseres Servos benutzt man ein 50Hz Signal, also ist die

Periodendauer 20ms lang. Bei einer Periodendauer von 20ms entspricht ein HIGH Pegel von 1000µs oder 1ms einer Drehung auf 0° also auf den linken Anschlag und ein HIGH Pegel von 2000µs oder 2ms einer Drehung auf 180° also auf den rechten Anschlag des Servos. Der Rest der Periodendauer ist logischerweise auf LOW Pegel.

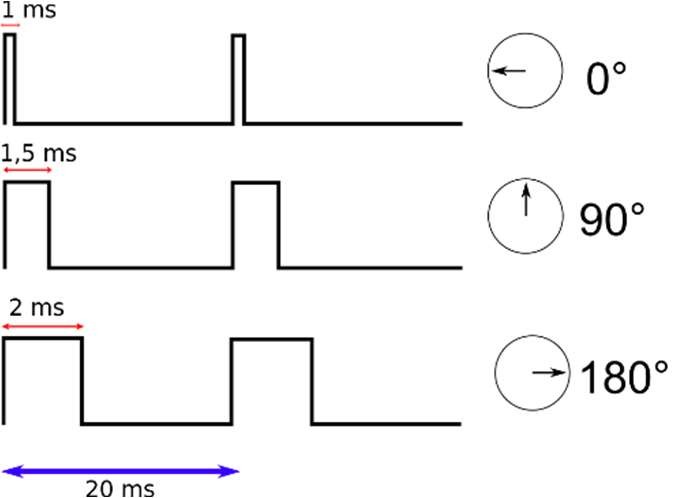


Abbildung 9 Servo PWM Ansteuern

(Quelle: Von Racso - Eigenes Werk, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=5525798)

In Abbildung 11 sieht man das 50Hz PWM Signal mit der jeweiligen Gradposition.

Ein großer Nachteil bei Modellbauservos ist, dass man nicht abfragen kann, ob der Servo überlastet ist und somit lässt sich auch nicht feststellen, ob der Servo überhaupt (noch) in der Lage ist auf eine bestimmte Gradzahl zu drehen.

Es gibt Digital- und Analogservos. Analogservos haben eine niedrigere

Winkelauflösung als Digitalservos. Außerdem wird die Servoposition bei Digitalservos schneller und präziser eingestellt. Man kann Digitalservos auch umprogrammieren, sodass man die Geschwindigkeit oder den Überlastungsschutz zu seinen Gunsten anpassen kann. Auch wenn Digitalservos technisch nur Vorteile erbringen, so sind der Bedarf an Hilfsenergie und der Preis viel höher als bei Analogservos. Wir verwenden einen Analogservo welcher mit ungefähr 7V angesteuert werden muss.

## Spannungsregler (Schmölzer)

Ein Spannungsregler dient zur Stabilisierung einer Spannung, um eine Schaltung zu versorgen. Meistens regelt der Spannungsregler eine Gleichspannung. Mit Stabilisierung ist das Ausgleichen von Schwankungen der Spannung einer Batterie oder dem Stromnetz gemeint. Spannungsregler können auch andere Aufgaben übernehmen, nämlich Strombegrenzung oder Abschaltung einer

Ausgangsspannung. Heutzutage werden integrierte Schaltungen als

Spannungsregler verwendet. Integrierte Standardspannungsregler sind Kurzschluss- und Überlastungssicher. Jedoch sind einige nicht gegen Verpolung geschützt.

Spannungsregler können auch unbeabsichtigt schwingen. Dieses Schwingen ist nicht nur gefährlich für den Spannungsregler, sondern können sie auch unerwünschte hochfrequente Schwingungen und Transienten in der Last erzeugen. Eine Transiente ist eine Schwingung, welche sich nach einer Zeit wieder einschwingt und weggeht. Neben Gleichspannung können natürlich auch Wechselspannungen geregelt werden.

Es gibt viele verschiedene Spannungsregler, welche es dem Benutzer ermöglichen die Spannung auf einen konkreten Wert zu stabilisieren. Wir verwenden den integrierten Spannungsregler LS7805

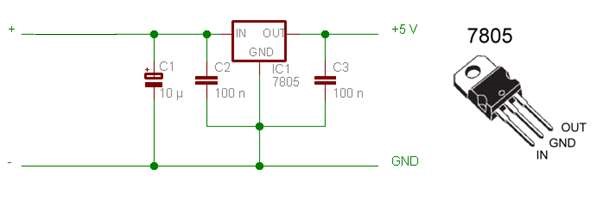


Abbildung 10 Schaltung Spannungsregler

(Quelle: https://www.mikrocontroller.net/articles/Controller\_an\_230V)

Die Schaltung, welche in Abbildung 12 zu sehen ist, regelt jede Eingangsspannung auf 5V. Außerdem werden 2 Kondensatoren mit 100nF verwendet und ein Elko mit

10µF. Ein Spannungsregler benötigt externe Kondensatoren, um unerwünschtes Schwingen und ein besseres Verhalten bei Lastsprüngen zu vermeiden. Die Kondensatoren müssen mit möglichst kurzen Leiterbahnen an der Steuerung angeschlossen werden, da lange Leiterbahnen zu einer großen Leitungsinduktivität führen und diese Leitungsinduktivität ebenfalls zu Schwingungen führen könnte. Die Kondensatoren dienen auch zur Glättung der Spannung. Der Kondensator C3 verbessert das Regelverhalten bei schnellen Lastwechseln.

Diese Schaltung kann laut Datenblatt jede Spannung von 7V bis 18V auf 5V runterregeln und der Ausgangsstrom beträgt dann 5mA. Die Toleranz bei der Ausgangsspannung ist laut Datenblatt bei 4,75V bis 5,25V. Diese Schaltung wurde auf einer Lochrasterplatine aufgebaut und regelt eine Gleichspannung von 7,2 auf

5V. Da wir den Arduino nicht mit einer Powerbank betreiben wollten und den Servo auch nicht vom Arduino betreiben wollen, benötigten wir eine Lösung zur

Stabilisierung von den 7,2V der Batterie, welche wir im Auto verbaut haben.

## Motoransteuerung

### Motor (Nikolic)

#### Allgemeines

Der Elektromotor wandelt elektrische Leistung in mechanische Leistung, da er ein sogenannter elektromechanischer Wandler ist. Genau sowas ist bei unserem Projekt notwendig da wir das Fahrzeug in Bewegung versetzten wollen, deshalb verwenden wir einen sogenannten Bürstenmotor. Der Elektromotor ist das Gegenstück zu einem Generator, der auch ähnlich aufgebaut ist. Stromdurchflossene Leiterspulen erzeugen ein Magnetfeld, durch das die gegenseitigen Anziehungs- und Abstoßungskräfte in Bewegung versetzt werden. Sie werden in vielen Geräten, Maschinen und Fahrzeugen als Antrieb verwendet.

#### Aufbau

Wie schon vorher erwähnt wurde bei diesem Projekt ein Bürstenmotor verwendet.

Dieser Motor besteht aus folgenden Bestandteilen:

* Stator

Der Stator ist der unbewegliche Teil des Elektromotors. Ein Bürstenmotor kann sowohl aus einem Elektromagneten als auch aus einem Dauermagnet bestehen. Üblicherweise befindet sich der Stator außerhalb und ist nur mit dem Gehäuse verbunden, aber es gibt auch Motoren, bei denen es anders verbaut ist. Dort liegt der unbewegliche Teil innerhalb und der Rotor rotiert sich um den Stator, wenn das der Fall ist, dann spricht man von einem Außenläufer.

* Rotor

Im Gegensatz zum Stator ist der Rotor der bewegliche Teil des Elektromotors, dieser wird auch manchmal als Anker und Läufer bezeichnet. Typischerweise besteht er aus einer Achse und aus einer Spule, durch die Strom fließt und den Rotor zu einem Elektromagneten umwandelt.

* Bürste

Bürsten kommen bei einem Bürstenmotor zum Einsatz. Mithilfe der Bürsten wird der Rotor über den Kommutator mit Strom versorgt. Diese Bürsten bestehen aus Grafit.

* Kommutator

Der Kommutator ist verantwortlich dafür, dass sich die Richtung des Storms andauernd verändert. Wenn sich die Stromrichtung ändert, dann ändert sich ebenfalls das Magnetfeld vom Elektromagneten, deswegen wird er auch Stromwender genannt. Das ist wichtig, damit der Motor nicht stehen bleiben kann und ständig in Bewegung bleibt. Vorwiegend besteht er aus einer Metallscheibe, die in zwei Teile, unterteilt ist, welche voneinander isoliert sind und diese drehen sich mit der Achse des Motors. Durch Bürsten bekommt der Kommutator seine Stromversorgung, welche jedes Mal nach einer halben Umdrehung des Motors kurz unterbrochen wird.

* Kondensator

Ein Kondensator ist ein Bauteil, welches Energie speichern kann und sie gezielt abgibt, wenn die Versorgung aus ist. Einige Elektromotoren besitzen einen

Betriebskondensator, welcher beim Starten des Motors hilft, ihn in die richtige Richtung zu drehen und somit auch gleichmäßig seine Kraft ausüben kann. Sie sind in den meisten Fällen außen am Gehäuse angebracht und meistens der Grund dafür, dass ein Elektromotor nicht anläuft.

#### Funktionsprinzip

In einem Elektromotor wird mithilfe des Magnetismus die elektrische Energie in mechanische Energie umgewandelt. Beim Magnetismus stoßen sich die gleichen Pole ab und die unterschiedlichen ziehen sich an. Nicht magnetisch geladene Sachen können durch elektrischen Strom magnetisch werden und auch die Polarität lässt sich ebenfalls beeinflussen. Ein einfacher Motor besteht aus einem beweglichen Teil, dem Rotor, welcher durch Strom magnetisch wird und einem festen Magnetischen Teil, dieser nennt sich der Stator. Wie oben schon erwähnt stoßen sich gleiche Pole ab, treffen sie nun aufeinander stoßen sie sich ab und der bewegliche Teil des Motors beginnt sich zu drehen. Damit sich der Motor nun ständig in eine Richtung drehen kann, wechselt sich die Stromrichtung automatisch nach jeder halben Umdrehung. Durch diese Methode wird sichergestellt das der Rotor nicht im Totpunkt stehen bleibt.

### Akku (Nikolic)

Für unser Projekt wurde ein NiMH-Akku mit einer Kapazität von 5000 mAh verwendet, diese spezielle Batterie, die wir benutzen, hat eine Spannung von 7.2 V, welche eine Leistung von 36 Wh aufbringt. Die Abkürzung NiMH steht für NickelMetallhydrid-Akkumulator und wurde entwickelt als umweltverträglicherer Ersatz für den Nickel-Kadmium-Akku (NiCd). Prinzipiell sind NiMH-Akkus gleich aufgebaut wie

NiCd-Akkus jedoch bestehet NiMH aus einem Metallhydrid, welches die Aufgabe hat Wasserstoff zu absorbieren und zugleich bildet es die eigentlich Elektrode. Dadurch kommt es zu einer positiven Auswirkung wobei bei gleichem Volumen der NiMHAkku eine höhere Kapazität aufweist als ein NiMH-Akku. Jedoch weisen diese Akkus auch einen Nachteil. Im Vergleich zum Vorgänger zeichnen sich NiMH durch eine deutlich höhere Selbstentladung aus und erreichen bei Raumtemperatur je nach Qualität des Akkus Werte von mehr als 50 % pro Monat. Die Entladung steigt hierbei mit der Nennkapazität. Nach sechs Monaten Lagerung beträgt die nutzbare Ladung eines solchen Akkus bei einem Selbstentladungswert von 50 % pro Monat nur noch ca. 2 %, das bedeutet sie ist im Grunde genommen leer und nach einem Jahr besitzt sie nur mehr 0.02 % der Restladung, also ist sie praktisch schon tiefentladen. Sollten NiMH-Akkus nicht gezielt allerspätestens alle sechs Monate aufgeladen werden, werden sie wegen der Tiefentladung nur eine sehr kurze Lebenserwartung aufbringen können. Sie besitzen außerdem noch eine Energiedichte pro Zelle von ungefähr 80 Wh/kg und sind ungeeignet, in Betrieb zu gehen bei Temperaturen von unterhalb 0 °C. Alleine schon in der Nähe des Gefrierpunktes zeigen sie einen eindeutigen Kapazitätsverlust und bei ungefähr -20 °C wird der Akku unbrauchbar. Empfindlich reagieren sie zudem noch auf falsche Polung, Überladung, Überhitzung und Tiefentladung bei Umpolung, welche auftreten kann, wenn die Zellen in Reihe geschalteten werden. Bei vollständigem Entladen nimmt die Lebensdauer ab wegen den beschränkten Ladezyklen von den durchschnittlichen 500 Ladezyklen. Möchte man die Solllebensdauer der Akkus verlängern dann, sollte man ein intelligentes Ladegerät benutzten welches von uns schon besorgt wurde.

### H-Brücke (Schmölzer)

Eine H-Brücke ist eine elektronische Schaltung, wo bei der Grundform fünf Zweipole, also fünf elektrische Bauteile mit jeweils 2 Anschlüssen, in der Form des Buchstabens H zusammengeschaltet sind.

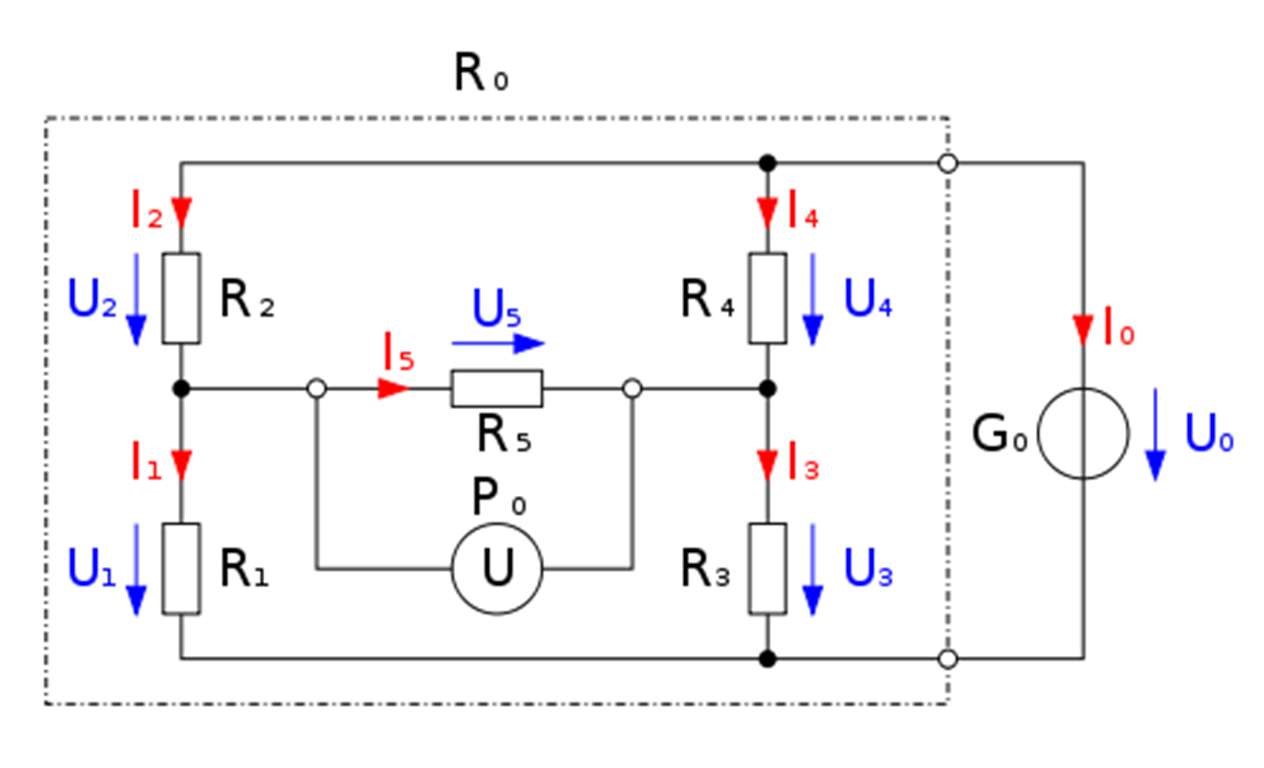


Abbildung 11 H-Brückenschaltung mit Spannungsquelle

(Quelle: https://de.wikipedia.org/wiki/Br%C3%BCckenschaltung#/media/Datei:WhBr\_Grundbild.svg)

Bei Abbildung 13 sieht man eine einfache Schaltung mit einer Spannungsquelle. In einer H-Brücke sind zwei Spannungsteiler mit einen sogenannten Brückenzweig verbunden. Mittels Einstellens der Widerstände kann man die Spannung, den Strom und sogar die Polarisation verändern. Es gibt 3 Varianten von H-Brücken:

* Viertelbrücke

Hierbei ist nur ein Widerstand variabel

* Halbbrücke

Hierbei sind 2 Widerstände variabel

* Vollbrücke

Hierbei sind 4 Widerstände variabel

Das Messobjekt, oder die Last wird auf einer Seite ein Teil des Spannungsteilers und noch mindestens ein Teil wird variabel gemacht. Somit kann man einen Nullabgleich des Stromes im Brückenzweig durchführen. Mit einen Nullabgleich kann man die Messung sehr präzise durchführen, da ein Nulldurchgang besser bestimmt werden kann als ein Maximum oder ein Minimum. Der Abgleich findet so lange statt, bis das Messinstrument im Brückenzweig die Nullanzeige genau genug anzeigt.

Bei der Messung von rein ohmschen Widerständen kann man eine H-Brücke mit Gleichspannung betreiben. Wenn man jedoch Impedanzen, also Spulen oder Kondensatoren messen will, so ist ein Wechselspannung Betrieb notwendig. H-Brücken kann man aber auch als Ansteuerungen für Gleichstrommotoren. Hierbei verwendet man Vierquadrantensteller. So eine Schaltung besteht aus einer H-Brücke mit 4 Halbleiterschaltern. Mittels dieser Schaltung kann man eine Gleichspannung in eine Wechselspannung mit variabler Frequenz und variabler Pulsbreite umwandeln.

