pamasol
Swiss Aerosol Solutions

Dokumentation über den Zusammenbau vom

NIBO Burger

Dokumentation über den Zusammenbau und die Programmierung des NIBO Burger Lehrling-Projektes für Automatiker EFZ.

Abteilung: Elektroabteilung
Verfasser: Jonas Bisig

Pamasol Willi Mäder AG
Driesbüelstrasse 2 – Postfach 157
8808 Pfäffikon SZ
Switzerland

Phone +41 55 417 40 40
Fax +41 55 417 40 44
www.pamasol.com
info@pamasol.com

Februar 2022

Zusammenfassung[1]

Der NIBO Burger ist ein Roboterbausatz der frei Programmierbar und mit 9 Sensoren ausgestattet ist. In dieser Dokumentation werden einige Einzelteile des NIBO Burgers beschrieben und deren Funktion erklärt. Zusätzlich wird die Programmiersprache und die Programmierumgebung erklärt.

Mit den 9 Sensoren kann der Roboter die Umgebung wahrnehmen und darauf reagieren. Die Sensoren bestehen aus 7 Sensor-Bricks, auf welchen immer ein LED und ein Fototransistor sitzt. Damit kann Objekte, welche sich nähern, erkennen. Mit den 3 Farbsensoren kann der Roboter alle Farben erkennen und auswerten.

Er besitzt zudem zwei Motoren, welche den Roboter antreiben. Im Set enthalten sind Zahnräder, welche zu einer Übersetzung von 25:1 oder 125:1 zusammengebaut werden können. Mit der Übersetzung von 25:1 kann der Roboter eine hohe Geschwindigkeit erreichen während mit der Übersetzung von 125:1 eine hohe Präzision erwartet werden kann.

Der Roboter enthält viele Mikrocontroller, welche ihn Steuern oder Programmieren lassen. Der Mikroprozessor auf dem NIBO Burger ist der Atmel ATmega16 mit einer Taktfrequenz von 16MHz. Für das Programmieren über USB-Anschluss ist der ATtiny44 zuständig und die Motoren werden über einen 74HC139 gesteuert. Der LM358 ist ein Operationsverstärker, welcher die Signale der Sensoren verstärkt.

Der Roboter ist mit Arduino Shields oder einem 8x8 Matrix LED-Display erweiterbar. In unserem NIBO Burger haben wir das LED-Display verbaut. Es wird mit einem zusätzlichen ATmega88A geliefert und man muss es selbst zusammenlöten.

Um den Burger zu programmieren, verwendeten wir die Programmierumgebung Atmel Studio, welche neu Microchip Studio heisst. Wir haben den NIBO Burger in der Programmiersprache C programmiert. Dabei mussten wir zuerst die vorgegebenen Exercise Tasks machen, bevor man zu den Master Tasks vorgehen konnte.

Die Aufgaben und den Umfang der Dokumentation werden auf der GitHub Seite https://github.com/pamasol/Lehrlingsprojekt-Nibo-Burger beschrieben. Dort kann man ebenfalls alle Tasks zur Kontrolle oder Inspiration anschauen. Wir benutzen die Software Git um unseren Fortschritt auf diese Webseite hochzuladen. Es gibt auch einen Installationsguide, welcher zeigt wie man Microchip Studio installiert und mit dem Burger verbindet.

[1] http://www.nicai-systems.com/de/nibo-burger

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Aufbau des Roboters

Der NIBO Burger ist in Roboterbausatz, welcher sehr einfach zusammenbaubar ist. Er ist programmierbar und kann mithilfe seiner Sensoren, die Umgebung wahrnehmen.

1.1 Mikrocontroller

1.1.1 Was ist ein Mikrocontroller?[2]

Ein Mikrocontroller ist ein winziger Computer, welcher auf einem einzigen Chip aufgebracht sind. In Abbildung 1 sieht man einen solchen Mikrocontroller. Es gibt sie in allen Grössen und Formen. Auf dem NIBO Burger ist der ATmega16A verbaut. Dieser Mikrocontroller hat eine Taktfrequenz von 16MHz. Die meisten Mikrocontroller haben einen internen Speicher, einen Zeitgeber, einen RAM Speicher und vieles mehr.

