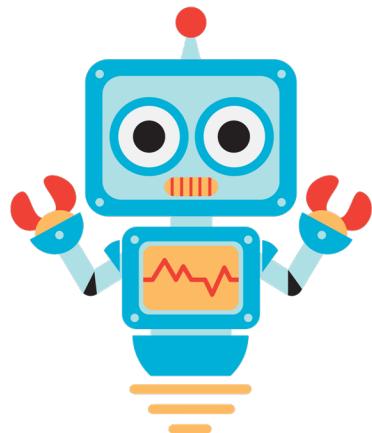


BERUFLICHE OBERSCHULE
ROSENHEIM

Alexander Stephan FT13a

Roboterentwicklung mit dem Arduino



betreut von
Stefan LANGGARTNER

01.10.2018

Inhaltsverzeichnis

1 Motivation	3
1.1 Einleitung	3
1.2 Warum ein Roboter?	3
1.3 Zielsetzung	4
2 Konzeption	5
2.1 Die Ausführungsmöglichkeiten	5
2.1.1 Zweibeinige Roboter	5
2.1.2 Zweirädrige Roboter	5
2.1.3 Roboter mit Raupenantrieb	6
2.2 Kriterien	7
3 Die Konstruktion	9
3.1 Überblick	9
3.2 Die Idee	9
3.3 Die wichtigsten Bauteile	10
3.3.1 Croduino Nova	10
3.3.2 Acrylhassis	11
3.3.3 Motortreiber	12
3.3.4 Powerbank	12
3.4 Zusammenbau	13
3.4.1 Vorbereitung	13
3.4.2 Bau der unteren Grundplatte	14
3.4.3 Einbau der Motoren	14
3.4.4 Montage der oberen Grundplatte	15
3.4.5 Befestigen des Stützrades	15
3.4.6 Installation des Servos	16
3.4.7 Montage der Räder	16
3.4.8 Das Gesicht des Roboters	17
3.4.9 Stromversorgung	18
3.4.10 Verkabelung	18
3.4.11 Letzter Schliff	19
4 Programmierung	20
4.1 Ansteuerung der Bauteile	20
4.1.1 Ansteuern des Motortreibers	20
4.1.2 Ansteuern des Servomotors	22
4.1.3 Auslesen des Ultraschallsensors	23

4.2	Netzwerkeinbindung	24
4.3	Hinderniserkennung und Vermeidung	24
4.4	Das fertige Programm	25
5	Mögliche Erweiterungen	26
5.1	Statusanzeige	26
5.2	Treppen- und Tischkantenerkennung	26
5.3	Helligkeitssensor	26
6	Herausforderungen	27
7	Fazit	27
8	Quellenverzeichnis	28
8.1	Buchquelle	28
8.2	Internetverweise	28
9	Anhang	29

1 Motivation

1.1 Einleitung

Immer noch zu viele Schüler haben das Problem, dass ihnen das Thema Programmierung zu trocken ist. Sie lassen sich abschrecken von abstrakten, kryptisch wirkendem Code. Dabei ist in der Realität zwar tatsächlich oft harte Denkarbeit involviert, aber es ist eben auch kein Hexenwerk. Deshalb ist es wichtig eine Verbindung zu schaffen; zwischen theoretischen Konstrukten und praktischer Anwendung. Konkrete Anwendungen machen es leichter Programme zu verstehen und motivieren durch unmittelbares Feedback. Vor allem aufgrund dieser Tatsache erfreuen sich sogenannte Physical Computing-Plattformen im Hobby- und Bildungsbereich immer größerer Beliebtheit. Sie erlauben, wie im Namen bereits angedeutet, physikalische Objekte, speziell elektronische Bauteile, auf unkomplizierte und intuitive Weise zu steuern. Ein populäres Beispiel für eine solche Plattform ist der Arduino¹, welcher ebenfalls die Basis für den hier vorgestellten Roboter.

1.2 Warum ein Roboter?

Als mir vorgeschlagen wurde einen Roboter für die Verwendung im Unterricht zu bauen, war ich sofort begeistert, da Roboter genau diese Idee verkörpern und Verbindung zwischen Lebewesen und Computern herstellen. In meinen Augen ist ein Roboter die ideale Möglichkeit, um Schüler für Technologie zu begeistern, da Roboter grundsätzlich Neugier wecken und Informationsverarbeitung so anschaulich wie möglich machen.

¹<https://arduino.cc>

Dabei tritt hier ein allgegenwärtiges Prinzip der Informationsverarbeitung auf, welches im konkreten Fall des Arduinos folgendermaßen aussieht:

- 1 Der Arduino bezieht Informationen aus der Außenwelt. Dazu werden Messgeräte genutzt, die in eine Schaltung eingebunden werden und entsprechend mit den Dateneingängen am Arduino verbunden.
- 2 Die eingehenden Daten werden mithilfe von Programmen und Algorithmen verarbeitet.
- 3 Ausgehend von der Auswertung der Daten werden Befehle an die entsprechenden Bauteile geschickt.

Allgemein lässt sich festhalten, dass dieser Ablauf dem bekannten EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe) gleicht, welches die Grundidee der Informationsverarbeitung verkörpert und auch unser Verständnis von Lebewesen maßgeblich prägt. Lebewesen erhalten ebenfalls Reize aus der Umwelt, verarbeiten diese und treffen davon ausgehend Entscheidungen, um ihr Überleben zu sichern.

1.3 Zielsetzung

Das Ziel dieser Seminararbeit ist letztendlich eine Möglichkeit für Schüler zu schaffen Funktionsweisen von Robotern zu verstehen und mit diesem gewonnenen Verständnis Logiken in Programmanweisungen umzusetzen. Dabei sollen die Schüler nicht nur generell im Umgang mit Elektronik geschult werden, sondern auch am Beispiel des Arduinos einen Einstieg in die Welt der Mikrocontroller-Programmierung finden. Darüber hinaus wird die Konstruktion des Roboters detailliert beschrieben und eine Schritt-für-Schrittanleitung bereitgestellt. Diese soll in erster Linie ein Ansatzpunkt für den Lehrer sein, um kostengünstig im Unterricht mit Robotern zu arbeiten, aber auch als Hilfe für interessierte Schüler dienen, die den Roboter erweitern oder die grundsätzlich die Konstruktion nach ihren Vorstellungen modifizieren wollen.

2 Konzeption

2.1 Die Ausführungsmöglichkeiten

Es gibt unzählige Roboterbauweisen, die sich jedoch stark in Aufwand und Komplexität unterscheiden. Dabei sollte man auch vor allem den Verwendungszweck berücksichtigen. Um nicht den Überblick zu verlieren werden im Rahmen dieser Arbeit die gängigsten Robotertypen kurz vorgestellt und deren Vor- und Nachteile diskutiert. Anschließend wird eine begründete Entscheidung für einen der Typen dargelegt.

