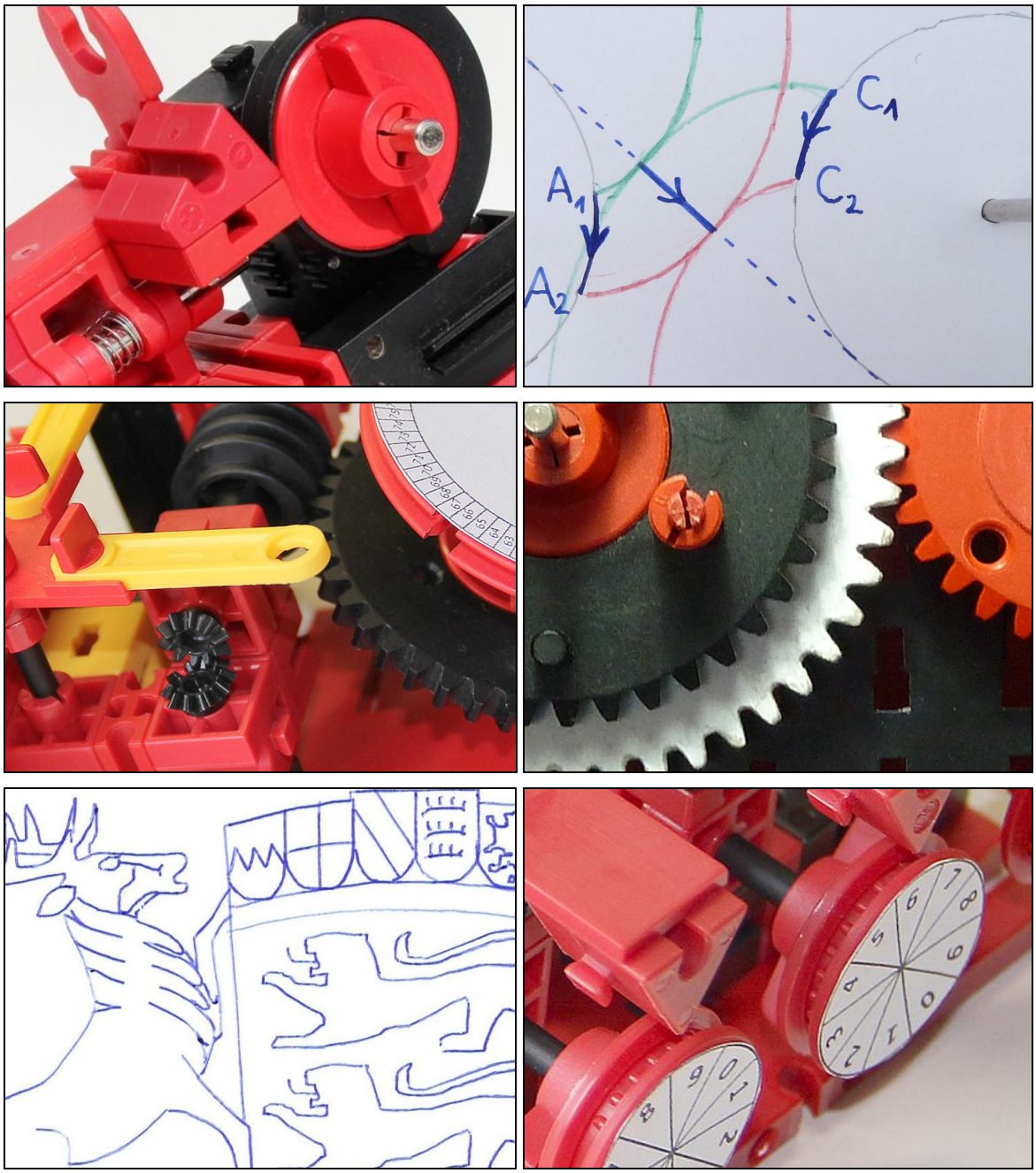
ft:pedia

Heft 1/2012



Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

Aufwind

Dirk Fox, Stefan Falk

Keine Frage – *wir* können uns unsere Freizeit kaum noch ohne fischertechnik vorstellen. Zumindest dann, wenn das Wetter schlecht und alle Bücher ausgelesen sind. Aber da sind ja noch die vielen anderen, die ohne solche glücklichen Bastelstunden ihr armseliges Dasein fristen müssen… sei es, weil sie fischertechnik nicht kennen, sei es, weil ihr Kinderzimmer so mit Computerspielzeug und Fertigware voll gestellt ist, dass nicht einmal mehr eine 1000er Box hineinpasst, oder sei es, weil sie einfach nur das Lieblingsspielzeug ihrer Kindheit aus den Augen verloren haben.

Die Zahl dieser verlorenen Seelen zu mindern war eines der Motive für die Gründung der ft-Community, des fischertechnik-Forums, der Convention und anderer ft-Modell-Schauen und schließlich auch der ft:pedia. Dieses vielfältige Engagement scheint Früchte zu tragen: Noch nie stieg die Zahl der in der ft-Community publizierten Modelle so rasch, nahmen die registrierten neuen Forums-Mitglieder so rapide zu – und war eine Convention so gestopft voll. Die Halle in Erbes-Büdesheim platzte 2011 schier aus allen Nähten.

Auch die ft:pedia erfreut sich reißenden Absatzes – in den ersten 12 Monaten erreichte sie eine verbreitete Auflage von insgesamt über 8.500 Exemplaren – davon fielen allein 3.500 Downloads in das erste Quartal 2012. Alles Hinweise auf die zarten Knospen eines großen „Comebacks“ von fischertechnik?