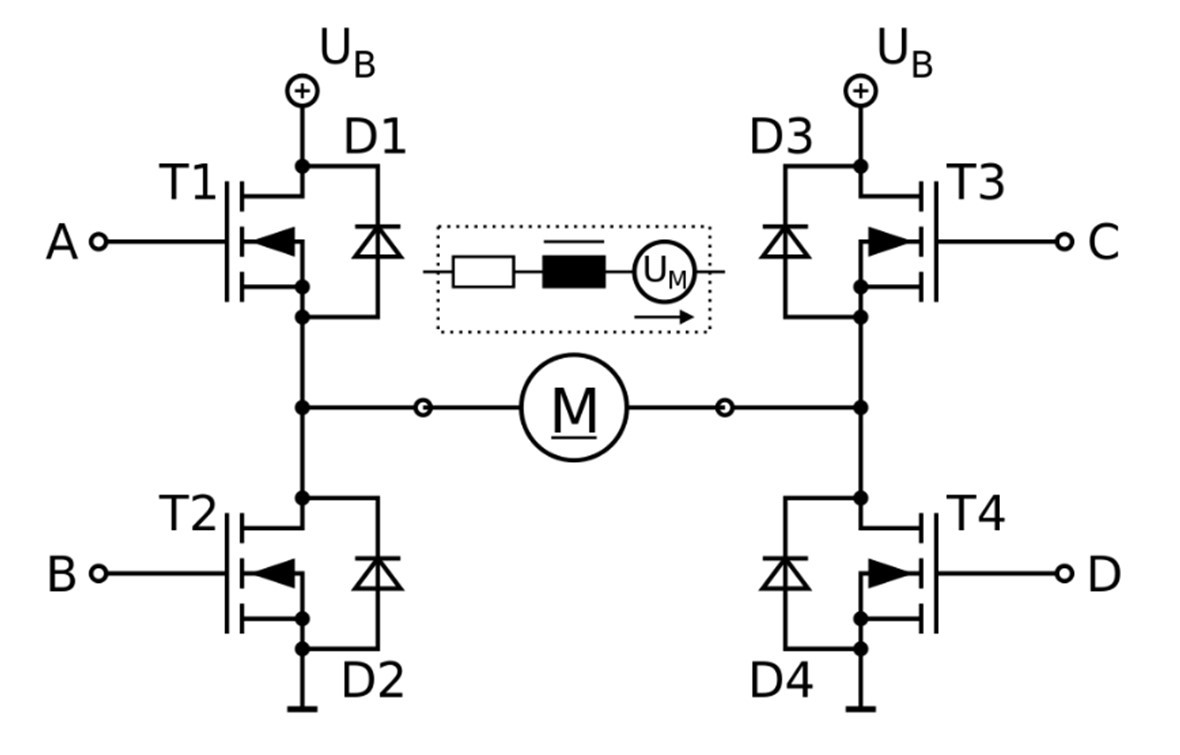


Abbildung 12 H-Brücke für DC-Motor

(Quelle: https://de.wikipedia.org/wiki/Vierquadrantensteller#/media/Datei:Vierquadrantensteller.svg)

Bei Abbildung 14 sieht man ein Beispiel für einen Vierquadrantensteller für einen

Gleichspannungsmotor. Die Aufgabe einer H-Brücke bei der Ansteuerung eines DCMotors ist die Beschleunigung und Bremsung in beide Drehrichtungen des Motors. Der Aufbau eines Vierquadrantenstellers besteht aus, zweimal zwei in Serie geschaltenen Transistoren mit jeweils einer Freilaufdiode in Sperrpolung. Eine Freilaufdiode wird auch Schutzdiode genannt und schützt vor Überspannung beim Abschalten einer Gleichspannungslast, also einen DC-Motor zum Beispiel.

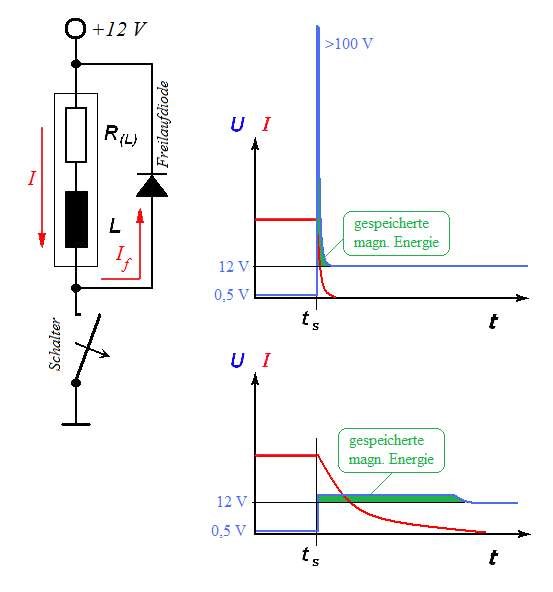


Abbildung 13 Strom und Spannung mit/ohne Schutzdiode

(Quelle: https://de.wikipedia.org/wiki/Schutzdiode#/media/Datei:Catchdiode.png)

Bei Abbildung 15 sieht man den Strom- und Spannungsverlauf an einer induktiven

Last. RL und L sind die induktive Last. Parallel dazu wird die Schutzdiode geschalten. Der Strom und die Spannung werden am Schalter gemessen. Der obere Verlauf beschreibt den Verlauf ohne Schutzdiode. Beim Abschalten sorgt die Spule für eine Selbstinduktion. Somit fließt der Strom in die ursprüngliche Richtung und es entsteht eine Spannungsspitze, die sich zur Betriebsspannung dazu addiert. Die Leitungen können dadurch geschädigt werden.

Der untere Verlauf beschreibt das Verhalten mit Schutzdiode. Mit einer Schutzdiode parallel zur Last begrenzt man die Spannungsspitze auf die Durchlassspannung der Diode (0,7V bei Silizium). Somit ist die Leitung geschützt.

Im Brückenzweig der H-Brücke liegt man den Gleichstrommotor (M oder das Ersatzschaltbild besteht aus einer Induktivität und einer Motorwicklung und der Spannungsquelle Um), welchen man ansteuern möchte.

Bei einen Vierquadrantensetzer gibt es vier Betriebsmodi:

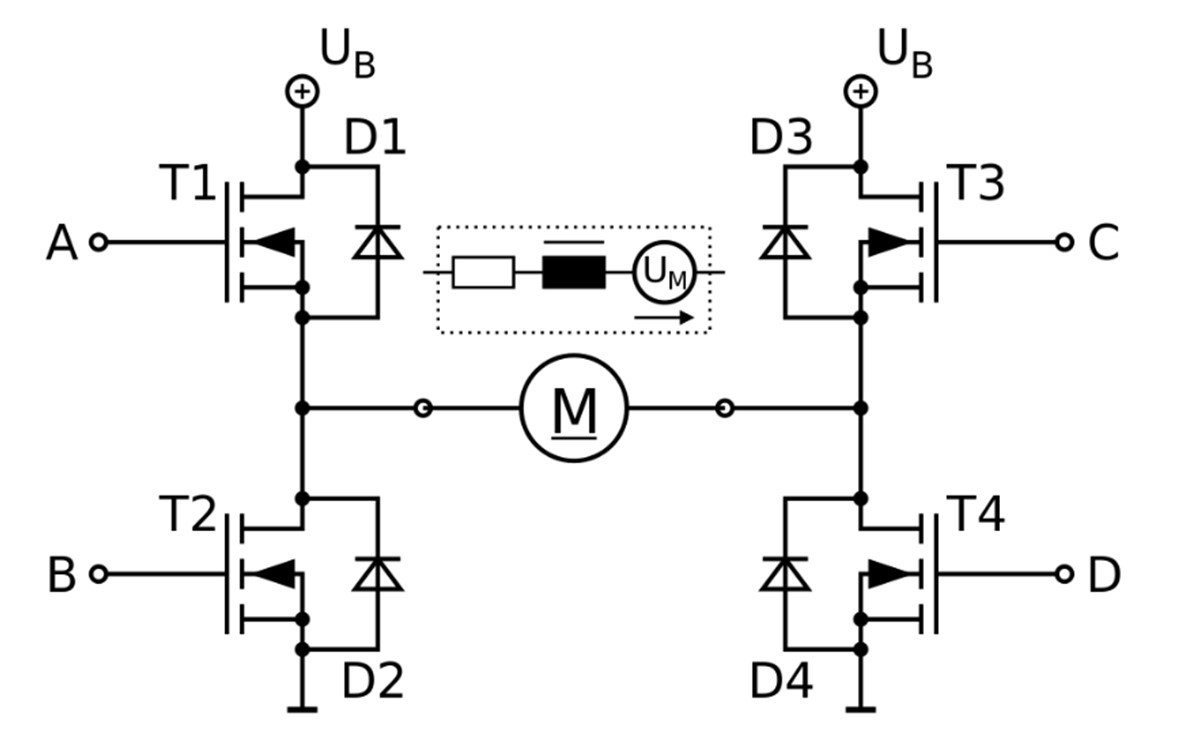


Abbildung 14 Vierquadranten Betriebsmodi Erklärung

* Tiefsetzsteller

Dieser Betrieb ist zum Antreiben da. Das heißt, der Motor nimmt Leistung auf. T4 wird durchgeschalten. Auf T1 wird ein PWM-Signal gelegt. Wenn T1 nun leitet, so liegt eine positive Spannung am Gleichspannungsmotor. Die

Induktivität des Motors magnetisiert sich und ein positiver Strom fließt. Der Motor erzeugt dadurch ein beschleunigendes Drehmoment. Schaltet T1 nun wieder ab, so wird Spannung durch die Motorwicklung induziert und der Strom fließt über D2 weiter. Die Magnetisierung des Motors nimmt wieder leicht ab. Die Beschleunigung wird stärker, wenn mehr Strom fließt. Mehr Strom fließt, wenn die „Leitphase“ länger im Verhältnis zur Sperrphase dauert. Wenn man T3 durchschaltet und das PWM-Signal auf T2 legt, so erreicht man eine entgegengesetzte Polarität.

* Hochsetzsteller

Der Hochsetzsteller-Betrieb dient zum Bremsen. Das heißt, der Motor gibt Leistung ab. T4 wird durchgeschalten. Das PWM-Signal wird hierbei an T2 gelegt. Wenn T2 leitet, so magnetisiert sich die wieder die Induktivität und es fließt ein negativer Strom. Der Strom hat zu UM eine entgegengesetzte Polung und der Motor gibt Leistung ab. Die abgegebene Leistung war davor im

Magnetfeld gespeichert. Wenn T2 sperrt, so induziert die Motorwicklung Spannung und der Strom fließt über D1 ab. Somit nimmt die Magnetisierung leicht ab. Die Energie aus dem Magnetfeld wird als Versorgungsspannung abgegeben. Mechanische Leistung wird vom Motor in elektrische Leistung umgewandelt. Der Motor bremst. Wenn man T3 durchschaltet und das PWMSignal auf T1 legt, so erreicht man eine entgegengesetzte Polarität.

* Notbremse

Bei der Notbremse leiten T2 und T4, dadurch wird der Motor kurzgeschlossen. Der vom Motor erzeugte Strom wird, nur durch die ohmschen Verluste begrenzt und die Leistung wird in Wärme umgewandelt. Hierbei muss man beachten, dass alle Bauteile die Werte aushalten und nicht kaputt gehen.

* Leerlauf

Beim Leerlauf-Betrieb leitet maximal nur ein Transistor. Nach dem Abbau des restlichen Magnetfeldes, fließt kein Strom mehr durch den Motor. Das bedeutet, dass weder beschleunigt noch gebremst wird.

Die verwendete H-Brücke ist „IBT-4“. Hierbei wird sie jedoch als Motortreiber bezeichnet.

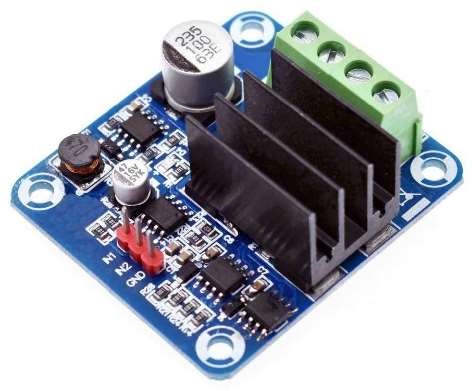


Abbildung 15 IBT-4 Motor Treiber

(Quelle:

https://www.aliexpress.com/item/32790123587.html?af=240682&utm\_campaign=240682&aff\_platform=aaf&utm\_mediu m=cpa&dp=201ca392e7ae5d4912448f999085a52f&cv=47843&mall\_affr=pr3&sk=VnYZvQVf&aff\_trace\_key=805dddee0965 4b2ca3e229d7e141bccf-1586792503787-05695-

VnYZvQVf&terminal\_id=f76789f2bd48422c9ae415878b845177&tmLog=new\_Detail&utm\_source=admitad&utm\_content=4

7843&aff\_request\_id=805dddee09654b2ca3e229d7e141bccf-1586792503787-05695-VnYZvQVf)

Die Pinbelegung dieser H-Brücke ist wie folgt:

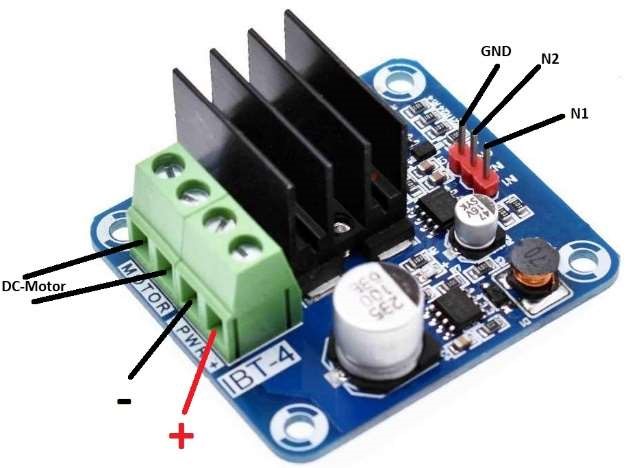


Abbildung 16 Pin-Belegung IBT-4

Bei „+“ und „-“ wird die Versorgungsspannung, also die Batterie angelegt. Bei den 2

Pins mit „DC-Motor“ wird der Motor angeschlossen. „GND“ wird mit dem GND Pin am

Arduino-Board verbunden. N1 und N2 sind für die Ansteuerung der H-Brücke mittels ARDUINO UNO nötig. Diese zwei Pins regeln die verschiedenen Modi der H-Brücke, mit einem PWM-Signal und werden im Programmcode definiert. Der Motor wird somit angesteuert und dreht sich in eine bestimmte Richtung (entweder vorwärts:

Tiefsetzsteller-modus; oder rückwärts: Hochsetzsteller-modus).

Diese H-Brücke kann im Bereich von 5-12V verwendet werden, weshalb sie für das

Auto, welches mit 7,2V betrieben wird, ausgewählt worden ist. Die „absolute Limitspannung“ der IBT-4 liegt jedoch bei 15V. Außerdem haltet die IBT-4 H-Brücke einen Strom von bis zu 50A aus. Ein weiterer Pluspunkt bei diesen Motortreiber ist die, im Verhältnis zu anderer, kleiner Dimensionierung. Er ist 48mm lang, 43mm breit und 23mm hoch. Diese Dimensionierung kommt dem Auto und dem Platzmanagments sehr zu gute.

## Blockschaltbild (Schmölzer)

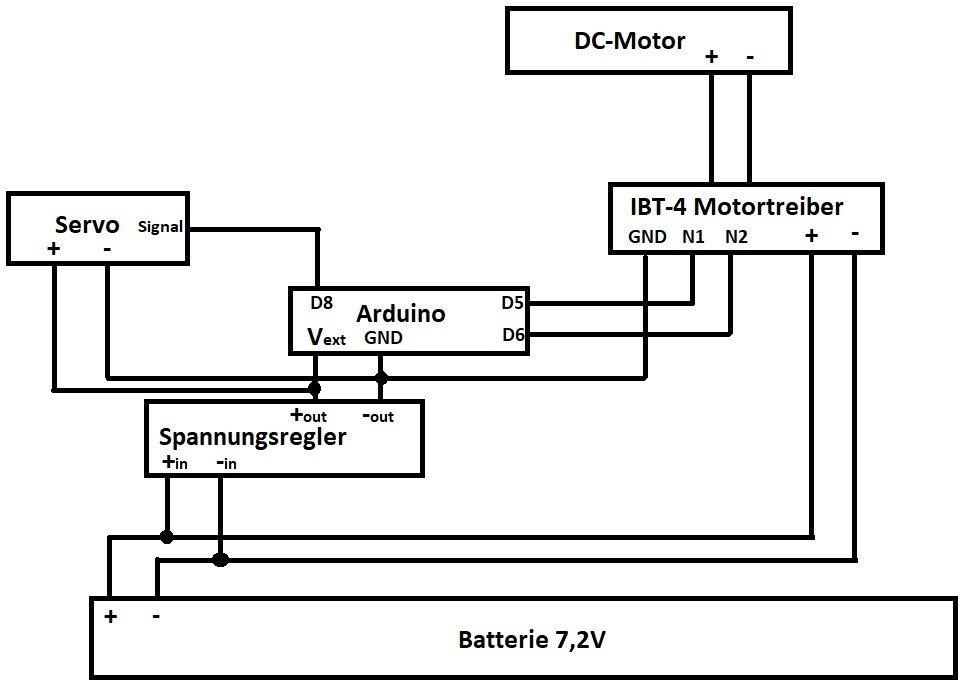


Abbildung 17 Blockschaltbild

Bei Abbildung 19 sieht man das Blockschaltbild zur Ansteuerung unseres Motors. Die

Batterie versorgt die H-Brücke mit den vollen 7,2V und gleichzeitig regelt der

Spannungsregler die 7,2V auf 5V runter, sodass man den ARDUINO UNO und den Servo extern versorgen kann. Der Arduino steuert das PWM, welches den Modus beim Motortreiber regelt. Am Ende steuert der IBT-4 Motortreiber den DC-Motor.

