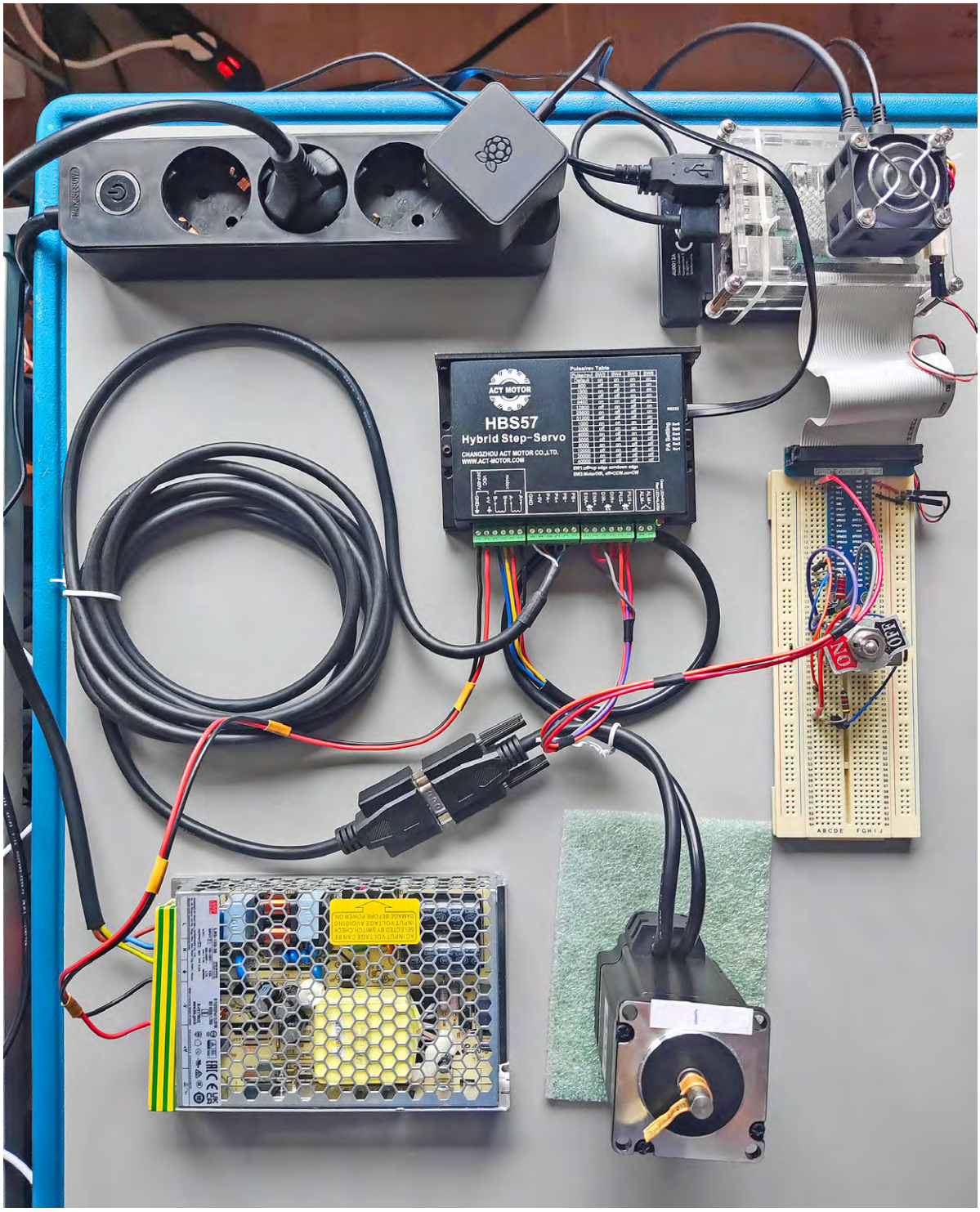
Schritt für Schritt

NEMA23-Motor und Controller/Driver mit RS232-Schnittstelle für Closed-Loop-Betrieb

Eine für Lernzwecke hervorragend geeignete Paarung aus NEMA23-Schrittmotor 23SSM8440-EC1000 und Controller/Treiber HBS57 wird in diesem Teil der Folge behandelt.



