NIBO BURGER

DOKUMENTATION

ZUSAMMENBAU UND PROGRAMMIERUNG

12.05.2022

Stefan Feier

Inhaltsverzeichnis

1. Erklärung des Roboters

1.1 Hauptplatinen

Der Roboter besteht auf zwei Hauptplatinen. Die obere Hauptplatine **(Bild 1)** ist für das Laden der Akkus zuständig. Auf dieser Platine auch ist die USB-Schnittstelle und die Steckplätze für das Maroon Shield zu finden. Auf der unteren Hauptplatine sitzt der Haupt IC, der ATmega16A. Dieser kann mit einem PC programmiert werden. Die beiden Motoren sind auch auf der unteren Platine befestigt.

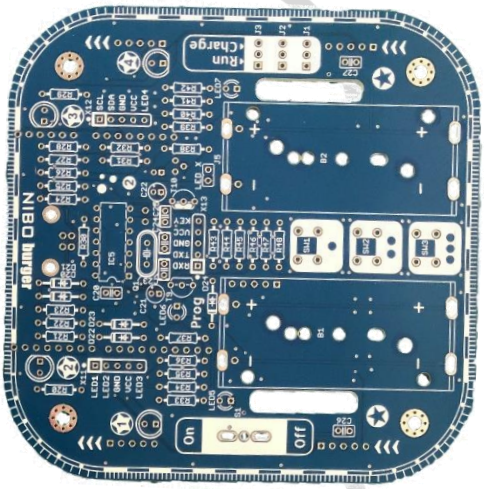


Bild 1: Hauptplatine 2

1.2 Sensoren

Auf den beiden Platinen gibt es insgesamt 13 Steckplätze für Erweiterungen. In diese können Farbsensoren **(Bild 2)**, Infrarotsensoren, LEDs oder ein Bluetooth Modul für die Steuerung über ein Smartphone eingesteckt werden.

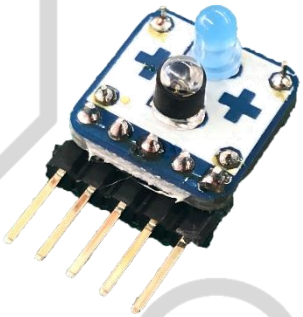


Bild 2: Blauer Farbsensor

1.3 ATmega16A

Das Gehirn des Roboters ist der ATmega16A **(Bild 3)**. Das ist ein Mikrocontroller mit 16K Bytes Speicher. Er hat 40 Pins und kann mit Frequenzen bis 16MHz verwendet werden. Im NIBO Burger wird er mit 15MHz verwendet (Quarz). Die Spannung beträgt 2.7V bis 5.5 V. Dabei verbraucht er maximal 0.6mA.

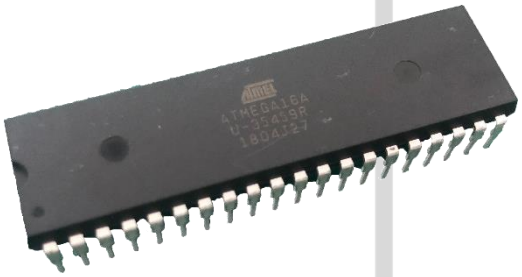


Bild 3: ATmega16A

1.4 Motorsteuerung

Um die Geschwindigkeit der Motoren zu regeln, werden die Motoren mit dem 74HC139 **(Bild 4)** und einem PWM (Pulsweitenmodulation) Signal **(Abbildung 4.1)** gesteuert. Das heisst, dass die Spannung immer von 0V auf 5V erhöht und wieder auf 0V abgesenkt wird. Je nachdem, wie lange die Spannung 5V beträgt (Pulsdauer), läuft der Motor schneller oder langsamer.

Bild 4: 74HC139

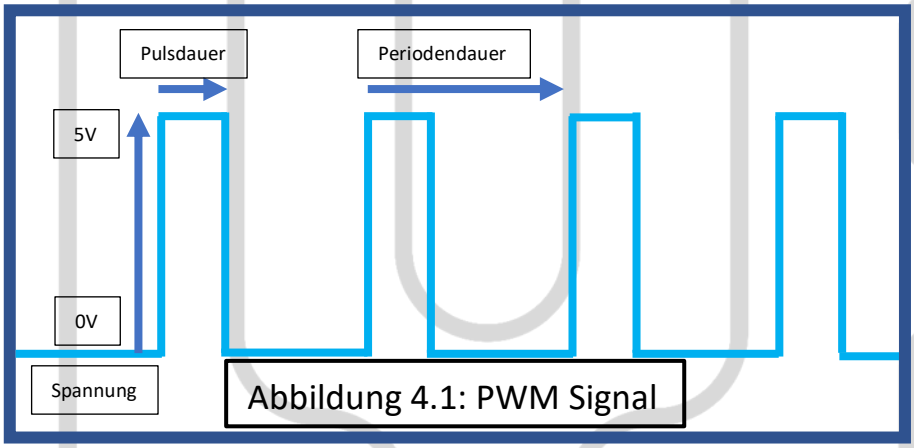


Abbildung 4.1: PWM Signal

1.5 Getriebe

Die Kraft der Motoren wird über das Getriebe **(Bild 5)** an die Räder weitergeleitet. Wenn man das normale Getriebe einbaut (125:1), fährt der Roboter langsam, kann aber sehr genau gesteuert werden. Wenn man die Anordnung der Zahnräder ändert, kann der Roboter sehr schnell fahren, ist aber nicht mehr so genau (25:1).