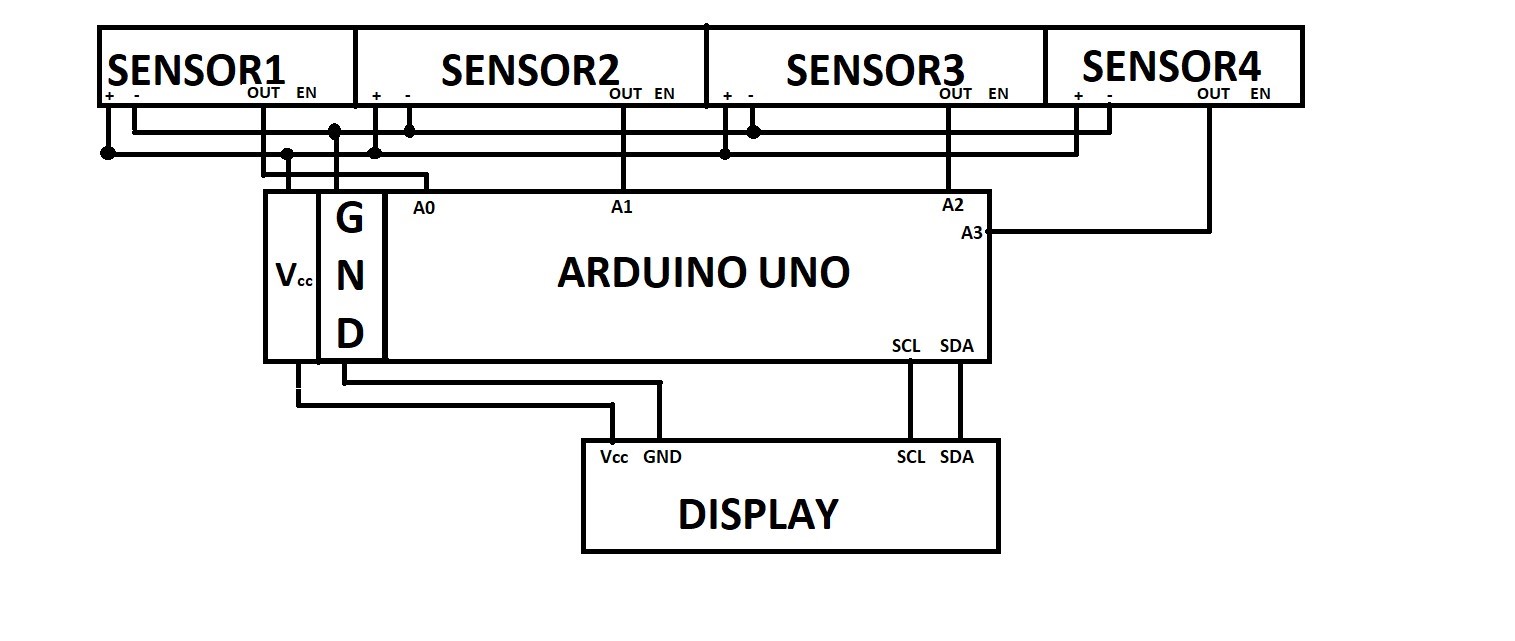


Abbildung 18 Blockschaltbild 2

Bei Abbildung 21 sieht man ein Blockschaltbild, welches die Sensoren und das

Display zeigen. Die Sensoren und das Display sind mit den GND und Vcc, also 5V

Pins vom Arduino UNO verbunden. Das Display kommuniziert mittels I2C mit den Arduino UNO, deswegen sind sie auf SCL und SDA verbunden. Die OUTPUT Pins der Sensoren sind auf Analogpins vom Arduino gesetzt. Der OUTPUT-Wert, der Sensoren, werden zu einer Distanz umgerechnet, welche dann am Display angezeigt werden. Wie das geschieht wird im Kapitel „Software“ genauer beschrieben.

# Software (Liewald)

## Arduino

Arduino ist eine Open-Source-Plattform für die Prototypen von Elektronik, die aus einem Mikrocontroller einer Programmiersprache und einer IDE besteht. Arduino ist ein Tool zum Erstellen interaktiver Anwendungen, welche diese Aufgabe für Anfänger vereinfacht, aber dennoch flexibel genug, damit Experten komplexe Projekte entwickeln können. Da Arduino open source ist bedeutet, dass es Arduino-Klone gibt, die man für Projekte auswählen können. Fortgeschrittenes Arduino Benutzer entwickeln Schilde für die verfügbaren Arduinos sowie völlig neue zu Arduino kompatible Boards, die auf verschiedene Aufgaben spezialisiert sind. Die absichtliche Einfachheit der Anwendung hat einen Zugang zum System für Leute ermöglicht, die noch nie daran gedacht hätten, einen Mikrocontroller zu verwenden oder zu programmieren.

Das Arduino UNO-Board basiert auf einem 8-Bit-Atmel-AVR-Mikrocontroller. Je nach Board finden man verschiedene Chips, darunter ATmega8, ATmega168, ATmega328, ATmega1280 und ATmega2560. Das Arduino-Board legt den größten Teil der Eingang und Ausgangspins des Mikrocontrollers frei, damit sie als Ein- und Ausgänge für andere Schaltungen verwendet werden können, die um den Arduino herum aufgebaut sind.

Wenn man sich http://arduino.cc/en/Main/Hardware ansieht, finden man eine große Auswahl an Arduino Bretter. Jedes von ihnen wurde für einen anderen Zweck entworfen und hat daher eine andere Anzahl der an Bord befindlichen Stifte und Elektronik.

Das Arduino-Projekt bietet auch Open-Source-Hardware und für diese stehen die Schaltpläne aller offiziellen Boards der öffentlich als EAGLE-Dateien (EAGLE ist eine PCB-Layout-, Autorouter- und CAM-Software von Cadsoft) zu Verfügung. Auf diese Weise, wenn keines der Board-Designs der Anforderungen zu entsprechen scheint kann man ein eigenes Arduino-kompatibles Board herstellen.

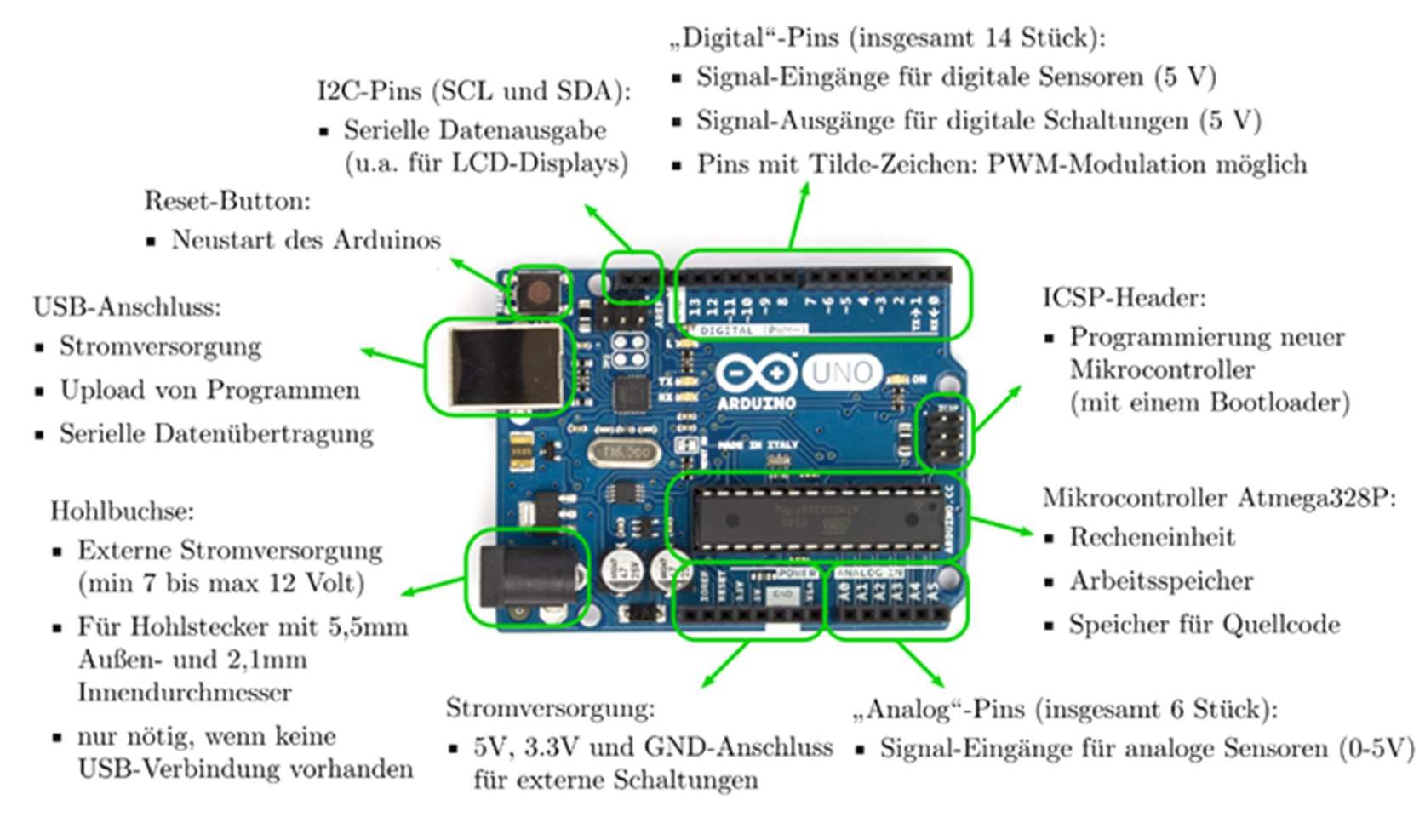


Abbildung 19 Arduino UNO

(Quelle: https://www.grund-wissen.de/elektronik/\_images/arduino-uno-mit-beschreibung.png)

## Arduino Eingang- und Ausgangpins

Der Arduino Uno verfügt über 14 digitale Eingangs- / Ausgangspins und 6 analoge

Eingangspins. Diese Pins entsprechen den Eingangs- / Ausgangspins des ATmegaMikrocontrollers. Die Architektur des Arduino-Boards legt diese Pins frei, sodass sie problemlos an externe Schaltkreise angeschlossen werden können. Arduino-Pins können in zwei Modi eingerichtet werden: Eingang und Ausgang.

### Digitale Pins

Arduino verfügt über 14 digitale Pins, die von 0 bis 13 nummeriert sind. Die digitalen Pins können als INPUT oder OUTPUT konfiguriert werden mit der Funktion pinMode(pin, INPUT);. In beiden Modi können die digitalen Pins nur digital senden oder empfangen Signale die aus zwei verschiedenen Zuständen bestehen: EIN (HOCH oder 5 V) und AUS (NIEDRIG oder 0 V). Als OUTPUT eingestellte Pins können bei Bedarf externe Geräte oder Schaltkreise mit bis zu 40mA Strom versorgen. Pins welche als INPUT gesetzt sind, sind bereit Spannung von den an sie angeschlossenen Geräten zu lesen. Stellt man diese Pins auf INPUT, damit man den Status eines Schalters, elektrische Impulse und andere digitale Signale lesen können. Sechs dieser Pins können auch als Pulsweitenmodulationspins (PWM) verwendet werden.

### Analoge Pins

Die Micro Controller welche im Arduino-Uno verwendet werden haben 6 analog zu digital Konverter (ADC) eingebaut. Die Funktion eines analog zu digital Konverters besteht darin, eine analoge Eingangsspannung umzuwandeln in einer digital zahl proportional zu einer Bezugsgröße (Referenzspannung) welche zumeist 5V beträgt. Der ATmega-Konverter auf dem Arduino-Board hat eine Auflösung von 10 Bit, was bedeutet, dass er einen integer bis zu 1023 ausgeben kann. Die Pins können mit dem Namen A0-A5 sowohl auf Eingang als auch auf Ausgang geschalten werden

### Pull-Up und Pull-Down Widerstände

Wenn man einen digitalen Wert von einem digitalen-Pin auslesen will wird eigentlich immer die Spannung gemessen, welche an diesem anliegt. Der Arduino unterscheidet hierbei zwischen 0 und 1. Eine 0 wird erkannt, wenn die anliegende Spannung 0V beträgt. Eine 1 wird erkannt, wenn die anliegende Spannung knapp der Versorgungspannung entspricht also ungefähr 5V. Um möglichst genau eine 0 und 1 erkennen zu können sollte die Spannung am Pin möglichst genau 0V oder 5V betragen. Wenn die Pins jedoch auf Eingang eingestellt sind, befinden sie sich in einem hochohmigen Zustand, dass bedeutet es braucht sehr wenig Strom, um den Pin von einem Zustand in den anderen zu bewegen. Dies kann verwendet werden für Anwendungen, welche dieses Phänomen ausnutzen, wie kapazitive

Berührungssensoren, aber für den allgemeinen Gebrauch bedeutet dies, dass der Zustand eines getrennten Eingangspins schwingt und zufällig zwischen 0 und 1 wechselt aufgrund von elektrischem Rauschen.

Wenn kein Eingang vorhanden ist, muss um dies zu vermeidenden der Eingang auf einen bekannten Zustand gebracht werden. Dies ist die Rolle der eingebauten Pull-Up Widerstände.

Wenn der Schalter mit dem Pull-Up Widerstands geöffnet ist, so ist die Spannung, welche am Eingang des Pins anliegt, 5V. Wenn der Schalter geschlossen wird, so kann der gesamte Strom über den Widerstand auf GND fließen, wodurch die Ausgangspannung auf 0 fällt da die Spannung nun gegen GND geht. Dadurch wird der Pin am Eingang immer in einen Zustand nahe der 0V oder 5V gebracht. Der Pulldown Widerstand funktioniert auf dieselbe Art, hierbei werden die 5V geschaltet sodass der Eingang auf GND gezogen werden muss. Schließt man nun den Taster, welcher zwischen 5V und GND ist so liegen dadurch 5V am Eingang des Pins an. Im ATmega-Chip sind 20K-Pull-up-Widerstände verbaut. Den Widerstand und damit der Modus indem der Pin arbeiten soll kann über die Software eingestellt werden.

Der Befehl: pinMode(HIGH); setzt den Pin in den Pull-up Modus, um den Pin im Pulldown Modus zu betreiben muss man lediglich das HIGH zu LOW ändern (pinMode(LOW);).

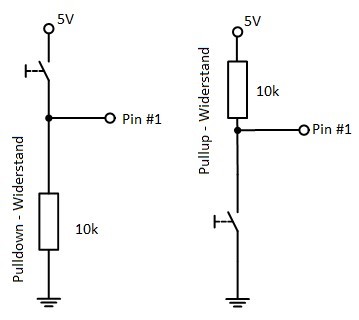


Abbildung 20 Pull-up und Pull-Down

(Quelle: https://www.mikrocontroller.net/attachment/364648/PullWiderstaende.png)

## Arduino DIE

Die Arduino IDE ist eine plattformübergreifende Anwendung und ist in Java geschrieben. Die IDE basiert auf Processing und wurde so entwickelt, dass man den mit der Programmiersprache C++ den Arduino Programmieren kann. Arduino IDE arbeitet mit dem AVR GCC-Compiler der den source welchen man in der IDE schreibt zu einem MaschinenCode umwandelt und ihm damit leserlich für Geräte wie den Arduino macht.

Die Hauptaufgabe dieser Software ist es das Programmieren eines Arduino so weit zu vereinfach, dass man zum Schluss mit einem Klick das Programm Kompilieren lassen und auf den Arduino hochladen kann.

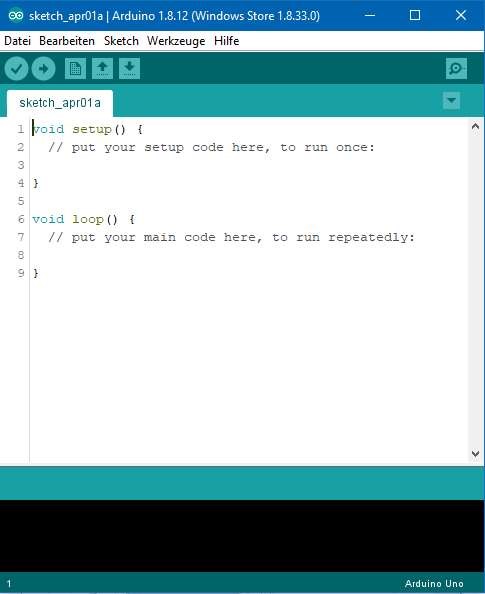


Abbildung 21 Arduino IDE

### Serial Monitor

Der serielle Monitor welcher in der Arduino IDE mit dabei ist, ist ein Tool zum Anzeigen der seriell gesendeten Daten vom Arduino zum Computer. Um diesen zu verwenden muss zuerst die Baudrate eingestellt werden, mit dem Befehl Serial.begin(9600); wird die Baudrate auf 9600 eingestellt. Das heißt es werden damit 9600 Bits pro Sekunde übertragen. Der serielle Monitor ist wichtig beim Programmiren und Debuggen eines neuen Programmes da der Micro-Prozessor zum Beispiel mit dem Befehl

Serial.println(Variable); den Wert der von Variable auf der seriellen Schnittstelle aus.

Dadurch kann man einen Blick in den Zustand eines Programmes zu einem bestimmten Zeitpunkt werfen.

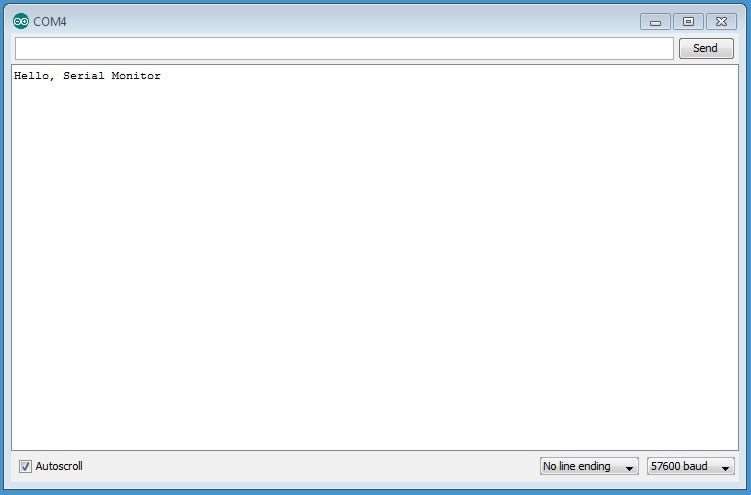


Abbildung 22 Serial Monitor

(Quelle: https://www.eltima.com/images/upload/products/spm/articles/arduino/ArduinoSerailMonitor01.jpg)

### Arduino Sprache

Die Arduino-Sprache ist C++ Wiring von Arduino ist eine Bibliothek von Kurzbefehlen, um Abläufe zu vereinfachen. Daher benutzt man die Bibliothek auch immer sobald man etwas in einem Sketch (Programm Datei) schreibt. Das ermöglicht Programme zum Laufen zu bringen mit nur zwei Funktionen: setup() und loop(). Es müssen bei jedem Arduino Programm beide dieser Funktionen mit inbegriffen sein selbst wenn man sie nicht benutz.