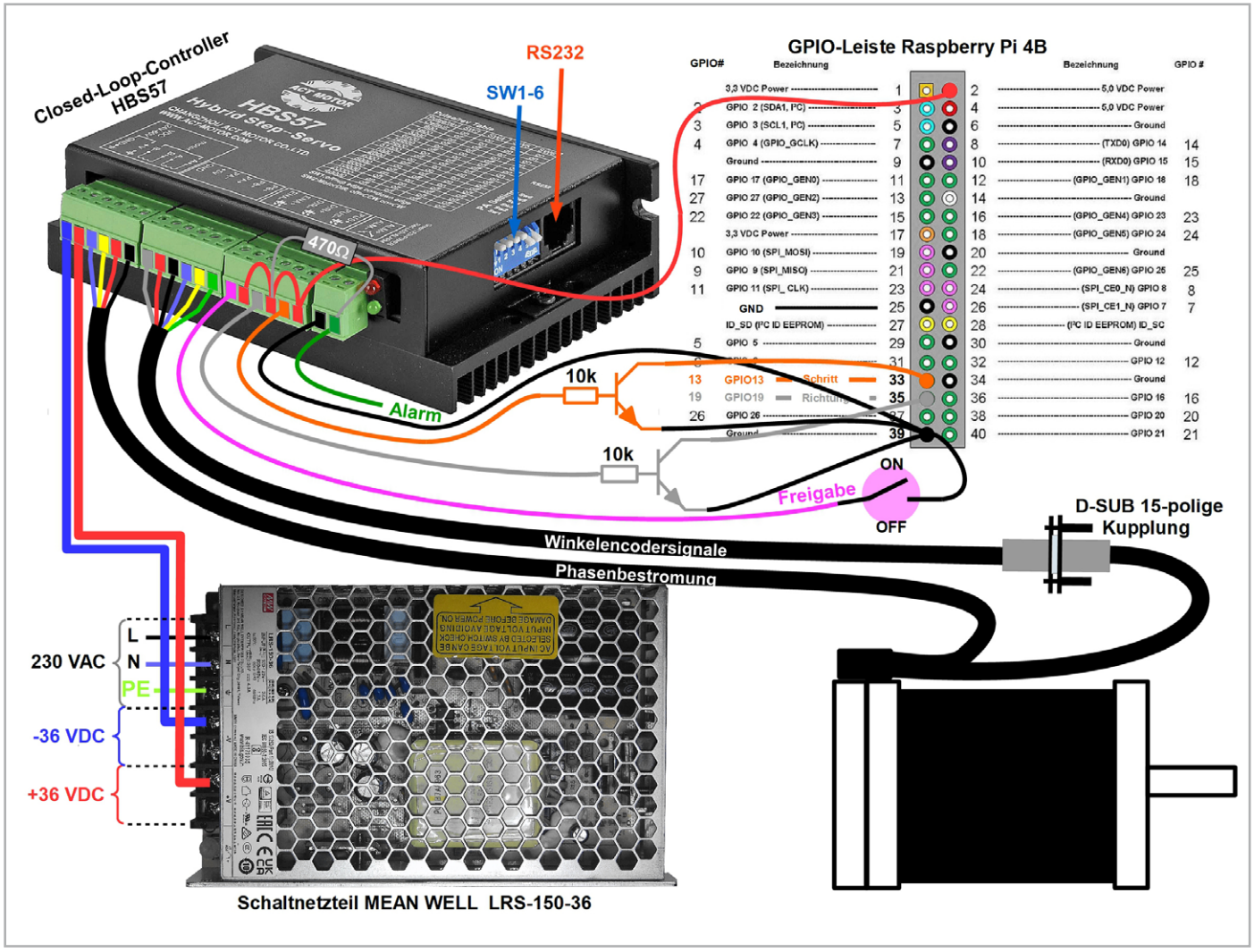


Bild 1: Der Experimentalaufbau als Verdrahtungsplan ohne den in die seitliche RJ12-Buchse eingesteckten RS232-USB-Adapter

Motor 23SSM8440-EC1000

Mit 1500 g Gewicht und den Abmessungen 120 x 47 x 47 mm ist der 2-Phasen-Motor ein kleines Kraftpaket. Eine maximale Phasenspannung von 2,4 V mit zugehörigem Phasenstrom von 4 A (0,6 Ω Wicklungswiderstand) verleiht dem Motor ein Arbeitsdrehmoment von 6,8 N·m bei einem Haltemoment mit unbestromten Phasen von 2,1 N · cm. Das Massenträgheitsmoment (Inertialmoment) des Rotors beträgt 480 g · cm² (siehe Kasten Elektronikwissen).

Den Motor verlassen zwei Kabel. Eines davon enthält vier Litzen, die zu den Phasenspulen führen. Das andere verbindet die Encodersignale mit einem 15-poligen D-Sub-Stecker, für den eine passende Kupplung mit Kabel im Lieferumfang enthalten ist, das die Litzen zum Anschluss an den Controller/Treiber HBS57 enthält.

Closed-Loop-Controller/Treiber HBS57

Der Closed-Loop-Controller/Treiber HBS57 zur Phasenbestromung unter Berücksichtigung der Positionsinformationen des im Motor integrierten Rotordrehwinkelencoders verfügt über eine serielle RS232-Schnittstelle. Über diese lassen sich die Betriebsparameter des Motors vielfältig beeinflussen und ein Testbetrieb des Motors konfigurieren. Der Controller/Treiber HBS57 verfügt über vier mit Steckkontaktblöcken versehene Kontakgruppen:

1. Bestromung der Phasen (GND, +V, B-, B+, A-, A+)
2. Rückführung der Encodersignale (GND, +5V, PA-, PA+, PB-, PB+)
3. Motorsteuersignale (ENA+, ENA-, DIR+, Dir-, PUS+, PUS-)
4. Alarmausgang (ALM-, ALM+)

Schaltnetzteil LRS-150-36

Das Schaltnetzteil LRS-150-36 liefert 36 VDC mit einem Strom von bis zu 4,3 A entsprechend ca. 150 W an den Anschlussklemmen GND und +V der linken Schraubkontakte.

Eine detaillierte Darstellung der gesamten Anordnung aus Netzteil, Motor, Closed-Loop-Controller/Treiber und der GPIO-Leiste eines Raspberry Pi 4B ist in Bild 1 zu sehen. Daraus lässt sich auch die Verdrahtung im Detail entnehmen.

Inbetriebnahme

Eine erste testweise Inbetriebnahme des Motors ist auch ohne Ansteuerung über den Raspi möglich. Dazu bedarf es einer kleinen Windows-Software, die über die RS232-Schnittstelle des Controllers mit diesem Kontakt aufnimmt.

Die serielle Schnittstelle

Heute haben immer weniger Computer eine serielle Schnittstelle nach dem RS232-Standard. Sie wurde weitestgehend durch USB-Interfaces (Universal Serial Bus) verdrängt, die aber nicht unmittelbar geeignet sind, mit dem Controller zu kommunizieren. Dazu bedarf es eines USB-RS232-Adapters, der den Anschluss von seriellen Geräten an einem USB-Port

Technikwissen

Wie kommt es zur Einheit kg · m² bzw g · cm² für das Massenträgheitsmoment?