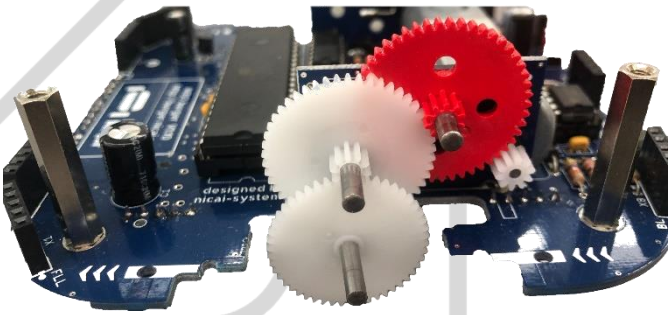


Bild 5: Getriebe

1.6 3.3V Spannungsversorgung

Die Sensoren werden mit 3.3V Spannung versorgt. Dafür müssen die 5V von den Akkus mithilfe vom TSC295033 **(Bild 6)** runter geregelt werden. Der maximale Strom beträgt 150mA.

Der TSC295033 ist ein «Low Dropout Voltage Regulator». Das heisst, dass die Ausgangsspannung sehr nahe an der Eingangsspannung liegen kann. z.B. Eingang = 2.7 V / Ausgang = 2.5 V

Bild 6: TSC295033

1.7 Funktion Infrarotsensoren

Die Infrarotsensoren senden ein Infrarotlicht aus. Wenn dieses Licht auf ein Objekt trifft, wird das Infrarotlicht zurückgestrahlt. Das zurückgestrahlte Licht wird vom Infrarotempfänger wahrgenommen **(Abbildung 7).** Diesen Wert kann man dann in seinem Programm verwenden, um den Roboter Objekten ausweichen zu lassen.

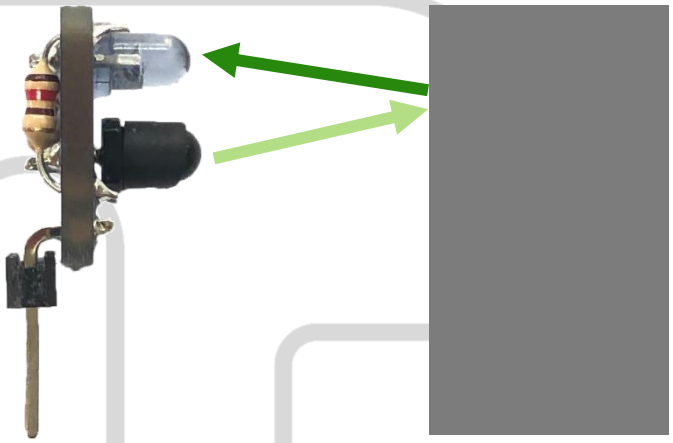


Abbildung 7: Funktion IR Sensor

2. Roboterbau

Als Erstes habe kontrolliert, ob alle Bauteile geliefert wurden. Danach habe ich die kleinen Bauteile auf zwei A4 Blätter festgeklebt und beschriftet, damit es einfacher ist, die richtigen Bauteile zu finden **(Bild 8)**.

Dadurch konnte ich in der Anleitung schauen, welches Bauteil ich benötige und dieses dann sofort einbauen, ohne viel zu suchen.

Die grossen Bauteile habe ich nicht auf ein Blatt geklebt, weil man diese auch ohne Sortieren gut findet.