Abbildung 1, Microcontroller

1.1.2 Eigenschaften eines Mikrocontroller[3]

Einen Mikrocontroller hat folgende typische Eigenschaften:

- besitzt einen Programmspeicher (früher ROM, EPROM, EEPROM, heute FLASH)
- einen Datenspeicher (RAM)
- eine Verarbeitungseinheit (CPU)
- digitale Eingabe- / Ausgabe- Ports (GPIO), oft analoge Eingabegeräte (ADC)
- einen oder mehrere Zeitgeber
- Kommunikationsbausteine (COM, UART, …)
- manchmal spezielle Bausteine für besondere Aufgaben z.B. LCD-Treiber

[2] https://ch.rs-online.com/web/generalDisplay.html?id=ideen-und-tipps/mikrocontroller-leitfaden#Was_ist_ein_Mikrocontroller?

[3] http://einsteiger.myavr.de/index.php?id=5

1.1.2.1 Eigenschaften des verbauten ATmega16[4]

Der verbaute ATmega16 Mikrocontroller hat einen EEPROM Speicher von 512 Bytes und einen Flash-Speicher von 16kB. Er besitzt eine Taktfrequenz von 16MHz, zwei 8-bit Zeitgeber/Zähler und einen 16-bit Zeitgeber/Zähler. Da die Zeitgeber aber nur mit einer Frequenz von 8MHz arbeiten und wir einen 15MHz Zeitgeber brauchen, ist auf dem NIBO Burger noch einen Quarz Oszillator verbaut, welcher dem ATmega16 und dem ATtiny44 die gewünschten 15MHz liefern. Es sind 4 PWM Signal Ausgänge und 32 Digitale Ein- und Ausgänge vorhanden. Der ATmega16 arbeitet mit 4.5 – 5.5V und max. 1.1mA. in Abbildung 2 kann man die verschiedenen Anschlüsse des ATmega16 sehen.

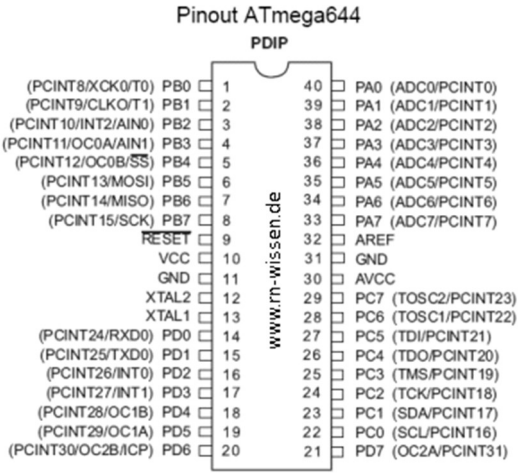

Abbildung 2, ATmega16 Pinout

1.2 Wie werden 3.3V auf der Platine generiert?[5]

Da auf dem NIBO Burger Sensoren verbaut sind, welche 3.3 Volt benötigen, wurde ein Spannungsregler eingebaut. Auf Abbildung 3 kann man die Bauweise eines solchen Spannungsregler begutachten. Der NIBO Burger wird mit 4 1.25 Volt Akkus betrieben. Der Spannungsregler erhält diese Spannung und wandelt sie dann auf 3.3 Volt um. Dabei gibt es einen Verluststrom, welcher die überschüssige Energie durch Wärme freisetzt. Es wurde der TSC 295033 als Spannungsregler verwendet, welcher eine Eingangsspannung von 0.3 – 30 Volt umwandeln kann.