Eine wichtige Kenngröße eines Schrittmotors ist das Massenträgheitsmoment **I** seines Rotors, da es zusammen mit dem Trägheitsmoment der Last das dynamische Verhalten des Motors, insbesondere beim Beschleunigen und Abbremsen, maßgeblich mitbestimmt. Es wird in der Einheit **kg · m²** oder **g · cm²** angegeben und beschreibt den Widerstand, den der Rotor einem beschleunigenden Drehmoment entgegensetzt.

Energieinhalt eines bewegten Massepunkts

Die Bewegungsenergie (kinetische Energie) eines geradlinig (linear) mit der Geschwindigkeit v bewegten Punktes P mit der trägen Masse m beträgt $E_{lin} = 0{,}5 \cdot m \cdot v^2$.

Soll dieser Massepunkt nun aber im Abstand r (Dimension m oder cm) mit der gleichen kinetischen Energie um einen Mittelpunkt rotieren, sich also auf einer kreisförmigen Bahn mit Radius r bewegen, hat er auf dieser Kreisbahn die Geschwindigkeit $v = \omega \cdot r$, wobei ω die Winkelgeschwindigkeit (Winkeländerung pro Zeiteinheit) ist. Seine Bewegungsenergie ist dabei $E_{rot} = 0{,}5 \cdot m \cdot r^2 \cdot \omega^2$.

Der Vergleich von E_{lin} und E_{rot} zeigt, dass an die Stelle der trägen Masse m bei der geradlinigen Bewegung das Trägheitsmoment (Inertialmoment) $m \cdot r^2$ bei der kreisförmigen Bewegung getreten ist. Die lineare Geschwindigkeit v hat ihre Entsprechung in der rotatorischen Winkelgeschwindigkeit ω.

Das Inertialmoment **I** hat deshalb die Dimension Masse · Abstand², also kg · m² oder g · cm².

des PCs ermöglicht. Leider hat der Lieferant von Motor und Controller/Treiber es nicht für nötig befunden, Informationen über die RS232-Schnittstelle und die zugehörige Software anzubieten. Auch telefonische Anfragen in der ACT-Niederlassung in Bremen wurden nicht beantwortet. Es war dort schlicht niemand ans Telefon gegangen. Es bedurfte also einiger Recherchen im Internet, um an das Fehlende zu gelangen.

Ein gefundenes, recht ausführliches Manual zum HBS57 brachte zwar einige wichtige Hintergrundinformationen, aber gerade das gesuchte Pinout des RS232-Controller/Treiber-Ports erwies sich als falsch. Dadurch wurde manche Stunde mit dem vergeblichen Versuch verschwendet, diesen in Betrieb zu nehmen.

Über ein Youtube-Video kam der Autor auf die Spur des chinesischen Unternehmens Changzhou Longs Motor Co. Ltd., das offensichtlich mit dem Motoren- und Controllerhersteller ACT in enger Verbindung zu stehen scheint. Jedenfalls kommen beide aus Changzhou und führen sowohl Controller als auch Motor unter der gleichen Typenbezeichnung (Bild 2).

Nach den Youtube-basierten Erkenntnissen scheint auch die Software namens ACT_HBS_Pro_turn.exe bzw. LONGS_HBS_Pro_turn.exe bis auf unterschiedliche Firmenlogos identisch zu sein.

Leider ist wegen einer falschen Zertifizierung die Homepage www.act-motor.com nicht zu erreichen, um zu prüfen, ob das Programm ACT_HBS_Pro_turn.exe dort verfügbar ist. Also wurde das Programm LONGS_HBS_Pro_turn.exe von der Serviceseite heruntergeladen. Das Pinout der seriellen Schnittstelle im Controller war aber auch hier nicht zu erhalten, konnte jedoch dem Youtube-Video entnommen werden und erwies sich als identisch mit dem aus dem Infoblatt.

Vom chinesischen Hersteller Leadshine ist ebenfalls ein Controller HBS57 erhältlich. Allerdings dient dieser der Bestromung von 3-Phasen-Schrittmotoren in Dreieckschaltung. Sehr merkwürdig!

Nach dieser Vorarbeit war es nicht mehr schwierig, die serielle Schnittstelle anzusprechen. Dazu verwendete der Autor ein RS232-USB-Interface mit den drei Drahtanschlüssen GND, RXD und TXD aus der Bastelkiste.

Wegen des Verständnisgewinns beim „Selbstbau" dieses Kabels wird dieser in Bild 3 und Bild 4 zusammengefasst und braucht nicht noch einmal im Text beschrieben zu werden.

Später stellte sich heraus, dass bei einem Lieferanten mit einem Zahlendreher in der Shop-Beschreibung („Anschlusskabel zur Verbindung des Motortreibers ACT HBS75 …") ein sehr preiswerter USB-RS232-Adapter (ca. 7 €) erhältlich ist. Diesen und die Selbstbauversion zeigt Bild 5.