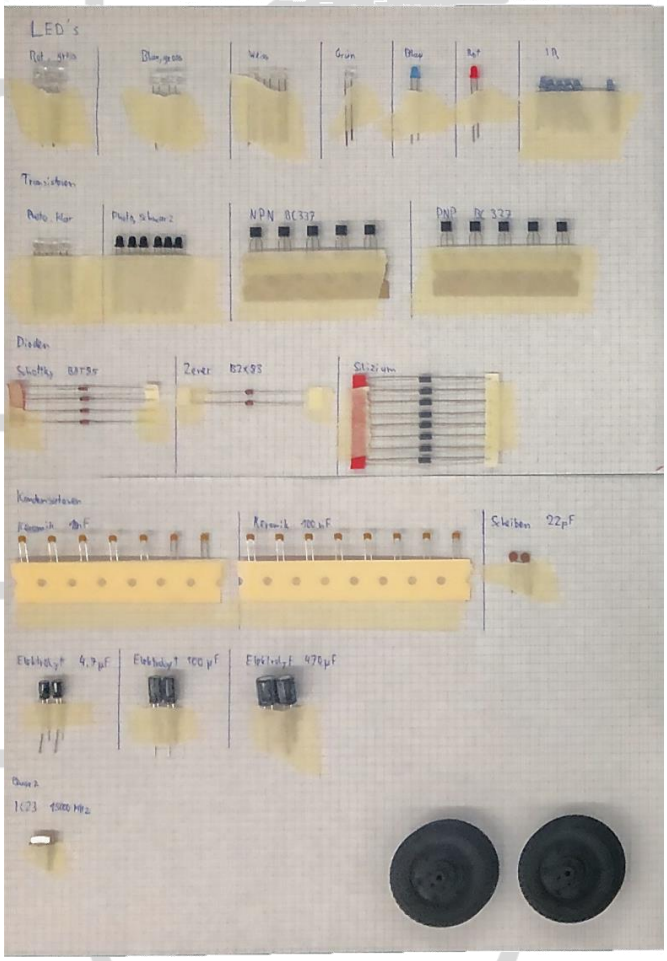


Bild 8: Sortierung der Bauteile

Der Zusammenbau des Roboters ist einfach, da es eine gute Anleitung dafür gibt. Dort wird Schritt für Schritt erklärt, welche Bauteile wo auf die Leiterplatte gelötet werden müssen und wie die fertigen Platinen zusammengeschraubt werden müssen.

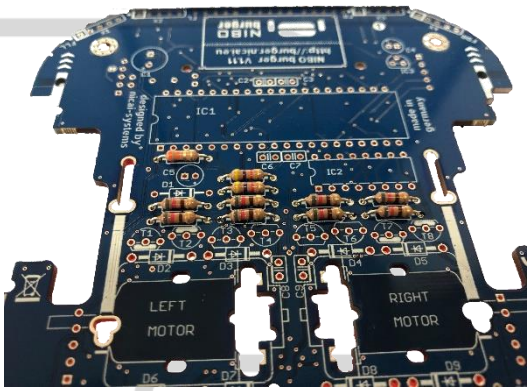


Bild 9: Platine 1

Als Erstes werden die kleinen Sensorbausteine bestückt und anschliessend gelötet. Nachdem die Zwischenplatinen und die Platinen für die Odometriesensoren bestückt und gelötet sind, beginnt man mit den beiden Hauptplatinen **(Bild 9 und 9.1)**. Auf diesen sitzen alle wichtigen Bauteile.

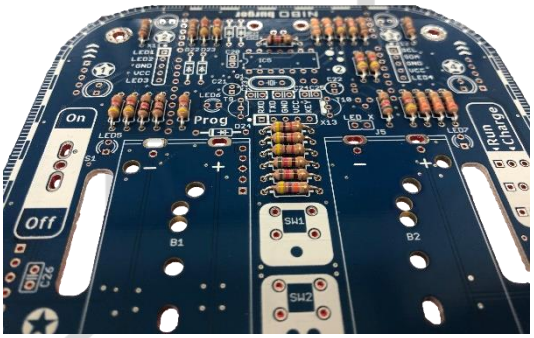


Bild 9.1: Platine 2

Beim Löten muss darauf geachtet werden, dass keine Pins miteinander verbunden werden. Es ist auch wichtig, dass alle Pins der Bauteile an die Platine gelötet werden und nichts vergessen wird. Ansonsten funktioniert der Roboter nicht.

Es werden zuerst alle kleinen Bauteile angelötet. Die grössten kommen erst am Schluss auf die Platine. So kann der Lötrahmen die Bauteile gut halten und das Löten ist einfacher.

Wenn alle Platinen gelötet wurden, baut man das Getriebe **(Bild 5)** ein und schraubt die Platinen dann zusammen.

Für den ersten Test ist ein Startprogramm auf dem ATmega16A vorinstalliert. Mit diesem Programm testet man die LEDs und Motoren und kalibriert die Farbsensoren.

3. Programmieren

Das Programmieren ist der spannendste Teil, weil man dort selbst etwas ausprobieren kann und sieht, ob das so funktioniert.

Der Roboter wird mit Microchip Studio in der Programmiersprache C programmiert.

Beim Programmieren hat man verschiedene Aufgaben, die man lösen soll. Zuerst gibt es 15 Übungsaufgaben, die noch relativ einfach sind. Nachdem die Übungsaufgaben gelöst wurden, kann man die grossen Aufgaben programmieren. Von diesen gibt es 5 Stück, welche immer schwieriger werden.

Ich habe zum Schluss noch ein finales Programm erstellt. Damit kann man eines der 5 Programme auswählen und ausführen. Beim Start wird auch noch die Batterieladung in % angegeben. Wenn der Wert unter 40% ist, schaltet der Roboter automatisch aus, weil unter einer gewissen Spannung die Sensoren nicht mehr richtig funktionieren. Das bedeutet auch, dass die meisten Programme nicht mehr laufen würden.

