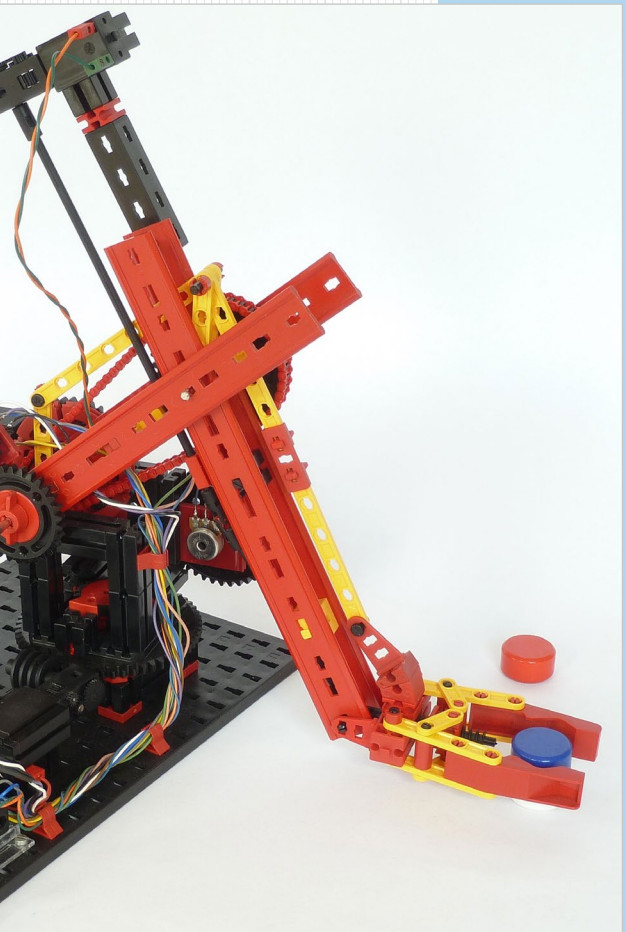
6 Der Greifer



Roboterarme bestimmen heute das Bild industrieller Fertigungsanlagen. Sie automatisieren Schweißprozesse, das Anbringen von Bauteilen und viele weitere sich wiederholende Tätigkeiten. Dabei arbeiten sie schneller und genauer als menschliche Arbeitskräfte, machen keine Fehler und benötigen keine Pausen. Schon im Jahr 1985 brachte fischertechnik einen Trainingsroboter auf den Markt, der es Heimcomputer-Hobbyisten ermöglichte, einen Einblick in die Welt der Industrieroboter zu erhalten und viel über ihre Mechanik und Programmierung zu erlernen [2].
Unser Greifer ist eine Hommage an diesen Trainingsroboter. Wir haben aber die Sensorik, Mechanik und vor allem die Programmierung vereinfacht. Das einfache und klare Arduino-Konzept zeigt hier seine ganze Stärke. Als Anwendung demonstriert unser Roboter die Lösung des klassischen Knobelspiels »Die Türme von Hanoi«.

6.1 Industrieroboter in Kinderzimmern

Wer schon immer gerne mit fischertechnik gespielt hat, weiß, wie viele kleine und große Entdeckungen man beim Experimentieren und Bau von Modellen machen kann. Tatsächlich haben Konstruktionssysteme ihr Innovationspotenzial in der Geschichte der Robotik noch deutlich eindrucksvoller unter Beweis gestellt. Nach heutigem ISO-8373-Standard ist ein Industrieroboter ein automatisch gesteuerter, frei programmierbarer Manipulator mit drei oder mehr Drehachsen. Laut der englischsprachigen Wikipedia wurde der erste solche Roboter im Jahr 1938 im *Meccano Magazin* vorgestellt (Abb. 6–1). Entwickelt wurde er von 1935 bis 1937 vom damals 21-jährigen Studenten *Griffith P. Taylor*, der ihm den Namen *Gargantua* gab [3, 4, 5]. Der frei programmierbare Ablauf der Bewegungen wurde auf einem Lochstreifen gespeichert. Um einen geordneten Stapel Bausteine zur abgebildeten Struktur aufzutürmen, benötigte Gargantua etwa 50 Minuten.

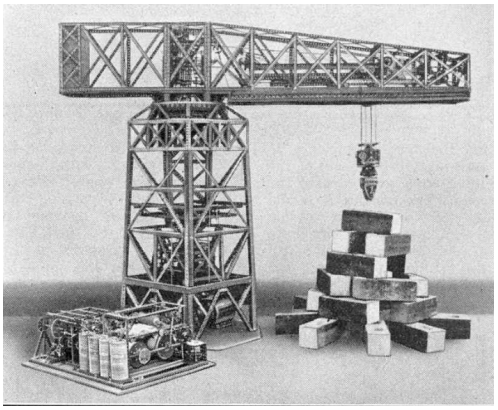


Abb. 6–1 Der Meccano-Roboterkran Gargantua aus dem Jahr 1938 [3]

Das erste Patent auf einen elektronisch gesteuerten Industrieroboter für den tatsächlichen Einsatz in der Industrie bekam *George Devol* (1912-2011) im Jahr 1954. Es dauerte allerdings noch bis zum Jahr 1960, bis das erste Exemplar verkauft wurde. In den folgenden Jahrzehnten verbreiteten sich die Roboter in Fertigungsanlagen immer schneller – vor allem in der Automobilindustrie.

Die ersten Heimcomputer kamen Ende der 70er-Jahre auf den Markt. Sie wurden in der ersten Hälfte der 80er-Jahre erschwinglich und eroberten die häuslichen Wohn- und Kinderzimmer. So kostete beispielsweise der Commodore C64 zu seiner Einführung im Jahr 1983 noch 1495 DM, doch halbierte sich der Preis noch im selben Jahr. Eine unvergleichlich kreative Pionierzeit begann.

Die Firma fischertechnik erkannte früh das Potenzial dieser Entwicklung und brachte schon 1984 einen genialen Computing-Baukasten auf den Markt, der die (elektro-)mechanischen Stärken des fischertechnik-Systems auf ideale Weise mit den neuen Möglichkeiten der Heimcomputer verband und Jugendlichen und Hobbyisten einen fundierten Einblick in die Welt der Automatisierung eröffnete (Abb. 6–2).