2.1.1 Zweibeinige Roboter

Zweibeinige Roboter sind an den Körperbau des Menschen angelehnt. Verfügen also über einen Kopf und Gliedmaßen. Aus diesem Grund wird diese Roboterart auch oft als humanoid bezeichnet. Als wesentlicher Vorteil ist hier die Vielzahl der Bewegungsrichtungen zu nennen. Außerdem gibt es solchen Robotern gegenüber einer höhere Akzeptanz in der Öffentlichkeit, da sie als niedlich empfunden werden. Nachteilhaft ist, dass diese Roboter wesentlich teurer in der Anschaffung sind und viel besser abgestimmt sein müssen, um ordentlich zu funktionieren. Der hohe Preis ergibt sich vor allem aus der Vielzahl an Motoren, die die Gliedmaßen und den Kopf steuern. Außerdem gestaltet sich aufgrund der in der Regel eher abgeschlossenen Konstruktion eine Erweiterung oftmals als schwierig. Eine günstige Alternative, um einen unkomplizierteren Einstieg bezüglich zweibeiniger Roboter zu erhalten bietet der Open-Source Roboter Otto², auf den ich im Laufe der Recherche gestoßen bin. Dieser kann sich auch gut für den Unterricht, da bereits augenscheinlich gut abgestimmte Libraries zur Verfügung gestellt werden und außerdem das komplette Chassis mithilfe von 3D-Druck hergestellt werden kann. Ebenfalls positiv anzumerken ist die verhältnismäßig gute Dokumentation.

2.1.2 Zweirädrige Roboter

Komplementär zum Konzept des zweibeinigen Roboters, gibt es auch das des Zweirädrigen. Diese haben erstmal nicht mehr viel mit der Anatomie des Menschen zu tun, sondern gleichen im Grunde genommen einem Fahrzeug. Dabei werden zwei Räder mit jeweils einem Motor versehen. Damit die Konstruktion an Stabilität gewinnt, beziehungsweise der Roboter leichter lenken

²<https://www.ottodiy.com/>

kann wird in der Regel an der Vorderseite des Roboters noch ein Kugellager oder ein Stützrad angebracht. Der Vorteil dieser Roboter ist nicht nur, dass sie relativ einfach zu bauen, sondern auch noch sehr kostengünstig sind. Fertige Chassis sind bereits für sehr wenig Geld zu haben und lassen sich leicht mit eigenen Bauteilen bestücken. Im Gegensatz zum humanoiden Roboter ist auch die Anzahl der Motoren sehr überschaubar und das Bewegungsmuster des Fahrzeugs sehr einfach, denn die Bewegung des Roboters resultiert ausschließlich aus der Drehrichtung der beiden Reifen. Damit geht auch eine einfachere Wartung sowie auch eine unkompliziertere Handhabung einher. Aus dieser Einfachheit ergeben sich allerdings auch einige Nachteile. Diese Art von Robotern ist nur für sehr begrenzte Einsatzgebiete geeignet. An Dinge wie Greifen ist hier oder Treppensteigen ist hier erst mal nicht zu denken. Auch gestaltet sich die Verwendung außerhalb der Wohnung oder eines ebenen Testgeländes als schwierig, da so gut wie keine Traktion vorhanden ist und selbst die kleinsten Hindernisse zur unüberwindbaren Mauer werden. Anzumerken wären an dieser Stelle auch noch die vierrädrigen Roboter, die aber in Ihrer Funktionsweise den zweirädrigen Robotern gleichen und dabei vor allem mit höheren Kosten verbunden sind, da mehr Motoren benötigt werden.

2.1.3 Roboter mit Raupenantrieb



Abbildung 1: Chassis mit Raupenantrieb

Wenn es in erster Linie um Geländetauglichkeit geht, sind Raupen eine gute Alternative zum Roboter auf Rädern. Dabei bewegen sich, wie man es beispielsweise von einem Panzer kennt, zwar auch Räder, nur dass diese wiederum eine Kette bewegen. Sie haben nicht nur den Vorteil, dass sie mit Unebenheiten besser klar kommen, da die große Auflagefläche mehr Traktion bietet, sondern können auch sehr hohe Geschwindigkeiten erreichen.

Allerdings besteht im gerade Hobbybereich der gravierende Nachteil, dass man für zuverlässige Ausführungen nicht nur relativ hohe Preise bezahlt sondern auch an bestimmte Hersteller gebunden ist.

2.2 Kriterien

Die wesentlichen Kriterien für einen Roboter für den Bereich Bildung nach ihrer Wichtigkeit:

- 1 Programmierbarkeit
- 2 Kosten
- 3 Nachhaltigkeit
- 4 Modularität

In erster Linie diente für mich die Programmierbarkeit der Roboter als wesentliches Kriterium, wobei natürlich auch immer ein gewisser preislicher Rahmen im Auge gehalten werden muss. Zuerst möchte ich darlegen, warum ich mich gegen einen zweibeinigen Roboter entschieden habe, welcher vielleicht zuerst am ansprechendsten wirkt. Vorneweg gilt hier die Einschränkung, dass hier nur auf den Roboter Otto eingegangen werden soll, denn die meisten anderen Ausführungen auf die ich im Laufe der Recherche sprengten den preislichen Rahmen. Für die komplexen Bewegungsabläufe der humanoiden Robotern, werden zahlreiche Bibliotheken benötigt. Die Implementierung wird schnell unübersichtlich, da die Befehle zahlreich und die dahinterstehenden Hardwareanweisungen oft unklar sind. Dazu kommt, dass sich der humanoide Roboter schlechter erweitern lässt, da schlachtweg kein Platz mehr im einmal gedruckten Chassis verfügbar ist.