An diesem Aufwind haben sicher die fischerwerke mit der Flexschienen-Innovation und dem Dynamics-Kasten großen Anteil. Und bald dürften auch die neuen Kästen dazu beitragen, die Anfang Februar auf der Spielwarenmesse vorgestellt wurden. Sie enthalten neben ansprechenden Modellen interessante Neuerungen. So positiv jedenfalls wurden neue Kästen von der kritischen Fan-Gemeinde seit langem nicht mehr aufgenommen.

Die Voraussetzungen für die weitere Ausbreitung des wundervollen „fischertechnik-Virus“ waren selten so gut wie heute. Wir hoffen, dass auch die Beiträge in dieser fünften Ausgabe der ft:pedia die Infektionsgefahr wirksam erhöhen.

Damit uns das auch weiterhin gelingt, brauchen wir allerdings euch:

- Bitte schreibt uns im Forum, was wir besser machen können, vor allem aber, welche Themen ihr euch für künftige Ausgaben wünscht.
- Bitte sprecht uns an, wenn ihr selber einen Beitrag schreiben möchtet. Auch „Erstschreiber“ brauchen keinerlei Scheu zu haben; wir unterstützen euch gerne.

Jetzt aber ran an die Kästen. Zu Chancen und Förderwirkungen fragt bitte eure Eltern oder Physiklehrer…

Beste Grüße,
Euer ft:pedia-Team

P.S.: Am einfachsten erreicht ihr uns unter ftpedia@ftcommunity.de oder über die Rubrik *ft:pedia* im Forum der ft-Community.

Inhalt

Termine

Was?	Wann?	Wo?
Fan Club Tag	08.07.2012 (10-16 Uhr)	fischerwerke Waldachtal
Convention 2012	29.09.2012	Erbes-Büdesheim

Hinweise

Andreas Tacke und seine Kinder warben am 02.12.2011 in einem n-tv-Bericht eindrucksvoll für fischertechnik (Video).

Mitte März sind die Fan-Club-News 1/2012 erschienen.

Impressum

http://www.ftcommunity.de/ftpedia

Herausgeber: Dirk Fox, Ettlinger Straße 12-14, 76137 Karlsruhe und Stefan Falk, Siemensstraße 20, 76275 Ettlingen

Autoren: Stefan Falk (steffalk), Dirk Fox (Dirk Fox), Thomas Püttmann (geometer).