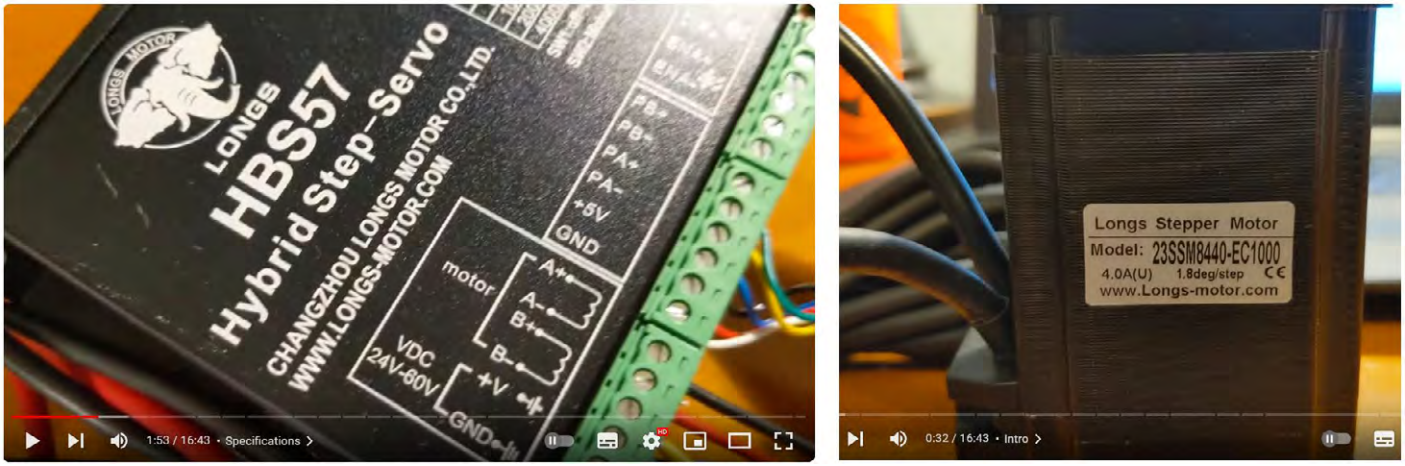


Bild 2: Der chinesische Hersteller Longs verkauft die verwendeten ACT-Komponenten unter gleichen Typenbezeichnungen und auch sonst allem Anschein nach identisch.

RS232-Modular-Stecker RJ12 (6P6C)

Pin-Nr.	Symbol	Zweck	Anmerkung	Farbe
1	n.c.	/	not connected	weiß
2	+5V	Versorgungs -spannung	für externe Geräte z.B. HISU	braun
3	TXD	senden (transmit)	/	grün
4	GND	Masse	0V	gelb
5	RXD	empfangen (receive)	/	rosa
6	n.c.	/	not connected	grau

Legende

HISU:	Handheld Intelligent Setting Unit (Intelligentes Handprogrammiergerät)
RJ12:	Registered Jack No. 12
6P6C:	6 Positions 6 Connections
RS232:	Recommended Standard 232 (serieller Schnittstellenstandard)
TXD:	Transmit Data (RS232 senden)
RXD:	Receive Data (RS232 empfangen)

Bild 3: In diesem Bild ist alles zusammengefasst, was man zur RS232-Schnittstelle des ACT-Treibers HBS57 wissen muss.

Achtung! Auf keinen Fall sollte man ohne Kenntnis der Pinbelegung der RS232-RJ12-Schnittstelle irgendeinen USB-RS232-Adapter verwenden. Es besteht die Gefahr der Beschädigung der beteiligten Geräte.

Debugging-Software

Die Debugging-Software LONGS_HBS_Pro_turn.exe ist eine ausführbare Datei, die nicht auf dem Computer installiert wird, sondern direkt nach ihrem Aufruf funktioniert.

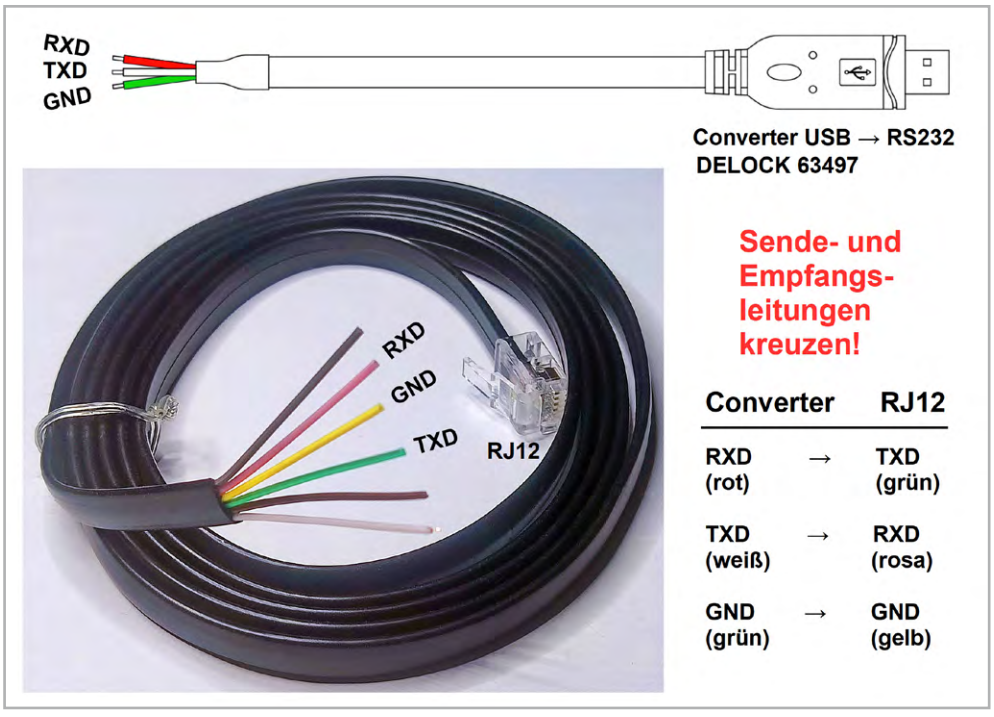


Bild 4: In einem ersten Anlauf wurde aus diesen Komponenten ein RS232-USB-Adapter selbst „gestrickt".

Bild 5: Links das von ACT erhältlich Adapterkabel und rechts die funktionsgleiche, aber nicht so schöne Do-it-yourself-Alternative.

