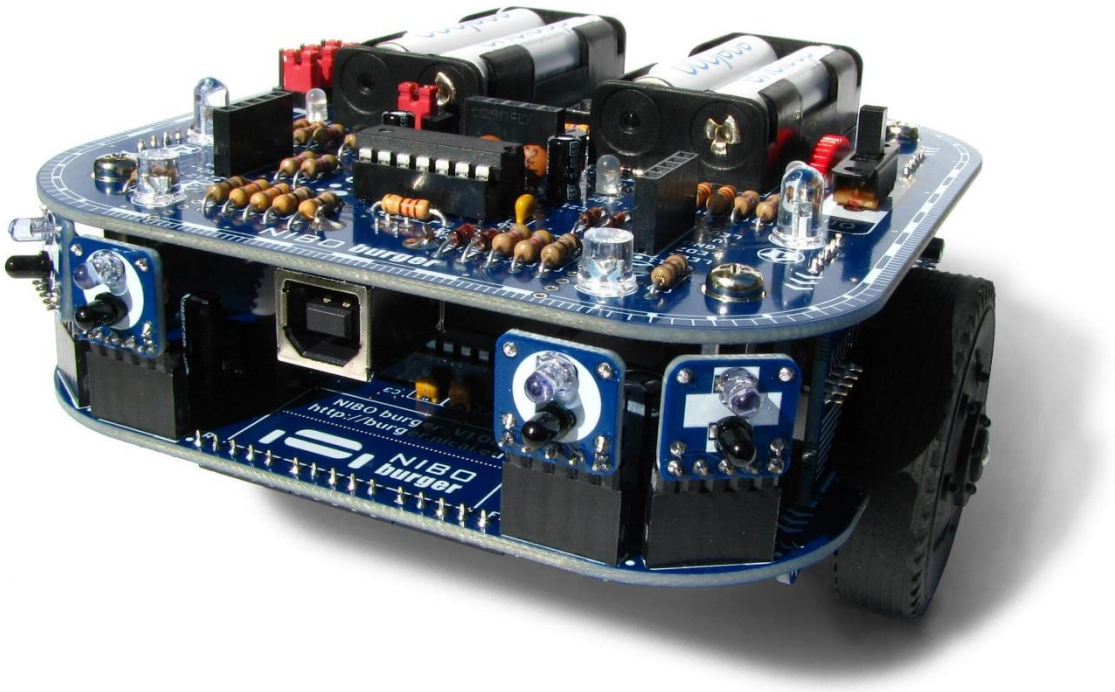
Nibo Burger



Verfasser: Nicolas Diethelm

Pamasol Willi Mäder AG

Driesbüelstrasse 2

CH-8808 Pfäffikon SZ
Schweiz

Tel. +41 55 417 40 40
Fax. +41 55 417 40 44

WWW.Pamasol.com
info@pamasol.com

Inhaltsverzeichnis

Bilder Inhaltsverzeichnis

1.0 Aufbau

1.1.1 Microcontroller[1]

Mikrocontroller oder auch MCU (microcontroller unit) genannt, sind Halbleiterchips welche Prozessor und Peripheriefunktionen enthalten. In vielen Mikrocontrollern befindet sich zugleich auch Arbeits- wie Programmspeicher. (Bild 1)

Peripheriefunktionen sind Funktionen, die nicht vom Prozessorkern zur Verfügung gestellt werden, sondern von zusätzlicher Hardware.