* setup() Funktion

Die setup() Funktion wird nur einmal beim starten des Programmes ausgeführt, dies geschieht sobald das Programm hochgeladen ist auf dem Arduino oder wenn er eingeschaltet wird. Die Funktion wird benutzt um Variable zu initialisieren wie pinMode().

* loop() Funktion

Die loop() Funktion läuft kontinuierlich in einer unendlichen Schlaufe so lange der Arduino eingeschaltet ist. In den meisten Programmen steckt der Großteil in dieser Funktion. Dadurch kann das Programm auf Ereignisse reagieren, Kontrollieren und Änderungen vornehmen. Die loop() Funktion wird erst nach der setup() Funktion ausgeführt.

### Datentypen

Um eine Variable zu benutzen muss man diese zuerst deklarieren und den Datentyp spezifizieren. Die Datentyp Spezifikation gibt an wie groß die Variable sein kann und damit auch wie viel Speicher sie braucht, um gespeichert werden zu können. Die Datentypen, die verwendet werden können, sind die gleichen welche auch in C++ verwendet werden können.

Für das Self driving Car sind daher folgende Datentypen relevant:

* byte kann einen Zahlenwert von 0 bis 255 haben also 1Byte groß
* int ist ein Integer kann einen Zahlenwert von -32,768 bis 32,767 haben also 2Byte groß
* float kann einen Zahlenwert von -3.4028235E+38 bis 3.4028235E+38 haben also 4Byte gorß

## I2C Bus

I2C ist ein serielles Kommunikation Protokoll bei welchem die Daten Bit für Bit über eine einzige Leitung übertragen werden (die SDA Leitung). Wie SPI ist I2C synchron, so dass die Ausgabe von Bits zwischen dem Master und dem Slave geteiltes Taktsignal mit der Abtastung von Bits synchronisiert wird. Das Taktsignal wird immer vom Master gesteuert.

Mit I2C werden Daten in Nachrichten übertragen. Nachrichten werden in Frames aufgeteilt. Jede Nachricht hat einen Adressframe, der die Binäradresse des Slaves enthält, und einen oder mehrere Datenframes, welche die übertragenen Daten enthalten. Die Nachricht enthält auch Start- und Stoppbedingungen, Lese- /

Schreibbits und ACK / NACK-Bits zwischen jedem Datenrahmen

* Startbedingung: Die SDA-Leitung wechselt von einem HIGH auf LOW, bevor die SCL-Leitung von HOCH auf LOW umschaltet.
* Stoppbedingung: Die SDA-Leitung wechselt von einem LOW auf einen HIGH, nachdem die SCL-Leitung von LOW auf HIGH umgeschaltet hat.
* Adressframe: Eine für jeden Slave eindeutige 7- oder 10-Bit-Sequenz, die den Slave identifiziert, wenn der Master mit ihm sprechen möchte.
* Lese- / Schreib-Bit: Ein einzelnes Bit, das angibt, ob der Master Daten an den Slave sendet (LOW) oder Daten von diesem anfordert (HIGH).
* ACK / Not-ACK(NACK)-Bit: Auf jeden Frame in einer Nachricht folgt ein Bestätigungs- / Nichtbestätigungsbit. Wenn ein Adressframe oder Datenframe erfolgreich empfangen wurde, wird vom empfangenden Gerät ein ACK-Bit an den Absender zurückgegeben.

### I2C Adressierung

I2C verfügt nicht über Slave-Auswahlleitungen wie SPI. Daher muss der Slave auf andere Weise darüber informiert werden, dass Daten an ihn gesendet werden, und nicht zu einem anderem Slave. Dies geschieht durch Adressierung. Der Adressframe ist immer der erste Frame nach dem Startbit in einer neuen Nachricht. Der Master sendet die Adresse des Slaves, mit dem er kommunizieren möchte, an jeden angeschlossenem Slave. Jeder Slave vergleicht dann die vom Master gesendete Adresse mit seiner eigenen Adresse. Wenn die Adresse übereinstimmt, sendet der Slave ein ACK-Bit an den Master zurück. Wenn die Adresse nicht übereinstimmt, tut der Slave nichts und die SDA-Leitung bleibt auf HIGH.

### Read-/Write Bit

Der Adressframe hat am Ende ein einzelnes Bit worüber der Slave darüber informiert, ob der Master Daten darauf schreiben oder Daten von ihm empfangen möchte. Wenn der Master Daten an den Slave senden möchte, ist das Lese- / Schreib-Bit LOW. Wenn der Master Daten vom Slave anfordert, ist das Bit ein HIGH.

### Single Master Modus

Für das Self driving Car wird der Single Master Modus verwendet. Dies ist die einfachste I2C-Buskonfiguration. Ein einzelner Master im Bus ist für die gesamte Kommunikation im Bus verantwortlich. Das Master-Gerät gibt die Adresse des jeweiligen Slave-Geräts an, auf das es Daten schreiben oder von dem es Daten lesen muss. Nur dieses bestimmte Slave-Gerät wird darauf reagieren.

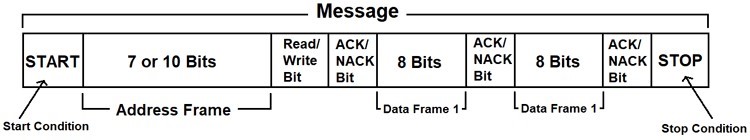


Abbildung 23 I2C Frames

(Quelle: https://i.stack.imgur.com/8QMAz.png)

## Arduino PWM

Die Pulsweitenmodulation (PWM) ist eine Möglichkeit, ein variables Signal auszugeben von einem gegebenen eingestellten Signal. PWM tut dies durch Ändern der Impulsbreite, wodurch wiederrum das Tastverhältnis eines Rechtecksignales geändert wird um die Leistung, die aus dem Pin des Arduino kommt zu regulieren. Es variiert, weil das Signal das Tastverhältnis annimmt und das ausgegebene

Leistungssignal mittelt. Umso größer das Tastverhältnis desto größer ist die Ausgangsleistung. Der Arduino UNO hat 6 Pins, welche ein PWM ausgeben könne, diese sind mit einer Welle (~) auf dem Board gegenzeichnet oder im Datenblatt aufgelistet. Die Spannung wechselt hier zwischen 0V und 5V in einem bestimmten

Verhältnis. Wenn das Tastverhältnis bei 50% also 1:1 liegt so gibt der Arduino eine Spannung aus welches die Hälfte der Zeit eins Takes der PWM auf 5V liegt und die andere Hälfte auf 0V. Im Code wird das mit Hilfe des Befehls analogWrite(Pin, Zahl); gemacht. Die Zahl kann hierbei ein Byte lang sein also von 0 – 255. Wenn man die Zahl 127 reinschreibt wird ein 50% Tastverhältnis (duty-cycle) ausgegeben. Bei 0 wird das Signal auf 0V bleiben und bei 255 werden konstant 5V ausgegeben ohne einen Wechsel auf 0V. Der einzige Nachteil bei diesem PWM Signal welches vom Arduino schon automatisch erzeug werden kann ist, dass die Frequenz des PWM

Signals nicht eingestellt werden kann, sondern fest vorgeben ist. Bei diesem Projekt

wird ein PWM Signal gebraucht um den Servo einzustellen und für die H-Brücke der Motorsteuerung.

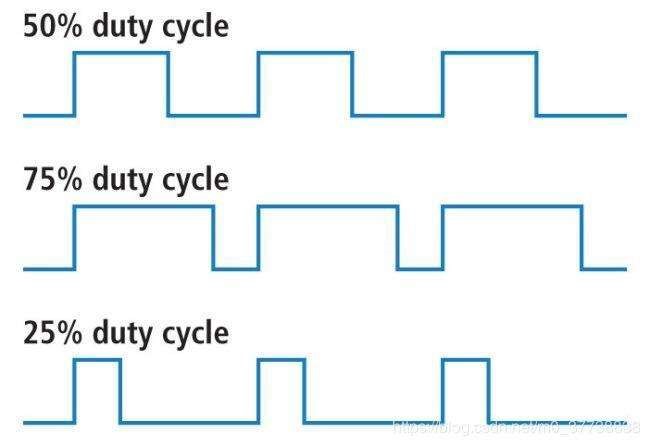


Abbildung 24 Arduino PWM

(Quelle: https://www.programmersought.com/images/61/354d87b8a0ca71d26b831f738de406fd.png)

## Bibliotheken

Bibliotheken bieten zusätzliche Funktionen, welche einfach in jedem Sketch verwendet werden können. Aber um eine Bibliothek verwenden zu können muss man diese zuerst in den Sketch einbinden. Dazu muss man die Bibliothek Datei, welche die Funktionen beinhaltet in den dafür vorgesehenen Bibliotheken Ordner, welcher sich im Installationspfad der Arduino IDE befindet. Die IDE hat dafür eine eingebaute Funktion, um das zu vereinfachen durch einfaches Einbinden einer Bibliothek durch Klicken. Danach um diese noch in einen Sketch zu Aktiviren muss man lediglich am Anfang der Datei mit #include und <Name> eingegrenzt den Namen der Bibliothek schreiben. Das ermöglicht dann beim Programmieren nicht nur auf die Vorgefertigten Funktionen, welche bei der Arduino IDE selbst mit hinein Programmiert wurden zuzugreifen, sondern auf alle Funktionen, welche in der neuen Bibliothek auch sind.

Die Standard Bibliotheken mit der die Arduino IDE kommen sind: EEPROM, Ethernet, Formata, GSM, LiquidCrystal, SD, Servo, SPI, SoftwareSerial, Stepper, TFT, WiFi und Wire. Die wichtigsten Funktionen dieser Bibliotheken kann man benutzen ohne die extra einbinden zu müssen wie zum Beispiel die oben genannten Funktionen Serial.begin(); oder analogWrite();.

## Servo Lenkung

Um einen Servo mit den Arduino anzusteuern gibt es eine Bibliothek, welche die passenden Funktionen dafür hat, um das ganze recht einfach umzusetzen. Um zu testen wie weit der Servo lenken darf, ohne in Konflikt zu kommen wurde ein kleines Testprogramm verwendet.

#include <Servo.h> damit wurde die Bibliothek geladen, welche alle Funktionen die wir brauchen Beinhaltet

Servo servoLenkung; Zu Beginn wird ein Servo für das Programm definiert. Der Servo trägt hierbei den Namen servoLenkung. Dies ist schon die erste Funktion aus der Servo Bibliothek.

byte x = 100; Dann wurde eine Variable mit den Datentyp byte verwendet

da bei der Bibliothek die Werte für die Position des Servos nur von 0°-180° ausgelegt sind und man daher nicht mehr als 180 braucht. Die Variable trägt den Namen x und stellt den Wert da auf welchem der Servo später ausgerichtet wird

void setup() Damit wird die setup() Funktion gestartet. Sie um fasst alles

innerhalb geschwungenen Klammerns {}

{

servoLenkung.attach(8); Diese Funktion aus der Servo Bibliothek richtet den Arduino Pin 8 auf OUTPUT. Des Weiteren wird der Servo mit dem Namen servoLenkung mit den Pin 8 für das Programm gespeichert.

servoLenkung.write(90); Diese Funktion schickt jetzt auf den Pin 8 ein PWM Signal,

welches in diesem Fall bei 90 ein Tastverhältnis von 50% hat da die Funktion write nur von 0-180 definiert ist wobei.

}

Mithilfe des Programmes und einigen Testversuchen sind folgende Werte herausgekommen:

101° ist der Wert, welcher im Programm eingestellt werden muss damit der Servo so wie er verbaut ist im Auto das Auto geradeaus fahren lässt.

136° ist der Wert, welcher das Auto maximal nach Links einlenken kann, ohne gegen das Plastik der Servohalterung des Autos zu drücken.

66° ist der Wert, bei dem der Servo gerade so nicht die Lenkung überdreht also der maximale Winkel der Rechtslenkung bevor es sich überdreht und wieder mehr Richtung links einschlägt.

## Motor ansteuern

Um den Motor anzusteuern und zu Regeln wurde eine H-Brücke verwendet diese schaltet große Ströme mit durch kleine Ströme also Signale am Eingang. Die H-Brücke kann Gleichstrom in 2 Polungen Steuern. Die Steuerung erfolgt mittels 2 verschiedener PWM Signale. Diese bestimmen ob der Motor vorwärts oder rückwärtsfährt und mit welcher Leistung der das macht. Dafür wurde ein kleines Testprogramm geschrieben, um zu testen wie der Motor reagiert also wie schnell er ist bei maximaler Auslastung und wie der Motor eingestellt werden muss damit das Auto nicht zu schnell ist.

const byte SENSOR\_PIN = 0; Um das Ändern der Geschwindigkeit und Richtung

etwas leichter zu machen als immer wieder ein neuen Code auf den Arduino hochzuladen, wurde hier eine Variable mit const byte also dem Datentypen Byte definiert. Da sich die Variable nicht ändern soll wird sie auf Konstant gestellt. Auf den Pin wird ein Potentiometer angeschlossen um das Programm zu Regeln.

|  |  |
| --- | --- |
|  |  |
| const byte L1 = 5;  6 ausgewählt | Da ein PWM Signal gebraucht wird wurden die Pins 5 und |
| const byte L2 = 6; | L1 und L2 sind die Namen der Variablen und soll auf die Pins der H-Brücke verweisen, welche verwendet wurde. L1 steht hier für das Schalten des Vorwärtsganges und L2 für den Rückwärtsgang. |
| void setup() | Die setup() Funktion wird gestartet und umfasst alles in den |

geschwungenen Klammern {}

{

pinMode(SENSOR\_PIN, INPUT); Pin für das Potentiometer wird auf INPUT gestellt

|  |  |
| --- | --- |
| pinMode(L1, OUTPUT); | Mit dieser Funktion werden die Pins 5 und 6 auf  OUTPUT im Arduino |
| pinMode(L2, OUTPUT); | gestellt da L1 und L2 für 5 und 6 steht. |
| Serial.begin(9600); |  |

Die Serielle Übertragung wurde mit einer Baudrate von 9600 gestartet.

}

void loop() Die loop() Funktion wird gestartet und umfasst alles in den

geschwungenen Klammern {}.

{

int sensorValue = analogRead(SENSOR\_PIN);

Ein integer welcher nur Lokal in der loop() Funktion gebraucht wird muss daher auch erst in dieser erzeugt werden. Dadurch kann dieser Integer auch nur innerhalb der loop() Funktion verwendet werden. Der Integer bekommt hier den Wert welcher durch die Funktion analogRead(); ausgelesen wird vom Pin 0 welches hierbei das Potentiometer darstellt. analogRead(); startet den analog digital Konverter welcher beim Arduino eine 10Bit Auflösung hat, daher werden die Werte welche sensorValue haben wird im Bereich von 0 bis 1023 liegen.

if (sensorValue < 512) Mit dieser if() Funktion welche alles in den

geschwungenen Klammern {} umfasst wir überprüft ob sensorValue einen Wert hat welcher kleiner als 512 ist also kleiner als die Hälfte. Falls dies zutrifft wird alles in der Klammer {} ausgeführt falls nicht wird diese Übersprungen.

{

int reversePWM = -(sensorValue - 511) / 2; Da ein PWM Signal welches in den

nächsten Zeilen mit der Funktion analogWrite(); geschrieben wird nur einen Wert von 255 haben kann aber der Wert welcher ausgelesen wird vom Potentiometer 0 bis 1023 sein kann wurde hier eine kleine Umrechnung für den Lokalen Integer Wert reversePWM eingerichtet.

|  |  |
| --- | --- |
| analogWrite(L1, 0); | Die Funktion soll den Wert von L1 also dem  Vorwärtsgang immer auf 0 Stellen bevor auf L2 also dem Rückwärtsgang ein Wert geschrieben wird da es sonst zu Hardware Problemen kommen könnte. |
| analogWrite(L2, reversePWM); | Der zuvor Umgerechnete Potentiometer Wert wird jetzt im angepassten Verhältnis als PWM Signal auf dem Pin 6 ausgegeben bis ein neues Signal drauf geschrieben wird. Der Motor fährt damit rückwärts. |
| Serial.println(reversedPWM); | Der umgerechnete Wert, welcher auf dem Pin |

6 ausgegeben wird, wird über den seriellen Monitor ausgegeben damit man sehen kann welchen Wert die Geschwindigkeit die umgesetzt wird hat.

}

else Mit else wird alles in den geschwungenen Klammern {} ausgeführt sobald

die Bedingung von oben (sensorValue < 512) nicht eintrifft, also sobald der gemessene Wert vom Potentiometer zwischen 512 und 1023 liegt.

{

int forwardPWM = (sensorValue - 512) / 2; Hier findet wieder dieselbe Umrechnung wie zuvor statt nur dieses Mal mit dem Integer forwardPWM da es sich hierbei um den Vorwärtsgang handelt.

analogWrite(L2, 0); Das PWM auf L2 also Pin 6 wird auf 0 gesetzte also

ausgeschaltet für den fall das hier noch ein PWM anliegen sollte.

analogWrite(L1, forwardPWM); Der zuvor Umgerechnete Potentiometer Wert wird

jetzt im angepassten Verhältnis als PWM Signal auf dem Pin 5 ausgegeben bis ein neues Signal drauf geschrieben wird. Der Motor fährt damit vorwärts.

Serial.println(forwardPWM); Der umgerechnet Wert, welcher auf dem Pin 5

ausgegeben wird, wird über den seriellen Monitor ausgegeben.