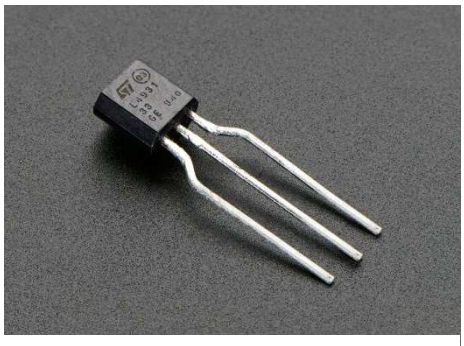

Abbildung 3, 3.3V Spannungsregler

[4] http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf

[5] https://www.mouser.com/datasheet/2/395/TS2950-51_B07-231899.pdf

1.3 Antrieb

1.3.1 Motor Steuerung mit H-Brücke[6]

Der NIBO Burger besitzt zwei Motoren, welche ihn antreiben. Mit der mitgelieferten Getriebezahnräder kann man eine 125:1 oder eine 25:1 Übersetzung zusammenbauen. Mit der Übersetzung von 25:1 kann der Roboter eine hohe Geschwindigkeit erreichen, während er mit der Übersetzung von 125:1 präzise eine Position anfahren kann. Auf unserem Modell haben wir die 25:1 Übersetzung eingebaut, um mit dem Roboter hohe Geschwindigkeiten fahren zu können. Die Motoren werden mit einer H-Brücke, in der sich 4 Transistoren befinden, angesteuert. Man kann die H-Brückenschaltung des NIBO Burgers in Abbildung 4 sehen. Die H-Brückenschaltung hat den Vorteil, dass der Motor sehr schnell die Drehrichtung ändern kann. Zusätzlich ist sie sehr kompakt und braucht nicht viel Platz auf dem Roboter. Durch die hohe Schaltgeschwindigkeit der Transistoren kann der Motor mit einem PWM Signal angesteuert werden. Der Motor erhält seine Spannung, indem die zwei diagonal gegenüberliegenden Transistoren schalten. Um die Drehrichtung zu ändern, schaltet man die anderen zwei Transistoren. Das bewirkt, dass an dem Motor die Spannungsrichtung umgedreht wird. Dieses Prinzip wird in der Abbildung 5 veranschaulicht.

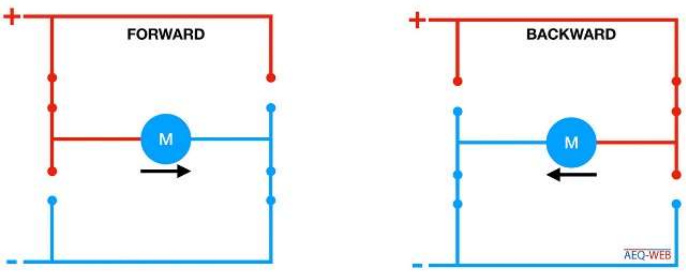

Abbildung 4, Beispiel H-Brücke

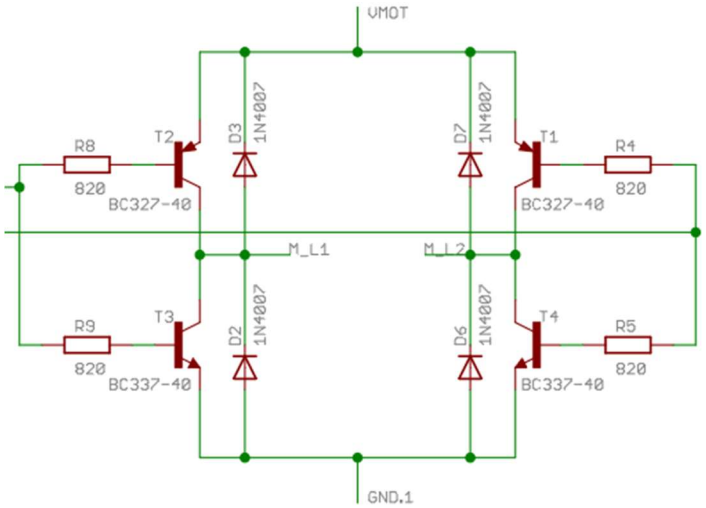

Abbildung 5, H-Brücke mit Transistoren auf dem NIBO Burger

[6] https://www.aeq-web.com/arduino-mosfet-h-bruecke-dc-motor-steuern-mit-drehrichtung-und-geschwindigkeit-vierquadrantensteller/

