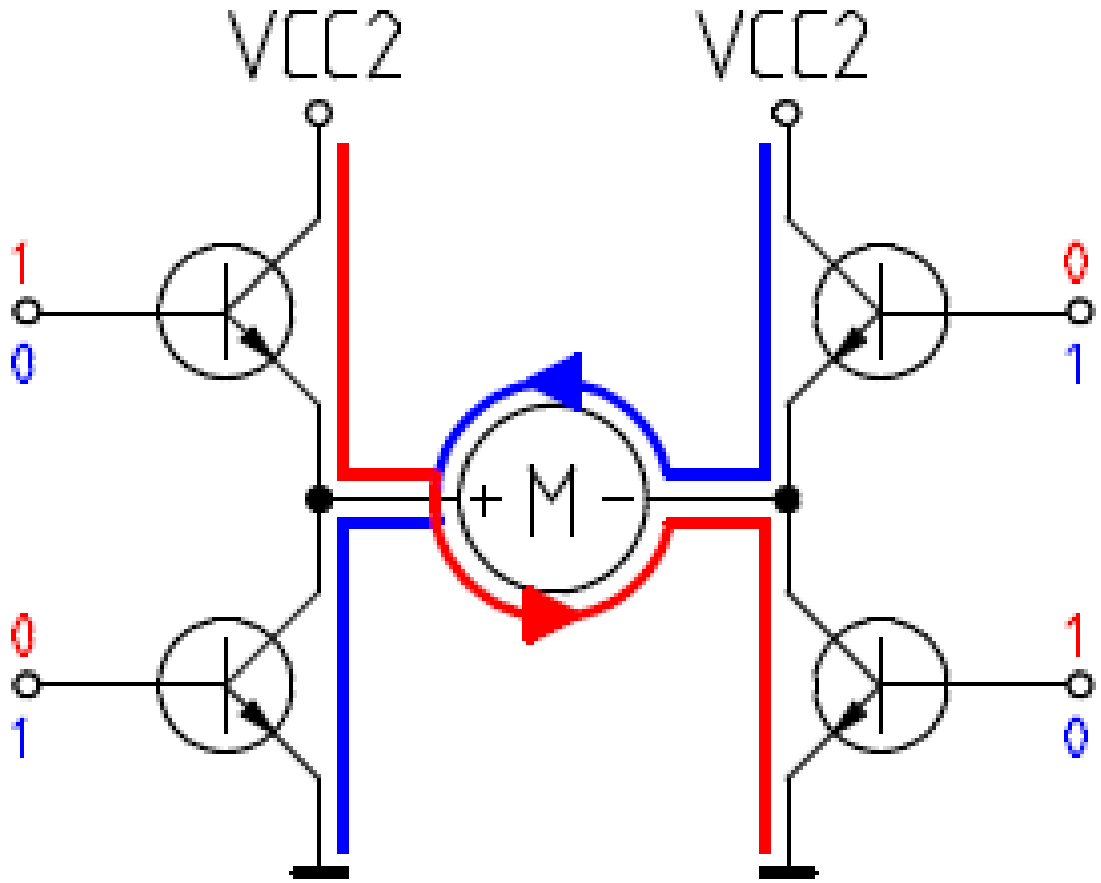
H-Brücke

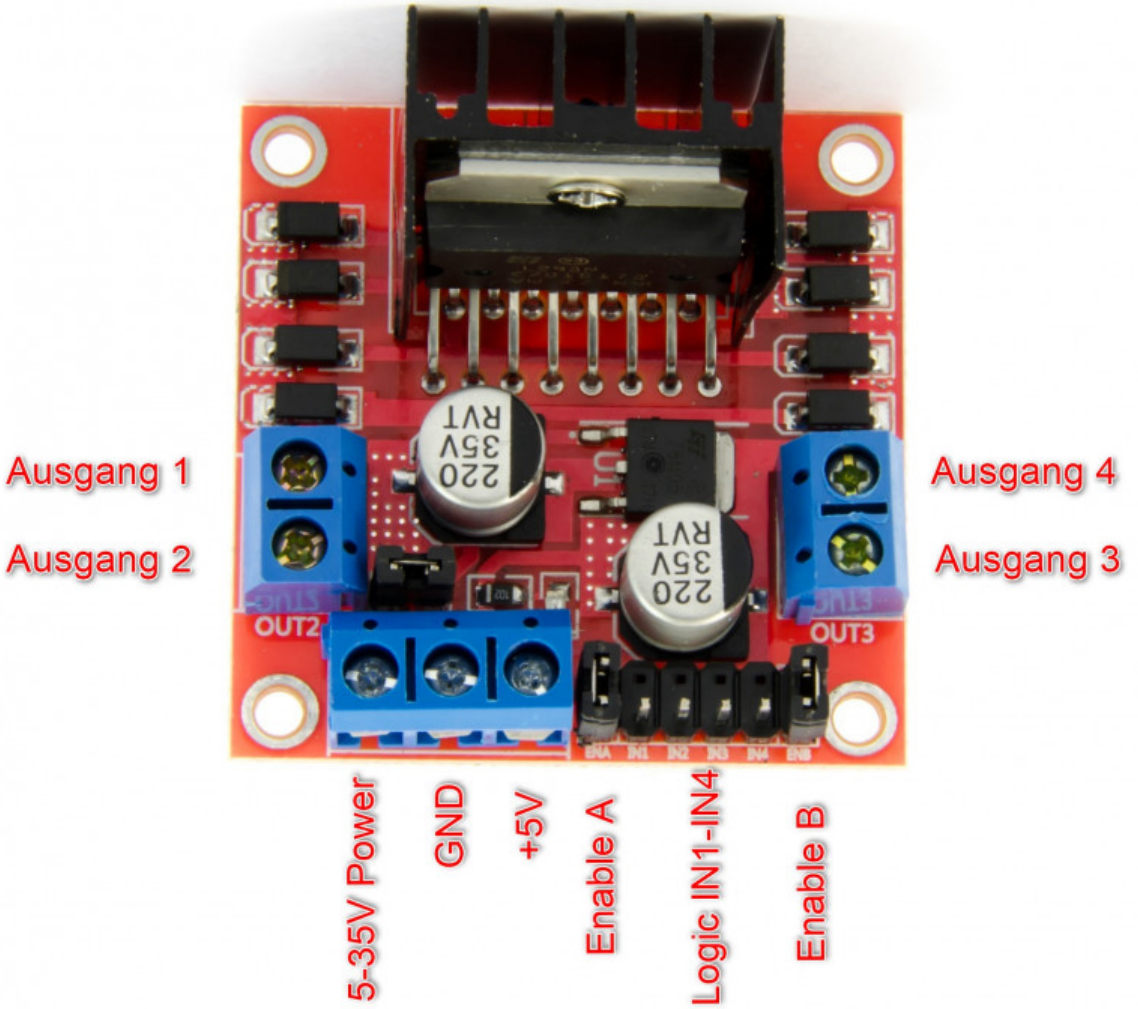
Eine Brückenschaltung – auch H-Schaltung, H-Brücke oder Vollbrücke genannt – ist eine elektrische Schaltung, bei der in der Grundform fünf Zweipole in Form des Großbuchstabens H zusammengeschaltet sind. Die Querverbindung heißt Brückenzweig.