Projekt

HP-GL-Plotter (Teil 2)

Dirk Fox

Im ersten Teil des Beitrags wurde die Konstruktion der „Hardware“ des HP-GL-Plotters vorgestellt [1]. In diesem zweiten Teil folgt eine Erläuterung des Steuerprogramms – der „Plotter-Software“ – in Robo Pro. Sie erlaubt das Einlesen und Plotten von (leicht modifizierten) HP-GL-Dateien.

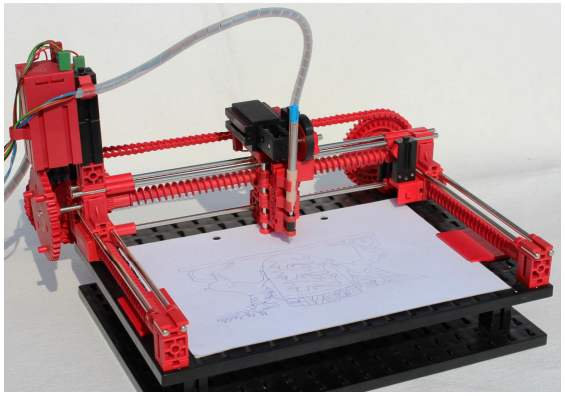


Abb. 1: Gesamtansicht des Plotters

Bei der Entwicklung des Steuerungs-Programms für den HP-GL-Plotter (Abb. 1) verfolgte ich dieselben Zielsetzungen wie bei der Plotter-Hardware [1]: Das Programm sollte

- keine Spezialinstallationen oder besondere Programmierkenntnisse voraussetzen;
- möglichst klein, einfach und verständlich sein, also mit wenigen, gut strukturierten Funktionen auskommen;
- leicht an Varianten oder auch ganz andere Plotter-Hardware angepasst werden können;
- ein möglichst standardisiertes Vektorgrafik-Dateiformat einlesen und verarbeiten können und
- insbesondere hinsichtlich der Plot-Geschwindigkeit optimiert sein.

Um diese Zielsetzung zu erfüllen, musste die Implementierung in Robo Pro erfolgen, auch wenn damit bestimmte Einschränkungen und Geschwindigkeitseinbußen verbunden waren.

Als Format für die einlesbaren Grafikdateien wählte ich die Plotter-Kommandosprache HP-GL/2 (*Hewlett-Packard Graphics Language*) [2], eine sehr intuitive und übersichtlich strukturierte Sprache für Stiftplotter, die sich Anfang der 90er Jahre als Standard-Format für Plot-Dateien von Vektorgrafiken durchsetzen konnte.

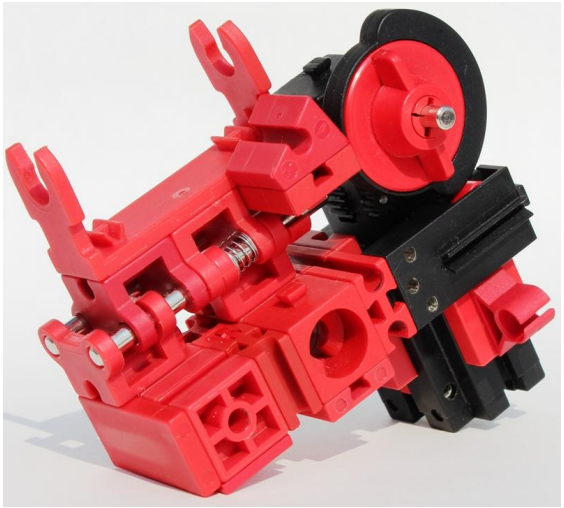


Abb. 2: Detailansicht des Schreibkopfs

Noch heute können nicht nur viele Grafik-Programme Vektorgrafiken als HP-GL-Dateien exportieren, sondern es finden sich auch zahlreiche Tools im Internet, die

andere Vektorgrafik-Formate, wie beispielsweise das *Drawing* Format (`.dwg`), das *Drawing Interchange Format* (`.dxf`) oder das *Design Web Format* (`.dwf`) in eine `.hpgl`- bzw. `.plt`-Datei konvertieren.