}

}

Mit diesem Programm kann man jetzt einfach an dem Potentiometer drehen, um die Geschwindigkeit und Richtung anzupassen, in der sich der Motor drehen soll.

## LC Display

Um ein kleines herkömmliches LCD-Display über den I2C Bus anzusteuern gibt es eine Arduino Bibliothek, welche auch standardmäßig schon im Library Ordner vorhanden sein sollte. Bei diesem Programm sollen die drei verschiedenen Distanzmessungswerte von den Infrarot Sensoren angezeigt werden. Da man so die Werte immer im Auge behalten kann wenn alles im Auto verbaut ist, da man ohne eine Physische Verbindung zwischen Arduino und Computer den seriellen Monitor nicht benutzen kann.

void Display() { Hier wird eine Funktion mit dem Namen Display() definiert

und wird später im loop() aufgerufen um die Werte welche auf dem Display angezeigt werden konstant zu

aktualisieren

|  |  |
| --- | --- |
| Serial.print("Sensor1"); | Sensor1 wird in dem seriellen Monitor geschrieben um die Werte, welche in den Monitor geschrieben werden auseinander zu kennen. |
| Serial.println(distance\_cm1); | Die gemessene Distanz des ersten Sensors in [cm] |

welche auf die Variable distance\_cm1 geschrieben ist wird auf dem seriellen Monitor ausgegeben um einen direkten Vergleich zu haben ob der am Display angezeigte Wert auch dem Tatsächlichen Wert entspricht welcher gemessen wurde.

distance\_cm\_1[0] = distance\_cm1 % 10;

distance\_cm\_1 ist ein Feld welches 3Byte lang ist. Mit dem % also der Rest Funktion von C wird der Wert distance\_cm1 also der Messwert durch 10 dividiert und der Rest welcher überbleibt wird in die Variable distance\_cm\_1[0] geschrieben. Wenn die gemessene Distanz des ersten Sensors den Wert 23cm hat wird dieser Wert durch 10 dividiert also 2 und 3 Rest. Der Rest wird dann auf die Variable distance\_cm\_1[0] geschrieben. Dies ist notwendig da man beim Schreiben auf solch ein LCD-Display nur einzelne Ziffern an einer Position schreiben kann.

|  |  |
| --- | --- |
| distance\_cm\_1[1] = distance\_cm1 / 10 % 10;  2. Und 3. Byte | Dasselbe wird auch hier für das |
| distance\_cm\_1[2] = distance\_cm1 / 100 % 10; | also für die 2. Und 3. Stelle da der Sensor bis zu 150cm weit messen kann. |
| Serial.println(distance\_cm2); | Dies wird für den Sensor 2 und |
| distance\_cm\_2[0] = distance\_cm2 % 10; | 3 wiederholt. |

distance\_cm\_2[1] = distance\_cm2 / 10 % 10; distance\_cm\_2[2] = distance\_cm2 / 100 % 10; Serial.println(distance\_cm3); distance\_cm\_3[0] = distance\_cm3 % 10; distance\_cm\_3[1] = distance\_cm3 / 10 % 10; distance\_cm\_3[2] = distance\_cm3 / 100 % 10;

lcd.setCursor(0, 0);

Mit der Funktion setCursor(); wird die Adresse des LCD-Displaychips welche für die Anzeige an der Stelle 2 in der Reihe 0 also das dritte Feld von oben zuständig ist als Ziel für den nächsten Befehl gewällt

lcd.print(distance\_cm\_1[2]); Diese Funktion schreibt den Wert der Variablen dann

auf die Position des Displays auf welcher der Cursor gesetzt ist

|  |  |
| --- | --- |
| lcd.print(distance\_cm\_1[1]);    lcd.print(distance\_cm\_1[0]); | und man Schreibt immer den nächsten Ziffernwert hin |
| lcd.setCursor(5, 0);  lcd.print(distance\_cm\_2[2]); lcd.print(distance\_cm\_2[1]); lcd.print(distance\_cm\_2[1]);    lcd.setCursor(10, 0); lcd.print(distance\_cm\_3[2]); lcd.print(distance\_cm\_3[1]); lcd.print(distance\_cm\_3[0]); | Dies wird für alle 3 Sensoren gemacht also für die restlichen beiden wiederholt. Die Cursor Positionen wurden so gewählt, dass sie ungefähr mit gleichem Abstand über das Display verteil aufscheinen. |
| lcd.setCursor(0, 1); | In diesem Fall bedeutet (0,1) das erste Zeichen in |

der zweiten Zeile.

lcd.print("cm"); Bei Wörtern kann man einfach die erste Position auswählen

und es wir normal aufgeschrieben.

}

## Der ganze Code

#include <SharpDistSensor.h> SharpDistSensor Bibliothek von für die Sensoren

wird eingebunden. Die Bibliothek wird verwendet um die Sensoren zu definieren und diese später auszulesen.

#include <Wire.h> Wire Bibliothek wird eingebunden.

#include <LiquidCrystal\_I2C.h> Vorher hinzugefügte LiquidCrystal\_I2C Bibliothek

wird eingebunden.

#include <Servo.h> Servo Bibliothek wird eingebunden.

LiquidCrystal\_I2C lcd(0x27, 16, 2); Hier wird festgelegt um was für einen Display

es sich handelt. In diesem Fall eines mit 16 Zeichen in 2 Zeilen und der HEX-Adresse 0x27. Für ein vierzeiliges I2C-LCD verwendet man den Code "LiquidCrystal\_I2C lcd(0x27,

20, 4)".

Servo servoLenkung; Ein Servo mit den Namen servoLenkung wird

definiert.

int distance\_cm\_1[3], distance\_cm\_2[3], distance\_cm\_3[3]; Für die LCD anzeige int distance\_cm1, distance\_cm2, distance\_cm3; byte Stufe\_1\_oben,Stufe\_1\_unten,Stufe\_2\_oben;

Pins, welche hier Definiert sind, wurden auf konstant (const) gestellt da sich die Pins nicht ändern müssen

const byte L1\_Output = 5; Arduino PWM output pin 5; (L1) const byte L2\_Output = 6; Arduino PWM output pin 6; (L2) const byte ServoLenkung = 8;

servoLenkung.attach(ServoLenkung); Das Setup enthält die Information, dass das Servo an der Steuerleitung mit Pin ServoLenkung(8) verbunden wird.

|  |  |
| --- | --- |
| const byte G\_Rueckwaerts = 30; Dies soll dazu dienen das Verhalten des | |
| const byte G\_Lenken = 50; | selbstfahrenden Autos schnell und  unkompliziert anpassen zu können. |
| const byte Links = 136; | Hier werden die Werte, welche der Servo benutzt |
| const byte Rechts = 66; | definiert. Diese stehen auch oben damit man sie falls |
| const byte Mitte = 101; | nötig schnell ändern kann. |
| byte Lenken; | Eine Variable welche für das Programm als |

Speicher verwendet wird wurde hier definiert.

Die Datentypen der Variablen sind alle so gewählt das sie nicht unnötig groß sind, um möglichst wenig Speicher zu besetzen

|  |  |
| --- | --- |
| const byte sensorPin0 = A0;  const byte sensorPin1 = A1; const byte sensorPin2 = A2;    void setup() { | Hier wurden die 3 Sensoren auf einen ADC gelegt |
| Serial.begin(115200); | Die Serielle Übertragung wird gestartet mit einer |

Baudrate von 115200

lcd.init(); // Im Setup wird das LCD-Display gestartet

lcd.backlight(); // Hintergrundbeleuchtung einschalten (lcd.noBacklight();

schaltet die Beleuchtung aus.

sensor1.setModel(SharpDistSensor::GP2Y0A60SZLF\_5V); sensor2.setModel(SharpDistSensor::GP2Y0A60SZLF\_5V); sensor3.setModel(SharpDistSensor::GP2Y0A60SZLF\_5V);

In der Bibliothek SharpDistSensor.h ist die Übertragungsfunktion des Sensors eingespeichert. Mit diesem Befehl werden jetzt 3 Sensoren erzeugt und einem Model also eine Übertragungsfunktion für die Errechnung des Abstandes zugewiesen.

|  |  |
| --- | --- |
| Stufe\_1\_oben = G\_Rueckwaerts + 10; | Hier werden die Variablen welche für die Grenzen |
| Stufe\_1\_unten = G\_Rueckwaerts; | der Geschwindigkeitsänderung wichtig |

sind berechnet

Stufe\_2\_oben = G\_Rueckwaerts + 30;

}

void loop() {

|  |  |
| --- | --- |
| Sensor(); | void Sensor(); wird ausgeführt |
| Lenkung(); | void Lenkung(); wird ausgeführt |
| Motor(); | void Motor(); wird ausgeführt |
| Display(); | void Display(); wird ausgeführt |

}

void Sensor() {

unsigned int distance1 = sensor1.getDist(); mit dieser Funktion

schickt der Arduino eine

unsigned int distance2 = sensor2.getDist(); Messanfrage an den Sensor.

Der gemessene

unsigned int distance3 = sensor3.getDist(); Wert wird dann auf den int distance3

geschrieben

distance\_cm1 = distance1 / 10; Da der Sensor in mm misst werden die

Werte

distance\_cm2 = distance2 / 10; durch 10 dividiert damit alles in cm zu

rechnen ist

distance\_cm3 = distance3 / 10;

}

void Display() { Siehe Kapitel: LCD Display

Serial.print("Distance 1");

Serial.println(distance\_cm1);

distance\_cm\_1[0] = distance\_cm1 % 10;

distance\_cm\_1[1] = distance\_cm1 / 10 % 10; distance\_cm\_1[2] = distance\_cm1 / 100 % 10;

Serial.print("Distance 2");

Serial.println(distance\_cm2); distance\_cm\_2[0] = distance\_cm2 % 10; distance\_cm\_2[1] = distance\_cm2 / 10 % 10; distance\_cm\_2[2] = distance\_cm2 / 100 % 10; Serial.print("Distance 3");

Serial.println(distance\_cm3); distance\_cm\_3[0] = distance\_cm3 % 10; distance\_cm\_3[1] = distance\_cm3 / 10 % 10; distance\_cm\_3[2] = distance\_cm3 / 100 % 10;

lcd.setCursor(0, 0); lcd.print(distance\_cm\_1[2]);

lcd.print(distance\_cm\_1[1]);

lcd.print(distance\_cm\_1[0]);

lcd.setCursor(5, 0); lcd.print(distance\_cm\_2[2]); lcd.print(distance\_cm\_2[1]); lcd.print(distance\_cm\_2[0]);

lcd.setCursor(10, 0); lcd.print(distance\_cm\_3[2]); lcd.print(distance\_cm\_3[1]); lcd.print(distance\_cm\_3[0]);

lcd.setCursor(0, 1); lcd.print("cm");

}

void Lenkung() {

if ((distance\_cm1 < G\_Lenken) && (distance\_cm2 > G\_Rueckwaerts))

{

Lenken = Rechts;

}

Die Variable Lenken wird auf den Wert für die Rechtslenkung gesetzt, wenn ein Hindernis zu knapp auf der linken Seite des Autos ist also der Sensor1 weniger als 30cm und Sensor2 mehr als 30cm misst. Dadurch Lenkt das Auto nach rechts wenn es zu nahe an einem Hindernis an der linken Seite ist.

else if ((distance\_cm3 < G\_Lenken) && (distance\_cm2 > G\_Rueckwaerts))

{

Lenken = Links;

}

else if() ist so wie eine if() Bedingung welche aber nur ausgeführt wird falls die darüber liegende if() Bedingung nicht eingetroffen ist. Die Variable Lenken wird auf den Wert für die Linkslenkung gesetzt, wenn ein Hindernis zu knapp auf der rechten Seite des Autos ist also der Sensor3 weniger als 30cm und Sensor2 mehr als 30cm misst. Dadurch Lenkt das Auto nach links wenn es zu nahe an einem Hindernis an der rechten Seite ist.

else if (distance\_cm2 < G\_Rueckwaerts)

{

Lenken = Mitte;

}

Die Variable Lenken wird auf den Wert für die Mittenlenkung gesetzt, wenn ein Hindernis zu nahe an der Vorderseite der Autos ist. Also während er im Rückwärtsgang ist. Dadurch fährt es gerade nach hinten und hat mehr weg um das Hindernis zu umfahren.

else {

Lenken = Mitte; Wenn keiner der Bedingung eintrifft also alles Frei ist

fährt er ebenfalls gerade aus. Das macht er solange bis wieder ein Hindernis erkannt wird

}

servoLenkung.write(Lenken); Der aktuelle Wert, welcher gerade in der Variablen Lenken steht wird auf den Servo übertragen und somit beginnt erst das eigentliche Lenken.

}

void Motor() {

if (distance\_cm2 < G\_Rueckwaerts)

{

analogWrite(L2\_Output, 0); analogWrite(L1\_Output, 20);

}

In diesem Teil wird der Rückwärtsgang eingeleitet sobald der gemessene Wert von Sensor 2 also dem Mittlernen Sensor an der Front unter dem Grenzwert fällt in diesem Fall 30cm.

else if ((distance\_cm2 < Stufe\_1\_oben) && (distance\_cm2 > Stufe\_1\_unten))

{

analogWrite(L2\_Output, 20); analogWrite(L1\_Output, 0);

}

Hier wird die langsamste also die sogenannte erste Stufe der Geschwindigkeit eingestellt, falls der Abstand vorne größer ist als 30cm aber kleiner als 40cm.

else if ((distance\_cm2 < Stufe\_2\_oben) && (distance\_cm2 > Stufe\_1\_oben))

{

analogWrite(L2\_Output, 30); analogWrite(L1\_Output, 0);

}

Hier wird die mittlere also die sogenannte zweite Stufe der Geschwindigkeit eingestellt, falls der Abstand vorne größer ist als 40cm aber kleiner als 70cm.

else if (distance\_cm2 > Stufe\_2\_oben)

{

analogWrite(L2\_Output, 40); analogWrite(L1\_Output, 0);

}

}

Hier wird die schnellste also die sogenannte dritte Stufe der Geschwindigkeit eingestellt, falls der Abstand vorne größer ist als 70cm.

Dieser Code würde im Normalfall nicht sehr gut funktionieren da es viele Bereiche gibt wo die Entscheidungsgrenze zwischen 2 zuständen zu knapp bei einander ist. Ein Beispiel dafür ist die Vorwärts- oder Rückwärtsgang Entscheidung. Normallerweiße laut der Logik der Codes sollte das Auto im Bereich von 30cm Abstand zur Wand vor ihm stehen bleiben und nur noch sehr schnell zwischen vor und zurück herumschalten. Da das Auto, aber Träge ist nicht sofort stoppen kann sind das nur Bereiche, in denen es umschaltet zwischen Vorwärts- und Rückwärtsgang aber noch nicht vor und zurückfährt. In der Realität sieht fährt das Auto immer ein paar Zentimeter vor und zurück. Wenn das noch mit einer erzwungenen Lenkung kombiniert wird kann es sich aus jeder Situation freifahren solange es genug Platz zu den Wänden hat und die Hindernisse sichtbar sind.

Wegen Zeitmangel durch Schließung der Schulen konnte der 4. Sensor nicht eingebaut werden, daher gibt es noch kein Programm, welches das Auto auch gegen Hindernisse von hinten schützt.

# Mechanik und Gehäuse (Nikolic)

## Auswahl

Am Anfang war erst zu überlegen, wie das Projekt aussehen sollte, nach einigem Überlegen wurde beschlossen, dass das Gehäuse die Form eines Autos haben soll. Als Nächstes war die Frage, ob es sinnvoller wäre, ein Gehäuse von Grund auf selbst zu bauen, da man dieses leichter den Vorstellungen und Anforderungen anpassen können würde oder ob es besser wäre, einfach einen Modell-Bausatz zu besorgen und dieses dann den Anforderungen nach zu modifizieren und bearbeiten. Nachdem beschlossen war wie, die Diplomarbeit aussehen soll, begann die Suche nach einem passenden Bausatz. Schnell wurde bemerkt, dass nicht irgendein Bausatz genommen werden konnte aufgrund der Tatsache, dass ein Arduino uno, mehrere Infrarot Entfernungsmesser und diverse andere Elektronikkomponenten Platz benötigen. Nach einigem Suchen auf diversen Internetseiten und verschieden

Modellbaugeschäften wurde ein passender Kandidat beim Elektronik-Versandhändler „Conrad Electronic“ gefunden, und zwar ein Chassis vom Hersteller „Tamiya“ mit dem Namen „TT-01 TYPE-E“.



Abbildung 25 TT-01 TYPE-E Bausatz

Interessant war der Bausatz vom Hersteller „Tamiya“ aus mehreren Gründen. Erstens, das Modell-Auto hat genügend Platz für alle benötigten Elektronikkomponenten, es ist also groß genug für unser Vorhaben. Zweitens, als Antriebsart hatte es einen Allradantrieb mit funktionierendem Differenzialgetriebe, da die Diplomarbeit zügig einen Parkour durchfahren muss, könnten diese zwei Faktoren ziemlich hilfreich sein. Noch ein wichtiger Grund, der für diesen Bausatz sprach, war, dass er schon mit einem passenden Motor kam. So mussten wir nicht zusätzlich noch nach einem kompatiblen Motor suchen, der das Fahrzeug in Bewegung bringt. Leider kam das Fahrzeug nicht mit einem passenden Akku und dem passende Servo, diese Gegenstände wurden zusätzlich bestellt und im Laufe des Zusammenbaus hinzugefügt.



Abbildung 26 ausgepackter Bausatz

Wie der Bausatz ausgepackt aussah, kann man in der Abbildung 28 sehen. In der Schachtel befanden sich vier Felgen, vier Reifen, das Chassis, diverse Schrauben und andere Kleinteile, die für den Zusammenbau von Bedeutung waren.

## Einbau des Stirnrads

Sobald die Schachtel offen war, wurde schon mit dem Zusammenbau begonnen. Als Erstes setzte man das Stirnradgetriebe, welches bezeichnet war als GB5, in das

Chassis. Dafür wurde eine Achse (MA11), ein Plastik-Lager (RB1), ein StirnradMitnehmer (GB2) und noch ein hinteres Antriebsgelenk (C5) verwendet.