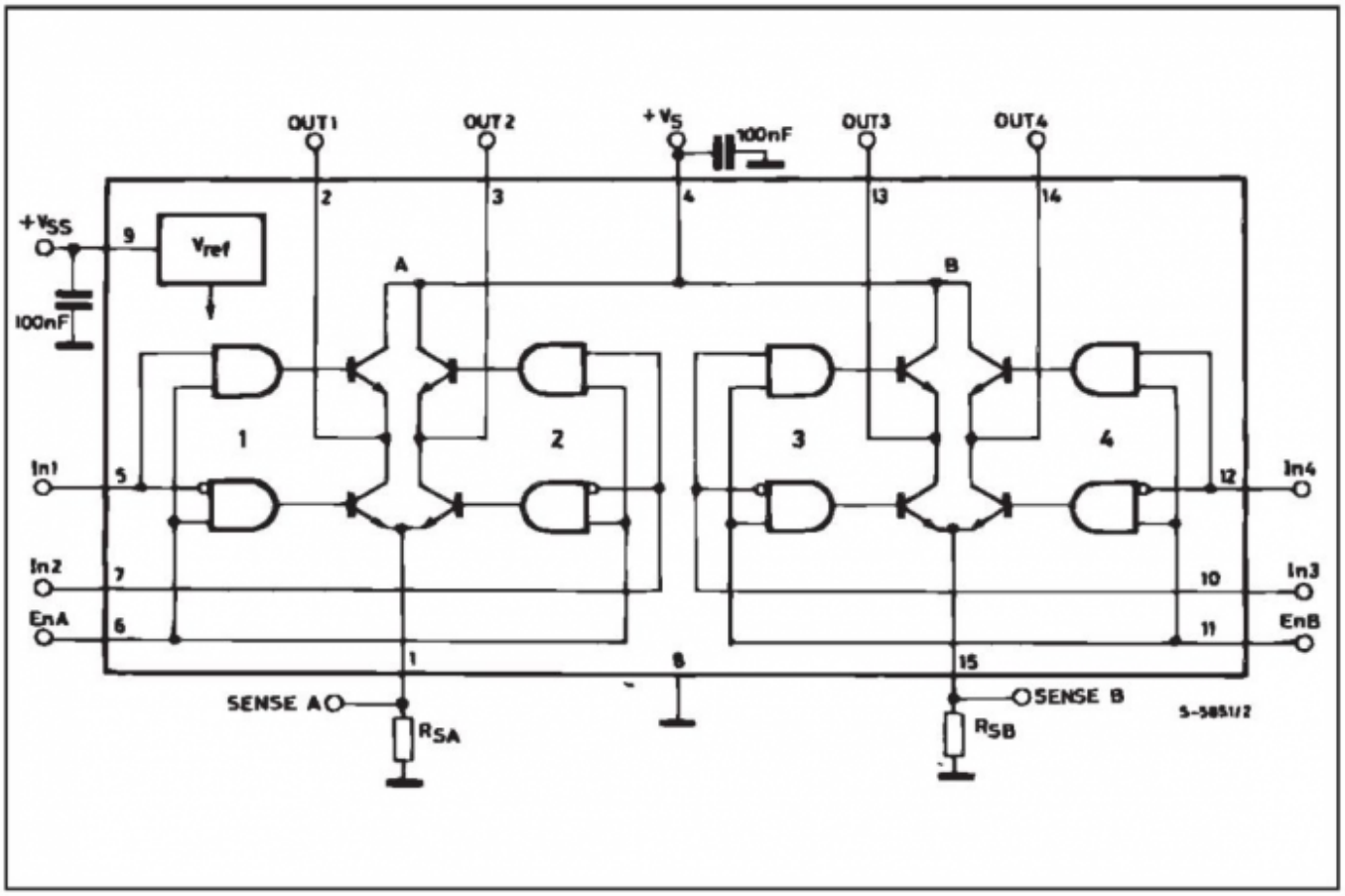
Mithilfe von einer H-Brücke **kann man die Stromrichtung beim Ausgang selbst bestimmen**. Das heißt, schließt man als Ausgang einen Motor an, so dreht sich der je nach Eingänge nach links oder nach rechts. Dazu muss man nur wissen, wie man die Eingänge entsprechend belegen muss. Dazu wiederum muss man zuerst verstehen, wie eine H-Brücke funktioniert. In den nachstehenden Bildern wird die Funktionsweise der H-Brücke erläutert.