HP-GL

Geschichte

HP-GL wurde von Hewlett-Packard als Sprache zur einheitlichen Ansteuerung der zahlreichen Plotter-Modelle entwickelt. Die älteste mir bekannte Referenz ist das „HP-GL Programmer's Reference Manual" aus dem Jahr 1984 [3], aber die Ursprünge von HP-GL reichen in die frühen 70er Jahre zurück. Die erste Version von HP-GL stammt von Norm Johnson und Dale Shapper; sie diente zur Ansteuerung der Flachbett-Plotter-Modellfamilie HP 9872, den ersten Mehrfarb-Plottern (Abb. 3).

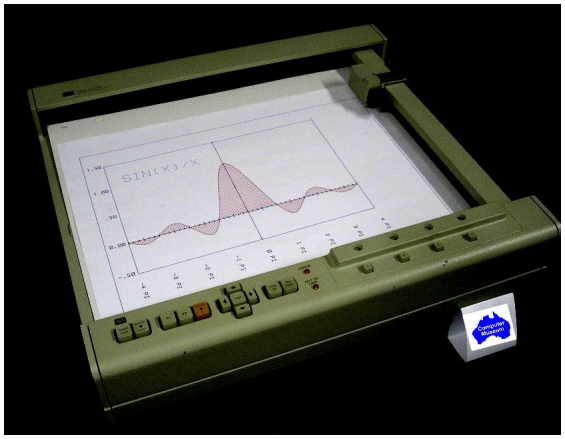


Abb. 3: Flachbett-Plotter HP 9872 (Quelle: www.hpmuseum.net)

Der Befehlsumfang von HP-GL, der ersten „High-Level"-Plotter-Sprache überhaupt, wurde in den folgenden Jahren parallel zur technischen Entwicklung der Hardware erweitert, was allerdings zu Inkompatibilitäten mit älteren Plottern führte. Als die 16-bit-Adressierung der Firmware, die Koordinaten mit einem Maximalwert von 32.767 unterstützte, an ihre Grenzen stieß, implementierte HP 1990 eine neue 32-bit-Firmware und spezifizierte dazu – als Teil von Version 5 der *Printer Command Language* (PCL) zur Ansteuerung des LaserJet III – eine vereinheitlichte Sprachversion: HP-GL/2. Diese Spezifikation setzte sich in den 90er Jahren als Industriestandard durch und wurde zu einem Standard-Austauschformat für Vektorgrafiken.

Struktur

Die Spezifikation von HP-GL/2 umfasst insgesamt 63 Befehle [4]. Sie lassen sich fünf funktionalen Gruppen zuordnen: Konfigurationsbefehle, Vektorgrafik-Kommandos, Vieleck-Befehle, Schriftzeichen-Kommandos und Linien- bzw. Füllattribut-Befehle.

Alle Befehle haben denselben Aufbau: Sie beginnen mit einem aus zwei Großbuchstaben bestehenden Befehls-Kürzel, gefolgt von durch Kommata getrennten Parametern (z. B. den Koordinaten eines Punkts oder dem Radius eines Kreises). Abgeschlossen wird jedes Kommando üblicherweise durch ein Semikolon.

Alle Plotter-Befehle orientieren sich an einem XY-Koordinatensystem, dessen Nullpunkt (0, 0) bei der Initialisierung des Plotters auf die äußerste linke untere Ecke der Zeichenfläche gesetzt wird; dort wird auch der Stift positioniert. Das Koordinatensystem kann durch HP-GL-Kommandos auf der Zeichenfläche verschoben, an den Koordinatenachsen gespiegelt und um den Nullpunkt gedreht werden.

Der Ausgangspunkt für den jeweils nächsten Plot-Befehl wird als *Current Active Position* (CAP) mitgeführt. Der Status des Stifts – „oben" oder „unten" – wird ebenfalls global gehalten (*Pen Status*), von ihm leitet sich ab, ob ein Punkt oder eine Linie geplottet oder lediglich der Stift bewegt wird. Die Schrittweite (*Plotter Unit*) von HP-GL-Plottern liegt bei einem 40stel Millimeter (0,025 mm).