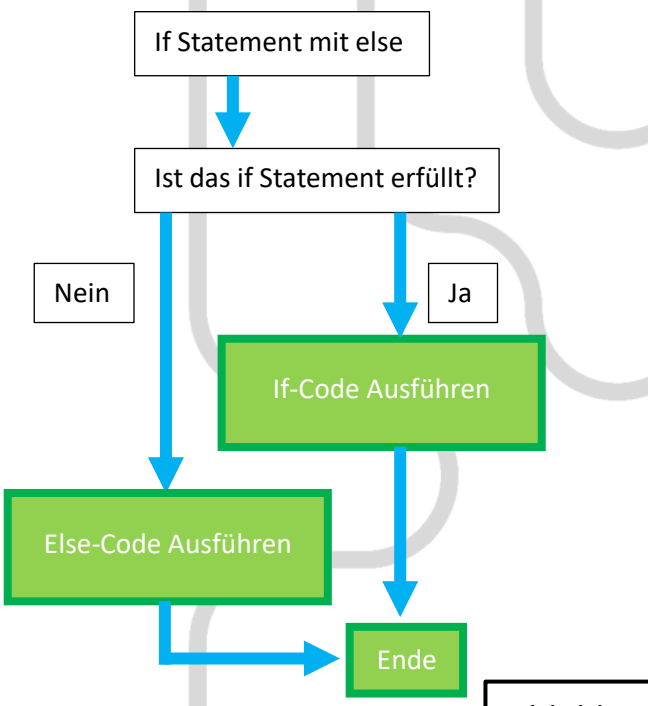
3.1 Anleitung zum finalen Programm

1. Taster 1 betätigen und gleichzeitig einschalten
2. Warten bis LED1 kurz blinkt
3. Taster 1 loslassen
4. Es wird die Batterieladung angezeigt
5. Ist die Batterieladung 40% oder grösser, schaltet der Roboter ein
6. Mit der «FN» Taste auf dem Maroon Shield kann das gewünschte Programm ausgewählt werden.
7. Das Programm mit der «Enter» Taste bestätigen
8. Das Programm ist geladen und kann verwendet werden.

4. Programmierbeispiele

4.1 If Statement

Das If Statement habe ich in meinen Aufgaben am häufigsten verwendet. Das if Statement funktioniert folgendermassen: Man kann eine Bedingung definieren und wenn diese erfüllt ist, wird der darunterliegende Code ausgeführt. Wenn die Bedingung nicht erfüllt ist, wird das ganze if Statement übersprungen. Mit dem Befehl «else» wird dann der darunterliegende Code ausgeführt, wenn das if Statement nicht erfüllt ist. **(Abbildung 10, 10.1 + 10,2)**



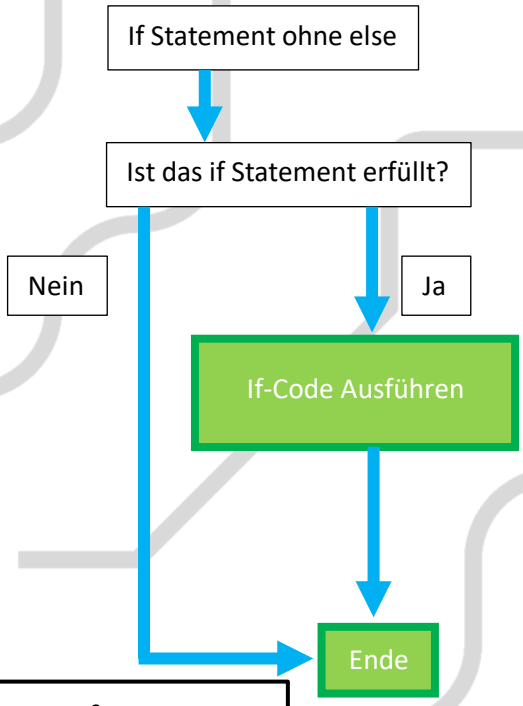


Abbildung 10: If Statement

```
if(y==x)    {
    led_set(1,1);
    delay(500);
    led_set(1,0);
    delay(500);
}
```