Doppelte H-Brücke (L298N)

Mit einer doppelten H-Brücke kann man nicht nur bei einem Ausgang die Stromrichtung bestimmen sondern bei 2 Ausgängen.



Schaltplan (L298N)



Funktionsweise (L298N)

Die H-Brücke (L298N) ist eine doppelte H-Brücke. Das heißt, mit diesem Bauteil kann man die Stromrichtung von 2 Ausgängen (z.B. 2 Motoren) beeinflussen.

Der Bauteil besteht aus insgesamt

- **6 logischen Eingängen**
  - 3 für Motor A (**In1, In2, EnA**)
  - 3 für Motor B (**In3, In4, EnB**)
- **4 Ausgängen**
  - 2 für Motor A (**OUT1, OUT2**)
  - 2 für Motor B (**OUT3, OUT4**)
- **1 Eingang (5V bzw. 12V) bzw. Ausgang (GND) für die Stromversorgung**

Des Weiteren sind in der doppelten H-Brücke (L298N) noch

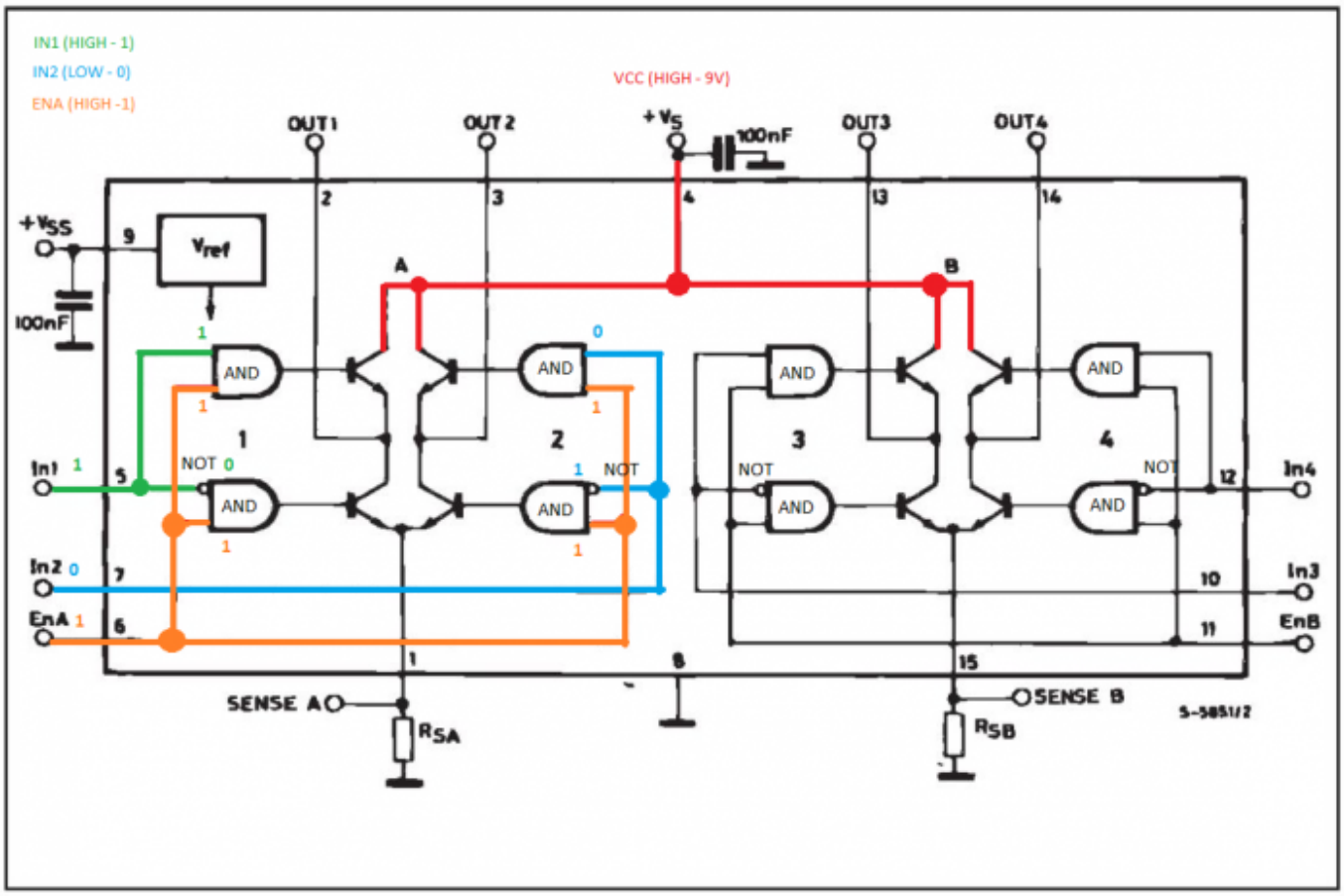
- 8 Transistoren
- 8 logische UND-Gatter bzw.
- 4 NOT-Gatter verbaut.