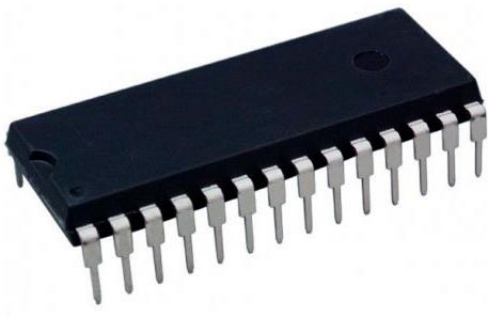


Bild 1: Microcontroller

1.1.2 ATmega16A-PU[2]

Das Gehirn des Nibo Burger ist ein ATmega16A-PU. Der Microcontroller besitzt einen Flash-EEPROM von 16K Bytes, einen EEPROM Speicher von 512 Bytes und einen SRAM Speicher von 1K Byte. Der Microcontroller besitzt eine 8-Bit Architektur, das heisst der Microcontroller kann in einem Takt 8-Bits gleichzeitig verarbeiten. Er kann mit einem Takt von 0-16MHz betrieben werden. Wir betreiben ihn mit einem Separaten Quarzoszillator der einen Takt von 15MHz angibt. Der Oszillator wird beim Nibo Burger an den 13 Pin (XTAL1) angeschlossen. Er hat zwei eingebaute 8Bit und einen 16Bit Timer/Zähler. Der ATmega16 hat 40 Pins, davon sind 32 Pins digitale Ein- oder Ausgangs Pins und 4 von den 32 Pins sind PWM Ausgänge. (Bild 2)

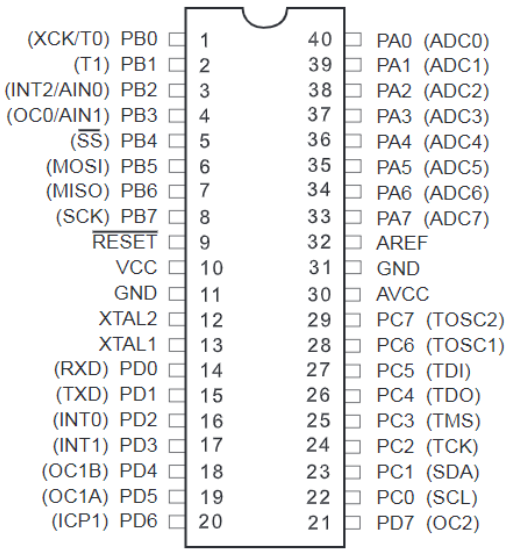


Bild 2: ATmega16a-PU Pin Layout

[1]https://de.wikipedia.org/wiki/Mikrocontroller#:~:text=Als%20Mikrocontroller%20(auch%20%C2%B5Controller%2C%20%C2%B5C,oder%20komplett%20auf%20demselben%20Chip.

[2] https://www.alldatasheet.com/datasheet-pdf/pdf/313643/ATMEL/ATmega16A-PU.html

1.1.3 EEPROM und Flash-EEPROM

EEPROM = Electric Erasable Programmable Read Only Memory

Bei EEPROMs kann der Speicherinhalt durch elektrische Signale gelöscht und anschliessend neu programmiert werden. Es gibt EEPROMs die in der Schaltung verweilen und solche die man zum programmieren aus der Schaltung entnehmen muss. (Oft zu finden in Steuerprogrammierbaren Steuerungen, SPS). (Bild 3)

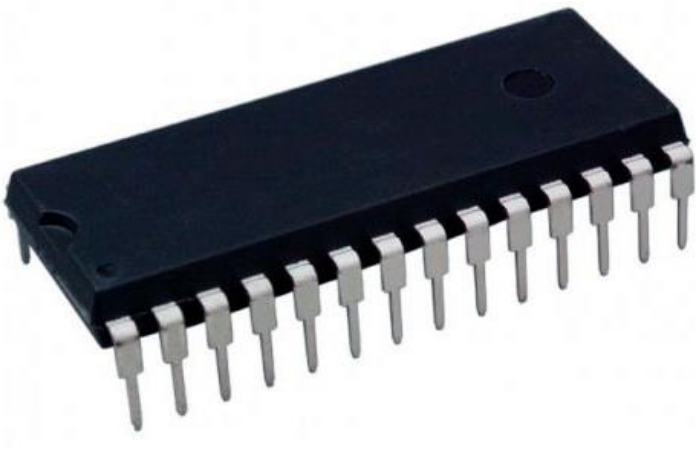


Bild 3: EEPROM

Flash–EEPROM = Flash Erasable Programmable Read Only Memory

Flash- EEPROMs können beliebig oft gelöscht und neu beschreiben werden. Bei Flash- EEPROMs erfolgt der Lese- oder Schreibvorgang über eine Serielle Schnittstelle. (Bild 4)