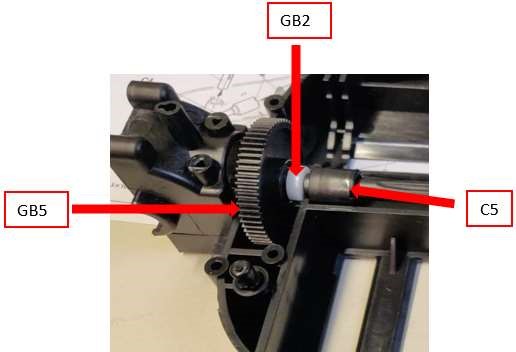


Abbildung 27 Einbau des Stirnrads

In Abbildung 29 sieht man, wie es aussehen sollte. Die Achse (MA11) und das PlastikLager (RB1) sind in der Abbildung nicht verzeichnet, weil sie nicht gesehen werden können, das Plastik-Lager (RB1) befindet sich im Stirnrad Mitnehmer (GB2) und die Achse (MA11) wurde in das passende Loch im hinteren Antriebsgelenk (C5) reingeschoben.

## Einbau der Antriebswelle

Nachdem das Stirnrad eingebaut war, wurde es Zeit für den zweiten Schritt. Der zweite Schritt war der Einbau der Antriebswelle, dafür benötigte man ein vorderes AntriebsGelenk (C4), zwei Plastik Stücke die später für die sichere Befestigung des Akkus relevant sein werden (B8), zwei Schneidschrauben mit den Maßen 3 x 10 mm (MA5) und zum Schluss noch eine Antriebswelle.

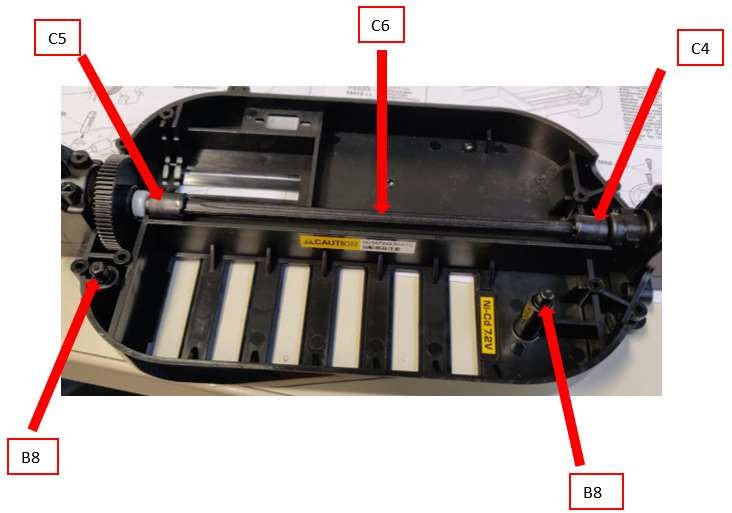


Abbildung 28 eingebaute Antriebswelle

In der Abbildung 30 sieht man, wie es zusammengebaut aussieht. Die Antriebswelle (C6) gehört in das vordere Antriebs-Gelenk (C4) und das hintere Antriebs-Gelenk (C5). Aber bevor man die Antriebswelle (C6) hineinlegt, sollten ihre Spitzen erst mit etwas Keramikfett eingefettet werden, damit es keinen direkten Plastik-an-Plastik Kontakt gibt, somit wird es sich besser drehen können und die Bauteile werden länger halten. Zum Schluss wurden dann die zwei Kunststoff-Stücke (B8) von oben in das dafür vorgesehene Loch gesteckt und anschließend wurden die passenden

Schneideschrauben (MA5) von unten festgedreht.

## Befestigung der Motor – Anbauteile

Als Nächstes wurden die Motor-Anbauteile befestigt. Für den dritten Schritt brauchte man den Motor, eine Motorplatte (MA16), ein Motorritzel (MA12) mit 19 Zähnen durch das die Kraft des Motors weitergeleitet wird an das Stirnradgetriebe, eine Madenschraube (MA7) mit den Maßen 3 x 3 mm, eine Motorhalterung (D6) und zum Schluss noch zwei Schrauben (MA6) mit den Maßen 3 x 6 mm.

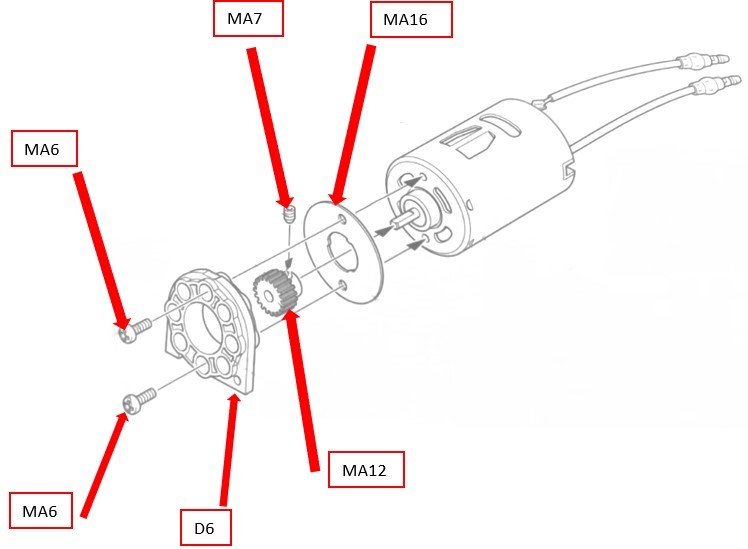


Abbildung 29 Zusammenbau der Motorbefestigung

(Quelle: Tamiya-KIT)

In der Abbildung 31 kann man sehen, wie es zusammengebaut werden muss, um das richtige Ergebnis zu erhalten. Für den Anfang legt man zuerst die Motorplatte (MA16) auf den Motor. Zu beachten ist, dass die zwei Löcher der Motorplatte genau über die Löcher des Motors gelegt werden. Als Nächstes steckt man das Motorritzel (MA12) über den Motor und befestigt ihn, indem man die Madenschraube (MA7) mit dem passendem Innensechskantschlüssel festdreht. Die Madenschraube ist allerdings auch mit dem Keramikfett einzuschmieren damit sie bei der Übertragung der Motorleistung nicht das Stirngetriebe zerstört. Zum Schluss wird dann die Motorhalterung (D6) an dem Motor mit den zwei 3 x 6 mm Schrauben (MA6) zugeschraubt.

## Motor Einbau

Als Nächstes war der Motor einzubauen. Hierfür waren lediglich nur vier Schneidschrauben (MA3) mit den Maßen 3 x 10 mm und ein Kunststoff Stück (D10) welches den Motor, das Stirngetriebe und die Motorritzel abdeckte notwendig.

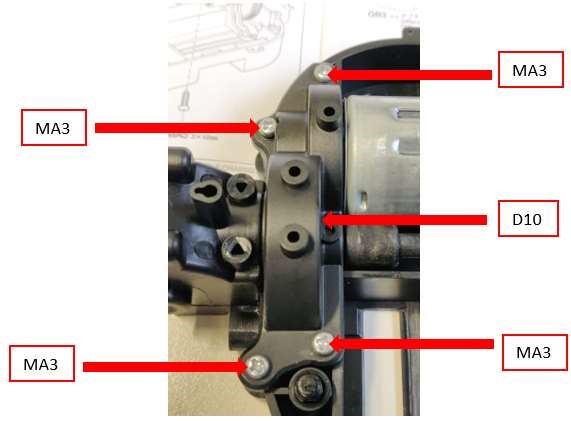


Abbildung 30 eingebauter Motor

In der Abbildung 32 ist zu sehen, wie es schlussendlich aussehen sollte. Als erstes wird der Motor an die passende Stelle gegeben, dabei sollte man aufpassen, dass das Stirngetriebe und die Motorritzel sich gut ineinander setzen lassen. Danach wird das Kunststoff Stück, welches alles abdecken soll (D10) draufgelegt und schraubt es an den oben gezeigten Stellen mit den vier Schneidschrauben (MA3) fest.

## Zusammenbau des hinteren Differenzialgetriebes

Nach dem Motoreinbau kommt der Zusammenbau des Differenzialgetriebes. Dafür werden drei Schneidschrauben (MA8) mit den Maßen 2 x 8 mm, eine Stern-Achse (MA13), zwei große Kegelräder (MA14), drei kleine Kegelräder (MA15), ein Differenzialgehäuse (GB3) und zum Schluss noch ein Tellerrad (GB4) benötigt.

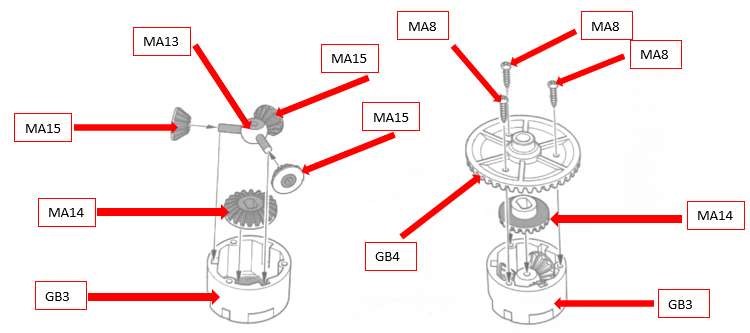


Abbildung 31 Differenzialgetriebe (hinten)

(Quelle: Tamiya-KIT)

In der Abbildung 33 kann gesehen werden, wie es zusammengebaut wird. Begonnen wird in der linken Hälfte. Das Differenzialgehäuse (GB3) wird zuerst von innen mit Keramikfett eingefettet und danach wird ein großes Kegellager (MA14) eingegeben welches ebenfalls geschmiert wird. Als Nächstes wird die Stern-Achse (MA13) genommen und werden die drei kleinen Kegelräder (MA15) darauf geschoben welche zuvor eingefettet wurden. Daraufhin wird die Achse in das Gehäuse gelegt. Beim nächsten Schritt wird ein weiteres eingefettetes großes Kegellager (MA14) genommen und auf die Achse gelegt. Zum Schluss wird dann der Tellerrad (GB4) draufgelegt, hier ist zu beachten, dass die Löcher vom Tellerrad und die vom Gehäuse übereinander liegen und mit den drei Schneidschrauben (MA8) festgeschraubt.

## Einbau des hinteren Differenzialgetriebes

Nach dem Zusammenbau kam der Einbau des hinteren Differenzialgetriebes. Dafür wurden zwei Metall-Lager (RB2), ein Plastik-Lager (RB1), eine Kegelritzel (GB1) und natürlich das davor zusammengebaute hintere Differenzialgetriebe verwendet.

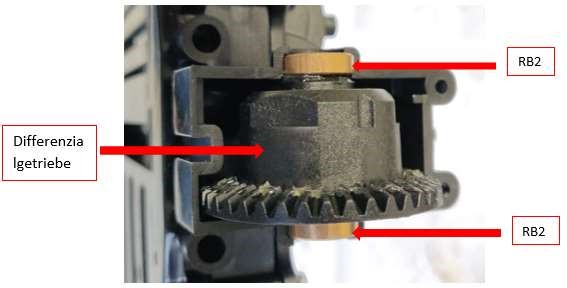


Abbildung 32 Einbau des Differenzialgetriebes (hinten)

In der Abbildung 34 ist zu sehen, wie es eingebaut aussieht. Als Erstes wird das Plastik-Lager (RB1) auf das hintere Antriebs-Gelenk aufgesteckt und anschließend kommt die Kegelritzel (GB1) darauf welche zuvor mit dem Keramikfett gut eingefettet wurde. Danach werden die zwei Metall-Lager (RB2) auf das zusammengebaute hintere Differenzialgetriebe gedrückt und das Differenzialgetriebe wird nochmal beim Tellerrad eingefettet, anschließend wird es einfach auf die Kegelritzel (GB1) gelegt.

## Anbringen der hinteren Querlenker

Für das Anbringen der hinteren Querlenker werden zwei Schneidschrauben (MA1) mit den Maßen 3 x 15 mm und eine Schneidschraube (MA2) mit den Maßen 3 x 12 mm gebraucht. Zusätzlich verwendet man noch die zwei obere Querlenker (B4), die zwei untere Querlenker (B14), eine Plastikabdeckung (A9) für das Differenzialgetriebe welche zugleich die oberen und unteren Querlenker haltet und noch eine Kunststoffhalterung (A10) welche später die Form des Autos halten soll.

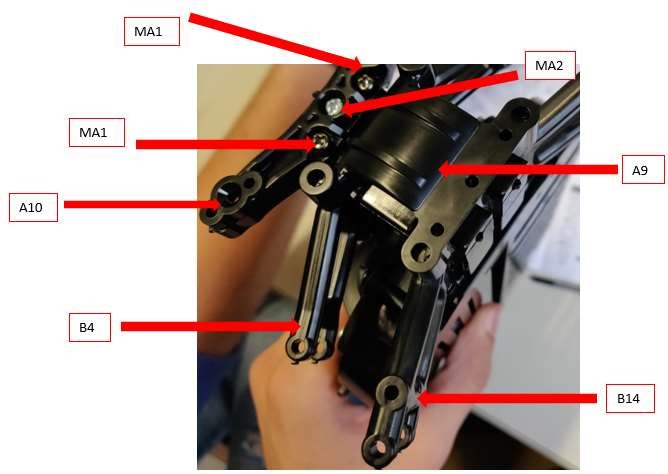


Abbildung 33 Anbringen der hinteren Querlenker

In der Abbildung 35 sieht man, wie es fertig ausschaut. Es wird begonnen mit den oberen Querlenkern (B4) und den unteren Querlenkern (B14) welche in die dafür vorgesehenen Lücke eingegeben wird aber zuvor werden die Spitzen der Arme eingefettet. Anschließend wird die Kunststoffabdeckung (A9) auf die oberen und unteren Querlenker darauf gegeben und die Plastikhalterung (A10) draufgelegt. Zum Schluss wird es mit den Schneidschrauben (MA1) und der anderen Schneidschraube (MA2) zugeschraubt.

## Hinterachsen

Die Hinterachse besteht aus drei Bauteilen und ist ganz simpel aufgebaut. Verwendet wurden zwei Plastik-Lager (RB1), zwei Radachsen (C2) und noch das Gehäuse (A6) der Achsen.

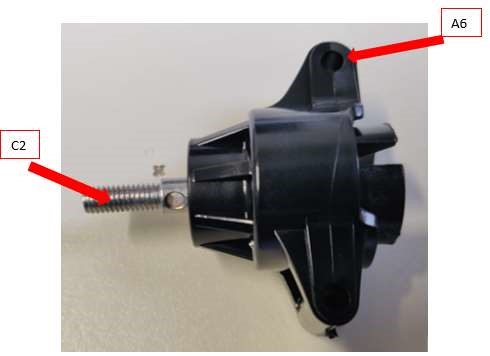


Abbildung 34 Hinterachse

In der Abbildung 36 ist eine von zweien fertiggestellten Hinterachsen zu sehen. Folgendermaßen wird es zusammengefügt, zuerst wird das Plastik-Lager (RB1) etwas eingefettet mit dem Keramikfett und anschließend auf die Radachse (C2) durchgesteckt. Danach wird die Radachse mit dem Kunststoff-Lager in das Achsengehäuse (A6) hineingelegt.

## Einbau des hinteren Stoßfängers

Der Einbau des hinteren Stoßfängers ging ebenfalls schnell und schlicht. Für den Einbau wurden lediglich drei Schneidschrauben (MA3) mit den Maßen 3 x 10 mm, zwei Schneidschrauben (MA4) mit den Maßen 3 x 12 mm und noch eine Plastikplatte (B6) die den hinteren Stoßfänger darstellen sollte benötigt.

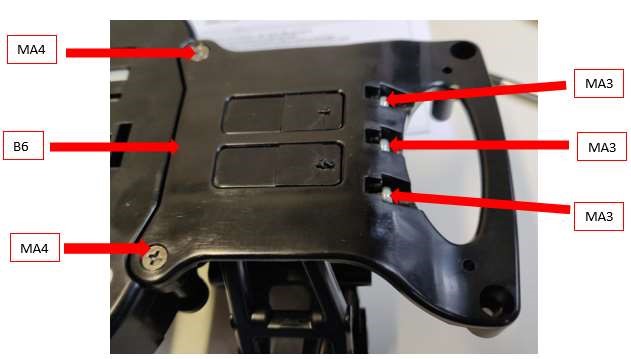


Abbildung 35 hinterer Stoßfänger

In der Abbildung 37 ist zu sieht, wie es angebracht aussehen sollte. Der Einbau ist zügig erledigt, als Erstes wird die Plastikplatte (B6) genommen und auf das Chassis draufgelegt, wichtig ist, dass sich die Öffnungen ohne Abstände überlappen. Als Nächstes werden die drei Schneidschrauben (MA3) an den oben dargestellten Stellen festgeschraubt, gefolgt davon werden dann die anderen zwei Schneidschrauben (MA4) ebenfalls zugeschraubt.

## Hinterachsen Einbau

Nach dem Einbau der hinteren Stoßstange wird nun die Hinterachse eingebaut. Hierfür wurden zwei Schneidschrauben (MA5) mit den Maßen 3 x 10 mm, zwei Schraubzapfen (MA9) mit den Maßen 3 x 32 mm, zwei Schraubzapfen (MA10) mit den Maßen 3 x 22 mm, zwei Achswellen (C1), zwei Getriebegehäuse-Gelenke (C3) und noch zwei Halterungen (B5) welche die Form des Autos halten soll.

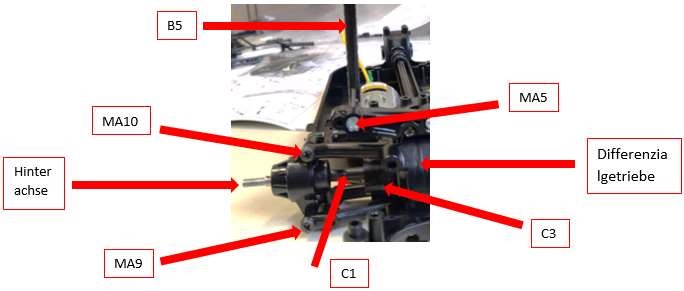


Abbildung 36 Einbau der Hinterachse

In der Abbildung 38 wird dargestellt wie alles eingebaut und verschraubt aussieht. Es sieht etwas komplizierter aus, als es in Wirklichkeit ist. Angefangen wird, indem das Getriebegehäuse-Gelenke (C3) etwas eingefettet mit Keramikfett wird und es anschließend in das Differenzialgetriebe eingeführt wird. Danach wird eine Achswelle (C1) genommen welche am Anfang und Ende eingeschmiert ist und es dann in das Getriebegehäuse-Gelenke (C3) hineingesteckt wird. Als nächstes wird die Hinterachse genommen, auf das Ende der Achswelle (C1) gehalten, die B4 und B14 in Position gebracht und mit den Schraubzapfen (MA10) und den Schraubzapfen (MA9) festgeschraubt. B4 und B14 sind in der Abbildung 36 zu sehen. Zum Schluss werden die Halterungen (B5) in die Vertiefungen eingeschoben, wo sie hingehören und mit den Schneidschrauben (MA5) angebracht.