Abbildung 10.1: If Programmierbeispiel

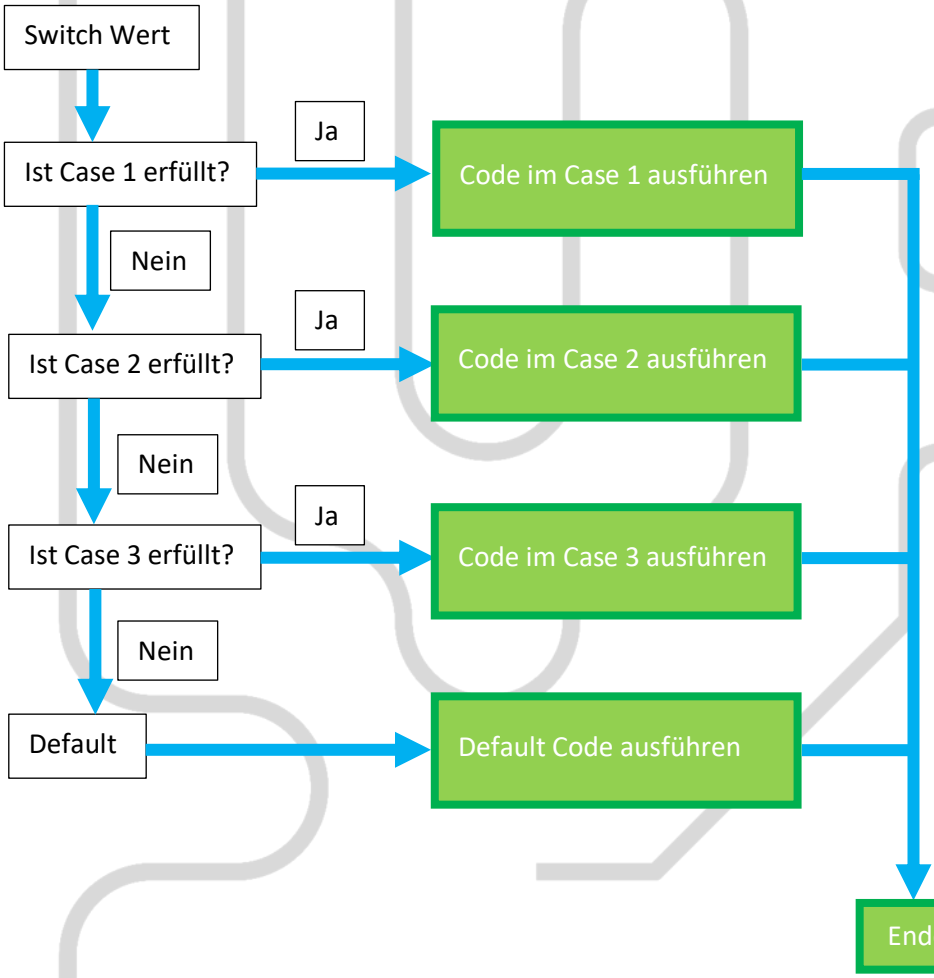
```
if(y==x)    {
    led_set(1,1);
    delay(500);
    led_set(1,0);
    delay(500);
}

else
{
    led_set(2,1);
    delay(500);
    led_set(2,0);
    delay(500);
}
```

Abbildung 10.2: If else Programmierbeispiel

4.2 Switch Case Statement

Beim case Statement hat man einen veränderbaren wert (z.B. value). Wenn dieser Wert zum Beispiel 3 ist, wird der Code im Case 3 ausgeführt. Man kann auch ein «default» definieren. Wenn kein Case zutrifft, wird der default case ausgeführt **(Abbildung 11 + 11.1)**.




```
int value=0;

switch (value)  {
    case 1:
    led_set(1,1);

    break;

    case 2:
    led_set(2,1);
    }

    break;

    case 3:
    led_set(3,1);

    break;

    default:
    led_setall(0,0,0,0);
}
```

Abbildung 11: Switch Case

Abbildung 11.1: Case Programmbeispiel

4.3 For Loop

Ein For Loop wird verwendet, um festzulegen, wie oft der Code ausgeführt wird. So kann man einen Code mehrmals wiederholen, ohne viel Platz im Code zu verbrauchen **(Abbildung 12 + 12.1)**.

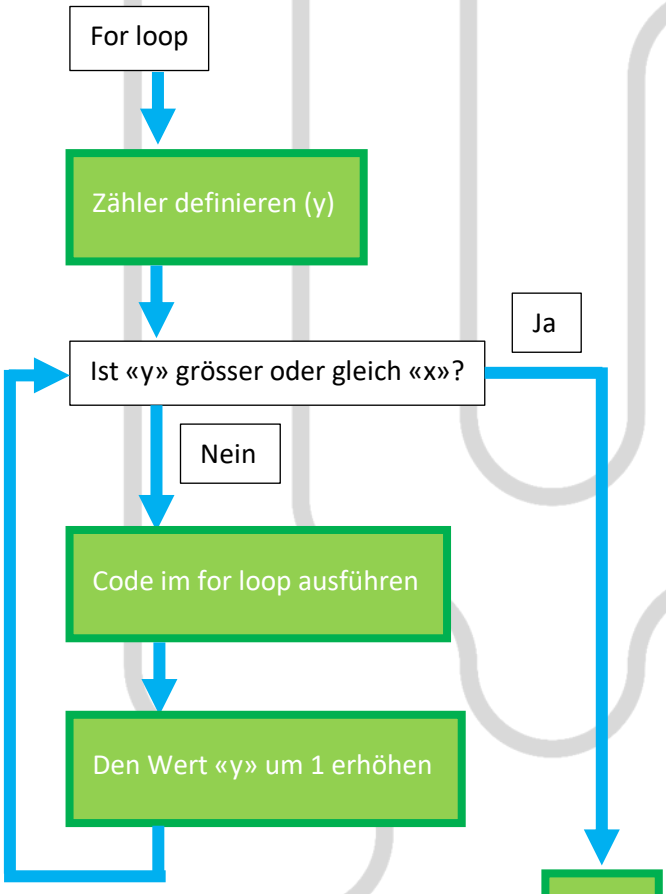


Abbildung 12: For Loop

```
for(int y=0; y<=x; y=y+1)    {
    led_set(1,1);
    delay(500);
    led_set(1,0);
    delay(500);
}
```

Abbildung 12.1: For Programmbeispiel

4.4 While Loop

Ein while loop ist ähnlich wie ein for loop. Der Unterschied ist, dass der Code im while loop so lange ausgeführt wird, bis eine Bedingung nicht mehr erfüllt ist und nicht, bis der Code x-mal durchgelaufen ist **(Abbildung 13 + 13.1)**.