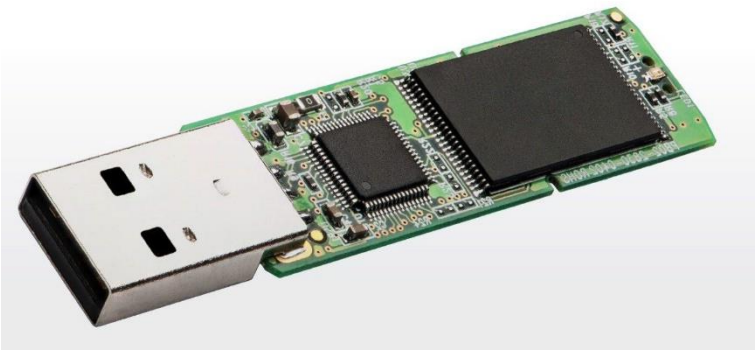


Bild 4: Flash-EEPROM

1.2.1 3,3 Volt[3]

Der Nibo Burger wird mit 4 mal 1,2 Volt Akkus betrieben, jedoch dürfen die Sensoren nur mit 3,3 Volt betrieben werden. Hier kommt jetzt der IC3 ins Spiel, der IC3 ist ein TS2950CT33, ein Spannungsregulator (Bild 5). Er reguliert die Spannung runter auf 3,3 Volt, jedoch gibt es beim runter Regeln einen Verlust, der in Wärme verloren geht.

Bild 5: TS2950CT33

1.3.1 IR-Sensor[4]

Infrarot Sensoren oder IR-Sensoren sind strahlungsentfindliche optoelektronische Bauelemente, die infrarot Strahlung erfassen können. Ihre spektrale impfindlichkeit im infraroten Wellenbereich liegt bei 780 nm bis 50 µm. Die IR-Sensoren haben einen Sender und einen Empfänger. Der Sender strahlt Infrarotstrahlung aus, ist ein Objekt vor dem Sensor, werden die Infrarotstrahlen reflektiert und vom Empfänger empfangen. (Bild 6)

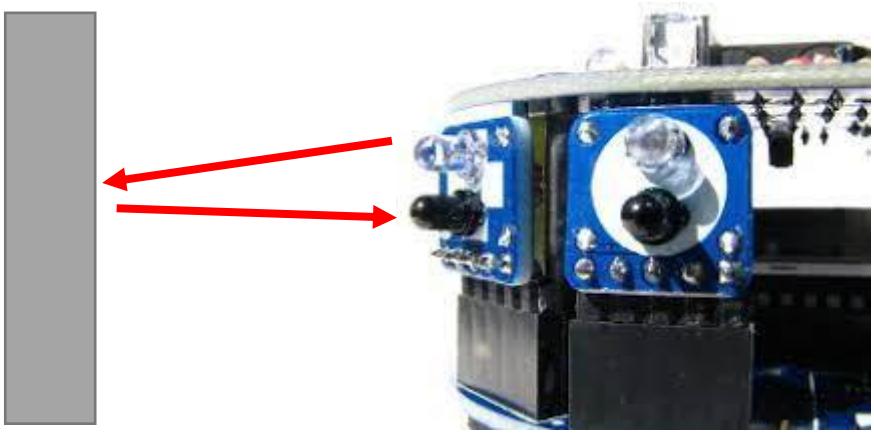


Bild 6: IR-Sensor Reflektion

[3] https://www.mouser.de/datasheet/2/395/TS2950_F14-1918547.pdf

[4] https://www.infratec-infrarot.ch/sensorik/service-support/glossar/infrarotsensor/

