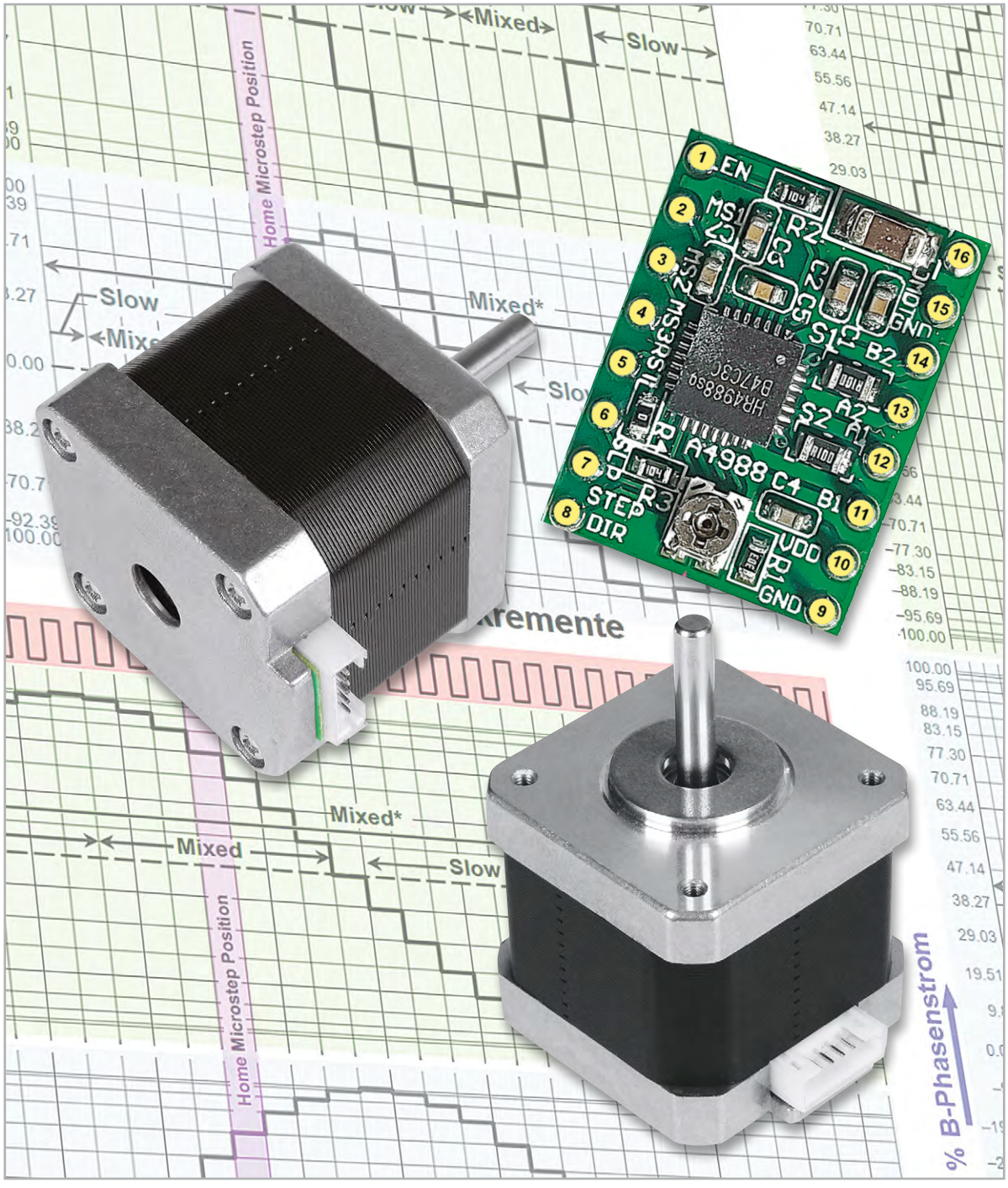
Schritt für Schritt

Steppermotoren zur präzisen Positionierung – Motortreiber A4988 am Raspberry Pi

In diesem Teil der Folge über Schrittmotoren wollen wir uns den bipolar angesteuerten Typen im NEMA-Format zuwenden. Der Schrittmotor wird mithilfe eines Raspberry Pi angesteuert und nach einigen Vorbereitungen kann der Motor mithilfe eines Python-Sketchs in Betrieb genommen werden.



Steppermotoren – Performance durch Controller

Als geeignetes Testobjekt wurde der Motor 17PM-K077BP01 CN des Herstellers Minebea gewählt, dessen zwei Phasenspulen über zwei auf die Anschlussteckerleiste herausgeführte Mittelanschlüsse verfügen. Damit lassen sie sich auch für die unipolare Bestromung nutzen und eignen sich als universelle Studienobjekte. Als Treibermodul verwenden wir den ST-A4988 vom Hersteller Allegro Microsystem. Der Motor 17PM-K077BP01CN ist eine NEMA-17-Type. Er wird mit einem ca. 50 cm langen, 4-drähtigen Anschlusskabel mit Steckern und Ferritkern zur Entstörung geliefert. Bild 1 gibt einen schnellen Überblick. Durch einen Schrittwinkel von 1,8 Grad im Vollschrittbetrieb werden 200 Schritte pro Umdrehung der Motorwelle benötigt. Ein 14-zahniges Kunststoffrad erlaubt den Einsatz eines Zahnriemens zur Übertragung der Motorwellenrotation mit Untersetzung.

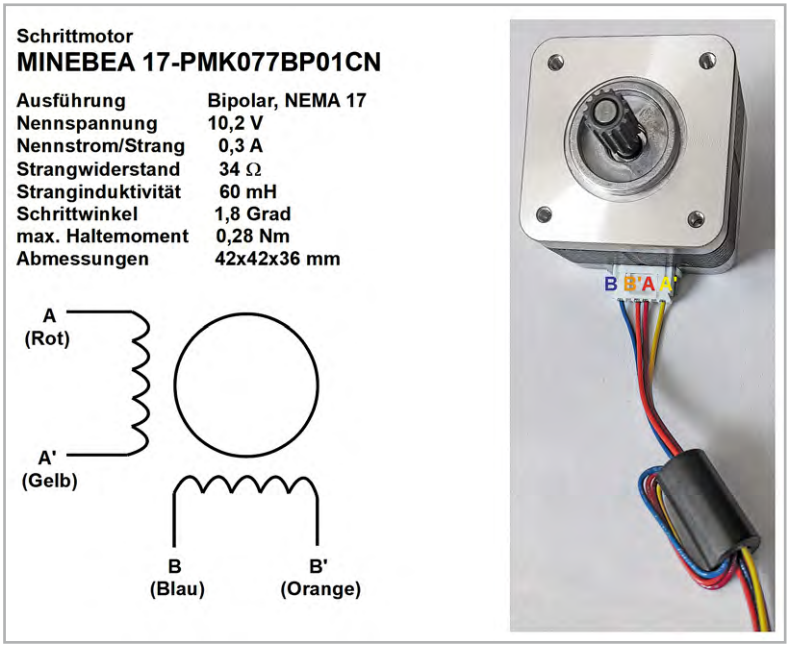


Bild 1: Mit diesem NEMA-17-Steppermotor mit herausgeführten Mittelanzapfungen der Phasen lassen sich vielfältige Experimente mit unipolarer und bipolarer Bestromung durchführen.