Auf eine DIN A4-Seite passen damit Grafiken mit einer Ausdehnung von etwa 16.000 x 11.000 Punkten. Durch Skalie-

rungskommandos kann die Schrittweite für jede der Achsen in einem festen Verhältnis vergrößert und so die Auflösung der Grafik verändert werden. Solcherart angepasste Schrittweiten werden *User Unit* genannt.

Minimaler Funktionsumfang

Einige HP-GL-Befehle haben bei unserem einfachen Stiftplotter keine Funktion und sind daher verzichtbar, wie beispielsweise *Select Pen* (SP), *Pen Width* (PW) oder *Velocity Select* (VS). Viele weitere Kommandos benötigen wir für das Zeichnen einfacher Vektorgrafiken nicht, wie z. B. die Kommandos für geometrische Formen (Vielecke, Kreise, Kreissegmente etc.), Ausfüllkommandos, die Auswahl von Linientypen oder das Plotten von Schriftzeichen – mehr dazu in der nächsten Folge des Beitrags.

Verzichten wir auch auf die Möglichkeit, eine Grafik zu skalieren oder zu spiegeln, genügen fünf HP-GL-Kommandos, um die meisten (insbesondere aus anderen Dateiformaten in HP-GL konvertierten) Vektorgrafiken zu plotten (Tabelle 1).

Nr.	Kürzel	Funktion	Param.
0	IN	*Initialize*	0
1	PU	*Pen Up*	0
2	PD	*Pen Down*	0
3	PA	*Plot Absolute*	2
4	PR	*Plot Relative*	2

Tab. 1: Die fünf wichtigsten HP-GL-Befehle

Die Auflösung des HP-GL-Plotters mit Encodermotoren ist mit einer *Plotter Unit* von 0,0208 mm etwas höher als die des HP-GL-Standards [1]. Konstruktionsbedingt bleibt beim Plotten auf DIN A5 jedoch ein Rand, sodass die maximale Größe einer Vektorgrafik bei 7.900 x 5.500 Punkten liegt. An diese Größe müssen HP-GL-konvertierte Vektorgrafiken, die meist auf eine DIN A4-Fläche zugeschnitten sind, angepasst werden.

HP-GL-Parser

Unser Hauptprogramm besteht im Wesentlichen aus einem Parser für die fünf HP-GL-Kommandos aus Tabelle 1 und ihre jeweiligen Parameter. Die Realisierung eines solchen Parsers in Robo Pro ist allerdings aus mehreren Gründen nicht ganz einfach:

- Robo Pro kann Daten nur im csv-Format (*Comma-Separated Values*) in ein (einspaltiges) Listenelement einlesen. In HP-GL-Dateien stehen Kommandos und Parameter (Radius, X- oder Y-Koordinate) jedoch meist ohne Trennzeichen nebeneinander. Um in Listenelemente importiert werden zu können, müssen die Werte durch Kommata getrennt und mit Titelzeile zeilenweise untereinander stehen.
- Robo Pro kann in einem Listenelement nur Gleitkomma- oder ganze Zahlen verarbeiten. Die HP-GL-Befehlskürzel (Tabelle 1) können daher nicht eingelesen werden.
- Die Spezifikation von HP-GL lässt Notationsvarianten zu. So können die Befehle durch Semikolon oder Zeilenumbruch voneinander getrennt werden, manchmal wird gar kein Trennzeichen verwendet. Die Parameter können mit oder ohne trennendes Komma angegeben werden; man findet auch Dateien, in denen bei wiederholten PA- oder PR-Kommandos die Koordinatenangaben ohne Befehlskürzel aufeinander folgen.

Daher muss eine HP-GL-Datei zunächst per Hand so „normiert“ und umformatiert werden, dass sie von dem Robo Pro-Parser eingelesen und interpretiert werden kann:

- **Schritt 1**: Einfügen von Leerzeichen (oder eines anderen Trennzeichens)

zwischen Befehlskürzel und Koordinaten, Ergänzung fehlender Befehlskürzel, Löschung nicht unterstützter HP-GL-Kommandos, Zeilenumbruch vor jedem HP-GL-Befehl.