Auch lassen sich beim Fahrzeug-Typ bessere Meilensteine definieren. Zuerst kann man beispielsweise den Roboter mithilfe eines Infrarot-Sensors eine markierte Strecke abfahren lassen. Später kann man dann Algorithmen implementieren, die eine selbstständige Fortbewegung ermöglichen, die unter Verwendung eines Motortreibers verhältnismäßig einfach zu programmieren sind und dennoch bei entsprechendem Interesse auch eine sehr hohe Komplexität an den Tag bringen können. Dies ist zwar beim humanoiden Roboter möglich, gestaltet sich aber aufgrund der Höhe zum Boden und der langsam, schwer zu steuernden Bewegungsroutinen wenig intuitiv. Letztendlich muss es eine Balance geben bei der der Roboter genug Funktionen hat, um Interesse zu wecken und gleichzeitig noch jeden Schüler die Möglichkeit bietet einen leichten Einstieg in die Welt der Robotik zu finden. Zuerst habe ich

zum humanoiden Roboter tendiert, da er auch optisch dem besser entspricht, was sich ein Schüler unter einem Roboter vorstellt. Außerdem ist er in der Lage zu tanzen, was Schülern sicher gut gefällt. Ein weiterer Vorteil ist, dass das Gehäuse im Fall Otto komplett mit 3D-Drucker erstellt werden kann, da der Roboter nur aus kubischen Objekten besteht. Allerdings sollte man hier auch anmerken, dass der Druckaufwand nicht zu unterschätzen ist. Die Druckdauer wird bei einem durchschnittlichen Drucker auf etwa 10 Stunden geschätzt, was definitiv eine große Hürde darstellt, wenn der Roboter beispielsweise in Bildungseinrichtungen gedruckt werden soll, da diese selten so lang belegt werden können. Außerdem ist er viel zu langsam und Erweiterungen aufgrund des sehr kleinen Gehäuses fast ausgeschlossen. Deshalb blieb für mich nur noch das Modell des zweirädrigen Roboters. Denn letztendlich erfüllt er alle oben aufgeführten Punkte. Er ist einfach zu steuern, da er nur aus zwei Motoren besteht, kann nach oben hin beliebig erweitert werden und bedient sich außerdem nur von gut verfügbaren, preisgünstigen Bauteilen. Bei der konkreten Planung des Roboters war es mir wichtig eine fundierte, gut dokumentierte Ausgangsbasis zu haben. Deshalb habe ich mir das Buch 'Roboter bauen mit Arduino' von Markus Knapp, einem langjährigen Robotik-Enthusiasten beschafft. In dem Buch wird detailliert ein Roboter beschrieben, der die genannten Punkte größtenteils und bietet dazu allerlei nützliches Zusatzwissen.

3 Die Konstruktion

3.1 Überblick

An dieser Stelle soll zuerst ein Überblick über die verwendeten Bauteile gegeben werden.

- 1 x Croduino Nova
- 1 x Acrylchassis
- 2 x GM3 224:1 90 Grad Getriebemotor
- 2 x 2-5/8SServo Kunststoffrad schwarz
- 1 x Ultraschall Entfernungsmesser HC-SR04
- 1 x Servo Hitec HS-311
- 1 x Motortreiber LN298N
- 1 x kleines Breadboard

3.2 Die Idee

Der Aufbau des Roboters lässt sich folgendermaßen umreißen: Zwischen zwei durch eine Steckverbindung zusammengehaltenen Acrylplatten befinden sich zwei Getriebemotoren, an denen sich jeweils ein Rad befindet. An der Unterseite des Chassis befindet sich ein Batteriefach, dass für die Stromversorgung dient. Auf der Oberseite wird nun die Schaltung in Form eines Breadboards angebracht, auf dem sich auch der Croduino befindet. An der Vorderseite des Roboters befindet sich dann wiederum ein Servomotor der den an einer weiteren Acrylplatte befestigten Ultraschallsensor bewegen kann. Durch diese Beweglichkeit ergibt sich für den Roboter nicht nur ein größeres Sichtfeld sondern auch eine präzisere Erfassung der umliegenden Objekte, da deren Plastizität besser erkannt werden kann.

3.3 Die wichtigsten Bauteile

3.3.1 Croduino Nova

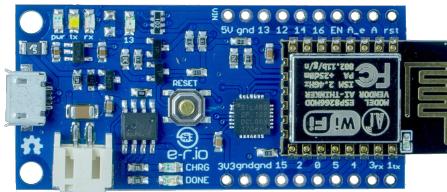
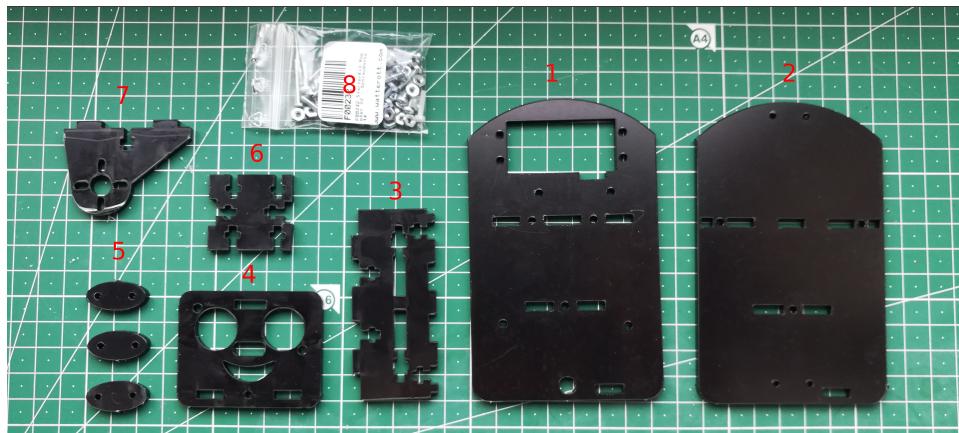


Abbildung 2: Croduino Nova

Mikrocontroller für den Heim- und Bildungsbereich wie den Arduino gibt es seit einiger Zeit. Allerdings sind Ausführungen wie der Arduino Uno nicht mehr wirklich zeitgemäß, besonders für Anwendung wenn man nicht nur mit einem Breadboard experimentieren will, sondern damit tatsächlich ein konkretes Projekt umsetzen will. Einerseits sind Uno's und ihre zahlreichen Klone relativ groß und verschwenden damit wertvollen Platz, andererseits fehlt ihnen werksseitiges WiFi ein wesentliches Feature, dass im Zeitalter des "Internet of Things" unverzichtbar ist. Zwar kann man dieses mit einem teuren WiFi-Shield nachrüsten oder alternativ einen ESP8266 dazukaufen. Genau dieser wird aber auch beim Arduino Nova verwendet, da dieser mittlerweile genug Leistung besitzt, um auch als eigenständiger Microcontroller verwendet werden zu können. Dabei werden trotzdem alle Ein- und Ausgänge bereitgestellt, die man auch von Klassikern wie dem Uno gewöhnt ist. Allerdings ist beachten, dass der Nova nicht sofort steckbrettkompatibel ist und entsprechende Pin-Header erst angelötet werden muss.