Treibermodul

Das Treibermodul ST-A4988 besteht aus einer kleinen Platine, deren Herzstück der DMOS-Treiber-Chip A4988 des Herstellers Allegro Microsystems ist. Für Spulenströme bis 1 A ist der Betrieb des Chips ohne Kühlkörper möglich. Um auf der sicheren Seite zu sein, empfiehlt sich jedoch das Aufkleben des mitgelieferten kleinen Kühlkorpers (Bild 2).

Trotz seiner geradezu winzigen Abmessungen hat es das Modul ganz schön in sich. Das Prinzipschaltbild in Bild 3 aus dem Allegro-Datenblatt soll einen knappen Überblick ermöglichen. Man sieht, dass die Phasenspulen des Steppermotors direkt an den Chip angeschlossen werden können. Die erforderlichen zwei H-Brücken, bestehend aus je 4-DMOS-Feldeffekttransistoren, schalten äußerst verlustarm, wodurch ihre Erwärmung sehr gering ausfällt. Die Spannungsversorgung der Brücken zur Bestromung der Phasenspulen sollte aus einem getrennten Netzteil erfolgen, das die erforderlichen Spannungen und Ströme zuverlässig liefern kann. Angesteuert wird der Chip von einem Microcontroller, dessen Aufgabe hier der Raspberry Pi 4B oder ein Raspberry Pi Pico übernimmt.

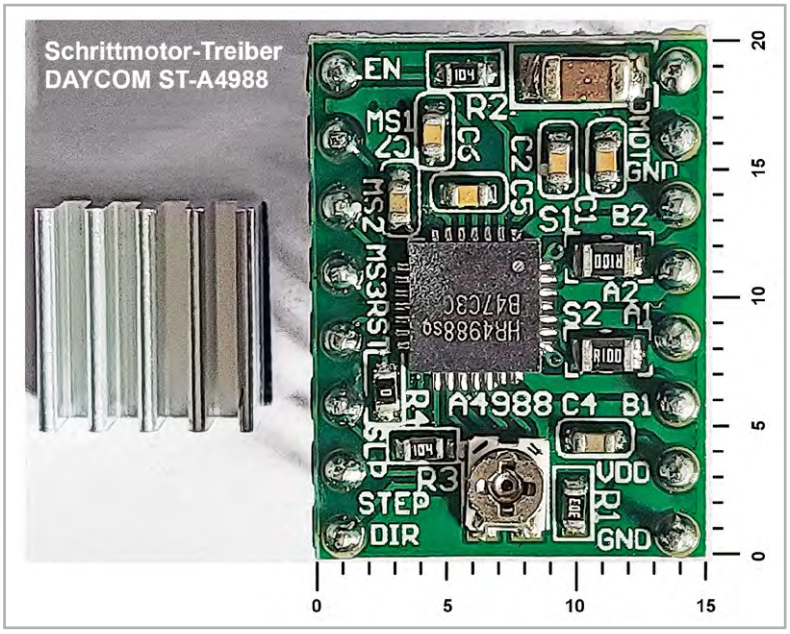


Bild 2: Briefmarkenklein, aber oho! Auf der 3 cm² großen Treiberplatine spielt der Allegro-Stepperchip A4988 die zentrale Rolle.

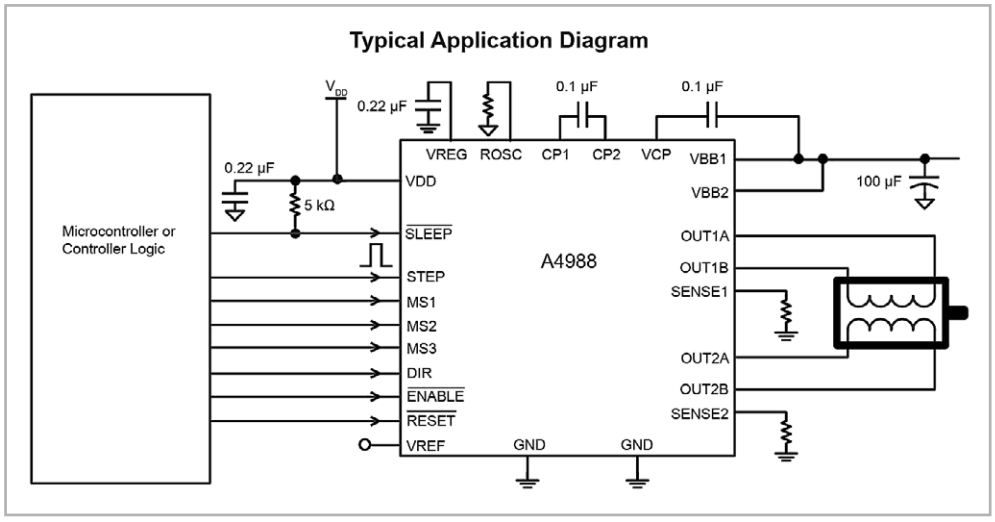


Bild 3: Dieses typische Anwendungsbeispiel lässt erkenen, dass sich der Chip A4988 leicht über die als Output geschalteten GPIO-Ports des Raspberry Pi 4B oder des Raspberry Pi Pico steuern lässt.