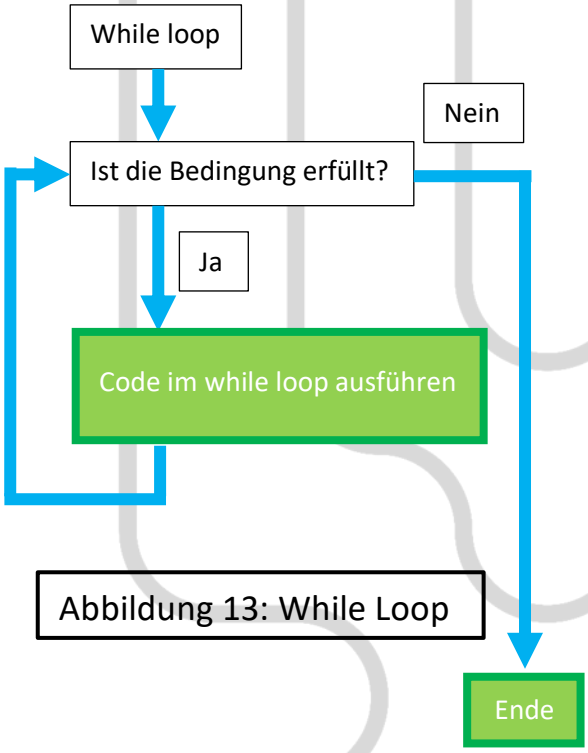


Abbildung 13: While Loop

```
int y = 0;

while(y<=x) {
    led_set(1,1);
    delay(500);
    led_set(1,0);
    delay(500);
}
```

Abbildung 13.1: While Programmbeispiel

4.5 Break Statement

Das break Statement kann bei allen Loops verwendet werden. Beim break Statement wird der loop verlassen, sobald es aktiv ist, auch wenn die Bedingung für das normale Verlassen des loops noch nicht erfüllt ist. **(Abbildung 14)**

```
for(int y=0; y<=x; y=y+1)   {
    led_set(1,1);
    delay(500);
    led_set(1,0);
    delay(500);
    if(y==30)   {
        break;
    }
}
```

Abbildung 14: Break Programmierbeispiel

5. Maroon Shield

Das Maroon Shield ist ein Display, welches auf den NIBO Burger oder ein Arduino aufgesteckt werden kann. Es hat ein 8x8 Pixel Display und 2 Taster.

5.1 USART (Universal Synchronous/Asynchronous Receiver/Transmitter)

Über USART kommuniziert das Maroon Shield mit dem NIBO Burger. USART ist eine elektronische Schaltung mit der 2 Geräte miteinander kommunizieren können. Dabei gibt es 2 Verbindungen: RX und TX. RX ist der Empfängeranschluss und TX ist der Sendeanschluss. Beim Anschliessen muss darauf geachtet werden, dass TX von Gerät 1 zu RX auf Gerät 2 verbunden wird. TX von Gerät 2 muss natürlich auch auf RX von Gerät 1 geführt werden. Das macht man darum, weil Gerät 1 auf TX sendet und Gerät 2 auf RX auf Daten wartet. **(Abbildung 15)**

Beim USART gibt es, im Gegensatz zum einfacheren UART, ein Zeitsignal. Damit muss das andere Gerät die Baud Rate nicht kennen. Zusätzlich ist die Übertragungsrate bei USART höher als beim normalen UART. Diese beträgt bis zu 4 Mbps.