1.4.1 Moaroon Shield[5]

Auf dem Nibo Burger haben wir ein Maroon Shield (8x8 Matrix), jedoch haben wir nur einen Microcontroller der 16 Ausgänge hat, um das Maroon Shield zu steuern. Um nicht jedes LED einzeln ansteuern zu müssen, unterteilt man das Matrix Display in Reihen (Bild 7). Will man z.B. das erste LED oben links ansteuern, muss man die Reihen A und 1 schalten. Es gibt aber ein Problem, wenn man jetzt die Reihe B ansteuern will, überlappen die Signale von der Reihe A. Für dieses Problem gibt es das Multiplexverfahren. Das Multiplexverfahren ist, dass nur eine Reihe auf einmal geschaltet wird, nach einer Millisekunde wird die vorherige Reihe wieder ausgeschaltet und die nächste beschalten (Bild 8). Für das menschliche Auge ist Beschaltung der Reihen zu schnell um es wahrzunehmen, wir sehen nur dass die LEDs dauerhaft eingeschaltet sind. Um die Beschaltung wahrzunehmen muss sie unter 14 mal pro Sekunde sein, erst dann kann das menschliche Auge die Beschaltung wahrnehmen.

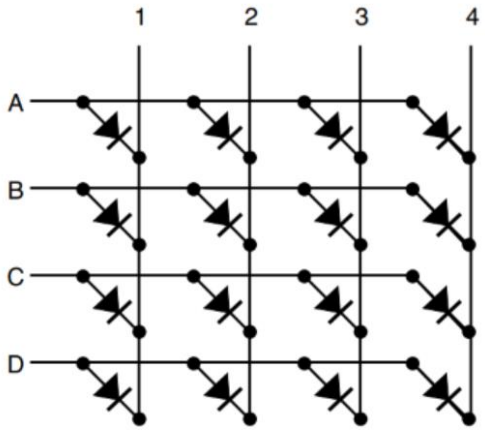


Bild 7: 4x4 Matrix

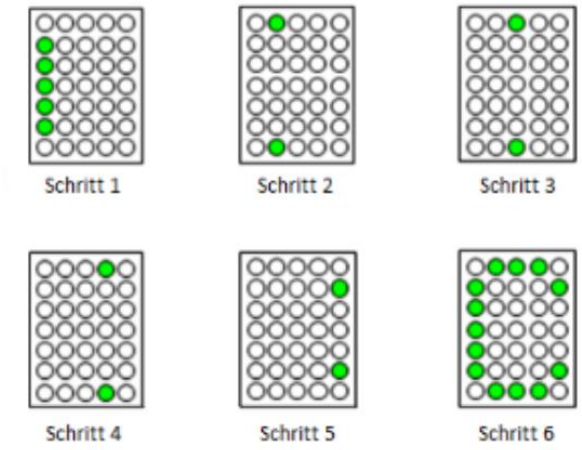


Bild 8: Matrix Multiplexverfahren

[5] https://de.wikipedia.org/wiki/Multiplexverfahren

https://de.wikipedia.org/wiki/Bildfrequenz#:~:text=Das%20menschliche%20Auge%20nimmt%20bei,auf%2016%20Bilder%20pro%20Sekunde

2.0 Programmieren

2.1.1 Programmierungsumgebung[6]

Zur Programmierung des Nibo Burgers haben wir Microchip Studio verwendet (Bild 9). Microchip Studio ist ein gratis Programm zum programmieren von Geräten, die AVR oder SAM unterstützen. In Microchip Studio kann man in C oder C++ schreiben. Für die Programmierung des Nibo Burgers haben wir den WinAVR Compiler verwendet, da der ATmega16A-PU zur AVR Familie gehört. Der WinAVR Compiler ist ein open Source Entwiklertool für die Atmel AVR Microchip Serie.