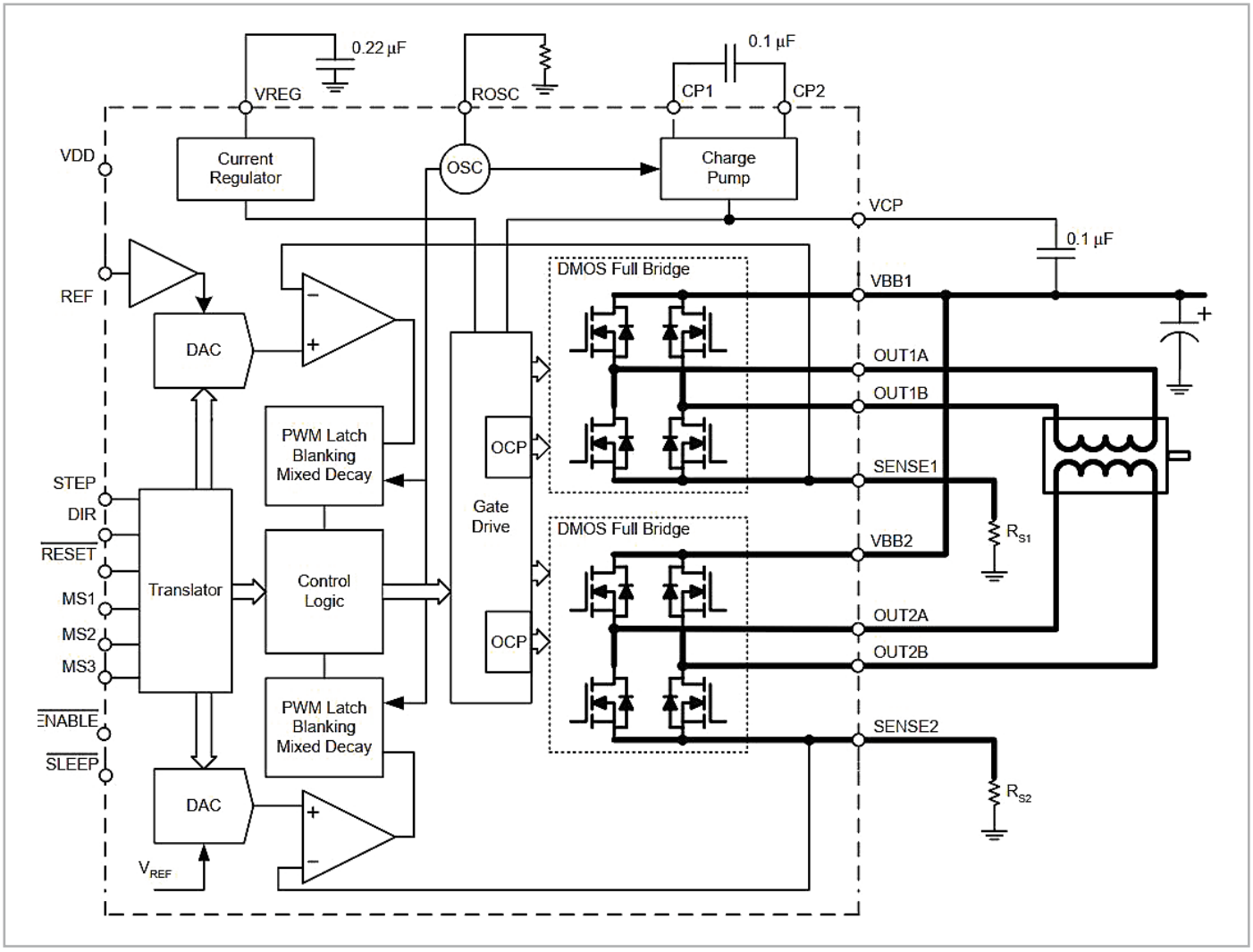


Bild 4: Durch die im Chip integrierten beiden Vollbrücken für die bipolare Phasenbestromung vereinfacht sich die Motoransteuerung erheblich.

Hinweis: Weil die Pins STEP (Schritt) und DIR (DIRection = Richtung) intern nicht auf HIGH- oder LOW-Potential gezogen werden, sollten sie nicht im schwebenden Zustand (floating) belassen, sondern stets mit dem gewünschten Potential angesteuert werden.

Eine wesentliche Schaltungsgruppe des Chips ist der sogenannte Translator (Übersetzer). Er bereitet die digitalen Ansteuersignale so auf, dass der Chip seine Aufgaben erfüllen kann (Bild 4). Die H-Brücken sind fett hervorgehoben. Die Widerstände RS1 und RS2 arbeiten als Fühler (Sensor) für den Spulenstrom. Der Spannungsabfall an ihnen wird von zwei Komparatoren als Abschaltkriterium herangezogen, um eine Überlastung von Motor und Chip durch Überströme zu vermeiden.

Bild 5 zeigt die Anschlussbelegung der Treiberplatine. Wir wollen die Anschlussfunktionen systematisch durchgehen.

Stromversorgung

Die Platine erfordert zwei Spannungsversorgungen. An Pin 16 (VMOT) und 15 (GND) wird das Gleichspannungsnetzteil zur Speisung der Phasenspulen des Motors angeschlossen (8–35 V, 2,5 A). Im Datenblatt des A4988 wird empfohlen, hier noch einen Entkopplungskondensator (50–100 µF) vorzusehen. An Pin 10 (VDD) und Pin 9 (GND) wird die Speisespannung für die Chip-Logik in Höhe von 3–5,5 V angelegt. Der Betriebsstrom für die Chiplogik liegt zwischen 5 und 10 mA und kann leicht vom Raspberry Pi übernommen werden. Auch hier empfiehlt sich ein Stützkondensator von 10–22 µF, der Spannungsspitzen und -einbrüche ausgleicht.

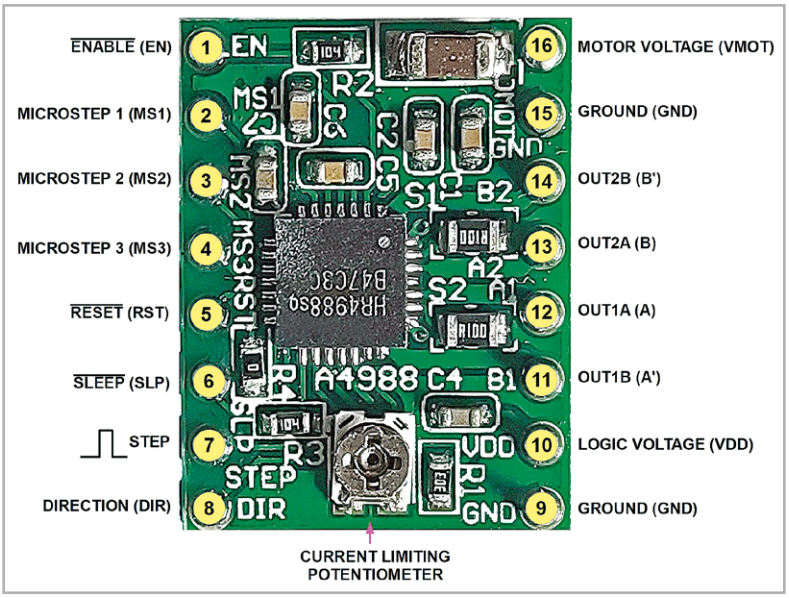


Bild 5: Das Pinout der Treiberplatine mit dem Potentiometer zur Einstellung des maximalen Spulenstroms

Microstep Selection Pins

Diese mit MS1 (Pin 2), MS2 (Pin3) und MS3 (Pin 4) bezeichneten Anschlüsse liegen unbeschaltet über interne Pull-down-Widerstände auf Massepotential und erzeugen so eine Bestromung der Motorspulen für den Vollschrittbetrieb mit 200 Steps/Umdrehung. Die weiterhin möglichen Betriebsarten Halbschritt (400 Steps/Umdrehung), Viertelschritt (800 Steps/Umdrehung), Achtelschritt (1600 Steps/Umdrehung) und Sechzehntelschritt (3200 Steps/Umdrehung) werden durch Anlegen von Logikpegeln an die MS-Pins gemäß Tabelle 1 gewählt.