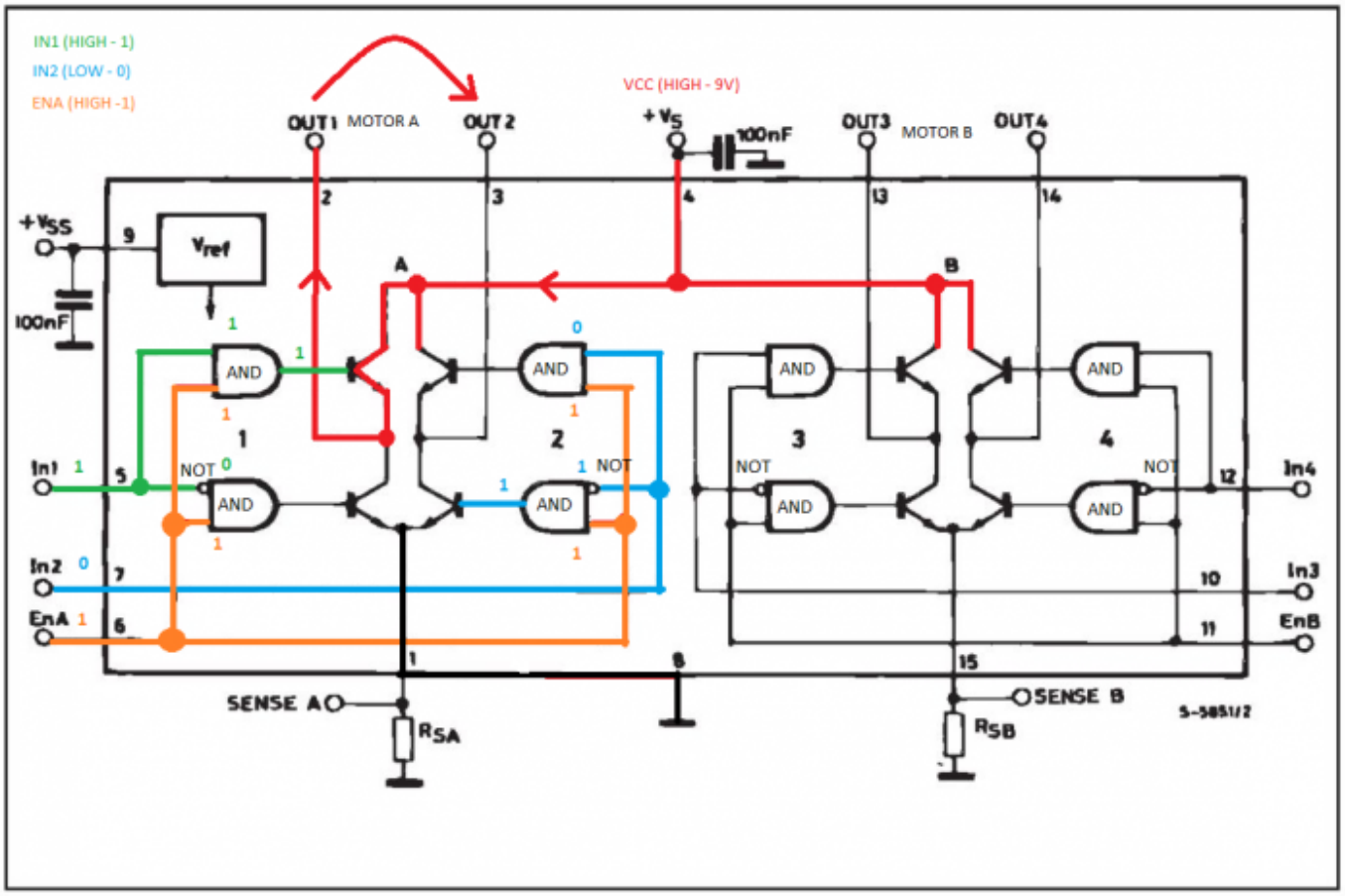
Beispiel

Betrachten wir nun mal eine Brücke und deren Eingänge bzw. Ausgänge.