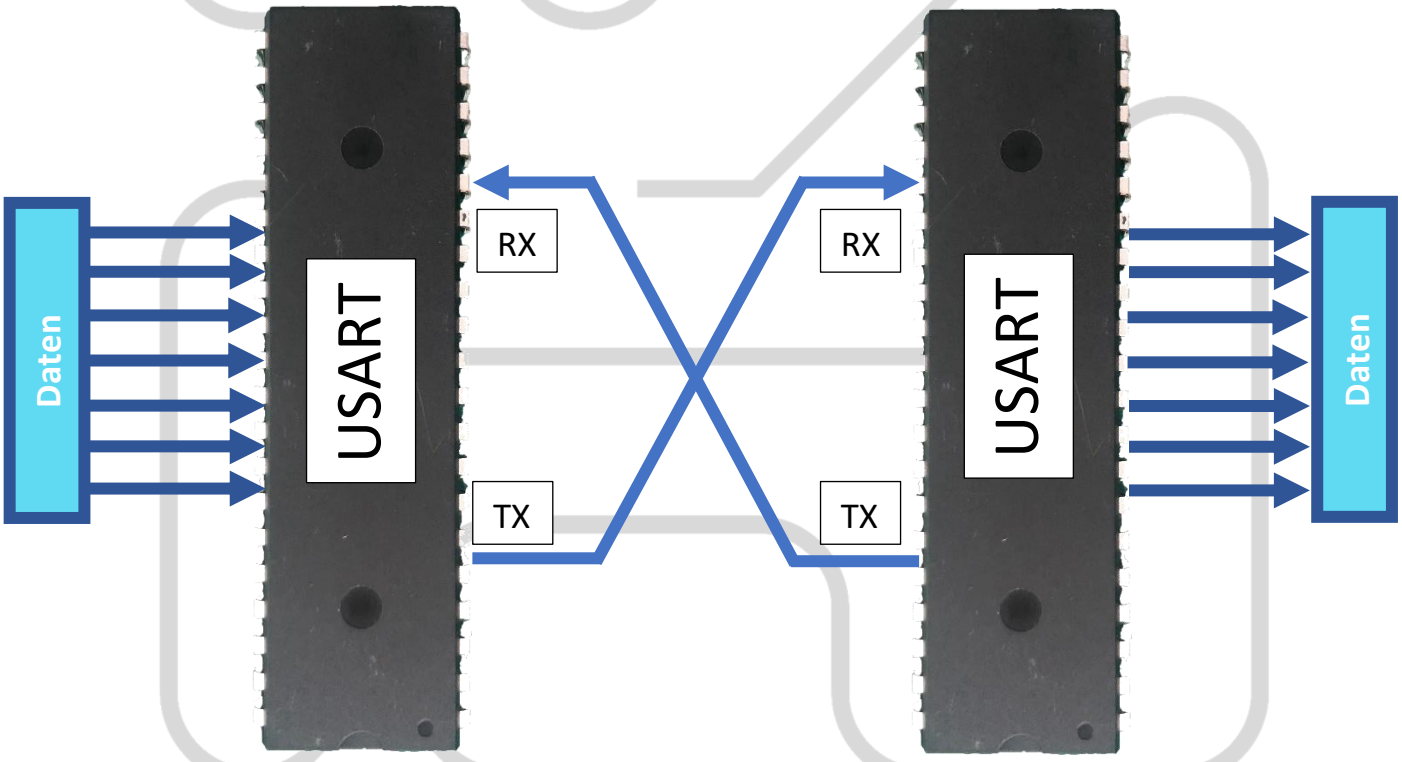


Abbildung 15: Erklärung von USART

6. 3d Drucken

Weil mein Roboter immer nach vorne umgekippt ist, wenn er über eine kleine Kante gefahren ist, habe ich eine Stütze mit dem 3d Drucker gedruckt. Dafür musste ich ein Foto von der unteren Platine in Fusion360 importieren. Damit konnte ich das 3d Modell so zeichnen, dass keine Bauteile oder Lötstellen im Weg waren. Weil mir die Stütze so noch nicht gefiel, habe ich diese ein wenig verziert und «Tesla» darauf geschrieben (Ist ja ein Elektrofahrzeug).

(Abbildung 16)

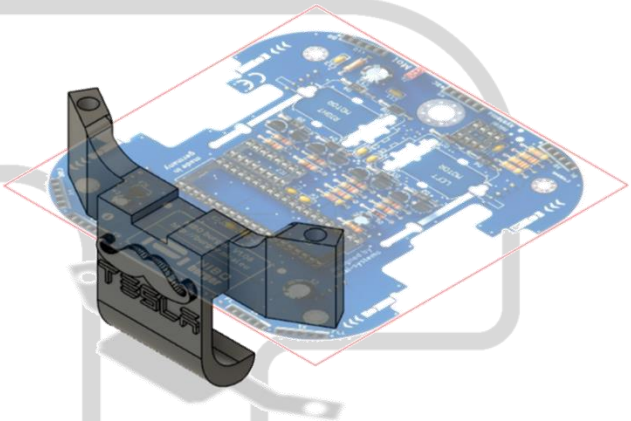


Abbildung 16: 3d Zeichnung

Als Zweites habe ich noch die Ladefläche von einem Tesla Cybertruck ausgedruckt, damit der Roboter kleine Gegenstände transportieren kann.

Auf **Bild 17** sieht man den fertigen Roboter.

Bild 17: Fertiger Roboter

7. Probleme

Beim Zusammenbauen und Programmieren hatte ich ein paar Probleme:

1. Die lade-LED hat nicht geblinkt, wenn der Roboter über USB angeschlossen und die Jumper auf «charge» gesteckt wurden. Das Problem war, dass die Leitung vom ATmega16A zum LED auf GND gezogen war und der Microcontroller darum die LED nicht ausschalten konnte. Das Problem konnte ich beheben, indem ich den Widerstand zwischen Microcontroller und LED aus und wieder einlötete. Ich weiss bis heute nicht, wie die Leitung mit GND verbunden war.

2. Der rote Farbsensor hat das eigene Licht (rot) wahrgenommen und einen gültigen Wert ausgegeben. Bei der Aufgabe ‘Rabbit warren’, wo er aber grünes Licht erkennen sollte, gab der Sensor immer 0 aus. Ich habe versucht, einen anderen Sensor einzubauen, das hat aber nicht funktioniert. Erst als ich die Sensoren nochmals kalibriert habe, hat der Sensor einen gültigen Wert ausgegeben.

3. Ich habe versucht, die Taster auf dem Maroon Shield in mein Programm zu integrieren. Weil es dafür aber keine Dokumentation gibt, musste ich in den libraries nachschauen. Nach langem Suchen habe ich herausgefunden, wie man mit den Tastern LEDs ansteuern kann. Leider ist es aber so, dass die Taster immer wieder ihren Wert senden, auch wenn diese nicht mehr betätigt sind. Darum leuchten die LEDs immer weiter, bis man den anderen Taster ein paar Mal betätigt hat.