3.3.2 Acrylchassis

Beim Chassis gibt es viele Varianten. Es gibt mittlerweile viele kommerziell verfügbare Chassis, allerdings ist die Entscheidung in diesem Fall auf eine Open-Source-Hardware Lösung gefallen, die auch in Knapp's Buch als Basis für den Roboter dient. Dabei gibt es einerseits die Möglichkeit das Chassis komplett fertig mit Chassis beim Elektronikversand Watterott³ zu bestellen. Bei Watterott ist natürlich auch ein Schraubenset, dabei welches auch zum Bau dieses Roboters verwendet wird. Alternativ kann sich das Chassis natürlich auch selbst lasern. Die benötigten Layout-Dateien sind auf der GitHub-Seite des Autors⁴ zu finden und dürfen selbstverständlich nach Belieben modifiziert werden. Somit kann der Roboter auch in Zukunft einfach erweitert werden, ohne dass man an ein bestimmte Konzeption dauerhaft gebunden ist.



- | | |
|----------------------|-----------------------|
| 1 Obere Grundplatte | 5 Stützradmontierung |
| 2 Untere Grundplatte | 6 Kleiner Motorhalter |
| 3 Großer Motorhalter | 7 Servomontierung |
| 4 Sensorhalterung | 8 Schraubenset |

³<https://watterott.de>

⁴<https://github.com/markusk>

3.3.3 Motortreiber

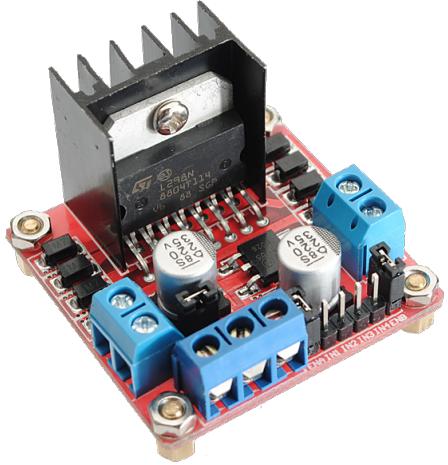


Abbildung 3: LN298N H-Brücken IC

Beim Motortreiber gibt es diverse Lösungen. Man kann sich entweder eine eigene Schaltung bauen oder einen bereits fertigen kaufen. Ich habe mich für den LN298N entschieden. Dieser H-Brücken IC bietet ein komfortables Gesamtpacket. Man hat jeweils zwei Ausgänge für zwei Wechselstrom-Motoren. Außerdem gibt es jeweils immer zwei korrespondierende Steuerpins und zusätzlich jeweils noch einen Enable-Pin der für die Geschwindigkeitsregelung zuständig ist. Als Eingangsspannung genügen dem Motortreiber 5V, wobei aber auch wesentliche höhere Eingangsspannungen kein Problem darstellen. Außerdem sollte man darauf achten, dass der Masse-Pin des Motortreibers mit dem des Arduino verbunden ist, um etwaigen Potentialunterschieden vorzubeugen und damit einhergehende Beschädigungen der Elektronik zu verhindern.

3.3.4 Powerbank

Zuerst habe ich zu einem Batteriefach mit Akkus tendiert. Allerdings liefern diese immer entweder zu viel Spannung oder zu wenig. Aus diesem Grund habe ich mich nach Alternativen umgeschaut und erhielt die Empfehlung eine Powerbank zu verwenden. Ich hatte zwar auch schon an eine Powerbank gedacht, war aber der Meinung, dass diese viel zu groß seien. Allerdings fand ich nach kurzer Recherche heraus, dass es mittlerweile extrem dünne Powerbanks gibt, die nicht größer als eine Kreditkarte sind. Zwar

sind diese verhältnismäßig schwach, aber dennoch mit 2500mHa mehr als ausreichend. Die Powerbank bietet den Vorteil, dass man den Arduino direkt mit einem USB-Kabel betreiben kann. Idealerweise verwendet man ein gewinkeltes Micro-USB Kabel, da man nur so ohne Biegung des Kabels an den USB-Anschluss des Croduinos gelangt.

3.4 Zusammenbau

3.4.1 Vorbereitung

Zuerst sollte man sich einen Arbeitsplatz einrichten und alle aufgeführten Werkzeuge bereitlegen.

- Spitzzange
- 2,5 mm Schlitzschraubenzieher
- Abisolierzange
- Pinzette
- Doppelseitiges Klebeband
- Lötkolben mit Lötzinn
- Doppelseitige Klebepads
- 3 mm Bohrer
- Schere

Außerdem müssen die einzelnen Acrylteile aus der Acrylplatte gedrückt werden. Dabei sollte vorsichtig vorgegangen werden, da Acryl ein spröder Werkstoff ist. Bei zu groben Vorgehen läuft man Gefahr, dass Bauteile abbrechen. Anschließend muss auf beiden Seiten die Folie abgezogen, die das Acrylglas vor Kratzern schützt. Wenn man Probleme hat die Folie mit den Fingern abzulösen, kann man auch die Spitze einer Schere zu Hilfe nehmen.

3.4.2 Bau der unteren Grundplatte

Zuerst steckt man die große und die kleine Motorhalterung und die große Motorhalterung in die Acrylplatte. Dann werden die M2,5-er Muttern in die kreuzförmigen Aussparung geschoben. Entweder bringt man hier viel Fingerspitzengefühl mit oder legt alternativ die Konstruktion schief und hält den Finger unter die Aussparung. So kann von oben bequem die Mutter hineingelegt werden. Alternativ kann man natürlich auch eine Pinzette verwenden. Dann müssen die 3x10mm Schrauben von unten in die Muttern geschraubt werden. Dabei sollten die Schrauben nicht zu fest angezogen werden, um sicher zu gehen, dass die Zugbelastung auf das spröde Acryl nicht zu hoch ist.

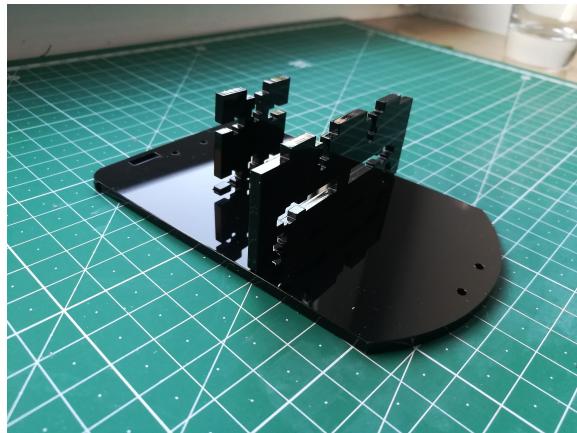


Abbildung 4: Untere Grundplatte

3.4.3 Einbau der Motoren

Als Erstes muss an die Kontakte der Motoren jeweils ein 5 cm langer Klin-geldraht gelötet werden. Dabei sollte zuerst die Isolierung an beiden Enden des Drahtes mit Hilfe einer Abisolierzange oder eines scharfen Messers entfernt werden. Dann verdreht man die frei-liegenden Kupferdrähte mit den Fingern, damit die Lötverbindung stabiler ist und keine einzelnen Drähte abstehen. Beim Löten selbst ist es sinnvoll den Draht nicht durch die Öse zu führen, sondern ihn flach, zur Motorarche parallel auf die Öse zu legen. Um die Motoren am Chassis zu befestigen müssen sie jeweils hochkant, mit der Achse nach außen zeigend, platziert und nach dem oben beschrieben Verfahren mit Muttern und Schrauben am kleinen Motorhalter befestigt werden.