Wenn eine USB-A-Buchse des PCs über eines der beschriebenen USB-RS232-Converter-Kabel mit der RJ12-RS232-Buchse des Controllers HBS57 korrekt verbunden ist, kann man das Programm LONGS_HBS_Pro_turn.exe aufrufen. Wie Bild 6 zeigt, fragt es zunächst, in welcher Sprache man es bedienen möchte. In der überwiegenden Zahl der Fälle wird man sich gegen die chinesische Variante und für Englisch entscheiden. Der darauf erscheinende Bildschirm fordert zur Suche des verwendeten COM-Ports auf. Das Ergebnis ist meist COM 3. Alsdann wird der Schiebeschalter rechts neben dem mit BaudRate beschrifteten Feld nach rechts geschoben (er nimmt bei erfolgter Aktivierung die Farbe grün an) und die Verbindung wird hergestellt. Das Textfeld rechts informiert nun darüber, welche Parameter aus dem Controller ausgelesen wurden.

Auto-Test

Für einen schnellen Auto-Test des Motors ohne Unterstützung durch den Raspi wird im gleichnamigen Feld zunächst der grüne Pfeil-Button „Download to Driver" gedrückt, wodurch die in diesem Feld eingestellten Parameter in den Controller geladen werden (Bild 7). Mit dem Schiebeschalter links im Feld „Auto_Test" wird der Motor mit diesen Parametern getartet.

Mit den voreingestellten Standardwerten sollte er nun mit den „Mäuseklaviertasten" SW3 bis SW6 in Stellung ON (Default) abwechselnd ca. 5 Sekunden lang 10 Umdrehungen nach rechts und anschließend nach links vollziehen. Wenn man mit den Parametern Accelaration (Beschleunigung), Position (Lage, 0-9000 Steps), Speed (Geschwindigkei, 0-1500) und Intervals (Zeitabstand zwischen den Drehrichtungs-

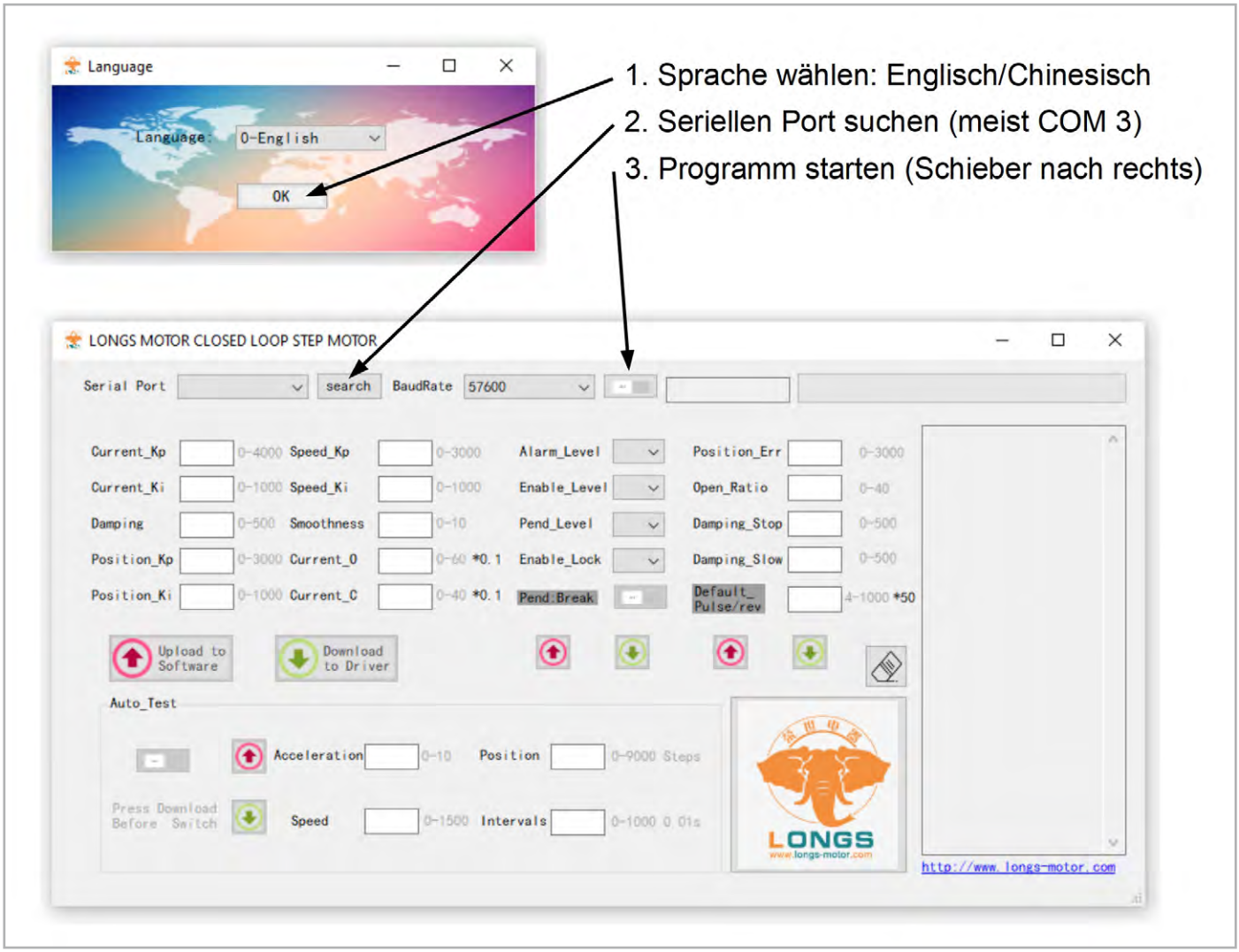


Bild 6: Weil das Debugging-Programm ACT_HBS_pro_Turn weder vom Hardwarelieferanten noch von ACT zu erhalten war, wurde die Version LONGS_HBS_pro_Turn verwendet.