Bild 9: Microchip Studio

2.2.1 If else

Das if-Statement funktioniert, indem man zuerst eine Kondition bestimmt, ab wann das if-Statement starten soll (Bild 10). Ist die Kondition richtig wird der Code in der if-Schleife ausgeführt. Ist die Kondition falsch wird das else-Statement ausgeführt (Bild 11).

```
if (Value == 2)
{
    Value++;
}else
{
    Value=2;
}
```

Bild 10: if-Statement

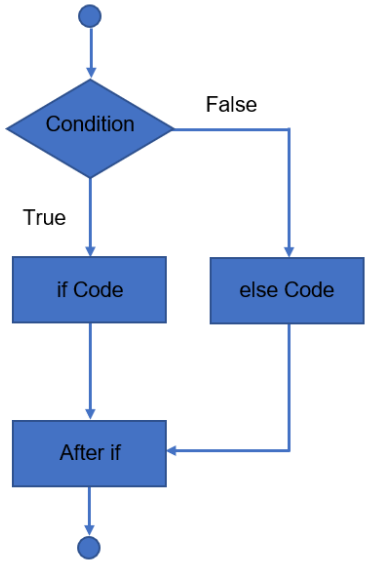


Bild 11: if-Statement Ablauf

[6] https://www.microchip.com/en-us/tools-resources/develop/microchip-studio#

http://winavr.sourceforge.net

2.2.2 For-Schleife

Eine for-Schleife funktioniert so, wenn die Kondition falsch ist, wiederholt sie ihren Mittelteil immer wieder bis die Kondition erfüllt ist (Bild 12). Am Ende eines Wiederholdurchgang wird die Zählerzahl meist inkrementiert oder dekrementiert. Bei der for-Schleife wird zuerst eine Zählervariabel definiert, die nur in der for-Schleife existiert. Als zweites wird definiert ab was für eine Kondition die for-Schleife aufhören soll sich zu wiederholen. Als letzter Schritt wird definiert, was mit der Zähler Zahl passiert, wenn die for-Schleife eine Wiederholung gemacht hat (Bild 13).

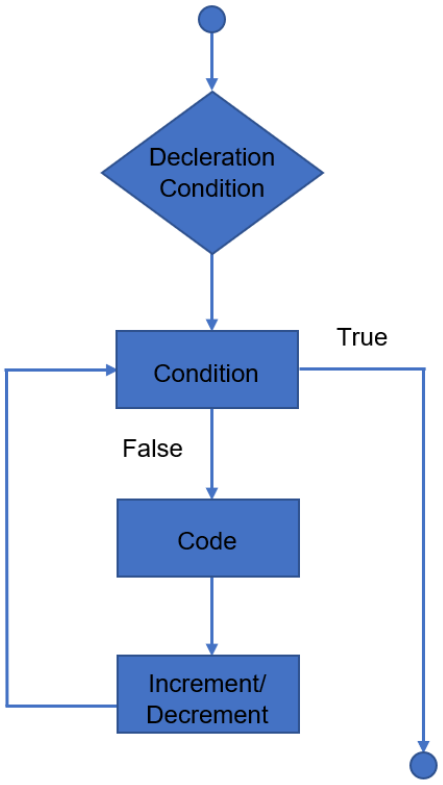


Bild 12: for-Schleife Ablauf

```
for (int i=0; i<10; i++)
{
    led_set(1,1);
    delay(200);
    led_set(1,0);
    delay(200);
}
```

Bild 13: for-Schleife Beispiel

2.3.1 Funktion

Eine Funktion ist ein Programmblock den man selber schreiben kann. Die Funktion muss man zuerst definiert haben, bevor man sie das erste Mal aufruft. Man kann z.b. eine Funktion für den Motorenstart machen, idem man den Speed für beide Motoren angiebt. Im void Loop kann man diese Funktion dann aufrufen mit der entsprechenden geschwindigkeiten der Motoren (Bild 14).