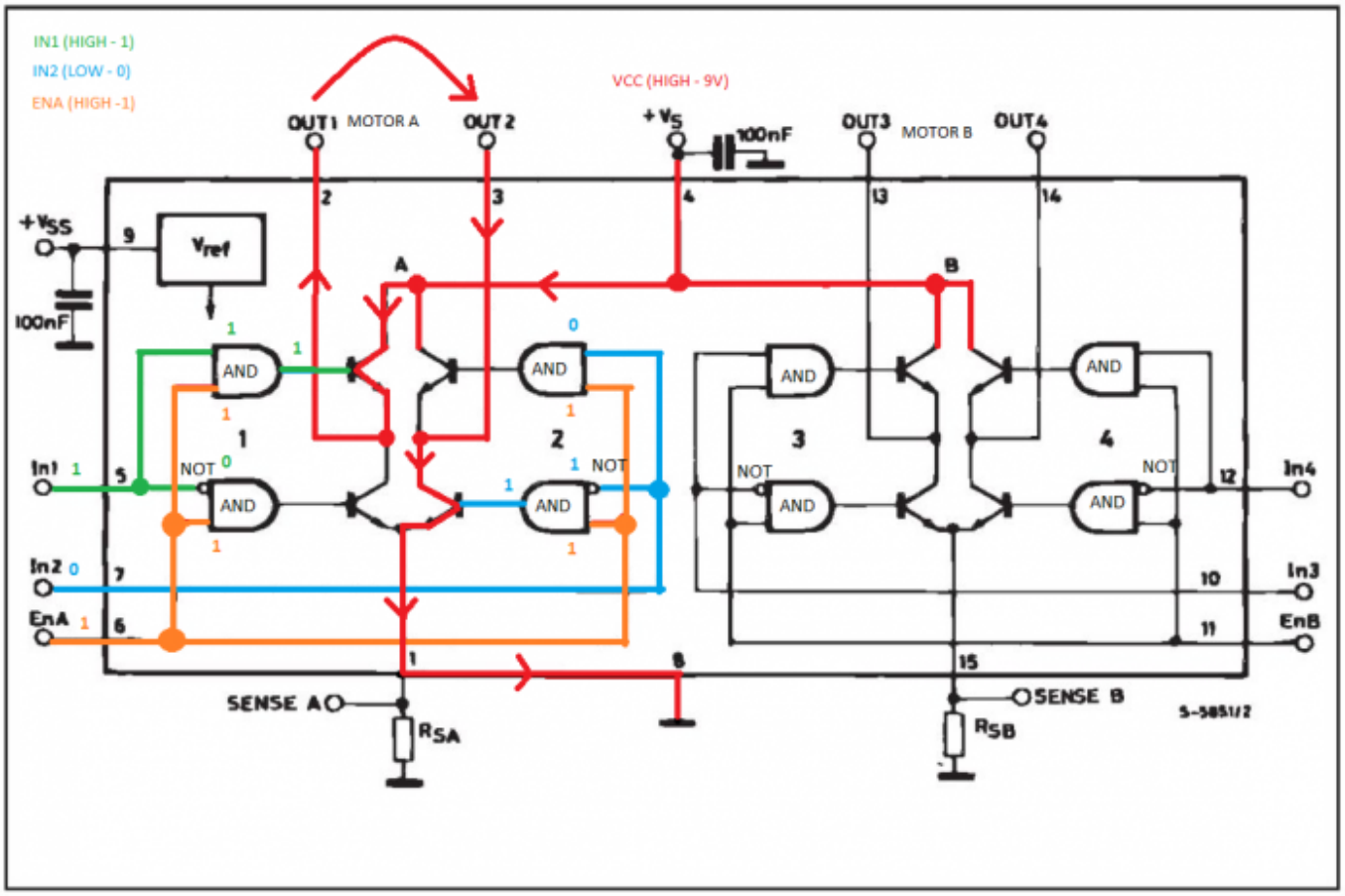
1. An den Eingängen **In1** und **EnA** wird eine **hohe Spannung (HIGH - ~5V)** angelegt.
2. An dem Eingang **In2** wird eine **niedrige Spannung (LOW - ~0V)** angelegt.

Somit ergibt sich folgendes Schaltbild:





Die oben angeführte Belegung der Eingänge (In1, In2 & EnA) hat zur Folge, dass sich der Motor aufgrund der Stromrichtung von OUT1 zu OUT2 von links nach rechts dreht. Würde man die In1 und In2 jeweils umgekehrt belegen, so würde sich der Motor von rechts nach links drehen.