Die Auswirkungen der gewählten Microstep-Auflösungen auf die Bestromung der Phasen A und B zeigt Bild 6. Die horizontalen mit Slow und Mixed bezeichneten Zeitabschnitte beziehen sich auf das erzwungene Abklingverhalten (decay) der Motorspulenströme nach einem ausgeführten Schritt. Man unterscheidet Fast (schnelles), Slow (langsames) und Mixed (gemischtes) Decay (Abklingen). Letztlich geht es darum, den stetigen Spulenstrom in den durch die Taktfrequenz und die Microstep-Einstellungen vorgegebenen Zeiten schnell genug abzubauen, um im nächsten Step für die FETs der H-Brücken gefahrlos mit dem erforderlichen neuen Spulenstrom fortfahren zu können. Im Datenblatt des A4988-Chips und in einem Application Report „Current Recirculation and Decay Modes“ von Texas Instrument (SLVA321a-2.pdf) werden weitere Informationen gegeben.

Die senkrecht violett hervorgehobenen Taktabschnitte kennzeichnen die sogenannte Home Microstep Position, aus der der Treiber nach dem Einschalten den Rotor von der 45°-Position ausgehend in Bewegung setzt. Über den Reset-Eingang (Pin 5 der Treiberplatine) wird der Translator in einen vordefinierten Home-Zustand versetzt. Alle STEP-Ansteuerimpulse werden so lange ignoriert, bis der Reset-Eingang auf HIGH gelegt wird.

Tabelle 1

MS1	MS2	MS3	Microstep-Auflösung
LOW	LOW	LOW	Vollschritt
HIGH	LOW	LOW	Halbschritt
LOW	HIGH	LOW	Viertelschritt
HIGH	HIGH	LOW	Achtelschritt
HIGH	HIGH	HIGH	Sechzehntelschritt

Steuer-Pins

Über die Steuer-Pins EN (ENable), RST (ReSeT) und SLP (SLeeP) lässt sich das Betriebsverhalten des A4988-Treiberchips steuern. Die Pins arbeiten mit inverser Logik (active low), d. h., sie aktivieren die mit ihnen verbundenen Funktionen, wenn ihr Logikpegel LOW (0) ist.

EN ist im LOW-Zustand aktiv und versetzt damit den Chip in den Betriebszustand. Erst wenn an EN ein HIGH-Potential angelegt wird, werden die Brücken-FETs alle abgeschaltet und damit die Bestromung der Motorspulen eingestellt. Damit ist dieser Pin besonders zur Schnell- oder Notabschaltung des Motors geeignet.

RST ist ebenfalls „active low“. Im LOW-Zustand werden alle STEP-Eingangsimpulse ignoriert. Zugleich wird der interne Translator auf einen Home-Zustand zurückgesetzt, von dem ausgehend der Motor nach dem Aufheben der Reset-Bedingung (RST → HIGH) startet.

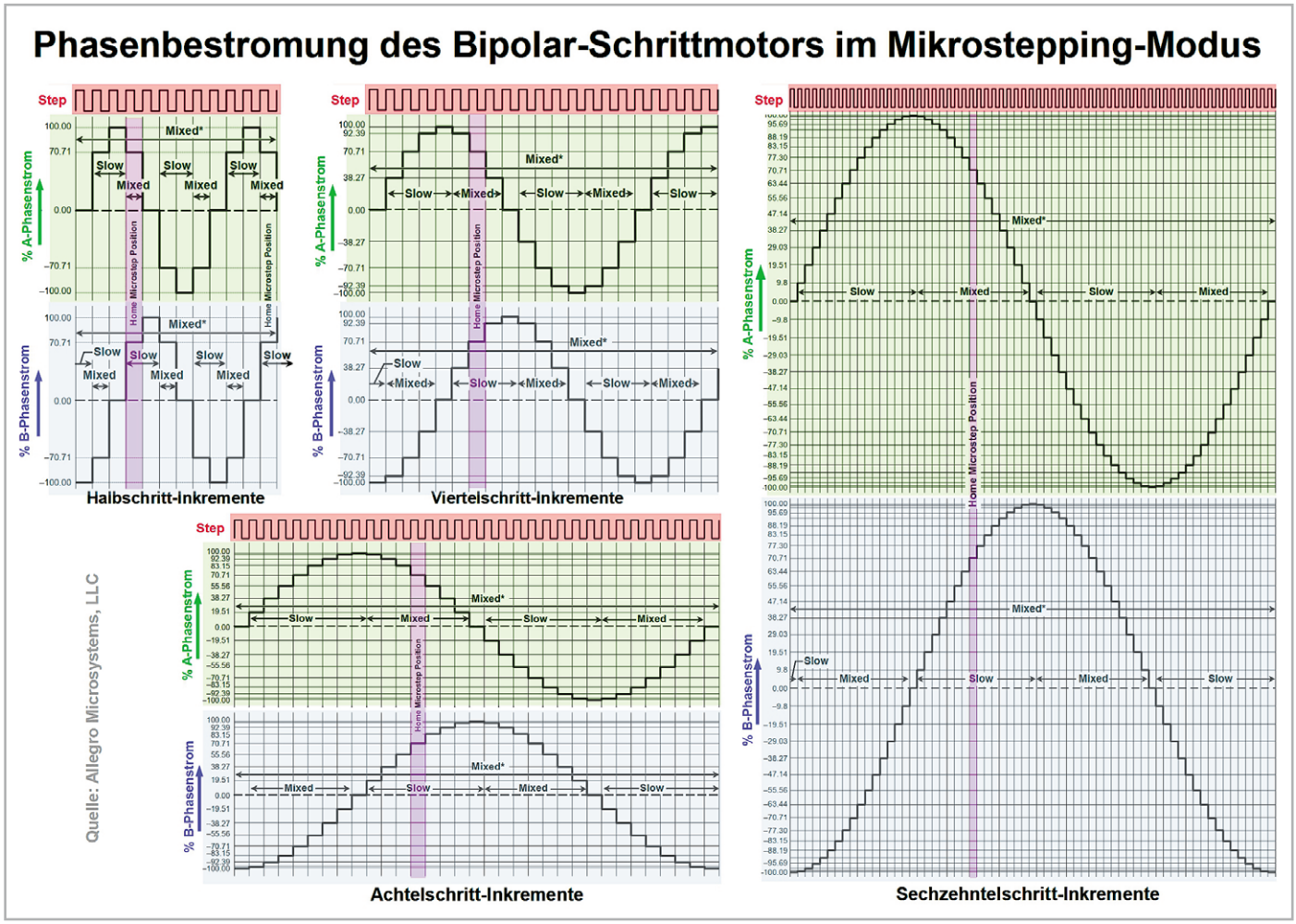


Bild 6: Die Bestromung der Phasen A und B im Halb-, Viertel-, Achtel- und Sechzehntel-Microstep-Betrieb

SLP minimiert im Active-Low-Zustand die Leistungsaufnahme des Motors im Stillstand durch Abschalten eines Großteils der inneren Schaltungsgruppen wie Ausgangs-FETs, Stromregelung und Ladungspumpe. Mit logischem HIGH-Pegel beaufschlagt wird der Chip in den normalen Betriebszustand versetzt. Nach dem Abschalten des Sleep-Zustands muss zur Stabilisierung der Ladungspumpe 1 ms vor der Abgabe des ersten Schrittkommandos gewartet werden.