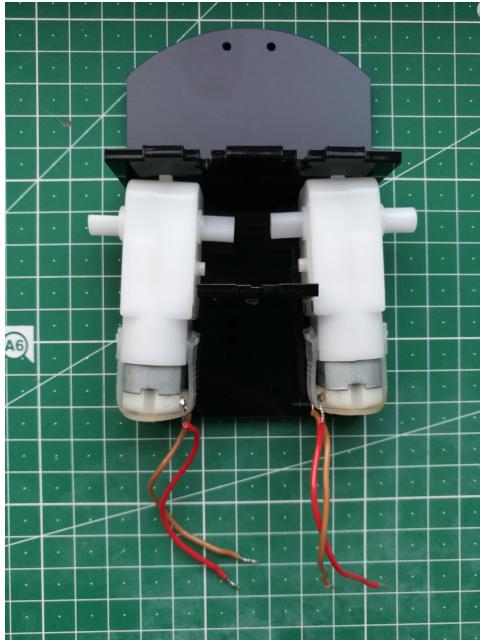


Abbildung 5: Position der Motoren

Beim großen Motorhalter gestaltet sich die Montage etwas schwieriger. Hier muss mit etwas Druck die 3x10mm Schraube in die Aussparung geschoben werden. Dabei sollte man einen etwas größeren Schraubenzieher wählen und dann leicht schräg an der Schraube anzusetzen.

3.4.4 Montage der oberen Grundplatte

Jetzt muss die obere Grundplatte in den kleinen und großen Motorhalter gesteckt werden. Dabei gibt es einmal zwei Schraubverbindungen zum großen Motorhalter und eine zum kleinen Motorhalter. Falls beim Festschrauben der Oberplatte an den Motorhaltern Probleme auftreten, kann auf eine Seite der Aussparung etwas Tesafilm geklebt werden, damit die Mutter an Ort und Stelle bleibt. Anschließend kann man bereits den Motortreiber an der Rückseite der oberen Grundplatte mit einem Klebepad befestigen.

3.4.5 Befestigen des Stützrades

Vorne am Gehäuse, direkt vor dem Servomotor, wird das Stützrad montiert. Dazu benötigt man zwei M2x18 mm Schrauben inklusive passende Muttern.

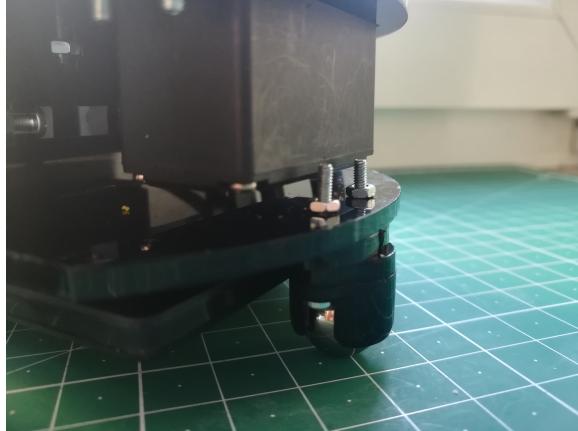


Abbildung 6: Vorderansicht des Stützrads

Da bei manchen Ausführungen der Acrylteile die Bohrungen an der Halterung für das Stützrad leicht verschoben sind, muss man die vorgesehenen Bohrungen am Gehäuse auf etwa 3 mm Durchmesser bringen. Nur so lassen sich in diesem Fall die Schrauben durch alle Bauteile führen. Wenn man merkt, dass das Stützrad etwas unsauber läuft, kann mit etwas Öl nachgeholfen werden.

3.4.6 Installation des Servos

Der Servo wird einfach in die vorgesehene Aussparung gelegt und mit einer oder mehreren Schrauben befestigt. Dann schraubt man auf den Servo das Servorad beziehungsweise das Servokreuz, wobei letzteres wohl die bessere Wahl ist, da die Bohrungen auf dem passenden Acrylteil auch kreuzförmig angeordnet sind. Falls das Servokreuz zu groß ist, sollte man gegebenenfalls die überstehenden Teile abschneiden, damit beim Bewegen des Servos nicht die Kabel im Weg sind.

3.4.7 Montage der Räder

Die beiden Räder werden jeweils auf die Achsen der Motoren geschoben. Um einem Durchdrehen der Räder vorzubeugen, können jeweils zwischen Rad und Achse zwei Klebepads geschoben werden. Es bestünde auch die Möglichkeit die Verbindung mit Heißkleber zu festigen, allerdings wäre dann wohl eine recht endgültige Entscheidung.

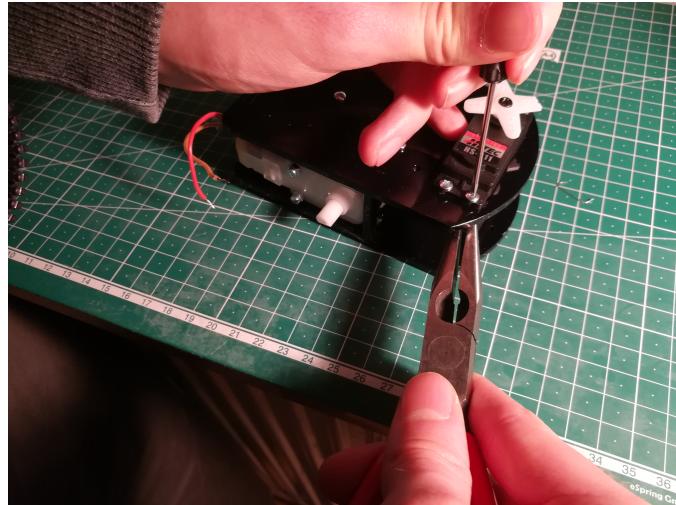


Abbildung 7: Mögliche Technik zum Festziehen der Schraube

3.4.8 Das Gesicht des Roboters



Abbildung 8: Verbindung von Servomontierung und Sensorhalter

Für den Ultraschallsensor benötigen wir das Acrylteil mit den runden Ausparungen. Außerdem benötigt man die Montierung für das Servokreuz. Das Gesicht wird wie gewohnt mit einer Schraube und einer Mutter an der Ser-