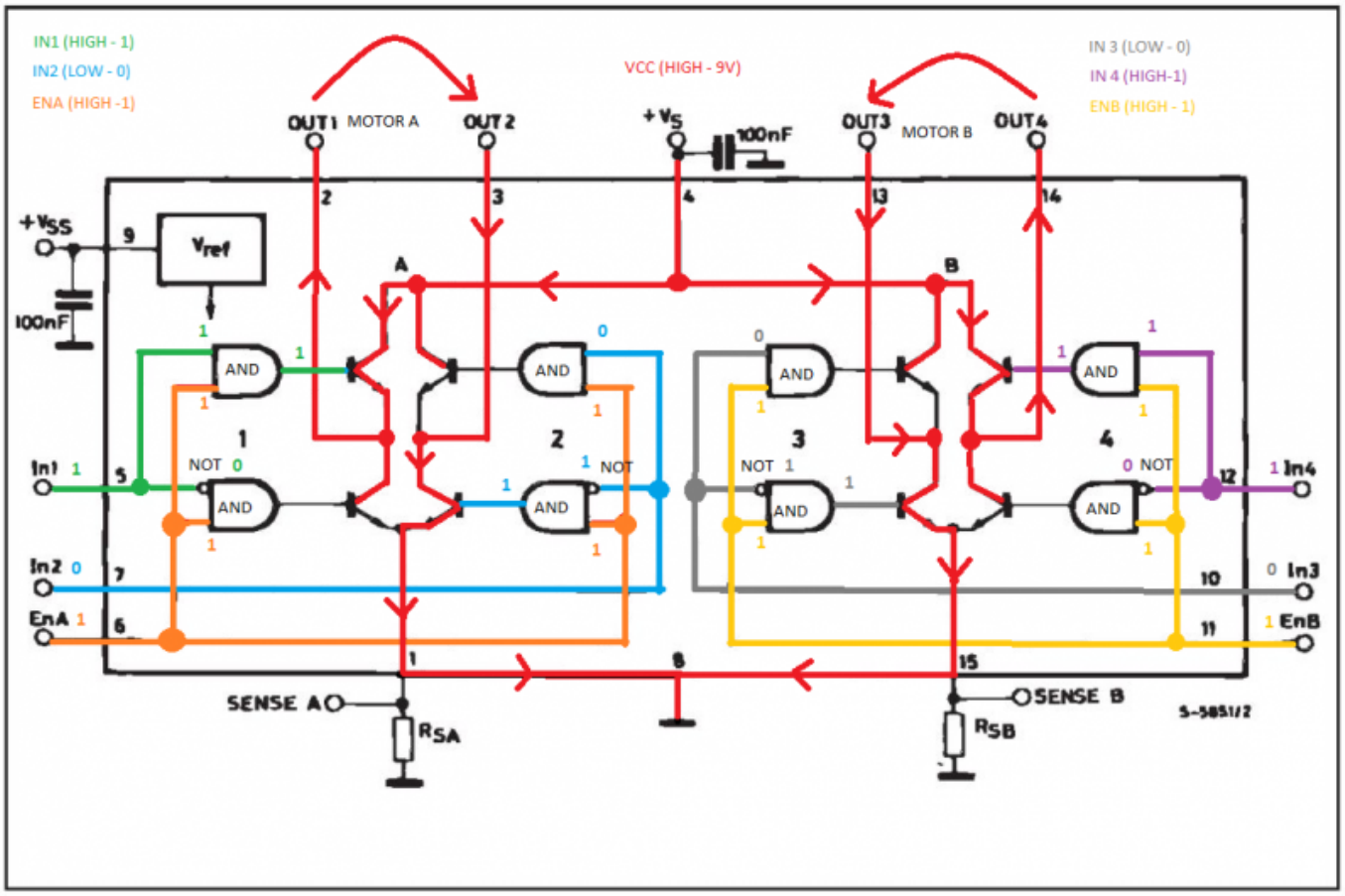
Beispiel

Motor A

1. An den Eingängen **In1** und **EnA** wird eine **hohe Spannung (HIGH - ~5V)** angelegt.
2. An dem Eingang **In2** wird eine **niedrige Spannung (LOW - ~0V)** angelegt.

Motor B

1. An den Eingängen **In4** und **EnA** wird eine **hohe Spannung (HIGH - ~5V)** angelegt.
2. An dem Eingang **In3** wird eine **niedrige Spannung (LOW - ~0V)** angelegt.



Ergebnis

Die oben angeführte Belegung der Eingänge hat zur Folge, dass sich der **Motor A von links nach rechts** bzw. der **Motor B von rechts nach links** dreht.

Schalttabelle (L298N) für MotorA

EnableA	In1	In2	Wirkung
0	0	0	kein Strom fließt - Motor läuft aus
0	1	0	kein Strom fließt - Motor läuft aus
0	0	1	kein Strom fließt - Motor läuft aus
0	1	1	kein Strom fließt - Motor läuft aus
1	0	0	kein Strom fließt - Motor läuft aus
1	0	1	Motor dreht sich von rechts (OUT2) nach links (OUT1)
1	1	0	Motor dreht sich von links (OUT1) nach rechts (OUT2)
1	1	1	Kurzschluss Motor - Motor stoppt

Programmcodes

Funktionen

Wie in anderen Programmiersprachen gibt es auch in Python die Möglichkeit Funktionen bzw. Unterprogramme zu schreiben.
Diese Funktionsblöcke werden nicht mit Klammern geöffnet bzw. geschlossen, sondern durch einen Doppelpunkt „:“ bzw. durch Einrückungen.
Besonders ist, dass jede Funktion mittels dem Schlüsselwort def definiert werden muss.

Beispiel test.py

```
#!/usr/bin/python

def add(a,b):
        return a+b

#MAIN PROGRAMM
if __name__ == "__main__":
        print add(10,5)
```

Man könnte nun eine zweites Python Programm schreiben, in dem man die Funktion add(a,b) von test.py aufruft.

Dafür gibt es nun 2 verschiedene Varianten

Variante 1 - test2.py

```
#!/usr/bin/python

from test import *

if __name__ == "__main__":
        print(add(10,39))
```

Variante 2 - test3.py

```
#!/usr/bin/python

import test

if __name__ == "__main__":
        print(test.add(20,30))
```

Für unser Projekt empfehle ich zwei Python-Programme

- 1 Programm für die Tastensteuerung (Also, wann wurde welche Taste gedrückt??) (z.B.: Robot.py)
- 1 Programm für die Ansteuerung des Motoren A und B (z.B.: Motor.py)

Motor.py

Im Python Programm Motor.py muss die Ansteuerung der Motoren mittels GPIO erfolgen. Dafür muss man zuerst die Bibliothek RPi.GPIO einbinden.