- **Schritt 2**: Einlesen der Datei in Excel, mit Leerzeichen, Komma und Semikolon als zulässigen Trennzeichen (drei Spalten).
- **Schritt 3**: Hinzufügen einer Titelzeile („Befehle“, „X“, „Y“) und Ersetzen der Befehlskürzel durch Ziffern: IN → 0, PU → 1, PD → 2, PA → 3, PD → 4.
- **Schritt 4**: Normierung der Koordinaten, sodass „0“ der kleinste Wert ist und Einfügen des Parameters „0“ in leere Zellen. Falls die größten Werte für X oder Y über die Zeichenfläche hinaus ragen, müssen die Koordinaten skaliert (durch einen geeigneten Faktor dividiert und gerundet) werden.[1]
- **Schritt 5**: Speicherung der Tabelle als csv-Datei ‚HPGL.csv’ mit Kommata als Trennzeichen zwischen den Werten der Spalten (andere Trennzeichen müssen ggf. mit einem Editor ersetzt werden).

Die Normierung und Umformatierung lässt sich auch durch ein Konvertierungsprogramm auf dem PC erledigen, das eine gegebene HP-GL-Datei einliest, die Koordinaten normiert und schließlich Befehle und Parameter im csv-Format abspeichert. Wegen der uneinheitlichen Syntax der HP-GL-Befehle ist das allerdings etwas knifflig – und sei dem geübten Leser als kleine Programmierübung überlassen.

Der HP-GL-Parser, unser Hauptprogramm, ist lediglich eine Schleife, in der das Listenelement „Befehl“ Wert für Wert ausgelesen und das entsprechende HP-GL-Kommando als Unterprogramm, ggf. mit den Listenelementen „X“ und „Y“ als Parameter, aufgerufen wird (Abb. 4).

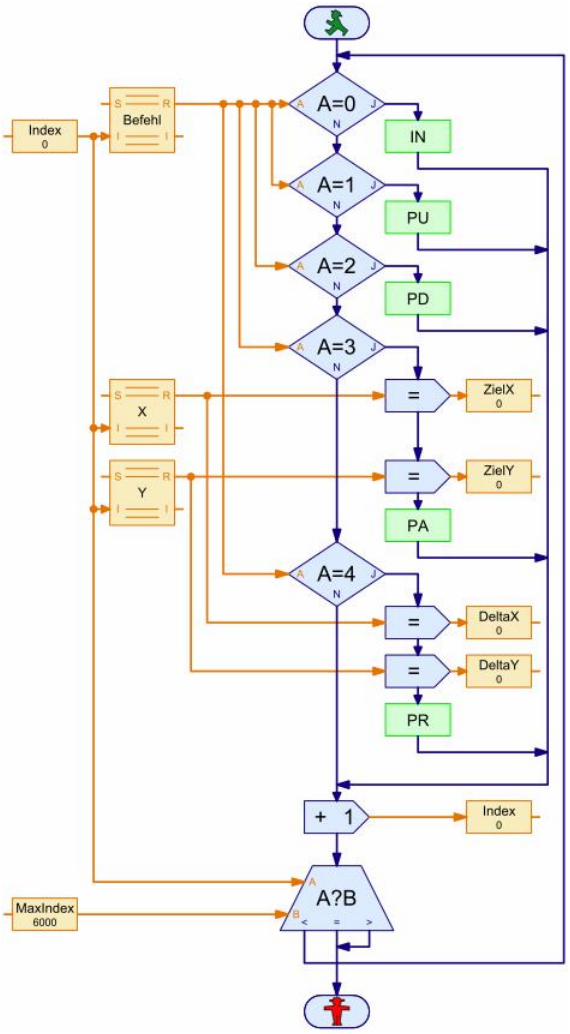


Abb. 4: HP-GL-Parser

In den Eigenschaften der Listenelemente muss dazu das Verzeichnis eingestellt werden, in dem Robo Pro die einzulesende csv-Datei findet (Abb. 5).