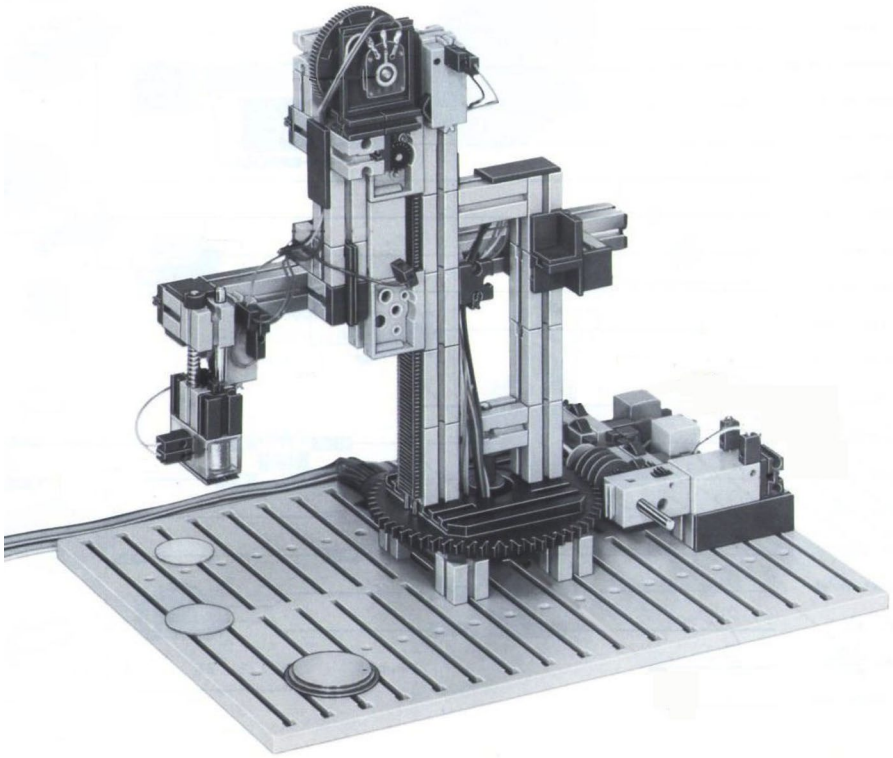


Abb. 6–2 »Die Türme von Hanoi« mit dem fischertechnik-Computing-Set aus dem Jahr 1984 [1]

Der Computing-Baukasten enthielt viele für die damalige Zeit spektakuläre Modelle, unter anderem einen Teach-in-Roboter, ein Grafik-Tablett, einen Polarkoordinaten-Plotter und einen Roboter, der die Lösung des Knobelspiels »Die Türme von Hanoi« vorführt.

Genau diese letzte, klassische Anwendung – »Die Türme von Hanoi« – greifen wir in diesem Kapitel auf. Unser Roboter stapelt allerdings die verschieden großen Scheiben der Türme nicht mit einem Elektromagneten um, sondern mit einer Greifhand.

Unser Greifer (Abb. 6–4) ist eine Hommage an den fischertechnik-Trainingsroboter, der schon ein Jahr später, 1985, dem Computing-Baukasten an die Seite gestellt wurde (Abb. 6–3). Bei beiden Robotern kommen vier Motoren zum Einsatz: drei S-Motoren zum Drehen des Körpers, des Oberarms und des Unterarms und ein kleinerer Mini- bzw. XS-Motor zur Steuerung des Greifers. Die Greifhand wird dabei stets parallel zur Grundfläche gehalten. Dies ist ein sehr guter Kompromiss zwischen Funktionalität und Komplexität.

Abb. 6–3 fischertechnik-Trainingsroboter aus dem Jahr 1985 [2]

Im Gegensatz zum fischertechnik-Trainingsroboter ist unser Greifer in der Lage, verschieden große Werkstücke sicher aufzunehmen und an anderer Stelle wieder abzusetzen. Erst diese Fähigkeit ermöglicht es, die Türme mit einer Greifhand umzustapeln.

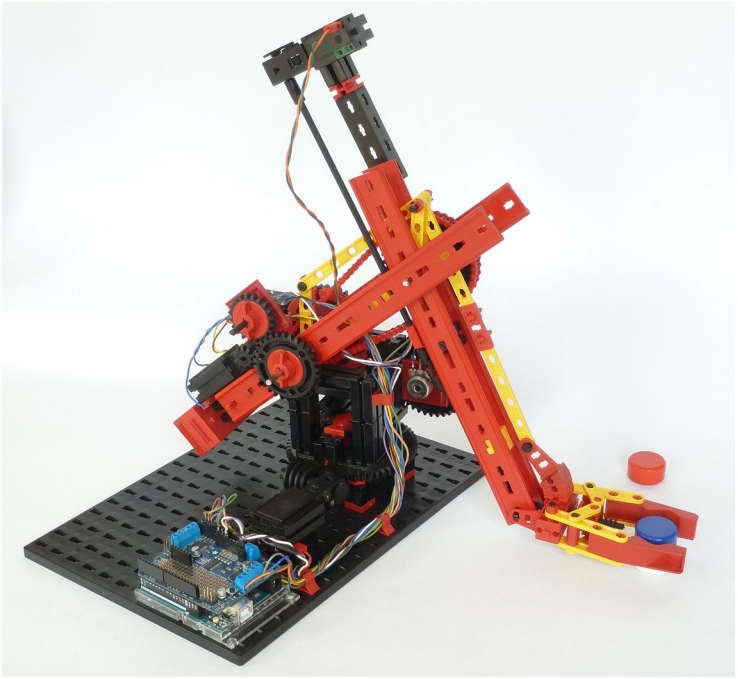


Abb. 6–4 Unser Arduino-gesteuerter Greifer

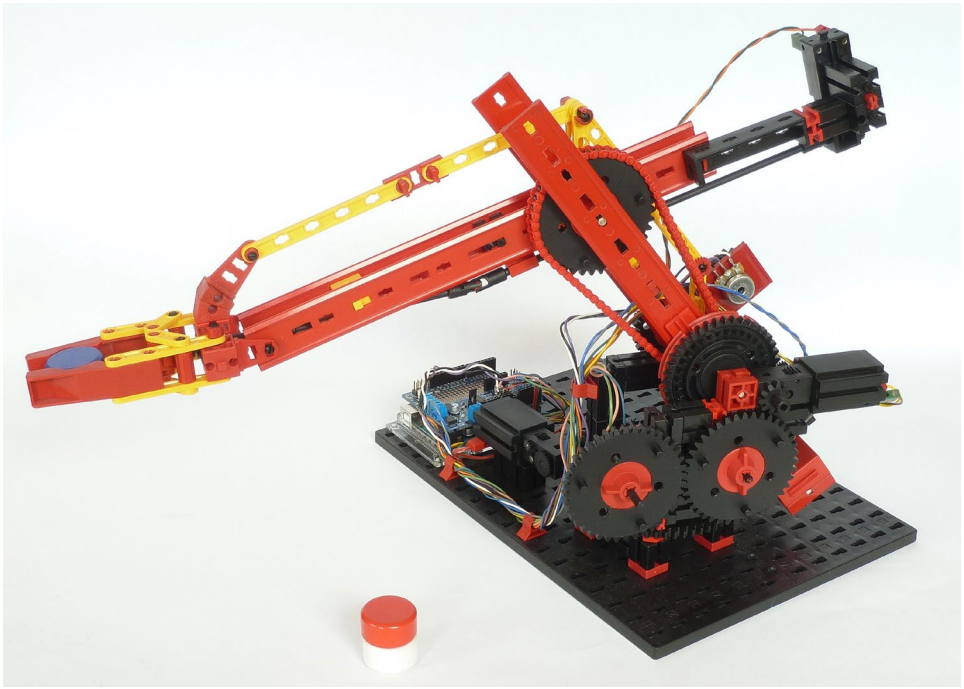


Abb. 6–5 Unser Greifer von der anderen Seite

Wer schon einmal das Innenleben von CD-Playern in den 80er-Jahren mit dem heutiger Player verglichen hat, kennt die Entwicklung bei solchen Produkten hin zu Effizienz, Übersichtlichkeit und Kostenersparnis. Diese allgemeine Entwicklung spiegelt sich auch beim Vergleich zwischen unserem Greifer und dem Trainingsroboter aus dem Jahr 1985 wider: Statt vier Tastern benötigen wir gar keinen, statt teurer Gabellichtschranken setzen wir als Drehgeber günstige Potenziometer ein, statt teurer Aluminiumprofile im Trainingsroboter verwenden wir günstige S-Laufschienen (36333).