vomontierung befestigt, welches dann wiederum mit zwei Schrauben in das Kreuz des Servos geschraubt wird. Anschließend muss man den Ultraschall-sensor nur noch durch die Augen schieben. Fixieren kann man diesen einfach mit etwas Isolierband. Auf dem Steuerkreuz kann man beispielsweise den Croduino mit einem kleinen Breadboard platzieren. Das bietet einerseits den Vorteil den Schwerpunkt weiter nach Vorne zu verlagern und so etwaiiges kippen zu verhindern. Andererseits kann man durch die etwas erhöhte Position den USB-Anschluss des Croduinos besser erreichen, der direkt auf der oberen Grundplatte des Roboters durch die Räder blockiert wäre.

3.4.9 Stromversorgung

Die Powerbank kann einfach unter mit ein paar Klebepads oder alternativ einem Kabelbinder befestigt werden. Anschließend muss man nur noch ein Micro-USB Kabel mit dem Croduino verbunden werden. Ansonsten reicht es auch für viele Anwendungen aus den Roboter einfach mit einem langen, flexiblen USB-Kabel zu betreiben, wenn der Computer ein entsprechend starke Stromstärke liefert.

3.4.10 Verkabelung

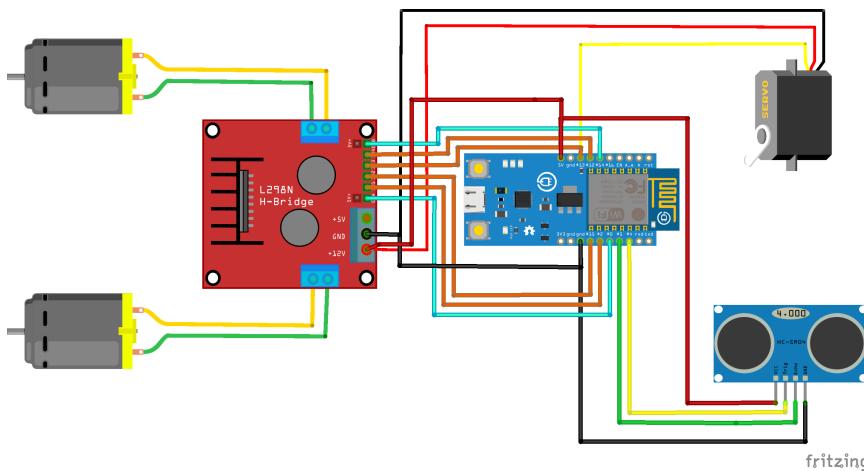


Abbildung 9: Verkabelung der Bauteile

Bei der Verkabelung kann wie gewohnt vorgegangen werden. Zu beachten ist, dass sich der Motortreiber und der Croduino das selbe Potential haben, also beide an einer gemeinsamen Masse anliegen.

Darüber hinaus fehlt für die Stromversorgung ebenfalls ein Pin. Das Problem habe ich gelöst, in dem ich eine Verzweigung aus drei Klingeldrähten gelötet habe, die jeweils mit dem GND sowie mit dem VCC Pin des Arduino verbunden sind. Bei den Servopins habe ich ebenfalls ein Stück Klingeldraht angelötet, da die Kabellänge ansonsten knapp geworden wäre.

3.4.11 Letzter Schliff

An dieser Stelle wäre der Roboter soweit betriebsbereit. Jetzt bleiben nur noch einige wenige kosmetische Korrekturen, wie zum Beispiel das Optimieren der Kabelführung. Natürlich kann auch die Position der Bauteile noch angepasst werden.

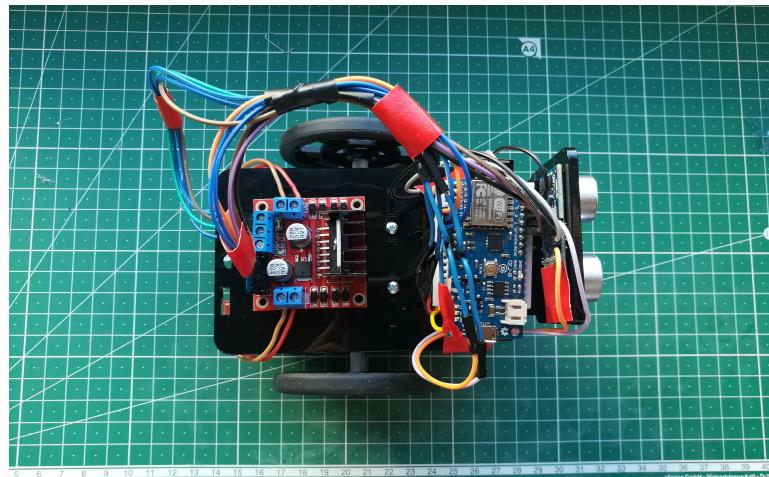


Abbildung 10: Mögliches Aussehen eines fertigen Roboters

4 Programmierung

4.1 Ansteuerung der Bauteile

4.1.1 Ansteuern des Motortreibers

Zuerst müssen die Pins für die beiden Motoren definiert werden. Dabei gibt es für jeden Motor jeweils zwei Pins für die Stromversorgung (IN1-4) deren Polarität die Fahrtrichtung bestimmt und einen Pin für die Geschwindigkeitsregelung (ENA + ENB).

```
#define MOTOR_A_ENABLE1 16
#define MOTOR_A_ENABLE2 14
#define MOTOR_A_SPEED 15

#define MOTOR_B_ENABLE1 2
#define MOTOR_B_ENABLE2 0
#define MOTOR_B_SPEED 12
```

Dann müssen alle Pins als Output gesetzt werden. Außerdem ist es an dieser Stelle sinnvoll die Motoren mit einer Geschwindigkeit zu initialisieren. Dabei setzt man ein PWM-Signal dessen Wert durch einen Integer zwischen 0 und 1023 angegeben wird. Dabei entspricht theoretisch 512 etwa 50 Prozent der Leistung, stellt aber tatsächlich circa die Schwelle da, ab der sich die Motoren anfangen zu bewegen.

```
void setup(){
    // Setze alle Motor-Pins als Ausgang
    pinMode(MOTOR_A_ENABLE1, OUTPUT);
    pinMode(MOTOR_A_ENABLE2, OUTPUT);
    pinMode(MOTOR_A_SPEED, OUTPUT);
    pinMode(MOTOR_B_ENABLE1, OUTPUT);
    pinMode(MOTOR_B_ENABLE2, OUTPUT);
    pinMode(MOTOR_B_SPEED, OUTPUT);

    // Setze die die Ausgangsgeschwindigkeit
    analogWrite(MOTOR_A_SPEED, 512);
    analogWrite(MOTOR_B_SPEED, 512);
}
```