Bild 7: Wenn die Verbindung zwischen der USB-Schnittstelle des Windows-PCs und der RS232-Schnittstelle des ACT-Treibers mit RJ12-Buchse hergestellt ist, liest die Software LONGS_HBS_pro_Turn.exe als Erstes die Treiberparameter aus.

änderungen, 0-100) etwas herumspielt, kann man einen recht guten Eindruck von den Fähigkeiten des Motors gewinnen. Zum Einschreiben eines geänderten Parameters in den Controller muss der Schiebeschalter im Feld „Auto-Test" zuerst deaktiviert, die grüne Pfeiltaste gedrückt und der Schiebeschalter wieder aktiviert werden. Darauf startet der Motor mit geänderten Parametern.

Jetzt ist auch ein guter Zeitpunkt, um sich von dem funktionierenden Nachholmechanismus für fehlende Schritte zu überzeugen. Dazu stellen wir Speed = 0, damit der Motor steht, weil er keine Schrittimpulse mehr erhält, aber weiterhin bestromt ist. Position_Err wird auf einen ausreichend großen Wert von z. B. 400 gestellt. Mit einer Zange drehen wir nun den Rotor um einen gewissen Winkel nach rechts oder links und sehen, dass mit dem Beenden der Dreh- und Haltekraft der Rotor wieder in seine Ausgangslage zurückkehrt (Bild 8).

Drehen wir den Rotor über die im Feld „Position_Err" vorgegebene Anzahl von Schritten, signalisiert uns das die rote Error-LED. Der Motor wird nun nicht mehr bestromt und lässt sich widerstandsarm drehen. Nach einem kurzen Ausschalten des 36-VDC-Netzteils ist der Alarm gelöscht.

Noch einmal: Dafür ist keine Ansteuerung durch den Raspberry Pi erforderlich!

Ansteuerung durch den Raspberry Pi

Nachdem man sich wie vorstehend beschrieben von der Betriebstüchtigkeit des Motors und Controllers überzeugt hat, kann man den Raspberry Pi „ans Ruder" lassen.

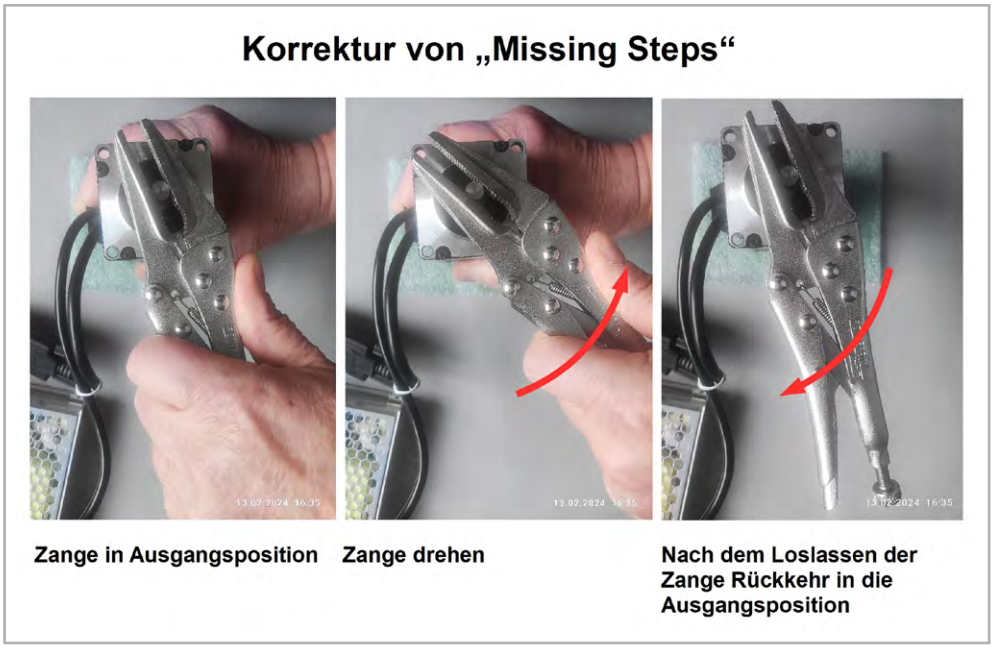


Bild 8: Mit einer Zange und etwas Kraft lässt sich der stehende bestromte Motor etwas verdrehen. Übt man keine Kraft mehr auf die Zange aus, wird diese wieder in die Ausgangsposition zurückgedreht. Damit wurden die „Missing Steps" nachgeholt.