1.3.2 PWM Signal[7]

Das PWM Signal, auch Pulsweitenmodulation genannt, besteht aus Pulsen mit verschieden langer Periodendauer. Das Signal besteht nur aus 2 Werten, die maximal Spannung und keine Spannung. Dabei bleibt die Frequenz der Pulse immer gleich, denn es wird die Dauer des Rechteckpulses moduliert. Mit dieser Technik kann man die Stärke einer Gleichspannung Regeln oder die Gleichspannung in eine Wechselspannung umformen. In Abbildung 6 kann man das Zeitdiagramm eines solchen PWM-Umformers sehen. Bei unserem Roboter wird ein PWM Signal verwendet, um die Transistoren der H-Brücke anzusteuern. Damit kann man die Drehgeschwindigkeit des Motors regulieren. Die Funktion des PWM Signals ist relativ einfach: Wenn man eine kurze Periodendauer der Rechteckpulse hat, ist die durchschnittliche Spannung tief und wenn man eine lange Periodendauer hat, ist die durchschnittliche Spannung hoch. Ein PWM Signal kann nur bei trägen Verbrauchern, wie zum Beispiel Motoren, eingesetzt werden. Da der Motor ein gewisses Momentum besitzt und sich auch noch spannungslos eine Zeit langdrehen würde, merkt der Motor nicht wenn man ihn mit einem gepulsten Signal ansteuert. Der NIBO Burger steuert das PWM Signal mit einer Frequenz von 14.7 kHz, das heisst, es gibt 14'700 Pulse in einer Sekunde.

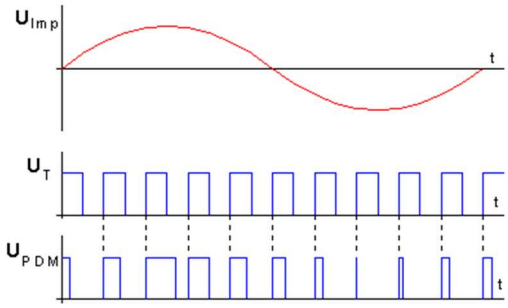

Abbildung 6, PWM-Signal

1.4 Sensorik

1.4.1 IR-Sensoren[8]

Die Infrarot Sensoren bestehen aus einer Infrarot LED und einem Fototransistor. Diesen IR-Sensoren Brick kann man in der Abbildung 7 sehen. Die LED's senden einen Infrarot-Lichtstrahl aus, der dann von einem Objekt reflektiert wird und vom Fototransistor erkannt wird. In Abbildung 8 wird dieses Prinzip mit einem Bild veranschaulicht. Die Fototransistoren reagieren je nach Typ auf ultraviolettes Licht, bis hin zu infrarotem Licht. Da unsere Fototransistoren auf Infrarotes Licht reagieren, sind sie mit einem schwarzen Kunststoff geschützt. Die schwarze Farbe dient dazu, gewisses Störlicht, zum Beispiel einer Lampe oder der Sonne, zu filtern. Zum zusätzlichen Schutz haben wir schwarze Schrumpfschläuche über die Fototransistoren gestülpt, so kann er nur auf Licht, welches von vorne kommt, reagieren. Wir können die Signale, die die IR-LED aussendet, nicht sehen, da die Wellenlänge von infrarotem Licht zwischen 780nm und 1mm liegen. Das für uns sichtbare Licht, hat Wellenlängen von 380nm bis 780nm.Wellenlängen die kürzer als 380nm sind, gelten als Ultraviolett.