Für die Grundlegenden Bewegungen des Roboters wurde jeweils eine Funktion angelegt. Dabei hängt die binäre Logik von HIGH und LOW davon ab, wie der Treiber verkabelt ist. Unter Umständen muss also die Logik für die Vorwärtsbewegung invertiert werden. Nach dem selben Ablauf lässt sich auch die Funktion für die Rückwärtsbewegung definieren.

```
void driveForward(){
    digitalWrite(MOTOR_A_ENABLE1, LOW);
    digitalWrite(MOTOR_A_ENABLE2, HIGH);
    digitalWrite(MOTOR_B_ENABLE1, LOW);
    digitalWrite(MOTOR_B_ENABLE2, HIGH);
}
```

Zum Lenken müssen sich die beiden Motoren in Unterschiedliche Richtungen drehen. Das wird durch eine Umkehrung des HIGH oder LOW Status von jeweils einem der Motoren realisiert.

```
void changeDirA(){
    digitalWrite(MOTOR_A_ENABLE1, !digitalRead(MOTOR_A_ENABLE1));
    digitalWrite(MOTOR_A_ENABLE2, !digitalRead(MOTOR_A_ENABLE2));
}
void changeDirB(){
    digitalWrite(MOTOR_B_ENABLE1, !digitalRead(MOTOR_B_ENABLE1));
    digitalWrite(MOTOR_B_ENABLE2, !digitalRead(MOTOR_B_ENABLE2));
}
```

Um den Roboter komplett zu stoppen müssen alle Enable-Pins der Motoren auf HIGH gesetzt werden. An dieser Stelle kann man einen Boolean einführen, dessen true oder false Wert Motor A oder B entspricht. Mit der Funktion stopAll() weisen wir somit beiden Motoren an sich nicht mehr zu bewegen.

```

void stopWheel(bool left){
    if(left){
        digitalWrite(MOTOR_A_ENABLE1, HIGH);
        digitalWrite(MOTOR_A_ENABLE2, HIGH);
    }
    else{
        digitalWrite(MOTOR_B_ENABLE1, HIGH);
        digitalWrite(MOTOR_B_ENABLE2, HIGH);
    }
}

void stopAll(){
    stopWheel(true);
    stopWheel(false);
}

```

4.1.2 Ansteuern des Servomotors

Zuerst muss definiert werden an welchem Pin das Steuersignal des Servos anliegt und ein entsprechendes Objekt erstellt werden. Außerdem benötigt man die Library servo.h, die das Objekt initialisiert.

```

#include <Servo.h>
#define SERVO_PWM 13 // Weise der Variable SERVO_PWM GPIO 13 zu
Servo servo1; // Erstelle Objekt namens servo1

void setup(){
    ...
    servo1.attach(13);
}

```

Prinzipiell ermöglicht der Servomotor eine Rotation um 180 Grad, allerdings würde das in unserem Fall eine zu hohe Belastung für die Verkabelung bedeuten. Zusätzlich sind für die meisten Anwendungen die Objekte neben dem Roboter nicht relevant, sondern hauptsächlich die in Hindernisse in Fahrtrichtung. Die Position wird dem Servo über einen Integerwert zwischen 0 und 180 mitgeteilt. Zuerst erstellt man eine Integer-Variable und weist ihr den Wert 90 zu, damit haben wir eine mittige Ausgangslage definiert. Anschließend iterieren wir mit zwei for-Schleifen zwischen zwei Schwellenwerten hin und her, was zu einer gleichmäßigen Rechts- und Linksbewegung des Servomotors führt. Nach jeder Addition oder Subtraktion des Positionswinkels und dem Servo Objekt übergeben. Unter Berücksichtigung der

Kabellängen hat sich eine Gesamtrotation um 60 Grad als günstig erwiesen. Damit sich der Servo nicht zu schnell bewegt und damit Hindernisse durch den Ultraschallsensor zuverlässiger erkannt werden, wurde die Schrittweite auf 0.1 reduziert. Wenn man eine durchgehende Servobewegung haben möchte, kann man die Methode turnServo() einfach im Loop des Arduino-Programms aufrufen.

```
void turnServo(){
    int pos=90; // Mittlere Position

    for(pos=60; pos<=120; pos += 0.1) {
        servo1.write(pos);
    }
    for(pos=120; pos>=60; pos -= 0.1) {
        servo1.write(pos);
    }
}
```

4.1.3 Auslesen des Ultraschallsensors

Ähnlich wie beim Servo benötigt man beim Ultraschallsensor auch eine Library, die rohen Messwerte der Zeitabstände zwischen Ein- und Ausgang der Ultraschallwellen in Längeneinheiten umrechnet⁵. Anschließend definiert man die beiden Pins ECHO und TRIG passend zur jeweiligen Beschriftung am Sensor. Anzumerken wäre, dass man in manchen Fällen die Pins vertauschen muss, um gültige Messwerte zu erhalten. Die Werte, die aus den Pins ausgelesen werden im nächsten Schritt dem Objekt sr04 übergeben. Dieses beinhaltet damit direkt auch den berechneten Abstandswert den man mit sr04.Distance() standardmäßig in Centimeter auslesen kann.

```
#include <SR04.h>

#define ECHO_PIN 4
#define TRIG_PIN 5

SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
```

⁵<https://www.waycon.de/produkte/ultraschallsensoren/messprinzip-ultraschallsensoren/>

4.2 Netzwerkeinbindung

Hierfür werden zwei WiFi-Libraries des ESP8266 benötigt. Einmal ESP8266WiFi, welche für die Bereitstellung eines lokalen Access Points dient und andererseits die ESP8266Server Library, welche den Server bereitstellt.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
```

Anschließend kann man den Server auf Port 80 starten

```
ESP8266WebServer server(80);
```

Jetzt kann den Access Point starten und die zugehörige SSID sowie das Passwort festlegen. Außerdem ist es sinnvoll die IP des Servers in der Konsole zu loggen, um den Server anschließend im Netzwerk zu finden.