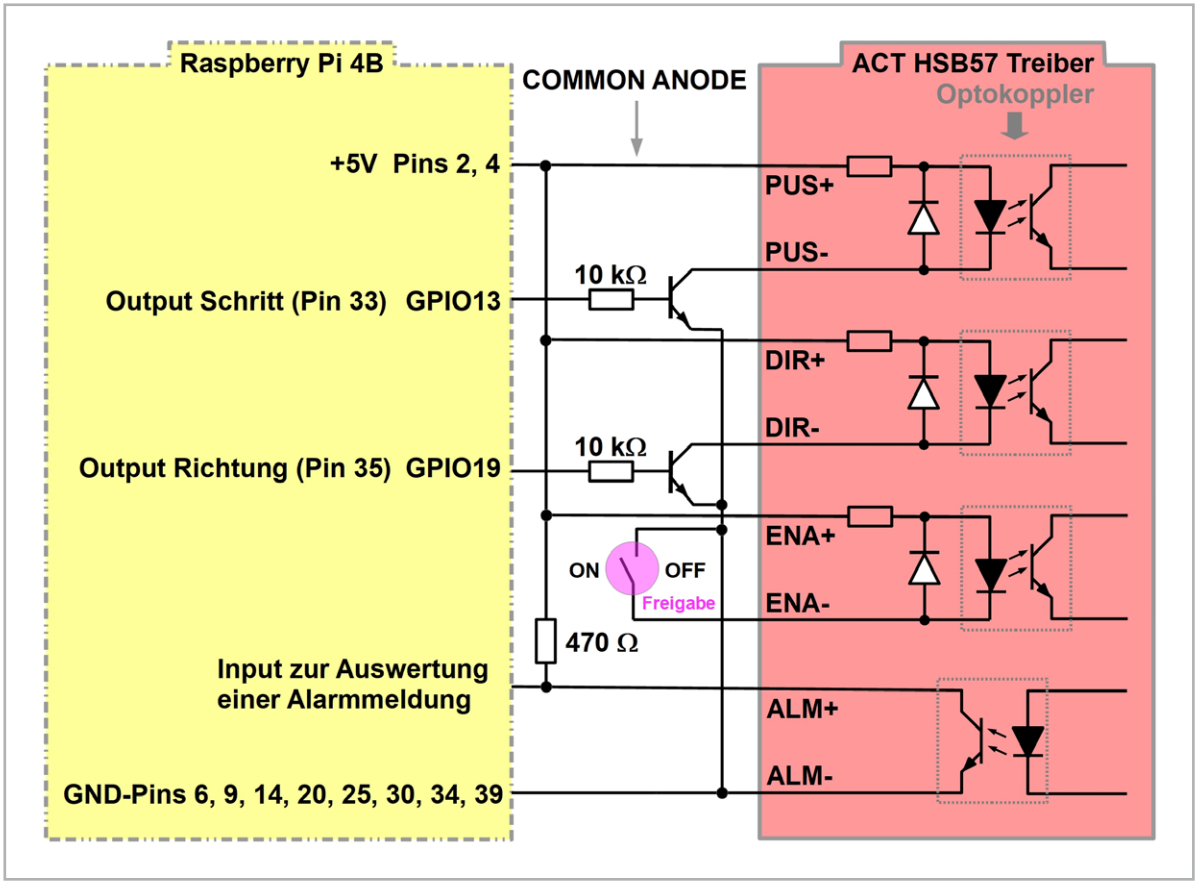


Bild 9: So steuert ein Raspberry Pi den Treiber ACT-HBS57. Weil die Treibereingänge zum Betrieb der Optokoppler-LEDs ca. 9 mA Strom benötigen, sollten die Ausgänge des Raspi über zwischengeschaltete Transistoren entlastet werden.

Wie man in Bild 9 sieht, werden die vom Raspberry Pi ausgegebenen Steuersignale dem ACT-HBS57-Treiber über galvanisch trennende Optokoppler zugeführt. Die Anoden der Sendedioden werden dabei auf Pluspotential gelegt, was die Bezeichnung „common anode" in der englischsprachigen Dokumentation erklärt.

Leider ist die Strombelastbarkeit der Raspberry-ausgänge auf 2-3 mA begrenzt. Die Stromaufnahme einer Optokoppler-LED liegt bei etwa 9 mA, sodass es sicherer ist, die Signaleingänge des HSB57 über Treibertransistoren anzusteuern. Mit den verwendeten npn-Typen BC547C mit einem Stromverstärkungsfaktor von 420 bis 800 muss ein GPIO-Ausgang nur noch ca. 10 bis 20 µA liefern, was ihn nicht stresst. Der Transistor kann einen Kollektorstrom von bis zu 100 mA liefern und ist den Eingangsströmen des HSB57 problemlos gewachsen. Mit einem auf HIGH geschalteten Raspberry-Port wird die betreffende Funktion ausgeführt.

Der Eingang ENA- (ENA: ENAble = Freigabe) des ACT-HSB57-Treibers kann über den Schalter Freigabe in Stellung ON aktiviert bzw. in Stellung OFF deaktiviert werden. OFF führt zur Abschaltung der Motorphasen und kann als Not-Aus verwendet werden. Die Motorwelle ist in diesem unbestromten Zustand einfach von Hand drehbar.

Ein vom ACT-HBS57-Treiber über Optokoppler ausgegebenes Alarmsignal lässt sich am Kontakt ALM+ abgreifen. Im Alarmfall leitet der Schalttransistor des Alarmausgangs, was durch den Raspberry Pi ausgewertet oder direkt zum Aktivieren eines akustischen oder optischen Signalgeräts verwendet werden kann.

Über DIP-Switch SW1 des HSB57 kann man auswählen, welche Flanke des Schrittsignals PUS+ (OFF = steigend ↑ oder ON = fallend ↓) den neuen Step auslösen soll. DIP-Switch S2 bestimmt die Drehrichtung des Motors: OFF = CCW (counterclockwise = entgegen dem Uhrzeigersinn), ON = CW (clockwise = im Uhrzeigersinn).

Python-Testskript

Diesmal soll der Test nicht darin bestehen, den Motor mit einer festen Geschwindigkeit nach links und rechts zu drehen. Vielmehr soll er nach dem Programmstart aus dem Stillstand mit zunehmender Geschwindigkeit nach rechts drehen und beim Erreichen einer Spitzendrehzahl wieder bis zum Stillstand abbremsen. Das setzt sich in die linke Drehrichtung fort, dann wieder in die rechte usw.

Das Python-Script in Bild 10 realisiert diese Funktion. Obwohl eigentlich recht gut kommentiert, eine kurze Beschreibung. In den Zeilen 1-7 das Übliche: Import benötigter Bibliotheken, Wahl der Broadcom-Pinbezeichnungen, Zuweisungen und Konfiguration der zugewiesenen Pins als Outputs (Ausgänge).