Step- und Richtungseingänge

Jede positive Flanke eines Impulses am STEP-Eingang treibt den Motor um einen Schritt (entsprechend der Microstep-Einstellung an MS1, MS2 und MS3) weiter. Je höher die Impulsfrequenz, desto schneller dreht der Motor. Seine Drehrichtung gibt der Logikpegel am DIR-Eingang vor. Ist er HIGH, dreht sich der Motor im Uhrzeigersinn, ist er LOW in Gegenrichtung.

Die STEP- und DIR-Eingänge werden intern nicht auf ein bestimmtes Potential gelegt, weshalb sie nicht unbeschaltet (floating) bleiben sollten!

Die Ausgangspins 12 und 11 werden mit Anfang (A) und Ende (A') der Phasenspule A verbunden, die Pins 13 und 14 mit Anfang (B) und Ende (B') der Phasenspule B. Jeder Ausgang kann Ströme bis zu 2 A führen. Die Höhe des Stroms hängt von der Stärke des Netzteils, der Chipkühlung und dem eingestellten Maximalstrom (current limit) ab.

Strombegrenzung: Vor der Inbetriebnahme des Motors muss der Strom durch die Motorspulen auf einen maximal zulässigen Wert in Höhe des Nennstroms (rated current) begrenzt werden. Dazu dient ein kleines Trimmpotentiometer am unteren Platinenrand (current limiting potentiometer), dessen Schleifer die Referenzspannung V_{REF} als Bruchteil der Versorgungsspannung für die Chiplogik V_{DD} abgreift.

Im Datenblatt des A4988-Chips wird der Zusammenhang im Vollschrittbetrieb zwischen maximalem Spulenstrom I_{MAX}, Referenzspannung V_{REF} und Sensorwiderstand R_S mit $I_{MAX} = V_{REF} / (8 \times R_S)$ beschrieben. Da die Stromfühlerwiderstände R_S in der Regel 0,05 Ω betragen, muss bei vorgegebenem maximalem Spulenstrom I_{MAX} die Referenzspannung auf $V_{REF} = 0{,}4\ \Omega \times I_{MAX}$ am Schleifer des Trimmpotentiometers eingestellt werden. Dazu empfiehlt es sich, das Voltmeter mit einer Krokodilklemme am blanken Schaft des Einstellschraubendrehers anzuschließen, um während der Drehung die Spannung auf dem Voltmeter ablesen zu können.

Für den verwendeten Schrittmotor Minebea 17-PMK077BP01CN mit einem Nennstrom von 300 mA ist das Trimmpoti also so lange zu verdrehen, bis an seinem Schleifer 0,4 V/A x 300 mA = 120 mV anstehen.

Weil die Referenzspannung ein Bruchteil von V_{DD} ist, muss die Justierung des maximalen Strangwicklungsstroms bei einer Änderung von V_{DD} erneut wie vorstehend beschrieben vorgenommen werden.

Die Vorgehensweise noch einmal zusammengefasst:

1. Aus dem Datenblatt des Schrittmotors Nennspannung und Nennstrom pro Strang entnehmen. Im vorliegenden Beispiel sind die relevanten Werte NEMA 17, 200 Schritte/Umdrehung, 10,2 V und 300 mA.
2. Den Treiber in den Vollschrittmodus versetzen (MS1, MS2 und MS3 unbeschaltet lassen).
3. Den Motor anhalten, indem der Step-Eingang auf Masse gelegt wird.
4. V_{REF} nach der Formel $V_{REF} = 0{,}4\ \Omega \times I_{MAX}$ berechnen
5. Spannung V_{REF} am Schleifer des metallenen Trimmpotentiometers beim Justieren messen und auf berechneten Wert einstellen.
6. Netzgerät zur Motorspeisung auf Strangnennspannung des Motors einstellen und an Pin 16 (+) sowie Pin 15 (-) anschließen.

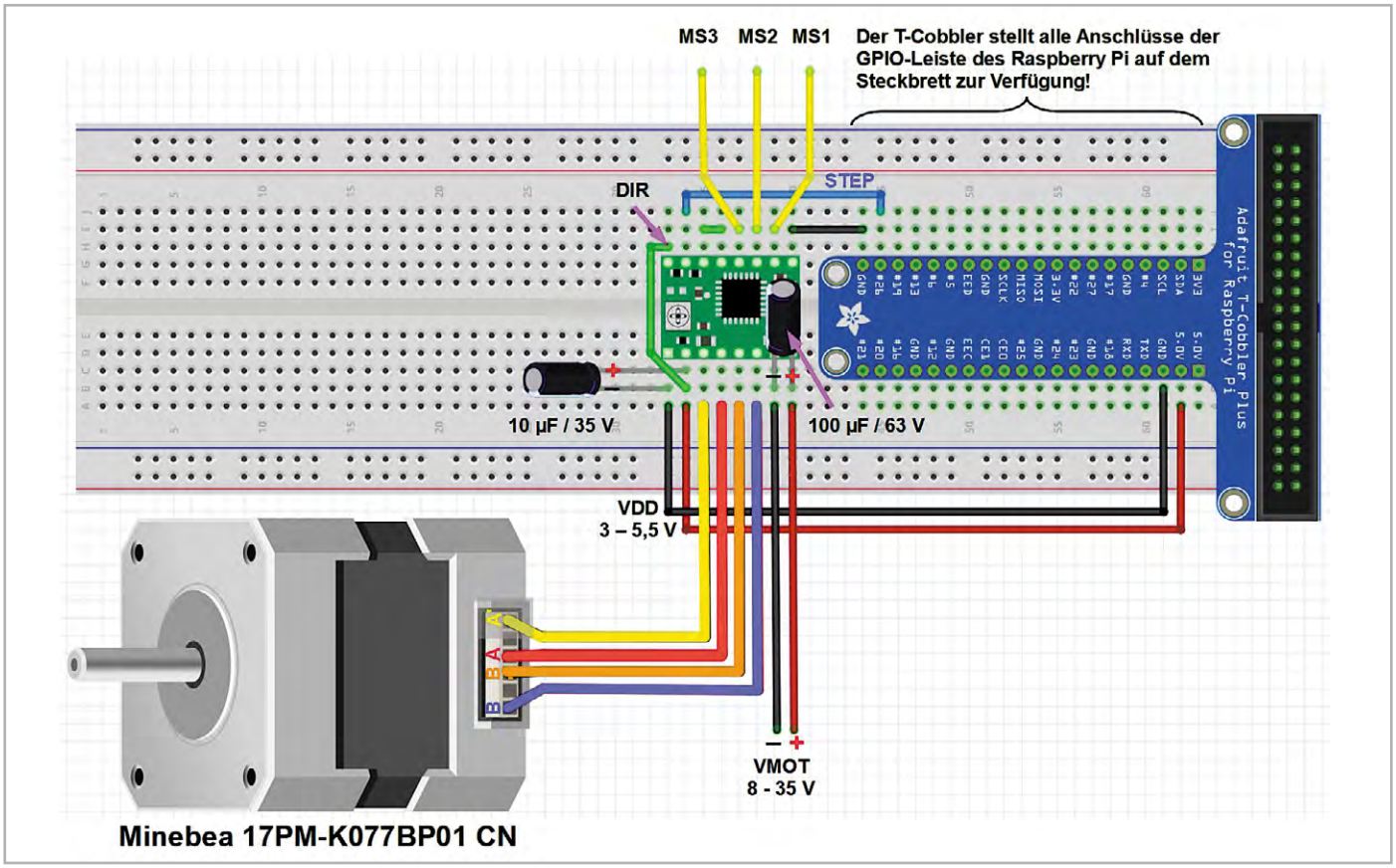


Bild 7: Mit dem Programm Fritzing lässt sich die Verdrahtung der Versuchsanordnung besonders übersichtlich darstellen.