Einbinden der Bibliothek

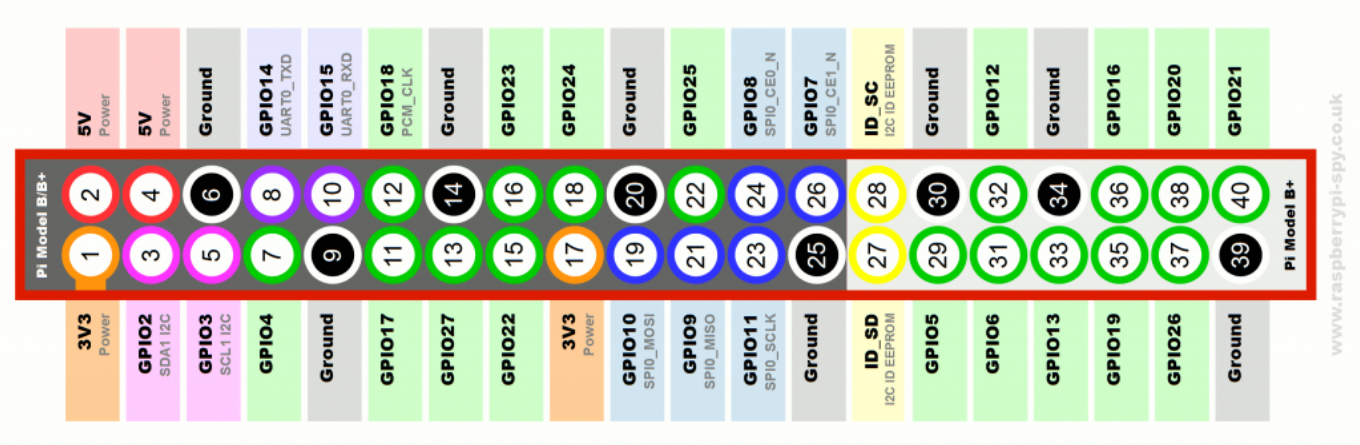
```
import RPi.GPIO as GPIO
```

Ansteuerungsmodus der GPIO Schnittstelle

Anschließend kann man bereits die GPIO-Codes im Programm definieren und ansteuern. Dafür legt man zuerst fest, ob man die GPIO-Schnittstelle mit den

- GPIO Nummern (**GPIO.BCM**) also **4,17 bzw. 27** für die GPIO4, GPIO17 bzw. GPIO27 oder
- GPIO Board Number (**GPIO.BOARD**) also **7,11 bzw. 13** für die GPIO4, GPIO17 bzw. GPIO27

ansteuern will.




```
GPIO.setmode(GPIO.BCM)

#oder
#GPIO.setmode(GPIO.BOARD)
```

GPIO Setup (Input or Output)

Als nächstes muss man in Python bekanntgeben, ob die GPIO Pins zur **Eingabe (GPIO.IN)** oder zur

Ausgabe **(GPIO.OUT)** verwendet werden.

```
#Zum Beispiel Pin GPIO4 als Ausgabe-Pin definieren
GPIO.setup(4, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
```

Signalübertragung auf GPIO Pins

Nachdem definiert wurde,

1. wie die GPIO Pins angesprochen werden und
2. ob diese als Eingabe oder als Ausgabe dienen,

kann man beginnen ein Signal auf die GPIO Pins zu legen bzw. zu empfangen.

Hier gibt es zwei Varianten der Signalübertragung

- GPIO.LOW (legt ein niedriges Signal)
- GPIO.HIGH (legt ein hohes Signal)

auf den GPIO Pin.

Senden eines Signals

```
    #Auf GPIO17 wird ein niedriges Signal (0) gesendet
    GPIO.output(17, GPIO.LOW)
    #Auf GPIO27 wird ein hohes Signal (1) gesendet
    GPIO.output(27, GPIO.LOW)
```

Empfangen eines Signals

```
    #Variable signal enthält das Signal an GPIO17
    signal=GPIO.input(17)
    #Prüfung auf ein hohes Signal
    if signal==GPIO.LOW:
         print("Empfang eines niedrigen Signals auf GPIO17")
    else:
         print("Empfang eines hohen Signals auf GPIO17")
```

PWM (Pulsweitenmodulation)