---

[7] https://de.wikipedia.org/wiki/Pulsdauermodulation

[8] https://de.wikipedia.org/wiki/Fototransistor

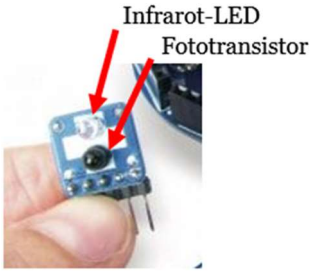

Abbildung 7, IR-Sensor Brick

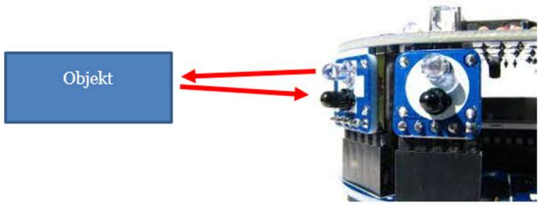

Abbildung 8, IR-Sensor Reflexion

1.4.2 RGB-Sensoren[9]

RGB steht für Rot, Grün und Blau, welche die drei Grundfarben des Lichts bilden. Die RGB-Sensoren Bricks sind gleich aufgebaut wie die IR-Sensoren. Der Unterschied dabei ist, dass es auf den verschiedenen Bricks ein rotes, ein grünes, und ein blaues LED hat. Zudem wurde ein Fototransistor verbaut, welcher auf sichtbares Licht und auf die entsprechende Farbe empfindlich ist. Alle 3 Bricks zusammen können dann jede Farbe, zum Beispiel auf einem Blatt Papier, erkennen und auswerten. Das geht, weil man mit den verschiedenen Werten der einzelnen Sensoren alle anderen Farben ausrechnen kann. Die Bricks sieht man in der Abbildung 9.

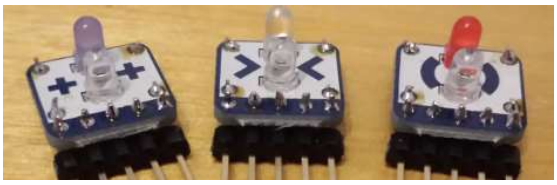

Abbildung 9, RGB-Sensor Bricks

1.5 Maroon SHIELD 8x8 LED-Display[10]

Da der NIBO Burger erweiterbare Module besitzt, haben wir noch das Maroon SHIELD 8x8 LED-Display dazu bekommen, welches in der Abbildung 11 abgebildet ist. Mit der mitgelieferten Anleitung war es einfach das Zusatzmodul zusammenzulöten. Als es fertig verlötet war, konnte man es einfach auf den NIBO Burger draufstecken. Gesteuert wird das LED-Display durch einen weiteren Mikrocontroller, den ATmega88A. Der NIBO Burger kann per seriellen Schnittstelle mit dem ATmega88A kommunizieren und neue Programme speichern. Mit diesem Display kann man viele verschiedene Sachen anzeigen lassen. Zum Beispiel kann man einen Schriftzug machen, verschiedene Animationen, einzelne LED's aufleuchten lassen und vieles mehr.

Abbildung 10, Maroon SHIELD 8x8 Matrix

[9] https://de.wikipedia.org/wiki/Farbsensor

[10] https://wolles-elektronikkiste.de/led-matrix-display-ansteuern

Die Ansteuerung der 64 LEDs auf dem Display erfolgt über 16 Mikrocontroller Ausgänge. Möglich ist das, weil nicht jede LED einzeln angesteuert wird. Das Display wird in Reihen und Spalten aufgeteilt. Als erstes werden alle Reihen auf LOW gesetzt und die Spalten auf HIGH. Wenn man nun die erste, oberste LED ansteuern möchte, muss man die gewünschte Reihe auf HIGH setzen und die zur LED gehörigen Spalte auf LOW. Wenn man nun mehrere Spalten auf LOW setzt, gehen auch mehrere LED's an. Sobald man aber eine zweite Reihe ansteuern möchte, geht das nicht mehr so einfach, denn die Signale überqueren sich. Darum nutzt man das Multiplexing Verfahren. Als erstes steuert man nur die erste Reihe an, wartet eine Millisekunde, dann steuert man die zweite Reihe an usw. Da unser Auge eine so kurze Zeit nicht erkennen kann, sehen wir dann alle Reihen gleichzeitig und es setzt die LED's zu einer Form zusammen.

In der Abbildung 10 eines 7x5 Matrix Displays sieht man sehr gut wie das Multiplexing funktioniert. Das Programm führt Schritt 1 bis 7 nacheinander ganz schnell aus und wir sehen das Endergebnis in Schritt 8.