Potenziometer kamen schon im Computing-Baukasten aus dem Jahr 1984 zum Einsatz, allerdings sind die heutigen Leitplastik-Potenziometer deutlich langlebiger als die damals verwendeten Kohleschicht-Potenziometer. Schon damals war ein Teach-in-Roboter eines der Hauptmodelle des Baukastens. Allerdings benötigte das Anlernen der Positionen noch eine Vielzahl von Tastern. Wir stellen hier ein einfacheres und schnelleres Verfahren dar, bei dem wir die Motoren auskoppeln und den Roboterarm manuell zu den gewünschten Positionen bewegen.

Die Laufschienen haben gegenüber den Aluminiumprofilen nicht nur den Vorteil, dass sie deutlich weniger kosten, sondern auch dass der Platz zwischen ihnen sinnvoll genutzt werden kann. So können wir den Unterarm unseres Roboters

ausbalancieren, indem wir den Greifer durch einen XS-Motor ansteuern, der auf der anderen Seite des Unterarms als Gegengewicht zum Greifer dient. Die entsprechende Antriebswelle verläuft zum Teil zwischen den Laufschienen. Auch den Oberarm haben wir durch ein Gegengewicht ausbalanciert. Damit wird es möglich, Schulter- und Ellenbogengelenke ohne Schnecken oder Schrauben direkt über U-Getriebe und Ritzel anzutreiben.

Den größten Fortschritt in Richtung Effizienz und Kostenersparnis gibt es jedoch bei der Steuerung. Der fischertechnik-Trainingsroboter wurde von einem der damaligen Homecomputer (Commodore, Schneider, Apple, Atari, Acorn, IBM) und einem fischertechnik-Computing-Interface gesteuert. Unser Greifer wird von einem autarken Arduino mit Motor Shield gesteuert. Der Arduino ist für Steuerungsaufgaben deutlich besser geeignet als die Kombination aus Heimcomputer und Interface, er ist kleiner und billiger als die damaligen Interfaces und lässt sich wesentlich komfortabler programmieren.

Wie schon oben beschrieben wird auch unser Greifer von vier Motoren bewegt. Damit kann er seine Hand überall in seinem Arbeitsbereich hinbewegen, die Ausrichtung der Hand bleibt dabei aber stets parallel zur Grundfläche und zeigt vom Roboter weg nach außen. Die meisten heute in der Industrie eingesetzten Roboterarme können auch die Ausrichtung der Hand frei bestimmen und setzen dafür zusätzliche Motoren ein.

Dass die Greifhand parallel zur Grundfläche bleibt, leistet die deutlich hervortretende, gelbe Verstrebung. Sie besteht aus zwei Parallelogrammen (Abb. 6–6). Das linke Parallelogramm sorgt dafür, dass die kurzen Streben oben im Ellenbogengelenk senkrecht stehen. Das rechte Parallelogramm bewirkt dann, dass der Anknüpfungspunkt der Streben an der Greifhand stets über der Drehachse der Hand liegt. Damit bleibt die Hand horizontal.

Zum Abschluss des Kapitels lassen wir unseren Greifer die Lösung des Spiels »Die Türme von Hanoi« demonstrieren. Dieses Knobelspiel ist zu einem Paradebeispiel geworden, wenn es darum geht, das Prinzip der Rekursion zu erklären. Und genau dafür ist unserer Meinung nach die Kombination aus einer klaren und direkten Umsetzung des Verfahrens in C und der konkreten, sichtbaren Demonstration durch den fischertechnik-Roboter ideal.