Eine andere Vorgehensweise besteht im direkten Messen des Phasenstroms nach den vorbereitenden Schritten 1. bis 3. und dessen Einstellung mit dem Trimmpotentiometer auf den Sollwert.

Wenn alle Vorbereitungen getroffen sind und die Verdrahtung gemäß Bild 7 vorgenommen wurde, kann der Motor in Betrieb genommen werden.

Das nur wenige Zeilen umfassende Python-Sketch in Bild 8 dient einzig und allein der Erzeugung von Step-Impulsen mit variabler Frequenz, um den Motor zu Drehschritten zu veranlassen. Dafür wird nur ein einziger GPIO-Port des Raspberry Pi verwendet, nämlich GPIO26. Drehrichtung, Microstepping, Sleep und Reset können hardwaremäßig auf dem Steckbrett vorgenommen werden. In einer verfeinerten Software können dies auch weitere Ports des Raspberry Pi abhängig übernehmen.

In Zeile 13 wurde als Sleep-Variable 13,5 Sekunden gewählt, was im sechzehnschrittigen Microstep-Modus (3200 Steps/Umdrehung) zu einer Achsendrehung des Motors mit der Geschwindigkeit eines Stundenzeigers führt. Die Achse vollzieht demnach eine volle Umdrehung in zwölf Stunden.

Ein weiteres Programmbeispiel, durch das der Steppermotor abwechselnd eine Volldrehung nach rechts und eine Volldrehung nach links vollzieht, zeigt Bild 9. Nach jeweils einer Vierteldrehung wird deren Vollzug in der Kommandozeile gemeldet.

Konstantspannung oder -strom?

Für einen Neuling auf dem Gebiet der Schrittmotoren kommt es bei der Interpretation der Daten eines Schrittmotors gelegentlich zu Verständnisproblemen. So ist z. B. bei dem nachfolgend verwendeten NEMA-23-Motor 5T5618S2404-A der Firma Nanotec aus München auf dem Typenschild zu lesen: „3.6A 1.25V“ (Bild 10). Dies sind die Nennwerte für Wicklungsspannung und -strom der beiden bipolar bestromten Statorspulen des Schrittmotors. Das dazu verwendete Treibermodul TB6600 ist mit einem Eingangsspannungsbereich von 9-42 V_{DC} spezifiziert. Wie passt das mit der Wicklungsnennspannung von 1,25 $V_{REF} = 0{,}4\ \Omega \times I_{MAX}$ V zusammen?

Die Antwort ist einfach: Direkt an der Wicklung darf höchstens eine Spannung von 1,25 V anliegen, die dann einen Wicklungsstrom von 3,6 A zur Folge hat. Das resultiert aus dem ohmschen Gesetz gemäß dem im Motordatenblatt angegebenen Wicklungswiderstand von 0,35 Ω: I = 1,25 V/0,35 Ω = 3,5714 A. Die dabei in jeder Wicklung umgesetzte Leistung ist dann P = 1,25 V x 3,6 A = 4,5 W. Sie darf nicht dauerhaft überschritten werden, weil dies zur Überhitzung der Wicklung führen würde. Daher darf bei der Bestromung der Spulen über eine Vollbrückenschaltung (H-Brücke), die mit einer Konstantspannung gespeist wird, diese den genannten Spannungsnennwert überschreiten.

Minebea_12h_simpel.py

```
1  from time import sleep       # Imp. Meth. sleep aus Bib. time
2  import RPi.GPIO as GPIO      # Namenszuweisung
3
4  GPIO.setmode(GPIO.BCM)       # Broadcom-Nummerierung der IO-Pins
5  GPIO.setwarnings(False)      # GPIO-Warnungen unterdrücken
6  STEP = 26                    # Zuweisung von 26 zu Variable STEP
7
8  GPIO.setup(STEP, GPIO.OUT)   # GPIO26 als Output definieren
9  GPIO.output(STEP, 0)         # GPIO26 auf LOW setzen
10
11 while True:                  # Endlosschleife einrichten
12     GPIO.output(STEP, 1)     # pos. Flanke auf GPIO26 als STEP
13     sleep(13.5)              # 13.5 sec. schlafen (entspricht
14                              # bei 16tel Microstep einer
15                              # Rotordrehung in 12 Stunden)
16     GPIO.output(STEP, 0)     # GPIO26 auf LOW setzen
```

Bild 8: Das „Progrämmchen“ Minebea_12_simpel erzeugt lediglich STEP-Impulse, die den Motor zu einem Rotationsinkrement veranlassen.

Minebea_230919_dir.py

```
1  from time import sleep; import RPi.GPIO as GPIO
2
3  GPIO.setmode(GPIO.BCM); GPIO.setwarnings(False)
4  STEP = 26; DIR = 19; links = 1; rechts = 0
5
6  GPIO.setup(STEP, GPIO.OUT); GPIO.setup(DIR, GPIO.OUT)
7
8  while True:
9      GPIO.output(DIR, links)
10     for i in range(3201):
11         GPIO.output(STEP, 1)
12         sleep(0.001)
13         GPIO.output(STEP, 0)
14         if i == 800:
15             print("Step ", i, "Vierteldrehung rechts")
16         elif i == 1600:
17             print("Step", i, "Halbdrehung rechts")
18         elif i == 2400:
19             print("Step", i, "Dreivierteldrehung rechts")
20         elif i == 3200:
21             print("Step", i, "Volldrehung rechts \n")
22     GPIO.output(DIR, rechts)
23     for i in range(3201):
24         GPIO.output(STEP, 1)
25         sleep(0.001)
26         GPIO.output(STEP, 0)
27         if i == 800:
28             print("Step ", i, "Vierteldrehung links")
29         elif i == 1600:
30             print("Step", i, "Halbdrehung links")
31         elif i == 2400:
32             print("Step", i, "Dreivierteldrehung links")
33         elif i == 3200:
34             print("Step", i, "Volldrehung links \n")
```

Kommandozeile

```
Step 2400 Dreivierteldrehung links
Step 3200 Volldrehung links

Step  800 Vierteldrehung rechts
Step 1600 Halbdrehung rechts
Step 2400 Dreivierteldrehung rechts
Step 3200 Volldrehung rechts

Step  800 Vierteldrehung links
```

Bild 9: Hier wird demonstriert, wie sich die Drehrichtung steuern lässt.