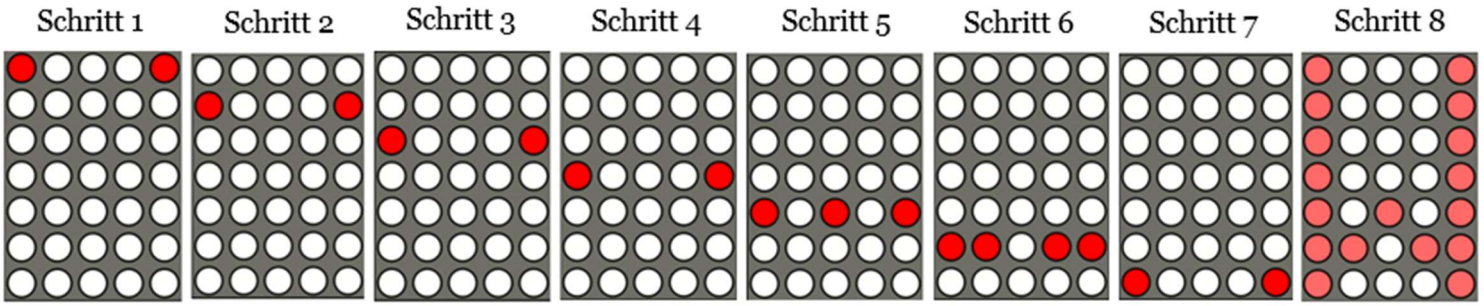

Abbildung 11, Multiplexing Beispiel an 7x5 Display

2 Programmieren und Software

2.1 Programmiersprache

Der NIBO Burger wird in der Programmiersprache C programmiert. Um das Programm zu schreiben, haben wir die Programmierumgebung Atmel Studio benutzt die heute Microchip Studio heisst. Das Logo des Programmes ist in Abbildung 13 zu sehen. Um die gemachten Aufgaben auf GitHub hochzuladen haben wir zusätzlich noch die Software Git verwendet. Git ist eine frei verfügbare Software, welche für die Versionsverwaltung von Dateien genutzt wird. Die Programmiersprache C ist recht einfach aufgebaut. Es gibt einen setup Teil und einen loop Teil. Der setup Teil wird bei Beginn des Programms einmal ausgeführt, der loop Teil wird immer wieder wiederholt. Im setup Teil werden die verschiedenen Ein- und Ausgänge deklariert und gewisse Bibliotheken implementiert. Im loop Teil schreibt man dann den eigentlichen Teil des Programmes.

2.1.1 Variablen

```
4   void setup()
5   {
6           led_init();
7   }
9   void loop()
10  {
11          led_set(2,1);
21  }
```

Abbildung 12, Beispielsprogramm

Abbildung 13, Microchip Studio Logo

Variablen werden zum Speichern von Werten verwendet. Variablen gibt es in verschiedenen Formaten. Zum Beispiel das Format Int, welches ganze Zahlen speichern kann. Die Grösse des Speichers hängt von dem Compiler des Programms ab. In unserem Fall ist der Speicher 32 Bit gross und kann somit Zahlen von 32700 bis -32700 speichern. Float ist auch ein Datentyp. Dieser kann gebrochene Zahlen speichern. Das Format der Variable muss man bei der Deklaration angeben, dass das Programm weiss, was in der Variable gespeichert werden kann. Die Variablen werden dann dynamisch im Programm abgeändert. Variablen sind nicht von überall erreichbar. Sind sie innerhalb einer Funktion deklariert, dann können sie auch nur in dieser Funktion aufgerufen werden. Wenn man sie aber ausserhalb von Funktionen deklariert, sind sie Global und somit von überall her abrufbar. In Abbildung 14 ist abgebildet, wie man variablen definiert.

```
3   int drehen = 130;
4   int gerade = 100;
5   int led;
```

Abbildung 14, Variablen

2.1.2 For-Schleife

Eine for-Schleife ist eine Widerholungssteuerung, die es uns ermöglicht, eine Schleife eine bestimmte Anzahl von Malen auszuführen. Die for-Schleife ermöglicht es uns, übersichtlich und kurz eine Schleife n-Mal auszuführen. In Abbildung 15 ist ein Beispiel einer for-Schleife abgebildet. In einer for-Schleife wird eine Schleifenvariable verwendet, welche die Schleife steuert.

```
96  for (led = 1; led <=4; led++)
97  {
98  led_set(led, 1);
99  delay(500);
100 }
```