```
void setup(){
    WiFi.mode(WIFI_AP); // Accespoint Mode setzen
    WiFi.softAP("edubot", "passwort123");
    Serial.println(WiFi.softAPIP());
}
```

Die Steuerung funktioniert in dem man an den Server eine HTML-Seite schickt und dann mit entsprechenden Links arbeitet, die im Browser beispielsweise mit einem Knopfdruck ausgelöst werden. Diese können dann mit dem server.on Befehl ausgelesen werden. Nachfolgend ein Beispiel, wie man einen Startknopf realisiert.

```
server.send(200, "text/html",
"<a href=\"/start\"><button>Start</button></a>");  
server.on("/start", driveForward)
```

4.3 Hinderniserkennung und Vermeidung

Im Grunde genommen funktioniert die Hinderniserkennung recht simpel. Wenn der Distanzwert des Ultraschallsensors unter 10 cm fällt, soll der Roboter rückwärts fahren, kurz anhalten und anschließend seine Richtung ändern. Ab diesem Zeitpunkt sollte das Hindernis nicht mehr in Fahrtrichtung sein und der Roboter kann wieder gerade ausfahren. Falls doch ein Hindernis im Weg ist, wird einfach die entsprechende if-Abfrage erneut ausgeführt. Am Ende muss erneut die changeDirA() Methode aufgerufen und damit die Logik an den Pins ebenfalls umgedreht werden.

```

void collisionHandling(){
    if(sr04.Distance() < 10){
        driveBackward();
        delay(2000);
        stopAll();
        delay(500);
        driveForward();
        changeDirA();
        delay(2000);
        changeDirA();
    }
    else{
        driveForward();
    }
}

```

4.4 Das fertige Programm

Prinzipiell kann mithilfe der vorgestellten Listings bereits ein funktionierender Roboter gebaut werden. Um den Roboter wirklich bequem steuern zu können habe ich die Netzwerksteuerung weiter ausgebaut und noch einen Geschwindigkeitsregler hinzugefügt. Unter anderem habe ich auch einen Handler für Fehler beim Laden der Seite hinzugefügt. Der vollständige Code ist auf meiner persönlichen GitHub-Seite zu finden.⁶

⁶<https://github.com/alexanderstephan/edubot>

5 Mögliche Erweiterungen

5.1 Statusanzeige

Hierfür bietet sich die Verwendung eines LCD-Display an. Auf diesem könnte man dann beispielsweise die Distanz zum aktuell erkannten Objekt ausgeben lassen. Außerdem könnte man diverse Debugging-Informationen anzeigen lassen. Möglich wären eine kurze Nachricht ob der Server gestartet ist oder ob der Roboter an sich betriebsbereit ist. Falls das Display groß genug ist, könnte man auch die IP-Adresse des Servers auf dem Display anzeigen lassen.

5.2 Treppen- und Tischkantenerkennung

Dafür wird ein weiterer Sensor in Bodennähe benötigt. Hierfür kann entweder erneut ein Ultraschallsensor oder auch ein günstigerer Infrarotsensor verwendet werden. Befestigt werden sollte dieser über dem Stützrad beziehungsweise direkt neben dem Servomotor.

5.3 Helligkeitssensor

Mit Hilfe eines Fotowiderstands kann man die aktuelle Umgebungshelligkeit auslesen. Dabei könnte man den Roboter immer zum hellsten Punkt im Raum fahren lassen. Man könnte beispielsweise den Roboter in einem abgedunkelten Raum hinter dem Lichtkegel einer Taschenlampe hinterherfahren lassen und hätte damit gleichzeitig eine einfache Möglichkeit der Steuerung.

6 Herausforderungen

Auch wenn mir der Bau und die Programmierung des Roboters sehr viel Freude bereitet hat, gab es durchaus auch problematische Stellen. Einerseits war es sehr schwer sich zunächst auf ein Design festlegen, da kein Budget vorhanden war um sich eventuelle Fehlritte zu erlauben. Auch die abweichenden Maße des Acrylhassis haben anfangs für Frustration gesorgt. Letztlich gab es aber doch meist sehr einfache Lösungen für die aufgetretenen Probleme. Auch die Netzwerkeinbindung gestaltete sich anfangs sehr mühselig, da ich eher ungeeignete Libraries verwendet habe und nicht speziell die offiziellen ESP8266 WiFi Libraries. Damit verknüpft war auch das Problem der Netzwerkeinbindung, die sich auf der Grund eines Gateways in unserem Schulnetzwerk als problematisch herausgestellt hat. Allerdings bietet der ESP8266 mit dem Accesspoint-Mode eine unkomplizierte Lösung mit dem Arduino direkt zu kommunizieren.

7 Fazit

Im Großen und Ganzen ist der Bau des Roboters ein voller Erfolg. Am Anfang hatte ich noch sehr viel Respekt vor dem Thema Robotik. Auch wenn es natürlich vor allem in Industrie und Medizin sehr anspruchsvolle Anwendungen für Roboter gibt, war es doch im Nachhinein erstaunlich simpel. Letztendlich hat man mit einigen wenigen, gut verfügbaren Bauteilen und einem günstigen Chassis bereits alles, um einen sich autonom bewegenden Roboter zu basteln. Trotz der verhältnismäßig simplen Konstruktion gibt es die Möglichkeit den Roboter beliebig zu erweitern und den Programmcode immer weiter zu optimieren und erweitern. Gerade bei der Hinderniserkennung und Umfahrung gibt es fast unbegrenzte Möglichkeiten für Schüler, um zu experimentieren und zu lernen. Diese Vielzahl an Möglichkeiten röhrt vor allem daher, dass man prinzipiell jede Art von Sensor auf irgendeine Weise in der Robotik nutzen kann. Genau wie Lebewesen können auch Roboter bessere Entscheidung fällen, je mehr Sinnesindrücke von ihrer Umgebung haben. Zusammenfassend kann man sagen, dass Roboter wesentlich mehr Anschaulichkeit bieten, als herkömmliche Breadboard-Schaltungen. Es macht eben einen Unterschied, ob man lediglich die Werte ausliest oder ob man eine tatsächliche Anwendung für Daten hat.

8 Quellenverzeichnis

8.1 Buchquelle

Knapp, 2016 Markus Knapp: Roboter bauen mit Arduino, Bonn 2016

8.2 Internetverweise

- 0 <http://www.clipartsuggest.com/images/6/robot-clip-art-book-covers-feJCV3-clipart.png> (Titelbild)
- 1 <https://arduino.cc>
- 2 <https://www.ottodiy.com/>
- 3 <https://watterott.de>
- 4 <https://github.com/markusk>
- 5 <https://www.waycon.de/produkte/ultraschallsensoren/messprinzip-ultraschallsensoren/>
- 6 <https://github.com/alexanderstephan/edubot>

9 Anhang