## Vorderer Querlenker

Der vordere Querlenker bestehet nur aus drei verschiedenen Bauteilen, aus einer Gelenkkugel (P7), den oberen Querlenker (B9) und einen unteren Querlenker (B10). Um ihn zusammenzubauen wird nur die Gelenkkugel (P7) eingefettet und in den Querlenker hineinpressen.



Abbildung 37 Querlenker

## Zusammenbau des vorderen Differenzialgetriebes

Für den Zusammenbau des vorderen Differenzialgetriebes werden zwei große Kegelräder (MA14) und drei kleine Kegelräder (MA15) verwendet. Zusätzlich werden noch die Stern-Achse (MA13), drei Schneidschrauben (MA8) mit den Maßen 2 x 8 mm, ein Differenzialgehäuse (GB3) und noch ein Tellerrad (GB4) eingesetzt. Der Zusammenbau des vorderen Differenzialgetriebes wird genau gleich ausgeführt wie der Zusammenbau des hinteren Differenzialgetriebes. Um den detaillierten Aufbau zu erfahren siehe Seite 60; Zusammenbau des hinteren Differenzialgetriebes.

## Einbau des vorderen Differenzialgetriebes

Um das Differenzialgetriebe einzubauen, benötigt man zwei Kunststoff-Lager (RB1), zwei Metall-Lager (RB2), eine Distanzscheibe aus Kunststoff (P3), eine Kegelritzel (GB1) und das zuvor zusammengebaute vordere Differenzialgetriebe. Der Einbau des vorderen Differenzialgetriebes ähnelt dem Einbau des hinteren Differenzialgetriebes jedoch gibt es einen Unterschied der bedeuten ist. Es wird genau gleich begonnen wie bei dem anderen Einbau jedoch wird nun nicht sofort die Kegelritzel (GB1) auf den Antrieb-Gelenk geschoben, sondern die Distanzscheibe aus Plastik (P3). Nach der Distanzscheibe wird noch ein Plastik-Lager eingeführt und danach wird der Einbau gleich fortgeführt wie bei dem hinteren Differenzialgetriebe. Für den detaillierten Einbau siehe Seite 61; Einbau des hinteren Differenzialgetriebes.

## Anbringen der vorderen Querlenker

Für diesen Arbeitsschritt werden zwei Schneidschrauben (MA1) mit den Maßen 3 x 15 mm und eine Schneidschraube (MA2) mit den Maßen 3 x 12 mm verwendet. Zudem noch zwei obere Querlenker mit Gelenkkugeln (B9), zwei untere Querlenker mit Gelenkkugeln (B10), eine Abdeckung (A9) für das Differenzialgetriebe und eine Plastikhalterung (A10). Dieser Aufbau ist dem Aufbau der hinteren Querlenker identisch, nur werden statt den oberen Querlenker (B4) und den unteren Querlenker (B14), obere Querlenker mit Gelenkkugeln (B9) und untere Querlenker mit Gelenkkugeln (B10) eingesetzt. Siehe Seite 62; Anbringen der hinteren Querlenker, für eine detaillierten Anleitung.

## Einbau des vorderen Stoßfängers

Um den vordere Stoßfänger einzubauen, werden zwei Schneidschrauben (MA4) mit den Maßen 3 x 12 mm, ein Kunststoff-Stoßfänger (B6) und drei Schneidschrauben (MA3) mit den Maßen 3 x 10 mm verwendet. Der Einbau des vorderen Stoßfängers wird genau gleich ausgeführt wie der Einbau des hinteren Stoßfängers, um die detaillierte Anleitung zu erblicken, siehe Seite 63; Einbau des hinteren Stoßfängers.

## Vordere Achsenschenkel

Dieser Zusammenbau ähnelt den Zusammenbau der Hinterachsen. Hierfür wurden zwei Radachsen (C2), zwei Radachsengehäuse (A3) und zwei Plastik-Lager (RB1) benötigt. Um den detaillierten Zusammenbau zu erfahren, siehe Seite 62; Hinterachsen. Zu beachten ist jedoch, dass statt dem Achsengehäuse (A6) das Radachsengehäuse (A3) eingesetzt wird. Die restlichen Schritte können genau gleich aufgebaut werden.

## Einbau der vorderen Achsenschenkel

Nachdem die vorderen Achsschenkel zusammengebaut wurden, sollten sie nun eingebaut werden. Dafür werden folgende Sachen benötigt, zwei Achswellen (C1), vier Kunststoffteile, die einen Dichtring darstellen sollen (A1), zwei Schneidschrauben (MA5) mit den Maßen 3 x 10 mm, zwei Schneidschrauben (MA3) mit den Maßen 3 x

10 mm, zwei Halterungen (B13) für die Form des Autos und zwei GetriebegehäuseGelenke (C3).

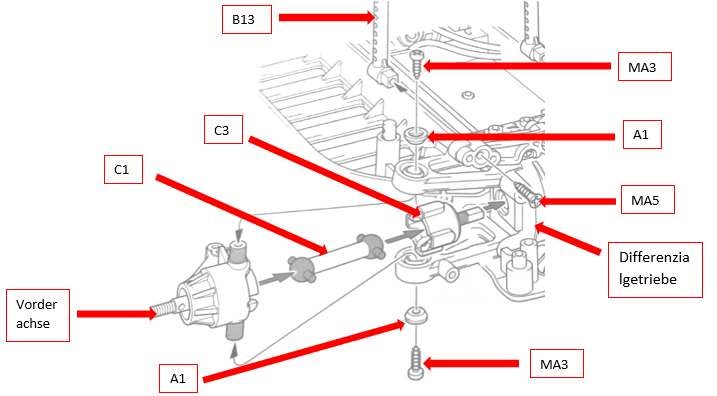


Abbildung 38 Einbau der Vorderarme

(Quelle: Tamiya-KIT)

In der Abbildung 40 ist abgebildet, wie die Bauteile zusammengefügt werden müssen. Der erste Schritt ist es, das Getriebegehäuse-Gelenke (C3) in das Differenzialgetriebe zu schieben und anschließend wird die Achswelle (C1), welche an den in der Abbildung zu sehenden grauen Stellen mit Keramikfett eingeschmiert wird, in das Getriebegehäuse-Gelenke (C3) gesteckt. Auf die Achswelle (C1) wird dann die Vorderachse aufgezogen und an den grau gekennzeichneten Flächen etwas eingefettet. Danach wird es über die oberen- und unteren Querlenker festgeschraubt mit den Schneidschrauben (MA3), zwischen den Schrauben und den Armen kommen jedoch Dichtringe (A1), die es ermöglichen sollen, dass sich die Vorderachse bewegen kann. Schlussendlich werden die Halterungen (B13) mit den Schneidschrauben (MA5) angebracht.

## Zusammenbau des Stoßdämpfers

Nun wird der Stoßdämpfer zusammengebaut. Der Dämpfer besteht aus sechs verschiedenen Bauteilen, und zwar einer Spiralfeder (MB5), einer Distanzscheibe (B15), ein oberes Befestigungsgelenk (B3), ein unteres Befestigungsgelenk (B7), ein Schutzrohr (B11) und eine Klammer (B12) die das obere Befestigungsgelenk (B3) mit dem Schutzrohr (B11) zusammenhalten soll.



Abbildung 39 Aufbau des Dämpfers

In der Abbildung 41 wird dargestellt, welche Bauteile benötigt werden, um den Dämpfer zu bauen. Als Erstes wird das Schutzrohr (B11) genommen und auf das obere Befestigungsgelenk (B3) aufgesteckt. Zu beachten ist, dass sich die Löcher, welche sich in beiden Bauteilen befinden, gegenseitig überlappen. Danach muss die Klammer (B12) genommen werden und durch die Lücke in beiden Bauteilen geschoben werden damit sie miteinander verbunden werden können. Dann wird die Distanzscheibe (B15) über das Schutzrohr (B11) gezogen gefolgt von der Spiralfeder (MB5). Zu guter Letzt wird dann das untere Befestigungsgelenk (B7) in das Rohr hineingeführt und um 90° nach rechts gedreht, um es zu fixieren.

## Einbau der Stoßdämpfer

Nun werden die vier Stoßdämpfer jeweils an jeder Achse eingebaut. Dieser Einbau war ziemlich simpel und wurde rasch erledigt. Dafür wurden lediglich acht Paßschrauben (MB2) mit den Maßen 3 x 14 mm gebraucht, welche an dem unteren Querlenker (B10/B14) und der Plastikhalterung (A10) befestigt werden sollen.

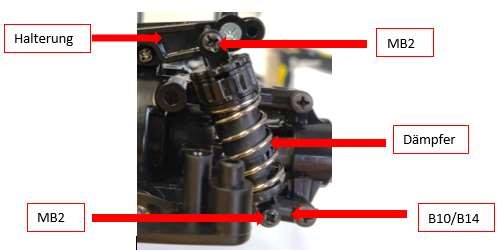


Abbildung 40 Einbau des Dämpfers

In Abbildung 42 kann beobachtet werden, wie der Dämpfer eingebaut ausschaut. Als erstes wird einfach der Dämpfer genommen und mit der Paßschraube (MB2) an der Halterung befestigt. Danach wird er mit einer weiteren Schraube (MB2) an den unteren Arm angeschraubt. An der Vorderseite des Autos wird der Dämpfer am Bauteil B10 montiert und an der Hinterseite wird es am Bauteil B14 angebraucht.

## Lenkgestänge

Jetzt wird das Lenkgestänge zusammengestellt. Das Lenkgestänge dient zum Bewegen der vorderen Achsen, welche durch einen Servo gelenkt werden. Dazu werden zwei Paßschrauben (MB2) mit den Maßen 3 x 14 mm benötigt außerdem noch eine Lenkstange (B1), ein Querlenker (D11) und zwei Plastik – Lager (A2).



Abbildung 41 eines der zwei Lenkgestänge

Wie man in der Abbildung 43 sehen kann, sind das alle Bauteile, die für diesen Vorgang benötigt werden. Dieser Schritt ist ziemlich simpel und schnell erledigt. Als erstes werden die zwei Plastik-Lager (A2) mit etwas Kraft in die große mittige Durchbohrung des Querlenkers (D11) gepresst. Nachdem das abgeschlossen wurde, wird nun die Lenkstange (B1) mit der Paßschrauben (MB2) festgemacht. Wichtig dabei ist, dass die Lenkstange (B1) unter dem Querlenker (D11) gehalten wird und die Paßschrauben (MB2) von oben festgeschraubt wird.

## Befestigung des Urethan–Stoßfängers

Als Nächstes wird der Schaumstoff an der vorderen Stoßstange befestigt. Für diesen Schritt werden zwei Schneidschrauben (MA3) mit den Maßen 3 x 10 mm gebraucht, ein Stück Schaumstoff und eine Halterung (D4) die es stabil halten soll.

In der unteren Abbildung 44 ist zu sehen, wie es montiert ausschaut. Dazu wird lediglich die Urethan-Stoßstange über die vordere Stoßstange (B6) draufgelegt. Zu beachten ist, dass der Schaumstoff über die Schraublöcher darübergelegt wird. Folgend wird die Halterung (D4) ebenfalls draufgelegt und mit den Schneidschrauben (MA3) festgeschraubt.



Abbildung 42 Urethan-Stoßstange

## Einbau des Lenkgestänges

Sobald die Schaumstoff-Stoßstange angebracht wurde, können nun auch die Lenkgestänge eingebaut werden. Hierfür werden bloß zwei Schrauben (MB1) mit den Maßen 3 x 12 mm, zwei Paßschrauben (MB2) mit den Maßen 3 x 14 mm, zwei Lagerzapfen (MB4) der Lenkung und natürlich die zuvor zusammengestellten Lenkgestänge benötigt.

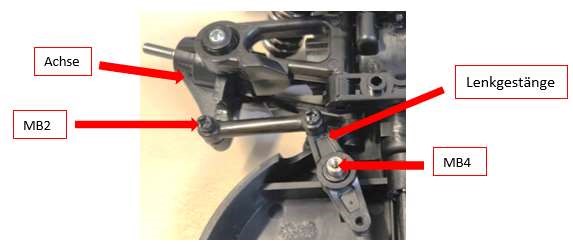


Abbildung 43 Einbau des Lenkgestänges

In der Abbildung 45 wird dargestellt, wie es nach dem Einbau ausschauen sollte. Leider können bei der obigen Abbildung nicht alle verwendeten Bauteile gesehen werden, deswegen wird der Einbau nun genau erklärt. Unter des Lagerzapfen (MB4) befindet sich eine Bohrung, in der die Schraube (MB1) ungefähr 7 mm mit einem Innensechskantschlüssel verschraubt wird, danach wird per Hand der Lagerzapfen (MB4) auf die Schraube (MB1) festgeschraubt. Nachdem dieser Teil erledigt wurde kann nun das Lenkgestänge auf den Lagerzapfen (MB4) darauf getan werden und an der Achse mit einer Paßschraube (MB2) dran geschraubt werden.

## Zusammenbau der Lenkanlage

Die Lenkanlage besteht aus mehreren Bauteilen, die an einen Servo verschraubt sind. Der Servo ist das wichtigste Bauteil der Lenkung, da er die Befehle vom Arduino erhält und die Vorderachsen lenkt. Neben den Servo gibt es noch eine Verbindungsachse (D7), eine Steuerachse (P6), ein Verbindungselement (P4), eine Distanzscheibe (P5), ein Steuerelement (P2), ein Kunststofflager (P1), zwei Paßschrauben (MB2) mit den Maßen 3 x 14 mm und zu guter Letzt eine Schneidschraube (MC5) mit den Maßen 3 x 10 mm.

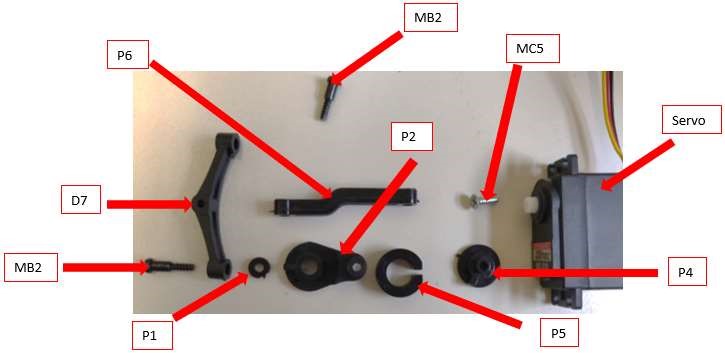


Abbildung 44 Lenkanlage

Wie in der Abbildung 46 zu sehen ist, sind viele Bauteile verwendet worden, um die Lenkanlage zu vervollständigen. Der erste Schritt hier ist es, das Verbindungselement (P4) auf den Servo zu pressen. Auf das Verbindungselement (P4) kommt dann die Distanzscheibe (P5). Danach wird das Kunststofflager (P1) genommen und in das größere Loch im Steuerelement (P2) hineingedrückt. Anschließend wird das Steuerelement (P2) durch das Kunststofflager (P1) mit einer Schneidschraube (MC5) auf die Distanzscheibe (P5) befestigt. Als Nächstes wird die Steuerachse (P6) mit der Paßschraube (MB2) an das Steuerelement (P2) daran geschraubt. Zu beachten ist jedoch, dass die kürzeren Teile der Steuerachse (P6) vor den längeren liegen und an diesem wird er auch befestigt. Nun wird die Verbindungsachse (D7) an dem längeren Teil der Steuerachse (P6) mit einer weiteren Paßschraube (MB2) festgemacht.

## Einbau der Lenkanlage

Für den Einbau der Lenkanlage wird nicht viel gebraucht und dieser Schritt wurde rasch abgeschlossen. Hierfür werden nur Halterungen (A4) für den Servo, zwei Kragenschrauben (MC2) mit den Maßen 3 x 10 mm, zwei Paßschrauben (MB2) mit den Maßen 3 x 14 mm und noch zwei Schneideschrauben (MA5) mit den Maßen 3 x 10 mm benötigt.



Abbildung 45 Einbau der Lenkanlage

In der Abbildung 47 ist zu sehen, wie die Lenkanlage eingebaut aussehen sollte. Begonnen wird, indem die Halterung (A4) an den Servo mit den Kragenschrauben (MC2) befestigt wird, danach wird der Servo in die passende Stelle in das Chassis gelegt und von unten mit zwei Schneideschrauben (MA5) daran geschraubt. Zum Schluss wird die Verbindungsachse (D7) mithilfe der Paßschrauben (MB2) an der linken und rechten vorderen Achse befestigt.

## Einbau der Räder

Nun werden die Räder an das Fahrzeug angebracht. Dazu werden die Räder benötigt, welche aus einer Felge und den Reifen bestehen, vier Steckachsen (MA11) mit den Maßen 2 x 10 mm, vier Sicherungsmuttern (MC9) mit einem Durchmesser von 4 mm, vier Plastiklager (RB1) und noch eine Kunststoffmutter (A7) die gleichzeitig als Distanzscheibe dient.