Abbildung 15, For-Schleife

2.1.3 Funktionen

Funktionen sind Codeblöcke, welche man beliebig mal ausführen kann. Der Vorteil dabei ist, dass man ein kompliziertes Programm in verschiedene Teile unterteilen kann und es somit einfacher zu lesen ist. Innerhalb der Funktion können Ein- und Ausgänge gesteuert werden und Variablen verändert werden. Sie Syntax zum Implementieren einer Funktion kann man in Abbildung 16 anschauen. Man kann bei der Aufrufung der Funktion einen Wert, zum Beispiel einer Variabel, mitgeben. Dieser Wert kann dann innerhalb der Funktion verarbeitet werden. Die Syntax des Aufrufens der Funktion wird in Abbildung 17 aufgezeigt. Mit dem Keyword «return» kann auch aus der Funktion einen Wert zurückgegeben werden.

```
3   void ExampleFunction() {
4       //ExampleCode
5
6       return ExampleVariable;
7   }
```

Abbildung 17, Funktion deklarieren Syntax

```
10  ExampleFunction(ExampleVariable);
```

Abbildung 16, Funktion aufrufen Syntax

2.2 State Machine[11]

Um uns die Mastertasks ein wenig zu erleichtern, wurde das Prinzip der Statemachine gezeigt. Mit einer Statemachine kann man Programme erstellen, die mit einer Abfolge von Fragen, Sachen herausfinden können. Dafür braucht man nur eine Variable, die mit Bedingungen geändert wird, und einen „Switch Case“. Zu Beginn setzt man die Variable auf 0. Im Loop verwendet man dann einen „Switch Case“, welcher den Wert der Variable überwacht. Wenn nun die Variable 0 ist, wird eine bestimmte Abfrage gemacht. Sobald diese zutrifft, setzt man die Variable auf 1. Jetzt gilt im „Switch Case“ der 1. Case und es wird etwas anderes abgefragt. Das geht dann immer so weiter, bis die Statemachine ihren gewünschte wert herausgefunden hat. Danach wird die Variable wieder auf 0 gesetzt und es beginnt von vorne. Eine einfachere Darstellung der Statemachine kann man in Abbildung 18 sehen.