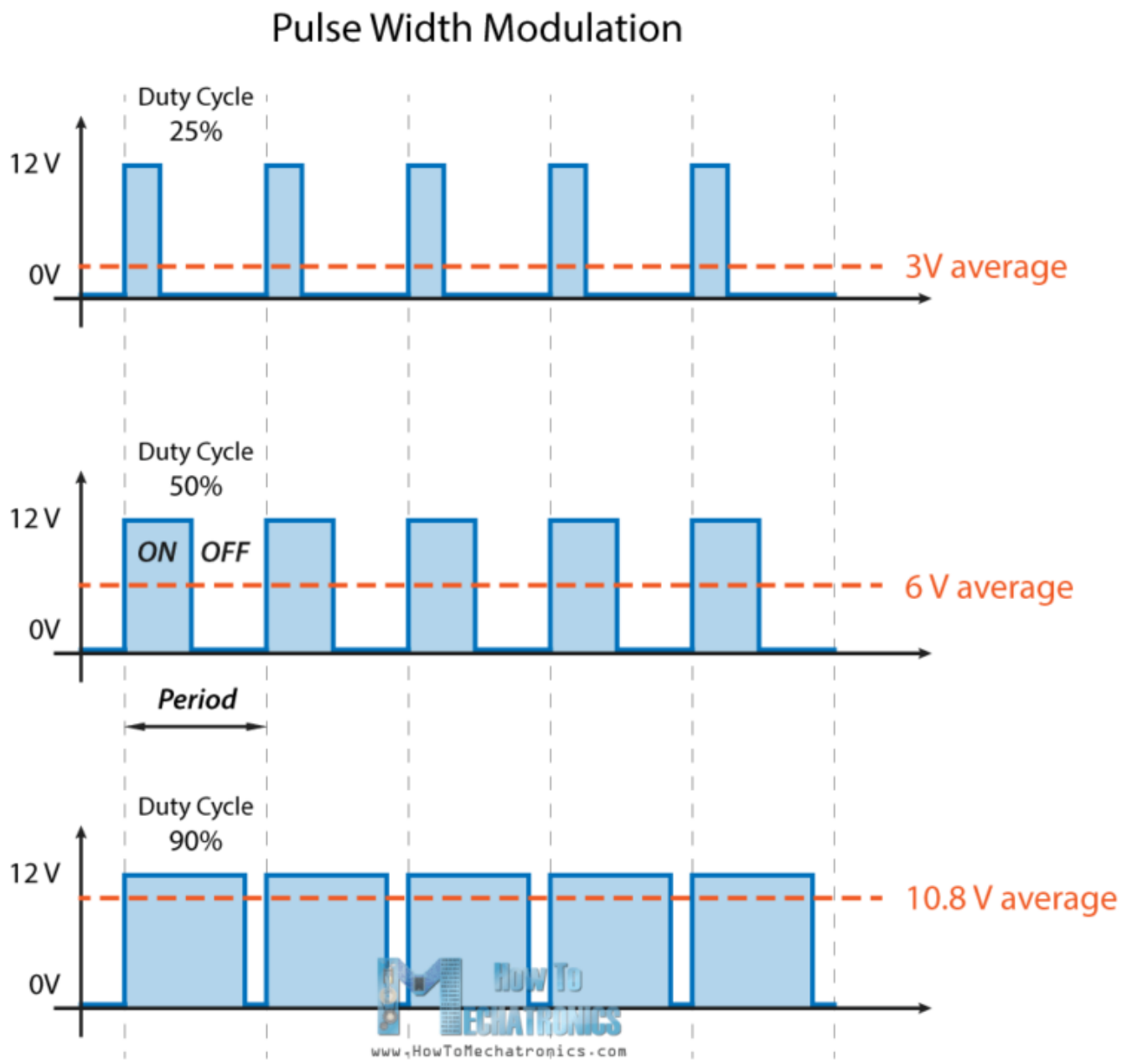
Über GPIO kann man nicht nur ganz niedrige (0V) oder ganze hohe Signale (3.3V) anlegen, man kann auch mithilfe von **PWM (Pulsweitenmodulation)** eine **Spannung die zwischen LOW und HIGH liegt erzeugen** und an einem GPIO Pin anlegen.

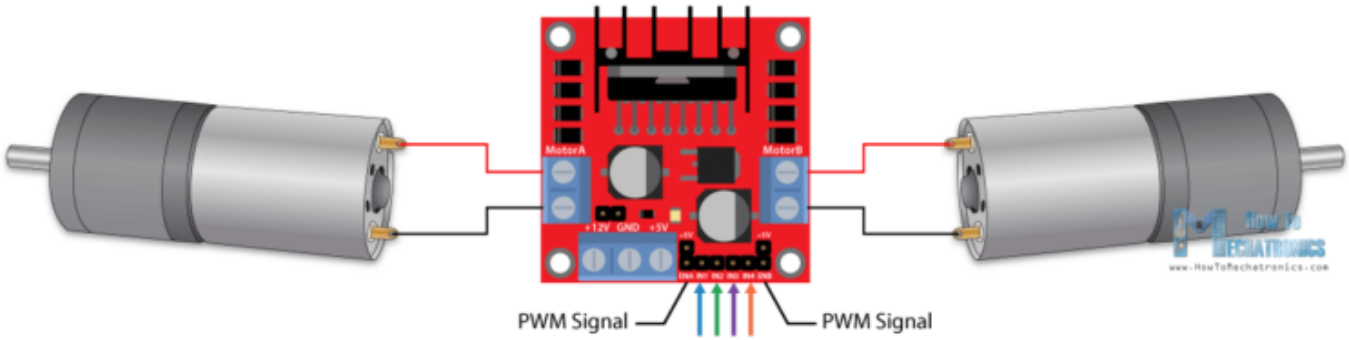
Damit kann man entweder **LEDs dimmen** oder **Motoren schneller bzw. langsamer** laufen lassen. Zweiteres kann für unser Projekt ganz hilfreich sein.
Dazu muss man die gewünschten GPIO Pins nicht nur als Input oder Output definieren, sondern zusätzlich bekanntgeben, dass man über diesen Pin auch weitere Signale (anstelle von GPIO.HIGH und GPIO.LOW) übertragen möchte.

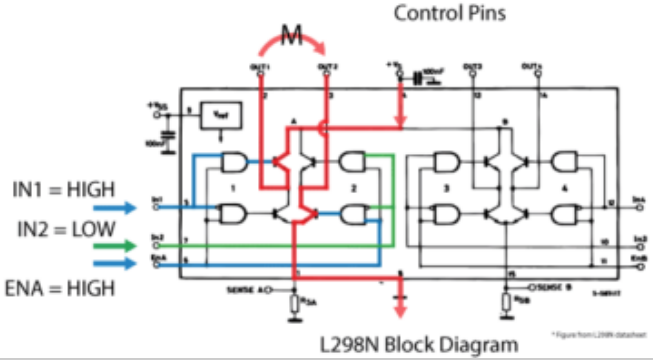
Dies geht wiefolgt:

```
EnableA=GPIO.PWM(4, 100)    #Frequency = 100
EnableA.start(10)            #10% von HIGH Signal (Möglich sind 0-100%)
```

Rein physikalisch wird einfach das Signal nicht durchgehend auf die Leitung übertragen sondern eben nur 10% der Zeit. Dadurch wird der Motor nicht ständig angetrieben sondern nur in 10% der Gesamtzeit, was sich wiederum auf die Geschwindigkeit des Motors auswirkt.







L298N Block Diagram

Video