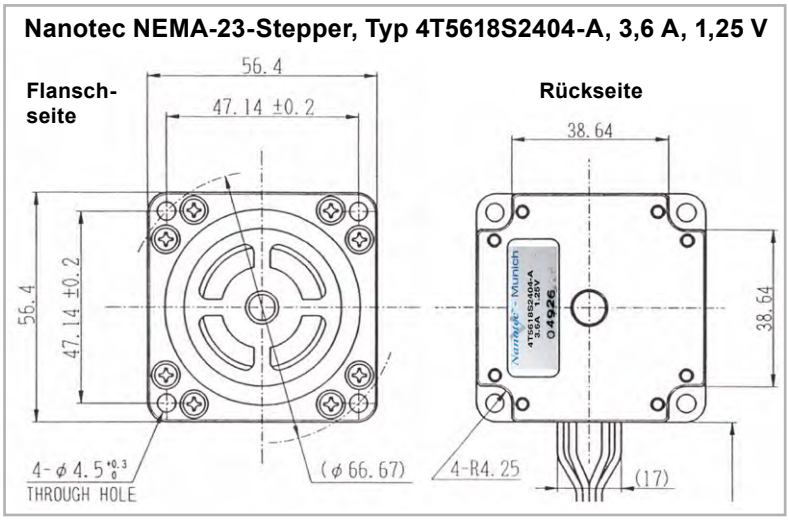


Bild 10: Maßzeichnungen aus dem Datenblatt des NEMA-23-Motors 5T5618S2404-A der Münchener Firma Nanotec

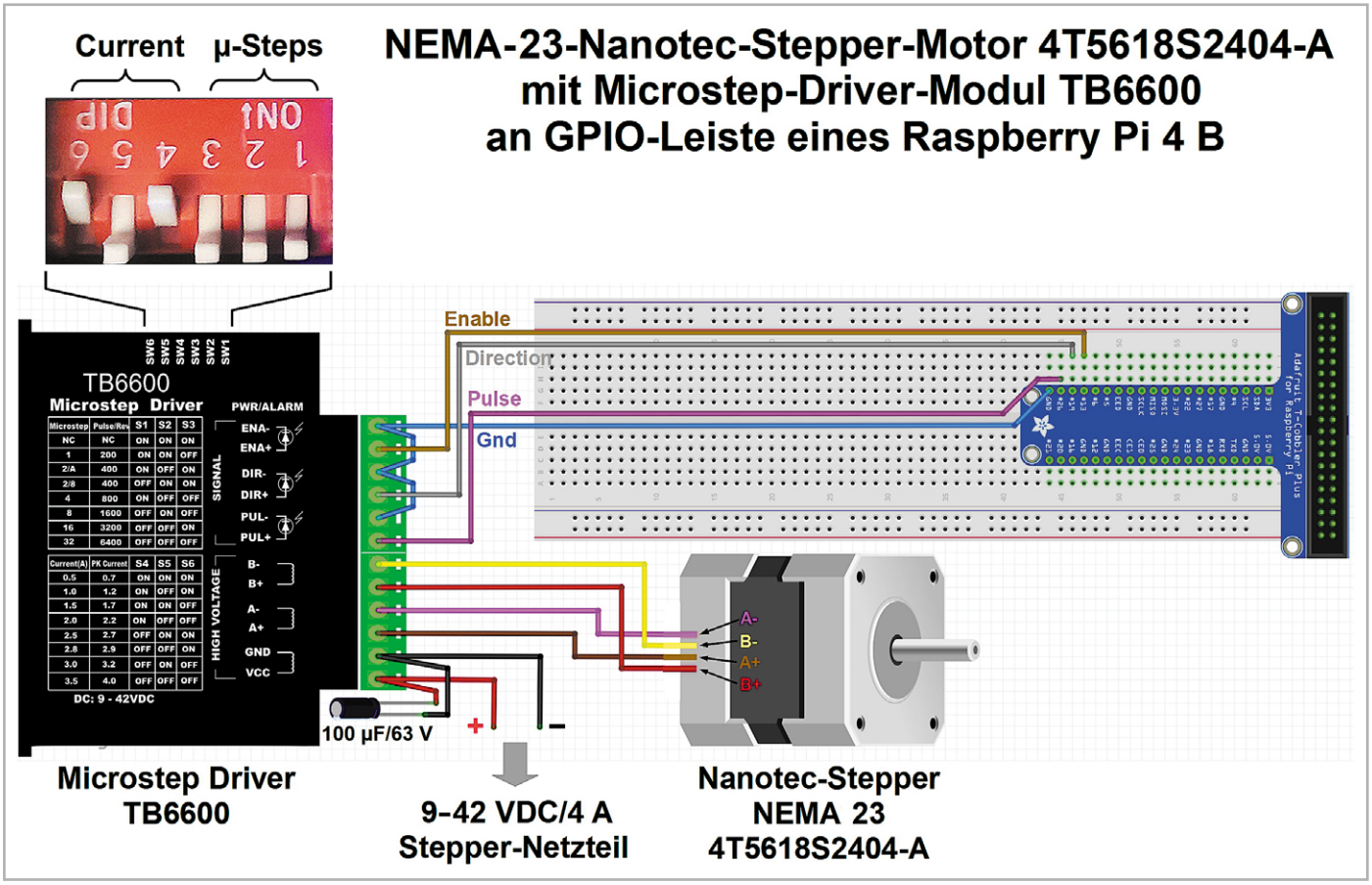


Bild 11: So werden Motor und Controller verdrahtet.