[11] https://de.wikipedia.org/wiki/Endlicher_Automat

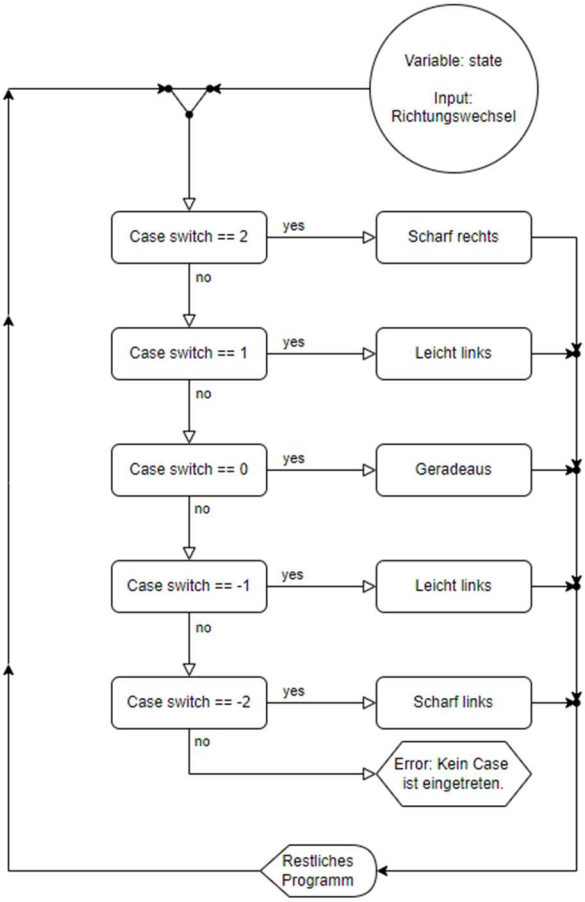

Abbildung 18, State-Machine

2.3 Git und GitHub[12]

Git ist eine Software, die zur Versionenverwaltung und Datei Teilung innerhalb eines Netzwerkes genutzt wird. Erfunden hat die Software Linus Torvalds, welcher auch der Erfinder des Betriebssystems Linux ist. Git ermöglicht es mehreren Entwicklern parallel an einem Projekt zu arbeiten. Git ist eine Open-Source Software. Das heisst, eine Open-Source-Software ist Computersoftware, die unter einer Lizenz veröffentlicht wird, bei der der Urheberrechtsinhaber den Benutzern das Recht einräumt, die Software und ihren Quellcode an jedermann und für jeden Zweck zu verwenden, zu studieren, zu ändern und zu verteilen. Das Logo von Git sieht man in Abbildung 19. Mit Git kann man auch eigene Projekte auf verschiedenen Plattformen wie GitHub oder GitLab teilen. Auf diesen Plattformen kann man das Projekt teilen, vergleichen und abändern. Man kann sich auch gegenseitig helfen und Kommentare schreiben. Das Logo von GitHub wird in Abbildung 20 aufgezeigt.

[12] https://de.wikipedia.org/wiki/Git

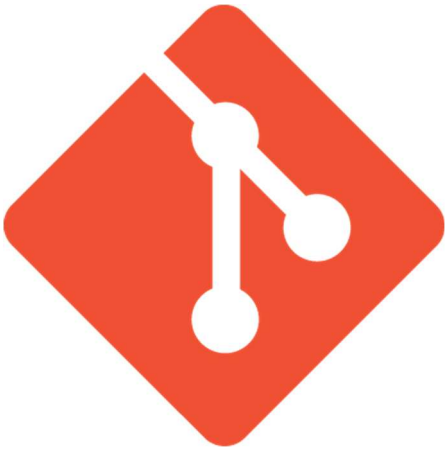

Abbildung 19, Logo Git

Abbildung 20, Logo GitHub

3 Schlusswort

Das NIBO Burger Lehrlingsprojekt hat mir sehr spass gemacht. Ich habe vieles dazu lernen dürfen. Ich habe gelernt, wie man Platinen nach Schema bestückt und zusammenlötet. Zudem konnte ich einen besseren Einblick in die Mikroelektronik erlangen. Auch Softwaretechnisch habe ich viel neues dazugelernt, wie zum Beispiel die Programmiersprache C. Ich fand es auch gut, dass wir mit der Software Git gearbeitet haben, diese habe ich nämlich vorher noch nicht gekannt. Es war ein wenig schade, dass sich das Getriebe des NIBO Burgers zum Teil verkantet hat. So konnten wir nicht genaue Fahrten machen da wir ihn immer wieder anschubsen mussten. Ich fand das Projekt trotzdem sehr unterhaltsam und lehrreich und es hat mir sehr Spass gemacht.

4 Quellenverzeichnis

Abbildung 1, https://www.conrad.ch/de/p/microchip-technology-atmega8535-16pu-embedded-mikrocontroller-pdip-40-8-bit-16-mhz-anzahl-i-o-32-154257.html

Abbildung 2, https://rn-wissen.de/wiki/index.php/ATmega16_ATmega32_ATmega644

Abbildung 3, https://www.adafruit.com/product/2166

Abbildung 4, https://www.homofaciens.de/technics-base-circuits-h-bridge_ge.htm

Abbildung 5, https://www.aeq-web.com/arduino-mosfet-h-bruecke-dc-motor-steuern-mit-drehrichtung-und-geschwindigkeit-vierquadrantensteller/

Abbildung 6, https://www.elektronik-kompendium.de/sites/kom/0401111.htm

Abbildung 7, https://www.pollin.de/p/bausatz-nibo-burger-nicai-systems-810806

Abbildung 8, http://www.nicai-systems.com/de/nibo-burger-spezifikationen

Abbildung 9, https://www.mikrocontroller.net/topic/375728

Abbildung 10, https://www.reichelt.com/de/en/maroon-shield-8x8-led-matrix-display-maroon-shield-p161083.html

Abbildung 11, https://upload.wikimedia.org/wikipedia/commons/c/ca/Dot_matrix.gif

Abbildung 13, https://www.microchip.com/en-us/tools-resources/develop/microchip-studio

Abbildung 17, https://www.mikrocontroller.net/articles/Statemachine

Abbildung 18, https://git.unileoben.ac.at/howto/git

Abbildung 19, https://commons.wikimedia.org/wiki/File:Octicons-mark-github.svg