Eine einfache Funktion dekrariert man mit void, danach kommt der Namen der Funktion, der frei wählbar ist, in der Klammer kann man variabeln definieren die danach in den Codeblock der Funktion eingesetzt werden.

```
void MotorStart(int speed)
{
    motpwm_setLeft(speed);
    motpwm_setRight(speed);
}


void loop()
{
    MotorStart(1023);
}
```

Bild 14: Funktion Beispiel

2.3.2 Funktion mit Rückgabewert

Man kann in einer Funktion auch einen Wert zurückgeben lassen, hier kommt die Funktion mit Rückgabewert ins Spiel. Diese Funktion kann man z.B. für Berechnungen brauchen, die anschliessend den gerechneten Wert zurückgibt (Bild 15).

```
int MotorSpeed(int speed)
{
    int Res=0;
    Res=speed-200;
    return(Res);
}


void loop()
{
    int Speed;
    Speed=MotorSpeed(1023);
    motpid_setSpeed(Speed);
}
```

Bild 15: Funktion mit Rückgabewert Beispiel

3.0 Persönliche Erfahrung

3.1.1 Aufbau

Bei Zusammenbau des Nibo Burgers hatte ich keine grossen Schwierigkeiten.

Das einzige Problem was beim Zusammenbau aufgetreten war, ist dass eines der Roten Zahnräder, einen minimal zu grossen Lochdurchmesser hatte, somit hielt das Zahnrad nicht auf der Achse und rutschte immer wieder ab(Bild 16). Um das Problem zu lösen, klebte ich das Zahnrad auf die Achse.

Ein hilfreicher Tipp beim Zusammenbau ist, die Teile nach dem Überprüfen auf ein Blatt zu kleben und in Sektionen aufzuteilen (Bild 17). So kann man die Bauteile leichter finden und es ist schwerer sie zu verlieren.

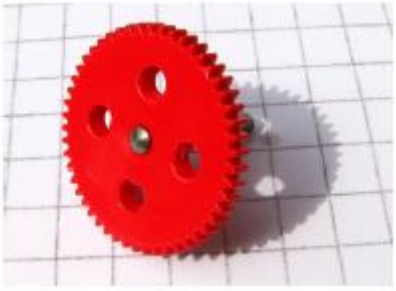


Bild 16: Zahnrad

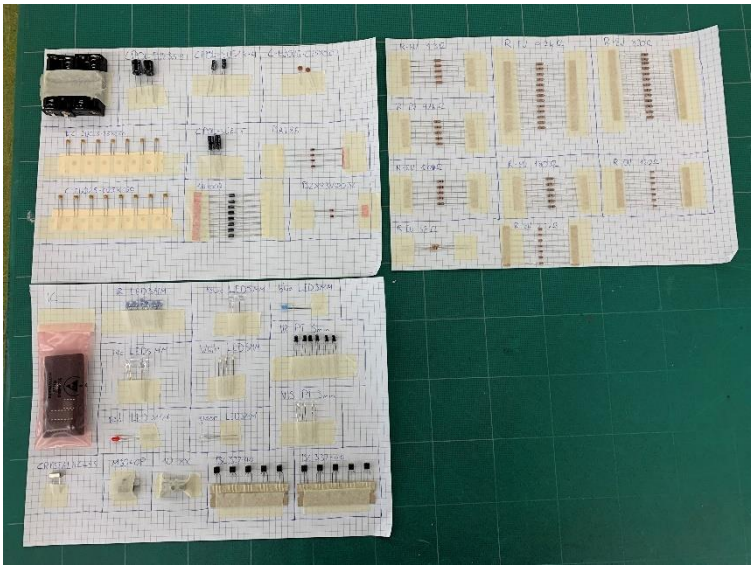


Bild 17: Überblick Tipp

3.1.2 Programmieren

Beim Programmieren des Nibo Burgers gab es schon mehr Schwierigkeiten.