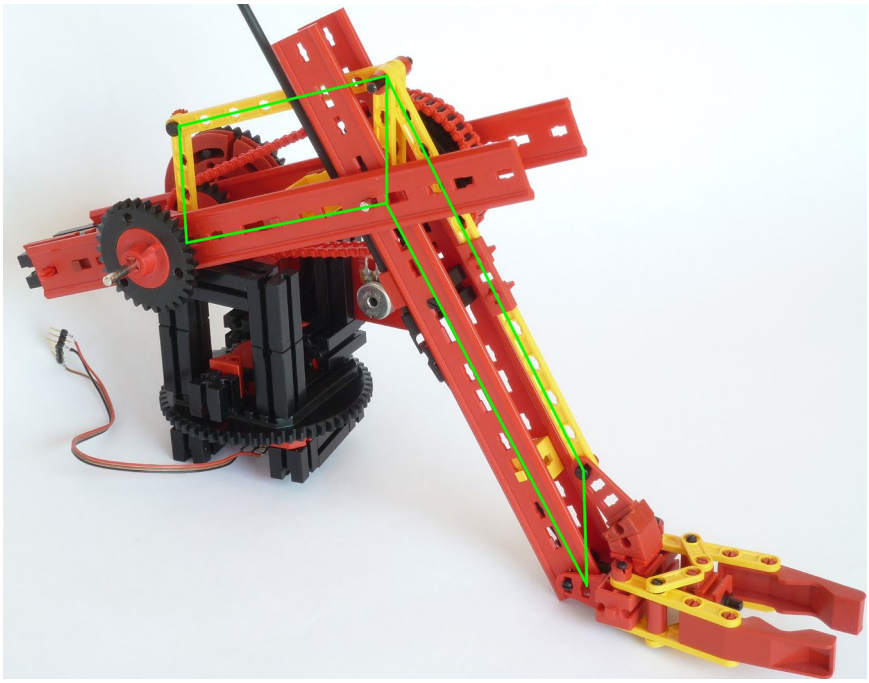


Abb. 6–6 Parallelführung der Greifhand

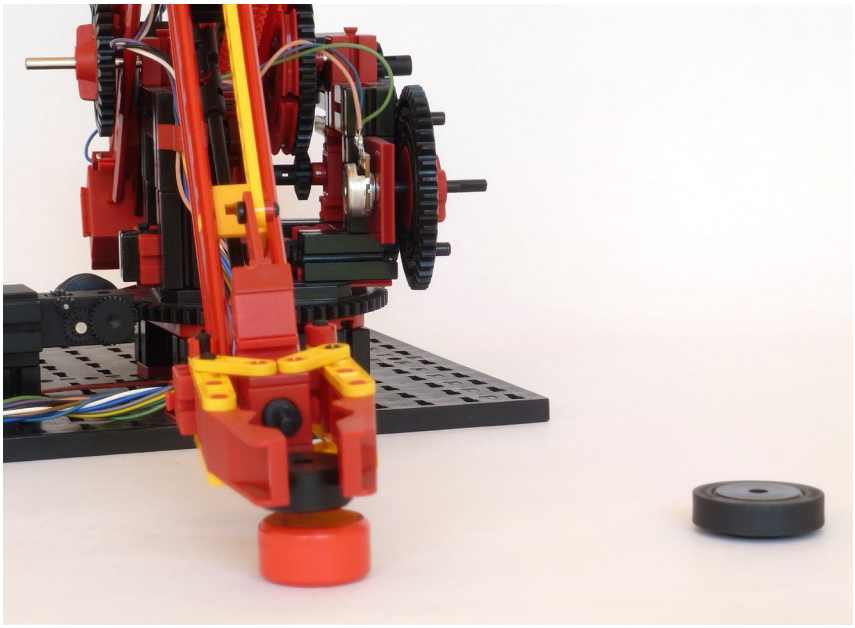


Abb. 6–7 Unser Greifer stapelt den Turm von Hanoi um.

Dank des einfachen Teach-in-Systems, der hohen Positioniergenauigkeit und der Fähigkeit, verschieden große Objekte zu greifen, kann der Roboter schnell für ganz andere Zwecke eingesetzt werden. Sei es, um Werkstücke zwischen Förderbändern zu bewegen, Fahrzeuge zu beladen oder gar Mühle zu spielen. Wie beim originalen fischertechnik-Trainingsroboter gibt es leicht umzusetzende Alternativen zur Greifhand: ein Elektromagnet oder ein Vakuumsauger, die wir beide im nächsten Kapitel vorstellen und einsetzen, oder auch ein einfacher Aufsatz zum Drehen von S-Riegeln. Der Fantasie sind hier keine Grenzen gesetzt.

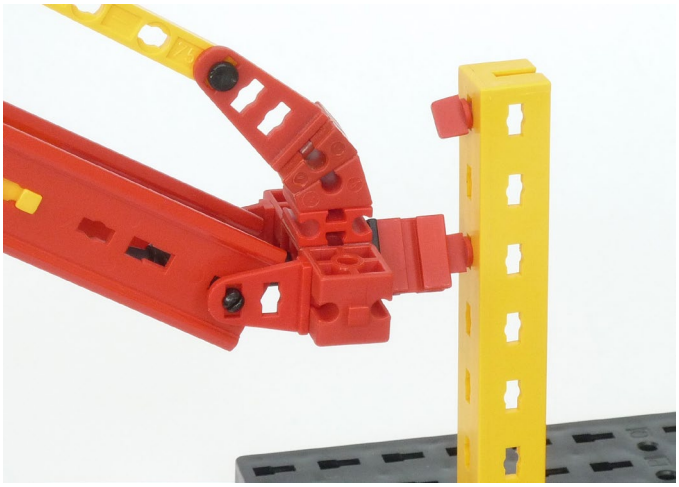


Abb. 6–8 Alternativer Aufsatz zum Schrauben von S-Riegeln

6.2 Steuerung mit Potenziometern

In diesem Abschnitt stellen wir die Methode vor, mit der wir die Gelenke unseres Greifers in diesem Kapitel und des Delta-Roboters im folgenden Kapitel ansteuern. Jede Drehachse ist mit einem Potenziometer ausgestattet. Einer der Außenkontakte wird mit Masse (GND), der andere mit 5 V verbunden. Das Potenziometer fungiert dadurch als Spannungsteiler. Die Spannung am mittleren Kontakt schwankt zwischen 0 und 5 V und ist näherungsweise proportional zum Drehwinkel. Dieser mittlere Kontakt wird mit einem analogen Eingang des Arduino verbunden. In unseren Sketchen können wir die Spannung an diesem Eingang messen. Wir erhalten durch einen Funktionsaufruf einen Wert zwischen 0 und 1023, den wir im Folgenden Potenziometerwert nennen.

Soll das Gelenk eine Bewegung ausführen, so geben wir einen Sollwert vor. Die Drehachse wird mit einem Motor so lange vor- oder rückwärts bewegt, bis

der Sollwert bis auf einen einstellbaren Fehler mit dem aktuellen Potenziometerwert (Istwert) übereinstimmt.

Diese Methode ist naheliegend, einfach und führt zu kurzen, übersichtlichen und verständlichen Programmen.

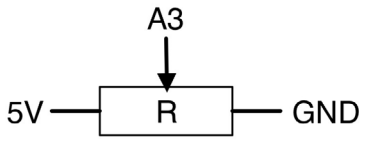


Abb. 6–9 Potenziometer als Spannungsteiler

Aufbau und Motorisierung

Wir verwenden hier die weitverbreiteten linearen Potenziometer PC16BU mit 10 kΩ der Firma OMEG. Die Achsen dieser Potenziometer haben genau wie die fischertechnik-Achsen einen Durchmesser von 4 mm und fügen sich damit problemlos ins fischertechnik-System ein.

Im aktuellen fischertechnik-Sortiment gibt es ebenfalls Potenziometer mit 4-mm-Achsen (32235). Diese originalen Potenziometer können auch verwendet werden; sie sind allerdings deutlich teurer.

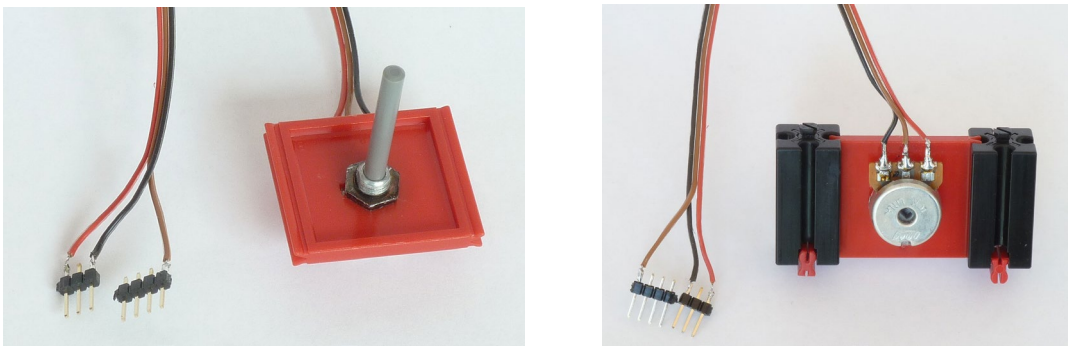


Abb. 6–10 OMEG-Potenziometer im Flachstein (32116)

Dass es Potenziometer im aktuellen fischertechnik-Sortiment gibt, hat den großen Vorteil, dass auch eine gebrauchsfertige Befestigungsmöglichkeit für Potenziometer zur Verfügung steht, nämlich der Flachstein (32116).