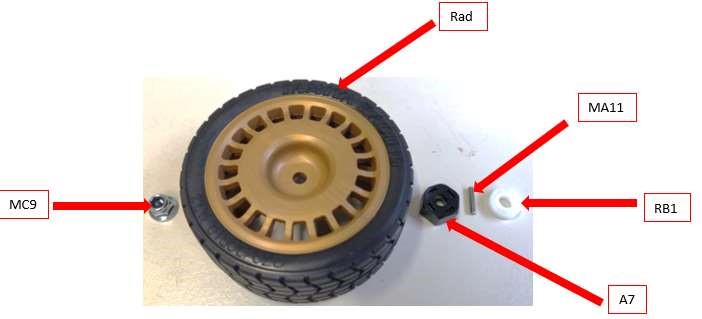


Abbildung 46 Zusammenbau eines Rads

In der oben dargestellten Abbildung 48 kann beobachtet werden, welche Bauteile benötigt werden für die Montage der Räder. Begonnen wird mit dem Plastiklager (RB1), das wird als erstes etwas eingefettet und auf die Radachse übergezogen, danach wird die Steckachsen (MA11) genommen und in die Lücke der Radachse hineingesteckt. Wenn das erledigt wurde, kommt nun die Kunststoffmutter (A7) zum Einsatz, sie wird einfach nur daraufgesteckt. Nach der Plastikmutter (A7) wird das Rad darauf gegeben und mit der Sicherungsmuttern (MC9) festgezogen. Das wird insgesamt viermal gemacht, bis alle Räder montiert sind.

## Karosserie

Nachdem das ganze Fahrzeug schrittweise aufgebaut wurde, wird nun die Karosserie überarbeitet und angebracht, um das Auto komplett zu haben. Die Karosserie wurde hergestellt von Tamiya und wurde einfach zu dem Bausatz hinzugefügt, dadurch musste die Karosserie des Fahrzeugs erst etwas überarbeitet werden, um für unser Projekt zu passen.

Als aller erstes wurden die vier Löcher an den passenden Stellen in die Karosserie gebohrt, um zu überprüfen, ob sich die Halterungen der Karosserie an der richtigen Stelle befinden. Nachdem das erledigt und sichergestellt wurde, dass soweit alles passt, wurden die Stellen ausgeschnitten, wo die Räder hinkommen, damit sie genug Platz haben, um sich zu drehen und problemlos zu lenken. Da die Form etwas kompliziert war zum Schneiden kamen drei Werkzeuge zum Einsatz. Eine Schere, ein Cutter und ein Multifunktionswerkzeug welches akkubetriebenen war vom Hersteller Dremel. In der Abbildung 49 kann man sehen, wie das Endergebnis aussah nach der Anpassung der Karosserie.



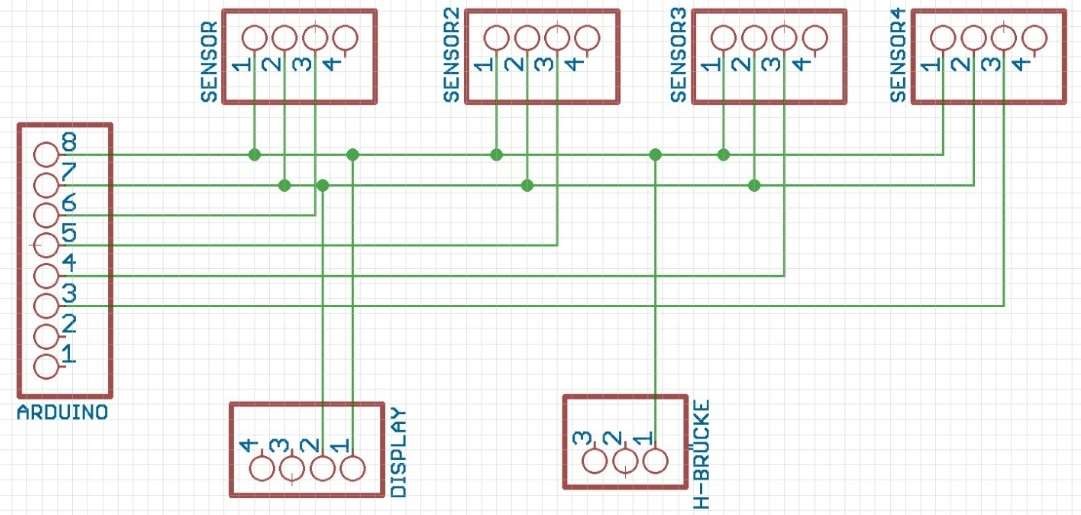
Abbildung 47 Karosserie nach der Bearbeitung

# Platinenanfertigung (Nikolic)

Nachdem das Projekt begonnen wurde, wurde schnell klar, dass viele Drähte zum Einsatz kommen würden. Der Arduino war das Gehirn des Autos und alles ging von ihm aus, also musste alles mit dem Arduino verbunden werden mit Drähten. Nach einer Weile kamen so viele Drähte zum Einsatz, dass sie durcheinander kamen. Um etwas Ordnung im Gehäuse zu schaffen und uns die Arbeit zu vereinfachen wurde entschieden, eine Platine zu fertigen.

## Eagle

Eagle ist eine Software der Firma „CadSoft“ und dient zum Designen einer Leiterplatine bevor sie gefertigt wird. Dieses Programm wird auch Electronic – Design – Automation (EDA) Programm genannt. Der Name Eagle ist nur eine Abkürzung und er steht für „Einfach Anzuwendender Grafischer Layout-Editor“. Diese Software ermöglichte uns, eine Leiterplatine zu designen, die nicht größer als 5 x 4.5 cm war. Über sie wurden alle elektronischen Komponenten miteinander verbunden ohne viel Durcheinander unterhalb der Karosserie zu haben. In der darunter dargestellten Abbildung ist der simple Plan zu sehen, welcher erstellt wurde.

 Abbildung 480 Beschrifteter Plan in Eagle

## Platine

Nachdem der Entwurf der Leiterplatte fertiggestellt wurde, konnte die Platine auch schon hergestellt werden. Bevor eine Platine geätzt werden konnte, musste erst ein Film hergestellt werden, auf dem der zuvor in Eagle hergestellte Plan der Verbindungen aufgedruckt wird. Anschließend wird der Film genommen, über die Kupferplatine gelegt und in der Dunkelkammer belichtet, sodass sich der Plan auf die Platine abbildet. Danach wird die Platine genommen und in der Ätzkammer mit einer starken alkalischen Lösung auf Ammoniakbasis bearbeitet, damit das freiliegende Kupfer beseitigt wird. Nach der Säurebehandlung wird die Leiterplatte erstmal in einer Maschine gereinigt, anschließend gebohrt und zum Schluss werden die benötigten Bauteile angelötet. In der Abbildung 51 kann das fertige Ergebnis betrachtet werden.

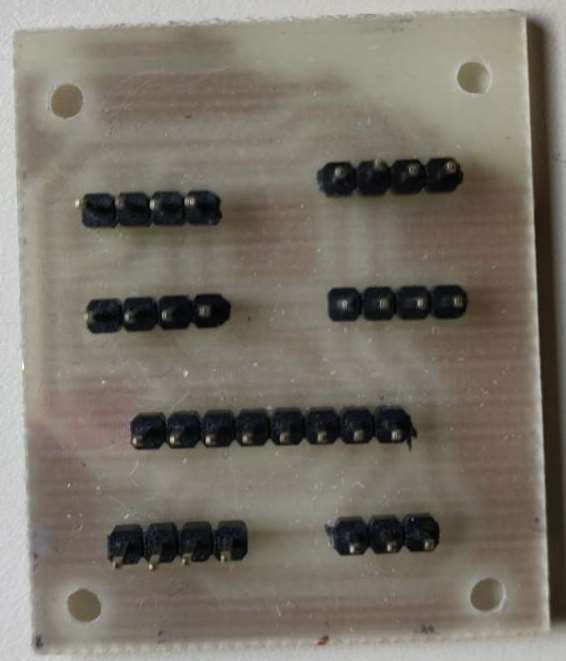
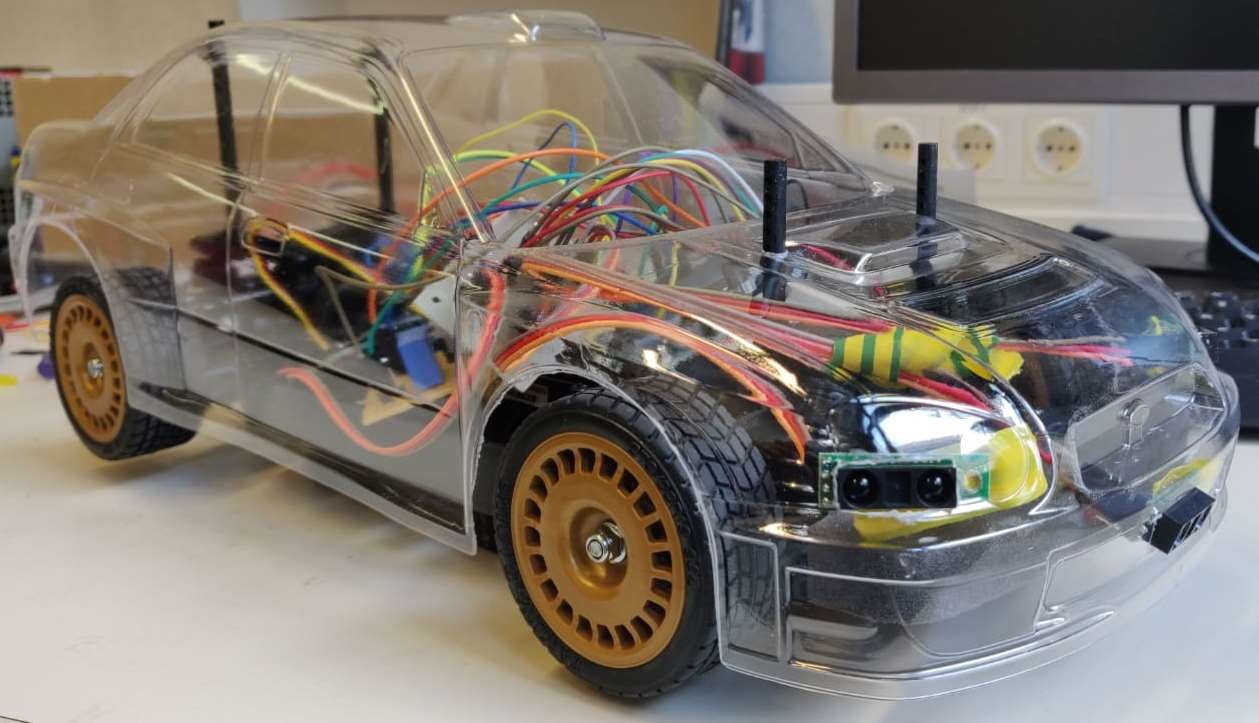


Abbildung 491 fertige Platine

# Komplettes Auto (Nikolic)

Nachdem alle Schritte nacheinander abgeschlossen wurden, konnten nun alle elektronischen Komponenten miteinander verbunden und angeschlossen werden. Sobald alles verkabelt wurde, konnten die Sachen provisorisch in das Gehäuse hineingelegt werden, um zu testen, ob alles funktionieren wird. Danach wurden noch die Sensoren an der Karosserie befestigt. Anschließend wurde das Fahrzeug getestet und Probleme, die aufgetaucht sind, sofort gelöst.

Abbildung 502 Komplettes Auto



Sensor

Sensor

Abbildung 52 zeigt das Projekt zu unserem derzeitigen Stand. Aufgrund des COVID19 Vorfalls konnte das Projekt leider nicht komplett fertiggestellt werden, deshalb sind die elektronischen Komponenten nur provisorisch gesichert. Vorne an der Karosserie sind zwei unserer drei Sensoren zu sehen, welche nur mit einem Isolierband befestigt worden sind. Geplant war es die Sensoren an einer von uns designten Halterung zu befestigen, aber leider kamen wir nicht dazu. Aus demselben Grund kamen wir nicht zur besseren Montierung des Arduinos, der H-Brücke und der Platine. Deswegen sieht es im Inneren des Fahrzeugs etwas chaotisch aus.

# Ausblick auf die Zukunft

## Hardware (Schmölzer)

Um das Auto besser lenken zu lassen, sollte man noch 2 Sensoren vorne anbringen.

Viel wichtiger jedoch ist der fehlende Infrarotsensor am hinteren Teil des Modelautos. Dieser dient zur Hindernisdetektion hinter dem Modellauto, falls es nach hinten fahren soll. Somit kann das Fahrzeug auch noch hinten fahren und Hindernissen ausweichen. Eine weitere vernünftige Erweiterung wäre ein Drehgeber. Ein

Drehgeber kann die Weite der Fahrt, also die Distanz und die Geschwindigkeit regeln und gut lesen. Somit kann man dann am Display anzeigen lassen, wie weit und wie schnell das Modellauto gefahren ist. Ein Drehgeber dient zur Regelung. Da der verwendete Servo nicht mehr vollständig ausschlagen kann, und das Modellauto somit nicht in scharfe Kurven lenken kann, sollte man einen besseren, leistungsstärkeren Servo einbauen.

# Quellennachweis

https://de.wikipedia.org/wiki/Nickel-Metallhydrid-Akkumulator https://www.elektronik-kompendium.de/sites/bau/1101251.htm https://de.wikipedia.org/wiki/Elektromotor

https://www.conrad.de/de/ratgeber/technik-einfach-erklaert/elektromotor.html https://de.wikipedia.org/wiki/Eagle\_(Software) (Arduino Datenblatt) https://www.arduino.cc/reference/en/ https://de.wikipedia.org/wiki/Fl%C3%BCssigkristallanzeige https://heimo.de/jpool/articles/lcd/tstn.shtml https://www.leifiphysik.de/optik/polarisation/ausblick/funktion-von-lcd-displays https://www.exp-tech.de/blog/arduino-tutorial-servo https://de.wikipedia.org/wiki/Servo https://de.wikipedia.org/wiki/Vierquadrantensteller https://www.sharpsde.com/fileadmin/products/Optoelectronics/Sensors/Specs/GP2Y0A60SZ LF\_TI\_OP13005EN.pdf (Infrarotsensor Datenblatt) https://www.exp-tech.de/blog/wiki-entfernungsmessung https://www.all-electronics.de/infrarot-abstandssensoren/ https://de.wikipedia.org/wiki/Abstandsregeltempomat https://de.wikipedia.org/wiki/Autonomes\_Fahren

https://www.digikey.at/de/articles/fundamentals-distance-measurement-gesture-recognitiontof-sensors

https://de.wikipedia.org/wiki/Rangordnungsfilter https://de.wikipedia.org/wiki/Entfernungsmessung

https://de.wikipedia.org/wiki/Elektrooptische\_Entfernungsmessung https://de.wikipedia.org/wiki/Pulsdauermodulation https://de.wikipedia.org/wiki/Spannungsregler https://de.wikipedia.org/wiki/Schutzdiode

# Abbildungsverzeichnis

Abbildung 1 GP2Y0A60SZLF ................................................................................................... 13

Abbildung 2 Schaltung selbstgebauter IR-Sensor .................................................................... 13

Abbildung 3 Träger des Sensors ..............................................................................................14

Abbildung 4 zeitliche Ablauf IR-Sensor .................................................................................... 15

Abbildung 5 Verlauf der Distanzmessung ................................................................................16

Abbildung 7 Flüssigkristallbildschirm ....................................................................................... 17

Abbildung 8 Funktionsweise LCD ............................................................................................. 17

Abbildung 10 Servoansteurerung .............................................................................................19

Abbildung 11 Servo PWM Ansteuern ...................................................................................... 20

Abbildung 12 Schaltung Spannungsregler ...............................................................................21

Abbildung 13 H-Brückenschaltung mit Spannungsquelle ....................................................... 25

Abbildung 14 H-Brücke für DC-Motor ...................................................................................... 26

Abbildung 15 Strom und Spannung mit/ohne Schutzdiode..................................................... 27

Abbildung 16 Vierquadranten Betriebsmodi Erklärung ........................................................... 28

Abbildung 17 IBT-4 Motor Treiber............................................................................................ 29

Abbildung 18 Pin-Belegung IBT-4 ........................................................................................... 30

Abbildung 19 Blockschaltbild .................................................................................................... 31

Abbildung 20 Blockschaltbild 2 ................................................................................................. 31

Abbildung 21 Arduino UNO ...................................................................................................... 33

Abbildung 22 Pull-up und Pull-Down ....................................................................................... 35

Abbildung 23 Arduino IDE ........................................................................................................ 36

Abbildung 24 Serial Monitor ..................................................................................................... 37

Abbildung 25 I2C Frames ........................................................................................................ 40

Abbildung 26 Arduino PWM ......................................................................................................41

Abbildung 27 TT-01 TYPE-E Bausatz ..................................................................................... 56

Abbildung 28 ausgepackter Bausatz ....................................................................................... 56

Abbildung 29 Einbau des Stirnrads ......................................................................................... 57

Abbildung 30 eingebaute Antriebswelle................................................................................... 58

Abbildung 31 Zusammenbau der Motorbefestigung ............................................................... 59

Abbildung 32 eingebauter Motor .............................................................................................. 60

Abbildung 33 Differenzialgetriebe (hinten) .............................................................................. 60

Abbildung 34 Einbau des Differenzialgetriebes (hinten) ..........................................................61

Abbildung 35 Anbringen der hinteren Querlenker ................................................................... 62 Abbildung 36 Hinterachse ........................................................................................................ 63

Abbildung 37 hinterer Stoßfänger ............................................................................................ 63

Abbildung 38 Einbau der Hinterachse ..................................................................................... 64

Abbildung 39 Querlenker ......................................................................................................... 65

Abbildung 40 Einbau der Vorderarme ..................................................................................... 67

Abbildung 41 Aufbau des Dämpfers ........................................................................................ 68

Abbildung 42 Einbau des Dämpfers ........................................................................................ 68

Abbildung 43 eines der zwei Lenkgestänge ............................................................................ 69

Abbildung 44 Urethan-Stoßstange .......................................................................................... 70

Abbildung 45 Einbau des Lenkgestänges ............................................................................... 70

Abbildung 46 Lenkanlage.......................................................................................................... 71

Abbildung 47 Einbau der Lenkanlage ...................................................................................... 72

Abbildung 48 Zusammenbau eines Rads ................................................................................ 73

Abbildung 49 Karosserie nach der Bearbeitung ...................................................................... 74

Abbildung 50 Beschrifteter Plan in Eagle ................................................................................ 75

Abbildung 51 fertige Platine ..................................................................................................... 75

Abbildung 52 Komplettes Auto ................................................................................................. 76