Die grössten Probleme gab es bei den Odometrie-Sensoren. Weil ich das schnellere Getriebe eingebaut habe, ist der Nibo Burger bei positiver Geschwindigkeit rückwärts gefahren. Darum musste ich die Geschwindigkeit im negativen Bereich eingeben, dass er vorwärts fährt.

Bei den Odometrie-Sensoren spielt die Richtung der Motoren eine Rolle. Da ich dachte die dass die Richtung egal sei, schrieb ich das ganze Programm mit positiven Werten, kommend von den Odometrie-Sensoren.

Nach einer Zeit fand ich selber heraus, dass die Werte in den Minusbereich gezählt werden. Später schaute ich noch in der Library nach und es war genau beschrieben, wie die Odometrie-Sensoren funktionierten (Bild 18).

Für das nächste Mal werde ich von Anfang an in den Librarys nachschauen, wenn ich etwas nicht weiss, wie es funktioniert.

Bild 18: Nibo Burger Library

4.0 Schlusswort

Ich habe durch das Nibo Burger Lehrlingsprojekt nicht nur viel Neues dazugelernt, sondern es hat auch viel Spass gemacht. Ich konnte durch dieses Projekt ein besseren Einblick in die Mikroelektronik erlangen. Zudem lernte ich wie man mit Git Bash und mit C umgeht.

Im Grossen und Ganzen konnte ich viel von dem Lehrlingsprojekt mitnehmen. Das Beste an dem Projekt fand ich, dass man eigene Ideen ausprobieren und umsetzen konnte.

4.0 Quellenverzeichnis

Bild 1:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fq.dota2.mk%2Fbestsellers%3D48062%2F&psig=AOvVaw2r57USfQ-jlXq4BAoxYtTY&ust=1653982567297000&source=images&cd=vfe&ved=0CAwQjRxqFwoTCNDQuojphvgCFQAAAAAdAAAAABAH

Bild 2:
https://www.alldatasheet.com/datasheet-pdf/pdf/313643/ATMEL/ATmega16A-PU.htm

Bild 3:
https://www.google.ch/aclk?sa=l&ai=DChcSEwjf8LTjn4f4AhUfkGgJHQJnAJ8YABBDGgJ3Zg&sig=AOD64_1zXI0PN-QpD8GaLVrNrFUdcVy1WQ&adurl&ctype=5&ved=2ahUKEwiPhKDjn4f4AhVPYxoKHQR2D3EQuxd6BQgBEJAE

Bild 4:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.pcwelt.de%2Fratgeber%2FDer-perfekte-Speicher-USB-Sticks-sind-klein-und-portabel-9029951.html&psig=AOvVaw1hvMLTfVd7J4cmUpB7qfh_&ust=1653986051392000&source=images&cd=vfe&ved=0CAwQjRxqFwoTCIiPwq7ohvgCFQAAAAAdAAAAABAE

Bild 5:
https://github.com/pamasol/Lehrlingsprojekt-Nibo-Burger/files/3652506/Doku_NIBOburger_20150909.pdf

Bild 6:
https://www.google.com/url?sa=i&url=http%3A%2F%2Fwww.nicai-systems.com%2Fde%2Fnibo-burger-spezifikationen&psig=AOvVaw2ewdgZi0hAuqkELCXGPgBM&ust=1653986555131000&source=images&cd=vfe&ved=0CAwQjRxqFwoTCNiG6J7qhvgCFQAAAAAdAAAAABAE

Bild 7:
https://github.com/pamasol/Lehrlingsprojekt-Nibo-Burger/wiki/8x8-pixel-matrix-display

Bild 9:
https://www.microchip.com/en-us/tools-resources/develop/microchip-studio

Bild 16:
https://github.com/pamasol/Lehrlingsprojekt-Nibo-Burger/files/3652506/Doku_NIBOburger_20150909.pdf

Bild 18:
https://docs.roboter.cc/niborobolib-3.6/niboburger/html/odometry_8h.html