Wie in Abb. 6–10 dargestellt, stecken wir das Poti in den Flachstein und schrauben es mit der Mutter fest. Anschließend löten wir ein 14 cm langes, dreiadriges Kabel an die Kontakte des Potenziometers. Die anderen Enden der Adern werden an eine vierpolige und an eine dreipolige Stiftleiste mit 2,54-mm-Raster gelötet (*Male Pin Headers*). In Abb. 6–10 ist der linke Kontakt des Potis über die schwarze Ader mit Masse (GND), der rechte Kontakt über die rote Ader mit 5 V und der mittlere Kontakt über die braune Ader mit Pin A3 verbunden.

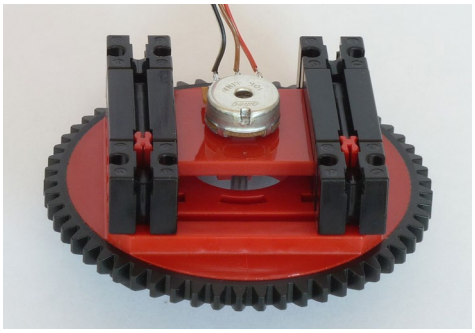


Abb. 6–11 Einbau des Potenziometers in den Drehkranz

In die Eckpositionen der roten Nuten des Drehkranzes schieben wir nun je einen BS 15. Anschließend stecken wir den Flachstein mit dem Potenziometer und den BS 30 ein.

Damit sich die Potiachse starr mit dem schwarzen Drehkranzoberteil dreht, schrauben wir auf ihr ein Zahnrad Z20 mit einer Nabe fest und sichern es durch zwei gegenüberliegende Sperrkeile. Für den Dauereinsatz sollten wir die Sperrkeile etwas stärker als abgebildet befestigen.

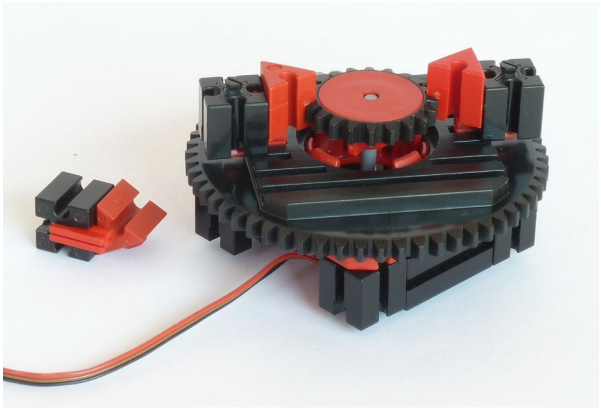


Abb. 6–12 Das drehbare Gelenk mit Potenziometer

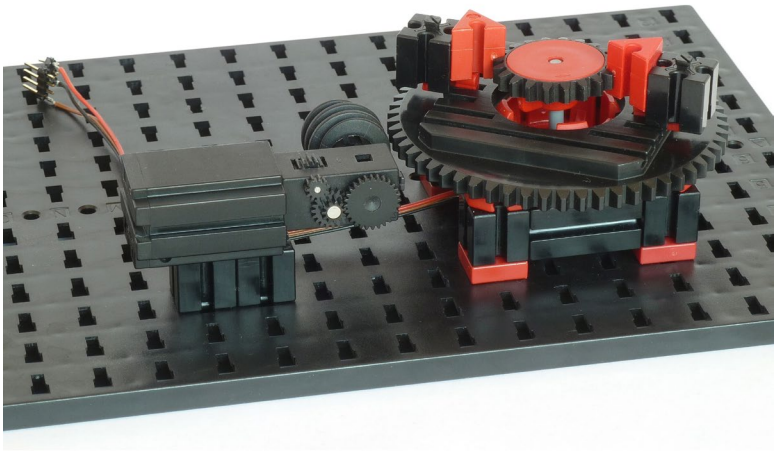
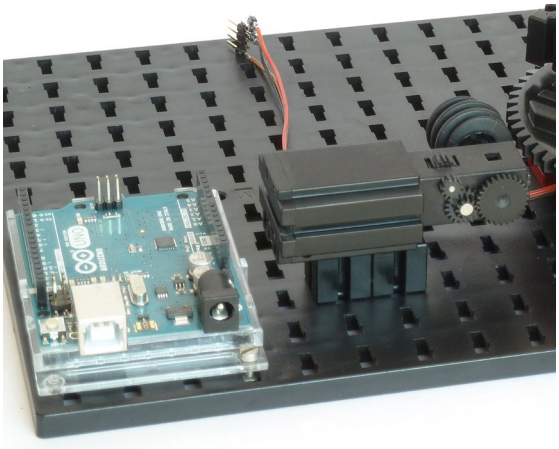


Abb. 6–13 Der ein- und auskoppelbare Antrieb des Gelenks



In eine Grundplatte 500 schieben wir nun in die Positionen D3, G3, D6 und G6 je einen roten BS 5 mit zwei Zapfen und in die Positionen K3 und L3 je einen BS 15 mit zwei Zapfen. Auf den vier BS 5 befestigen wir unseren Drehkranz mit dem Potenziometer und auf den zwei BS 15 einen S-Motor mit U-Getriebe und Rastschnecke (35072). Der Motor kann durch Verschieben einfach und schnell ein- und ausgekoppelt werden.

Abb. 6–14 Die Befestigung des Arduino Uno auf der Grundplatte

Anschluss der Elektronik

Ein originaler Arduino Uno R3 wird mit einer Grundplatte aus Plexiglas geliefert, die sich einfach mit einer M3-Schraube und einer passenden Mutter neben dem S-Motor auf der Grundplatte anschrauben lässt. Wer das Sunfounder-Gehäuse verwendet (siehe Kapitel 1), kann die Zapfen der angeklebten Bauplatten 30 × 45 in die Grundplatte schieben.

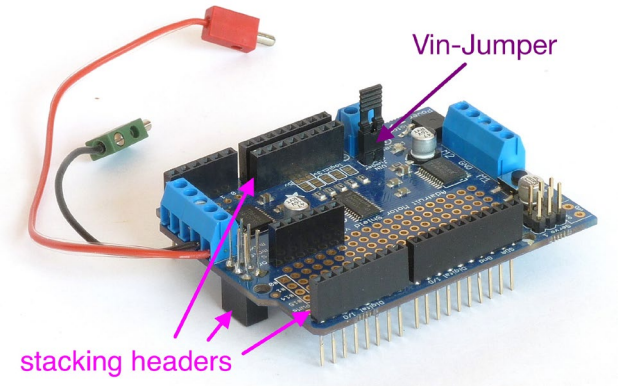


Abb. 6–15 Adafruit Motor Shield v2.3

Das Motor Shield v2.3 von Adafruit sollte man komplett mit sogenannten *Stacking Headers* versehen. Ein passendes Set gibt es in der Regel dort, wo man auch das Motor Shield erwerben kann. Durch diese Stift-/Buchsenleisten kann man Kabel oder weitere Shields ebenso auf das Motor Shield stecken wie auf den Arduino selbst.