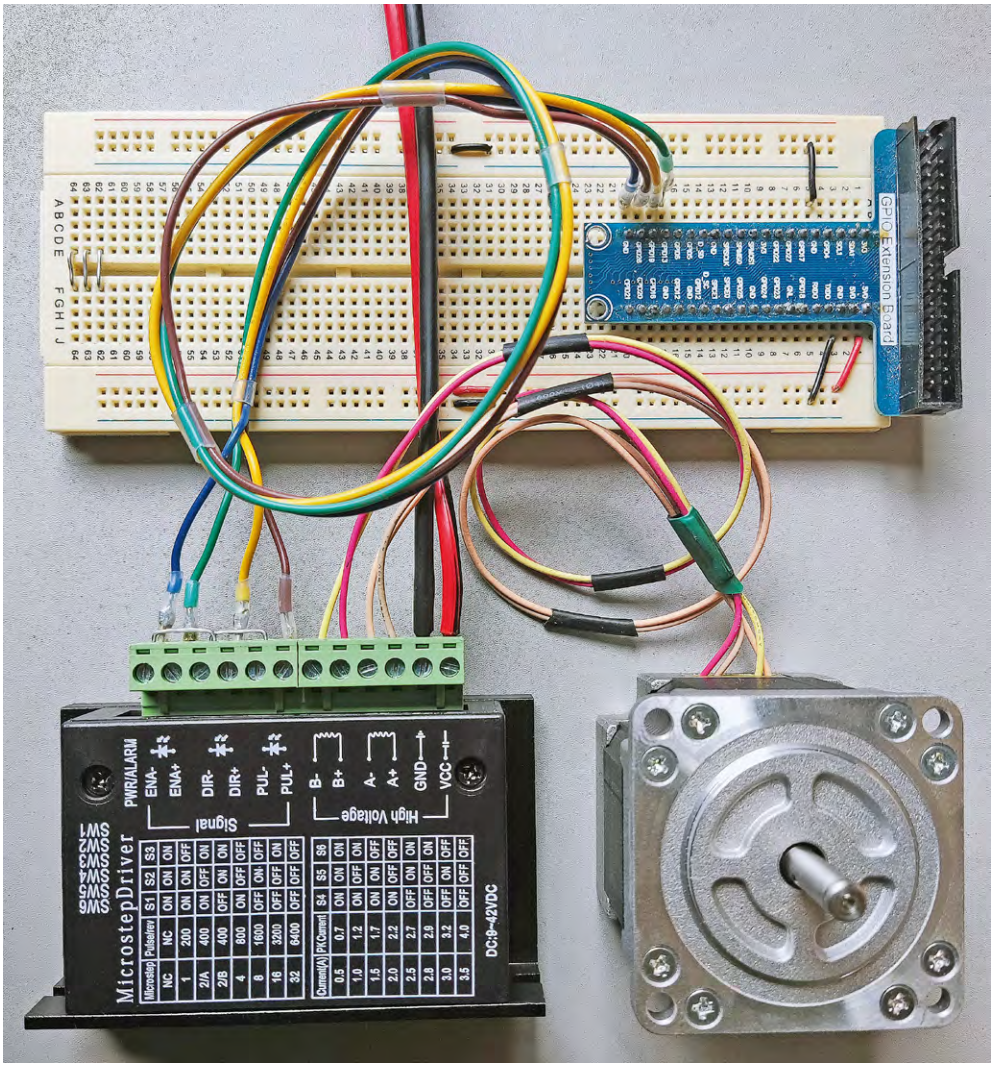


Bild 12: Die Verdrahtung in der Praxis

Bild 13: Das Python-Programm sollte leicht nachzuvollziehen sein.

Nanotec_4T56118S2404-A_231015.py

```
from time import sleep; import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM); GPIO.setwarnings(False)
STEP = 26; DIR = 19; ENA = 13; links = 1; rechts = 0; enable = 1; delay = 0.001

GPIO.setup(STEP, GPIO.OUT); GPIO.setup(DIR, GPIO.OUT); GPIO.setup(ENA, GPIO.OUT)

if enable == 1:
    GPIO.output(ENA, 0)
elif enable == 0:
    GPIO.output(ENA, 1)

while True:
    GPIO.output(DIR, links)
    for i in range(6401):
        GPIO.output(STEP, 1)
        sleep(delay)
        GPIO.output(STEP, 0)
        if i == 1600:
            print("Step", i, "Vierteldrehung rechts")
        elif i == 3200:
            print("Step", i, "Halbdrehung rechts")
        elif i == 4800:
            print("Step", i, "Dreivierteldrehung rechts")
        elif i == 6400:
            print("Step", i, "Volldrehung rechts \n")
    GPIO.output(DIR, rechts)
    for i in range(6401):
        GPIO.output(STEP, 1)
        sleep(delay )
        GPIO.output(STEP, 0)
        if i == 1600:
            print("Step", i, "Vierteldrehung links")
        elif i == 3200:
            print("Step", i, "Halbdrehung links")
        elif i == 4800:
            print("Step", i, "Dreivierteldrehung links")
        elif i == 6400:
            print("Step", i, "Volldrehung links \n")
```

Kommandozeile

```
Step 6400 Volldrehung links

Step 1600 Vierteldrehung rechts
Step 3200 Halbdrehung rechts
Step 4800 Dreivierteldrehung rechts
Step 6400 Volldrehung rechts

Step 1600 Vierteldrehung links
```

Anders ist es, wenn am Ein- oder Ausgang der Brücke eine Konstantstromquelle angeordnet ist. Diese Konstantstromquelle darf die Wicklung höchstens mit dem Nennstrom bestromen, wird aber natürlich selbst mit einer höheren Spannung als der Wicklungsnennspannung betrieben. Dabei gilt: Je höher die Betriebsspannung der Konstantstromquellen, desto größer ist deren Verlustleistung.

Beim erstgenannten Konstantspannungsbetrieb stellt sich also ein entsprechender Wicklungsstrom ein, beim zweitgenannten ergibt sich eine zugehörige Wicklungsspannung. Der Konstantstrombetrieb ist dem Konstantspannungsbetrieb hinsichtlich erreichbarer Drehzahl, dynamischen Verhaltens und Drehmoment im oberen Drehzahlbereich überlegen und kommt in modernen Treiber-ICs nahezu ausnahmslos zur Anwendung. In beiden Betriebsarten darf die maximal zulässige Verlustleitung in den Wicklungen nicht überschritten werden.

TB6600 ist ein preiswertes (ca. 5 Euro) und dennoch leistungsfähiges, universell verwendbares Schrittmotortreibermodul. Es wird durch ein separates Netzteil gespeist, das eine Gleichspannung von 9-42 V mit einem Spitzenstrom von 4 A liefern können sollte. TB6600 verfügt über drei Steuerungseingänge, nämlich

- ENA: ENAble = Modul aktivieren,
- DIR: DIRection = Drehrichtung,
- PUL: PULse = Schrittimpuls,

die mit drei jeweils als Ausgang geschalteten Ports der GPIO-Leiste des Raspberry Pi gemäß Bild 11 verbunden sind. Ein „Mäuseklavier" (DIP-Schalter, DIP: Dual Inline Package) an der Seite des Stepper-Moduls erlaubt die Einstellung der Mikrosteps und und des Wicklungsstroms.

Ein Foto des realen Aufbaus zeigt Bild 12. Der Wannenstecker am T-Cobbler wird über ein 20-poliges Flachbandkabel mit der GPIO des Raspberry Pi 4 B verbunden, die rot-schwarzen Kabel am oberen Bildrand führen zum Netzteil zur Motorspeisung. Wie in Bild 11 zu sehen ist, wurde an den DIP-Schaltern SW4 = ON, SW5 = OFF, SW6 = ON gewählt, was zu einem Wicklungsstrom von 1 A führt. Mit SW1 = OFF, SW2 = OFF und SW3 = OFF werden 32 zusätzliche Mikrosteps gewählt. Weil der Motor ohne Mikrosteps mit 200 Schritten à 1,8° eine ganze Drehung vollzieht, sind es jetzt 32 x 200 = 6400 Steps/Umdrehung.

Bild 13 zeigt das zugehörige Python-Programm. Zu erwähnen ist, dass bei einem passiven Enable-Ausgang (GPIO13 = 0) der Motor steht, ohne dass die Statorspulen bestromt werden. Das spart Energie und erlaubt die leichte, widerstandsarme Drehung des Rotors in eine Wunschposition. ELV