In Zeile 9 wird die Zahl der auszuführenden Schritte (number of steps: nst) vorgegeben. In Zeile 10 wird durch n = 0 festgelegt, dass die Anfangsdrehrichtung beim Eintritt in die Endlosschleife (Zeile 13 bis 30) rechts sein soll. In den Zeilen 13 bis 18 wird durch die Prüfung von n bestimmt, ob eine Links- oder Rechtsdrehung erfolgen soll, mit ent-

sprechender Konsolenausgabe. Das geschieht durch Modulo-2-Divisionen, mit denen festgestellt wird, ob der aktuelle Wert von n gerade (0, 2, 4, 6 ...) oder ungerade (1, 3, 5, 7 ...) ist. Bei geradem Wert von n wird in Zeile 14 der Variablen `Richtung = 0` zugewiesen, bei ungeradem Wert `Richtung = 1` und die Konsolenausgabe von print(nst, „Rechtsschritte“) bzw. print(nst, „Linksschritte“) veranlasst. Sollte die Drehrichtung nicht stimmen, kann sie mit DIP-Switch SW1 gewechselt werden.

In Zeile 19 wird das Ergebnis dieser Überprüfung dem DIR-Ausgang durch `GPIO.output(DIR, Richtung)` mitgeteilt und in der weiteren Abarbeitung der Endlosschleife verwendet. Es folgen zwei Schleifen 20 bis 24 und 25 bis 29, die die Beschleunigung und das Abbremsen in der jeweiligen Richtung bewirken. Dies geschieht dadurch, dass der jeweilige Schleifenindex als Kehrwert für die sleep-Anweisungen in den Zeilen 24 und 29 verwendet wird. In der ersten Schleife mit wachsendem Index i (i = 1 bis nst) wird wird der Schleifendurchlauf kürzer, was einer beschleunigten Drehung entspricht. In der zweiten Schleife wird der Schleifenindex j vom Endwert nst (Zahl der Schrittimpulse) der vorangegangenen Schleife ausgehend bei jedem Schleifendurchlauf um 1 bis auf den Endwert 1 dekrementiert. Der Kehrwert in der sleep-Anweisung wird also größer und die Drehgeschwindigkeit damit kleiner. Anschließend wird in Zeile 30 der Drehrichtungsindex n inkrementiert und wieder in Zeile 13 zurückgekehrt, wo ein neuer Schleifendurchlauf in der entgegengesetzten Drehrichtung beginnt.

Die DIP-Schalter SW3-6 werden in die Positionen on-on-on-off gebracht, was 1000 Impulsen/Umdrehung entspricht. Im Programmbeispiel wird durch die Wahl von nst = 10000 bestimmt, dass 10 volle Rotorumdrehungen beschleunigt und 10 volle Umdrehungen abgebremst in jede Drehrichtung vollzogen werden sollen. Das setzt sich so lange fort, bis das Programm vom Benutzer beendet wird.

Die Ansteuerimpulse für die Schritte des Steppermotors werden in den Zeilen 21 bis 23 bzw. 26 bis 28 gebildet. Aufgrund der Anweisungen `sleep(0.000001)` in den Zeilen 22 und 27 beträgt die Impulslänge 1 µs. ELV

ACT_HBS57_23SSM8440-EC1000_240216_Test.py

```
1  from time import sleep; import RPi.GPIO as GPIO  # Bibliotheken laden
2  GPIO.setmode(GPIO.BCM); GPIO.setwarnings(False)  # GPIO-Modus BCM; Warnungen unterdrücken
3
4  STEP = 13                    # Wertzuweisung für Variable STEP (step: Schritt)
5  DIR  = 19                    # Wertzuweisung für Variable DIR (direction: Richtung)
6  GPIO.setup(STEP, GPIO.OUT)   # GPIO-Pin STEP als Ausgang für Schrittimpulse festlegen
7  GPIO.setup(DIR, GPIO.OUT)    # GPIO-Pin DIR als Ausgang für Richtungsvorgabe festlegen
8
9  nst = 10000                              # number of steps (Schrittzahl) nach links/rechts
10 n = 0                                    # Startdrehrichtung "Rechts"
11
12 while True:                              # Endlosschleife
13     if n%2 == 0:                         # n gerade => rechts, n ungerade => links
14         Richtung = 1                     # Rechtsdrehung
15         print(nst, "Rechtsschritte")     # Konsolenausgabe
16     else:                                # n ungerade
17         Richtung = 0                     # Linksdrehung
18         print(nst, "Linksschritte")      # Konsolenausgabe
19     GPIO.output(DIR, Richtung)           # Drehrichtungswahl
20     for i in range(1, nst + 1, 1):       # Schleife für aktuelle Drehrichtung (Drehzahl steigt)
21         GPIO.output(STEP, 1)             # Stepimpuls an
22         sleep(0.000001)                  # Stepimpulslänge 1µs
23         GPIO.output(STEP, 0)             # Stepimpuls aus
24         sleep(1/i)                       # Abstand der Stepimpulse zunehmend
25     for j in range(i, 1, -1):            # Schleife für aktuelle Drehrichtung (Drehzahl fällt)
26         GPIO.output(STEP, 1)             # Stepimpuls an
27         sleep(0.000001)                  # Stepimpulslänge 1µs
28         GPIO.output(STEP, 0)             # Stepimpuls aus
29         sleep(1/j)                       # Abstand der Stepimpulse abnehmend
30     n += 1                               # Drehrichtungswechsel
```

Kommandozeile

```
>>> %Run ACT_HBS57_23SSM8440-EC1000_240216_Test.py

  10000 Rechtsschritte
  10000 Linksschritte
  10000 Rechtsschritte
  10000 Linksschritte
  10000 Rechtsschritte
```

Bild 10: Dieses kleine Python-Testskript lässt den Motor aus dem Stillstand auf eine Maximaldrehzahl beschleunigen und dann wieder auf null abbremsen. Anschließend wiederholt sich das Gleiche in die Gegenrichtung.