Den S-Motor schließen wir an den Motorausgang M4 des Shields an. Dabei muss die Polung beachtet werden. Bei uns ist das rote Kabel 14 cm und das schwarze 7 cm lang. Die Stiftleisten stecken wir so in die Buchsenleisten des Shields, dass die rote Ader mit 5 V, die schwarze mit Masse (GND) und die braune mit Pin A3 verbunden ist.

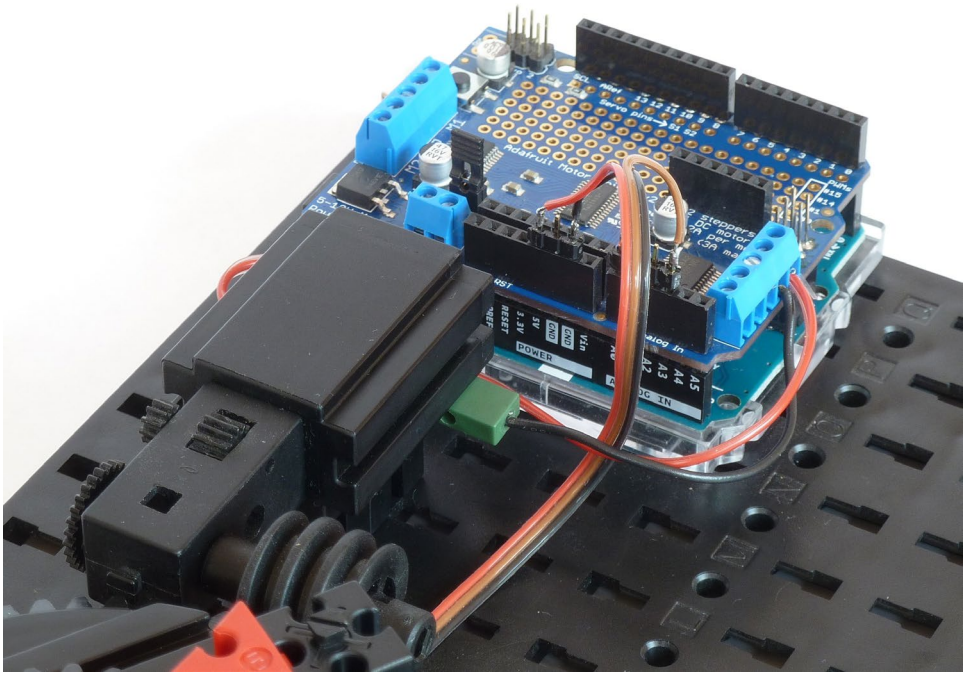


Abb. 6–16 Anschluss des Motors und des Potenziometers

Abb. 6–17 Adapter von 3,5-mm-Klinke auf 5,5-mm-Klinke

Zum Betrieb der Motoren benötigt das Motor Shield eine Gleichspannung von 9 V. Wer ein aktuelles fischertechnik-Steckernetzteil (505287) besitzt, kann dieses über einen Adapter von 3,5-mm-Klinke auf 5,5-mm-Klinke wie in Abb. 6–17 an den Arduino anschließen. Wichtig ist, dass der »Vin-Jumper« auf das Motor Shield gesteckt wird, dann versorgt das Steckernetzteil Arduino und Shield gemeinsam. Der Jumper wird zusammen mit dem Shield geliefert und ist in Abb. 6–15 deutlich zu erkennen. Statt des originalen fischertechnik-Netzteils können natürlich auch andere 9-V-Steckernetzteile verwendet werden. Zu beachten ist, dass an der Klinke innen Plus und außen Minus (GND) anliegt.

Schließlich verbinden wir den Arduino noch über ein USB-Kabel mit dem Computer.

Das Auslesen des Potis

In der Arduino-Entwicklungsumgebung stellen wir unter dem Reiter *Werkzeuge* das verwendete Arduino-Modell und den zugehörigen USB-Port ein. Wir öffnen unseren ersten Sketch *Drehkranz_Poti* von der Webseite zum Buch.