Ein paar Tage später habe ich herausgefunden, dass die Taster, nicht wie in der library beschrieben, einen Buchstaben (A, a, B, b) angesteuert wird, sondern mit den Zahlen (65, 97, 66, 98). Diese Zahlen sind die ASCII Zeichen von A, a, B und b.

4. Beim finalen Programm musste ich für die Programmauswahl den Namen des Programms auf dem Maroon Shield anzeigen lassen, damit man das richtige Programm auswählen kann. Während der Name angezeigt wird, habe ich ein delay gemacht, dass der loop den Namen nicht noch einmal anzeigt und es eventuell zu einem Overflow kommen kann. Um zum nächsten Programm wechseln zu können, kann man einen Taster betätigen. Mein Problem war, dass nicht erkannt wurde, dass der Taster gedrückt wurde, bis der delay abgelaufen ist und der loop von vorne begann. Das heisst, dass der ganze Text durchlaufen musste, bis man das nächste Programm auswählen konnte. Für einen

Namen brauchte das Display etwa 7 Sekunden. Im schlimmsten Fall müsste man bei 5 Programmen also 35 Sekunden warten, bis das richtige Programm ausgewählt ist.

Das Problem konnte ich folgendermassen beheben: **(Abbildung 18)**

1. Testen, ob Variable 0 ist
2. Maroon Shield löschen
3. Name wird ans Maroon Shield gesendet
4. Variable wird auf 1 gesetzt
5. Wenn Variable nicht 0 ist, diese bei jedem Durchgang um 1 erhöhen
6. 0.5 Sekunden warten
7. Wenn die Variable einen bestimmten Wert erreicht hat (Delay, bis der Name auf dem Maroon Shield durchgelaufen ist / 0.5 Sekunden * Durchgänge) wird diese wieder auf 0 gesetzt.
8. Beginnt wieder bei 1

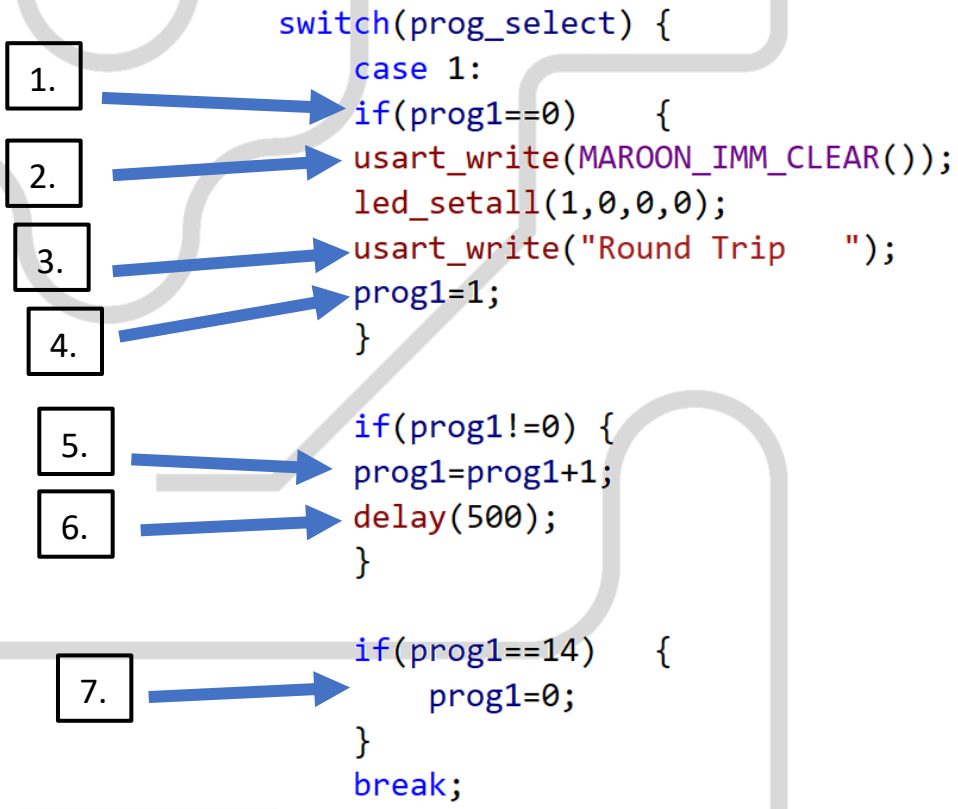


Abbildung 18: Code zum Anzeigen des Programms

8. Schlusswort

Mir hat das NIBO Burger Projekt sehr gut gefallen, weil man selbst etwas ausprobieren konnte und vom Löten bis zum funktionierenden Roboter dabei war. Ich denke, dass mir diese Arbeit sehr gut geholfen hat, die Programmiersprache C besser zu verstehen. Ich finde es auch gut, dass man mit dem Roboter machen kann, was man will. Man kann eigene Programme erstellen oder Zubehör dafür 3d Drucken.