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  Serial.println(analogRead(3));
  delay(200);
}
```

Listing 6–1 Sketch Drehkranz_Poti

Die Funktion `setup` wird nur einmalig ausgeführt, und zwar nach dem Hochladen eines Sketches, nach dem Start des seriellen Monitors, nach einem Reset oder direkt nach dem Anlegen der Versorgungsspannung. In ihr wird eine serielle Verbindung zur Datenübertragung zwischen dem Arduino und dem Computer initialisiert. Die Zahl 115200 gibt die Übertragungsrate in Bit/s an.

Die Funktion `loop` dagegen wird anschließend fortlaufend ausgeführt. In ihr wird zunächst mittels `analogRead(3)` der Potenziometerwert gemessen. Das Ergebnis ist ein Wert zwischen 0 und 1023, der durch die Funktion `Serial.println` über die serielle Verbindung zum Computer übermittelt wird. Anschließend wird 200 ms gewartet, und so geht es weiter: Auslesen und Übertragen erfolgen im Wechsel.

Durch Klicken auf den kreisförmigen Button mit dem Pfeil nach rechts in der Arduino-IDE wird der Sketch übersetzt und auf den Arduino hochgeladen.

Um die Messergebnisse sehen zu können, starten wir den seriellen Monitor der Arduino-IDE, indem wir auf den Button mit der Lupe oben rechts im Fenster klicken. Es öffnet sich ein Fenster wie in Abb. 6–18.

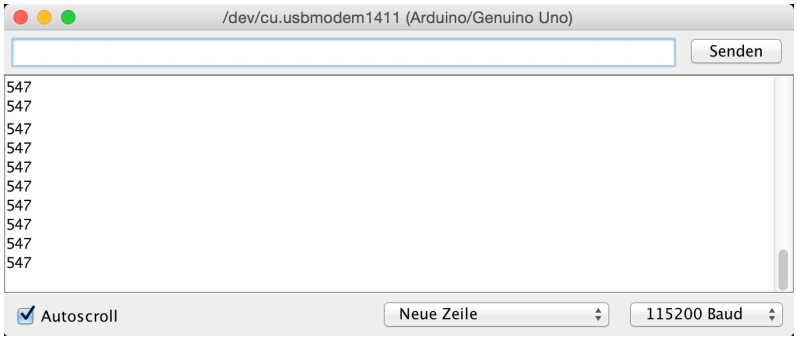


Abb. 6–18 Die gemessenen Potenziometerwerte werden auf dem seriellen Monitor angezeigt.

Drehen wir nun bei ausgekoppeltem Motor das Drehkranzoberteil im Uhrzeigersinn, so werden die angezeigten Werte größer. Sobald sich die Werte dem Maximalwert 1023 nähern, sollte man einmal vorsichtig bis an den Anschlag des Potenziometers weiterdrehen. Jetzt ist eine gute Gelegenheit, den Drehbereich des Oberteils durch Lösen der Sperrkeile anzupassen. Die Potis erlauben einen Drehwinkel von 270°. Drehen wir das Oberteil gegen den Uhrzeigersinn, so werden die Werte wieder kleiner. Auch hier sollte man das Potenziometer einmal vorsichtig bis an den Anschlag drehen.

Die Motorsteuerung

Wir sind jetzt bereit für die Motorsteuerung des Drehkranzes und laden dazu den Sketch *Drehkranz_Motor* von der Webseite zum Buch. Bevor wir ihn erklären, probieren wir ihn erst einmal aus. Dazu übersetzen wir ihn, laden ihn auf den Arduino hoch und starten den seriellen Monitor (Abb. 6–19).

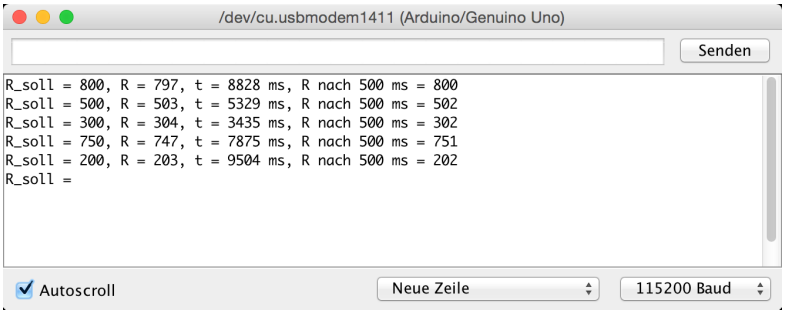


Abb. 6–19 Die Steuerung des Drehkranzes im seriellen Monitor

Nach der Eingabe eines Sollwertes bewegt sich der Drehkranz und kommt dann zum Stoppen. Im seriellen Monitor wird der Potenziometerwert angezeigt, kurz nachdem der Motor gestoppt wurde, sowie die Dauer der Bewegung. Dann wird 500 ms gewartet und der Potenziometerwert noch einmal ausgelesen. Dieser Wert dient zur Kontrolle, ob das unvermeidbare kurze Weiterlaufen des Motors nach dem Abschalten den Drehkranz über den gewünschten Sollwert hinaus dreht. Da wir eine Schnecke zum Antrieb des Drehkranzes verwenden, ist dies nicht der Fall.

Das Protokoll zeigt, dass der Sollwert durch unseren Sketch bis auf ±2 genau angefahren wird. Da unser Poti einen Drehbereich von 270° besitzt und wir 1024 Potenziometerwerte haben, erhalten wir eine Wiederholgenauigkeit von unter ±1°, die für unsere Zwecke vollkommen ausreichend ist. Wiederholgenauigkeit bedeutet hierbei, dass eine einmal angefahrene Position bei einem erneuten Anfahren mit dem gleichen Sollwert wieder erreicht wird. Es geht hier noch nicht darum, dass wir den gemessenen Potenziometerwert in den Drehwinkel des Drehkranzes umrechnen. Dabei müssten wir die kleine, aber für viele Zwecke nicht vernachlässigbare Nichtlinearität des Potis und des A/D-Wandlers im Arduino berücksichtigen. Darauf werden wir in Kapitel 7 genauer eingehen.

Schauen wir uns nun den Sketch an. In den ersten Zeilen sorgen die `#include`-Anweisungen dafür, dass die Bibliotheken eingebunden werden, die für die Kommunikation zwischen Arduino und Motor Shield benötigt werden:

```
#include <Wire.h>
#include <Adafruit_MotorShield.h>

Adafruit_MotorShield MotorShield = Adafruit_MotorShield();
Adafruit_DCMotor *M = MotorShield.getMotor(4);
```

Listing 6–2 Einbindung der Bibliothek MotorShield

Die dritte und vierte Zeile erzeugen die Objekte `MotorShield` und `M`. Wer mit objektorientierter Programmierung und dem Zeigerkonzept in C nicht vertraut ist, braucht hier keine Angst zu bekommen: Während die beiden Zeilen auf den Neuling durch das häufige Vorkommen des Wortes Motor Shield kryptisch bis unsinnig wirken können, ist die Anwendung der erzeugten Objekte und ihrer Methoden im weiteren Verlauf des Sketches intuitiver. Das Argument 4 des Methodenaufrufs `MotorShield.getMotor(4)` bedeutet, dass der Motor `M` am Motorausgang M4 des Motor Shield angeschlossen ist.

In der einmalig ausgeführten Funktion

```
void setup() {
  Serial.begin(115200);
  MotorShield.begin();
  M->setSpeed(0);
}
```

Listing 6–3 Initialisierung des Motor Shield

werden die serielle Schnittstelle und das Motor Shield initialisiert sowie die Geschwindigkeit des Motors `M` auf 0 gesetzt.

Wir schauen uns jetzt die fortlaufend ausgeführte Funktion `loop()` genauer an:

```
void loop() {
  Serial.print("R_soll = ");
  while (Serial.available() == 0);
  int R_soll = Serial.parseInt();
  Serial.read();
  Serial.print(R_soll);
```

Listing 6–4 Einlesen des Potenziometer-Sollwertes

Im ersten Teil wird zunächst der Potenziometer-Sollwert über den seriellen Monitor eingelesen. Dazu ist es wichtig, dass sich der serielle Monitor im Modus *Neue Zeile* befindet (Abb. 6–18). Was immer man in die Eingabezeile des seriellen Monitors tippt, wird erst in den seriellen Puffer des Computers übertragen, wenn die Eingabetaste gedrückt wurde. Durch die Zeile

```
while (Serial.available() == 0);
```

wartet der Arduino so lange, bis neue Zeichen im Puffer des Computers vorhanden sind. Die Funktion `Serial.parseInt` versucht dann, möglichst viele Zeichen im Puffer in eine ganze Zahl umzuwandeln. Das erste Zeichen, das nicht ins Muster passt, wird als Trennzeichen interpretiert. In unserem Fall wird es das Newline-Zeichen sein, das vom Drücken der Eingabetaste stammt. Dieses Zeichen selbst muss nun noch durch `Serial.read();` aus dem Puffer entfernt werden.

Mit dem nächsten Teil der Funktion `loop()` wird durch den Funktionsaufruf in der mittleren Zeile der Drehkranz gedreht:

```
unsigned long start = millis();
drehe(R_soll);
unsigned long dauer = millis() - start;
```

Listing 6–5 Drehen des Drehkranzes

Das Argument gibt den Sollwert an, der am Ende der Drehung ausgelesen werden soll. Die erste und die letzte Zeile dienen zur Zeitmessung: Die Funktion `millis()` liefert die Millisekunden zurück, die seit dem Start des Sketches vergangen sind. Der Datentyp ist `unsigned long`, also eine vorzeichenlose 32-Bit-Zahl.

Im letzten Teil der Funktion `loop ()` wird der Potenziometerwert angezeigt, kurz nachdem der Motor gestoppt wurde, die Dauer der Bewegung und der Potenziometerwert nach 500 ms:

```
  Serial.print(", R = "); Serial.print(analogRead(3));
  Serial.print(", t = "); Serial.print(dauer);
  Serial.print(" ms, ");
  delay(500);
  Serial.print("R nach 500 ms = "); Serial.println(analogRead(3));
}
```

Listing 6–6 Anzeige der Potenziometerwerte und der Bewegungsdauer

Sehen wir uns jetzt die entscheidende Funktion `drehe()` an:

```
void drehe(int R_soll) {

  int R;
  int toleranz = 3;
  boolean aktiv = true;

  while ( aktiv ) {
    R = analogRead(3);
    if (R > R_soll + toleranz) {
      M->setSpeed(255);
      M->run(BACKWARD);
    }
    else if (R < R_soll - toleranz) {
      M->setSpeed(255);
      M->run(FORWARD);
    }
    else {
      M->setSpeed(0);
      aktiv = false;
    }
  }
}
```

Listing 6–7 Funktion »drehe()«

Diese Funktion dreht den Drehkranz, bis der aktuelle Potenziometerwert bis auf eine einstellbare Toleranz mit dem übergebenen Sollwert übereinstimmt. Dazu werden zu Beginn die benötigten Variablen `R`, `toleranz` und `aktiv` deklariert. Obwohl die Variable `toleranz` vom Sketch nicht verändert wird, haben wir sie nicht als Konstante oder statische Konstante deklariert, da wir uns nicht darauf festlegen wollen, dass sie in erweiterten Varianten der Funktion unveränderbar bleiben muss.

In der `while`-Schleife wird zunächst der aktuelle Potenziometerwert ermittelt und überprüft, ob er größer als der Sollwert plus der Toleranz ist. Ist das der Fall, wird der Motor mit maximaler Geschwindigkeit rückwärts laufen gelassen. Andernfalls wird überprüft, ob der aktuelle Potenziometerwert kleiner als der Sollwert minus der Toleranz ist. Falls ja, wird der Motor mit maximaler Geschwindigkeit 255 vorwärts laufen gelassen. Ansonsten liegt der aktuelle Potenziometerwert im Toleranzintervall um den Sollwert. Der Motor wird dann gebremst und die boolesche Variable `aktiv` wird auf `false` gesetzt. Dadurch wird die Schleife beendet.

Unser Drehkranz mit Potenziometer und der einfachen Funktion `drehe()` kann in vielen verschiedenen Anwendungskontexten eingesetzt werden. Wir verwenden ihn im Folgenden als Basis für unseren Greifer.

6.3 Der Aufbau des Greifers

Der Körper

Der Körper des Roboters wird in der Robotik gelegentlich auch *Karussell* genannt. Der Drehkranz aus dem vorigen Abschnitt bildet die Grundlage. Wir entfernen allerdings zunächst einen der beiden Sperrkeile, den anderen befestigen wir etwas stärker. In den Abb. 6–20 und 6–21 sieht man die ersten beiden Baustufen.

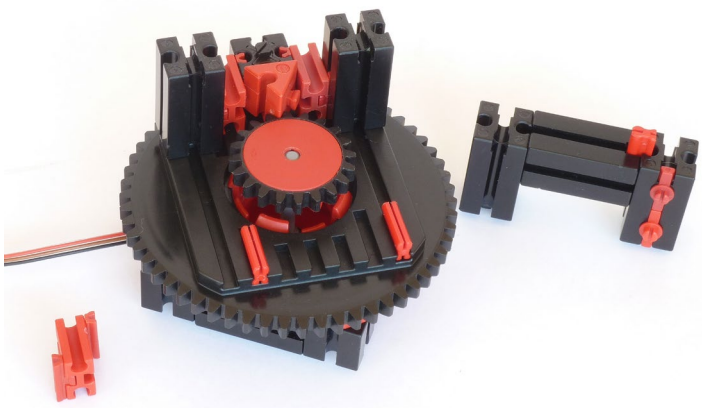


Abb. 6–20 Erste Baustufe des Roboter-Körpers

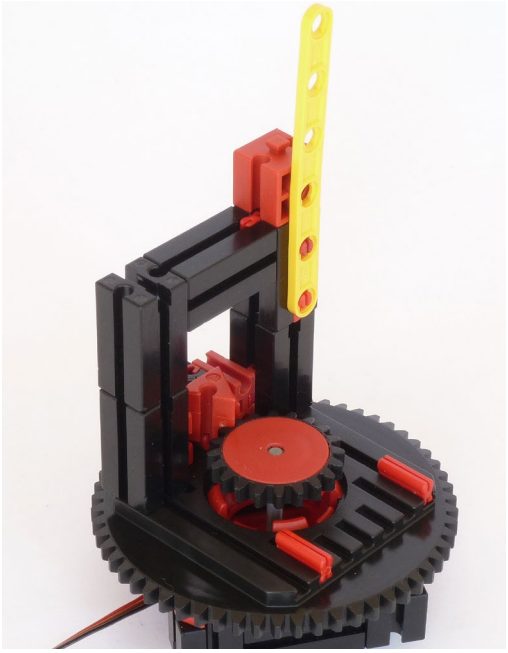


Abb. 6–21 Zweite Baustufe des Roboter-Körpers

Dem verbliebenen Sperrkeil gegenüber liegt ein Block mit den Lagern. Wir bauen ihn aus zwei Teilen zusammen.

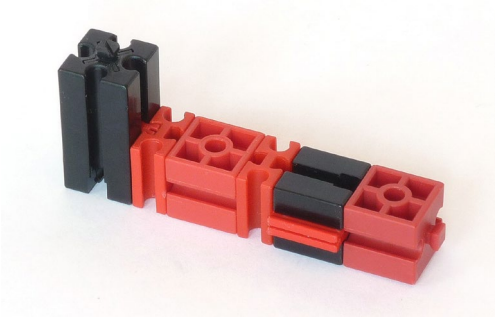


Abb. 6–22 Erster Teil des Lagerblocks

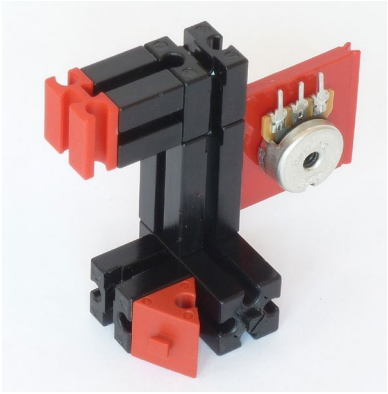


Abb. 6–23 Zweiter Teil des Lagerblocks