Gegebenenfalls muss auch die zulässige Maximalgröße angepasst werden, falls die csv-Datei mehr als die voreingestellten 6.000 Zeilen enthält, ebenso wie der Schleifenzähler *MaxIndex*.

[1] Hier ist Spielraum für Verbesserungen: Die Normierung der Koordinaten lässt sich natürlich auch in Robo Pro implementieren, indem man aus Maximum und Minimum der Koordinaten den Skalierungsfaktor bestimmt und damit die Koordinatenangaben multipliziert.

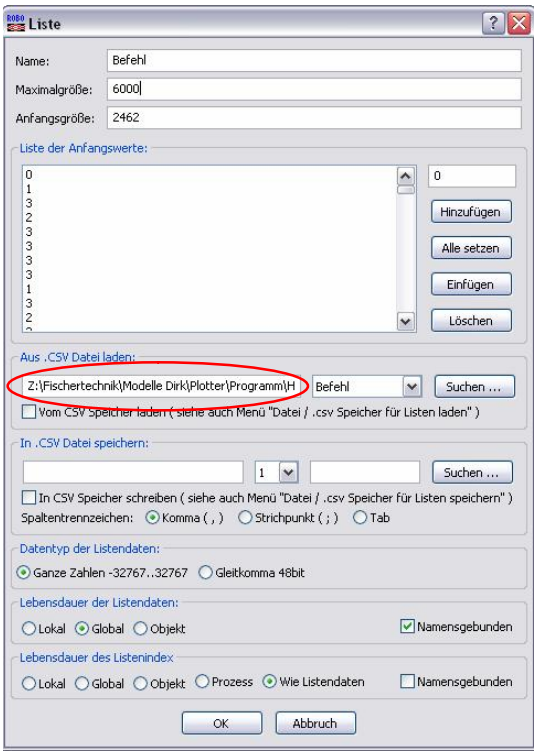


Abb. 5: Eigenschaften der Listenelemente

Variablen und Konventionen

Für die Ansteuerung des Plotters sind zwei Variablen und ein Status-Wert (Flag) von zentraler Bedeutung: die X- und Y-Koordinate (*X*, *Y*) der aktuellen Position (CAP) des Schreibkopfes und der Stiftstatus *PenUp* (1 = richtig, 0 = falsch, also „Stift unten“). Sie werden als globale Variablen mitgeführt und beim Start initialisiert: (*X*, *Y*) = (0, 0) und *PenUp* = 1.

Gemäß dem HP-GL-Standard ist die linke untere Ecke der Zeichenfläche des Plotters der Punkt (0, 0), an den der Zeichenstift bei der Initialisierung bewegt wird. Der maximale *X*- und der maximale *Y*-Wert werden als konstanter Punkt (*XMax, YMax*) im Programm festgelegt. Bei Verwendung der Encodermotoren liegt die rechte obere Ecke der Zeichenfläche des HP-GL-Plotters im Punkt (7.900, 5.500).

Ausgehend von der jeweils aktuellen Position des Schreibkopfes (*X*, *Y*) können beliebige Punkte im Abstand *DeltaX* und *DeltaY* (relative Koordinaten) angesteuert werden. Beide Werte können auch kleiner Null sein. Zur Vereinfachung des Programms arbeiten alle Befehle immer nur mit den Absolutwerten *DeltaXabs* und *DeltaYabs*. In welche Richtung eine Bewegung dann tatsächlich erfolgt, legen die beiden Vorzeichenvariablen *RichtungX* und *RichtungY* fest (1 = positiv, 0 = negativ). Von diesen Variablen hängt ab, in welche Richtung sich die Motoren drehen.

Bei meinen Tests musste ich feststellen, dass die Impulszählung der Encodermotoren im Online-Modus reproduzierbar und abhängig vom verwendeten Motor fehlerhaft ist. Die Zahl der Fehler verringerte sich, wenn ich die Motoren langsamer laufen ließ. Daher führte ich zwei weitere Variablen ein, *TempoX* und *TempoY*, über die die Geschwindigkeit der Encodermotoren gesteuert werden kann.

HP-GL-Kommandos

Version 1 der Plotter-Software[2] beschränkt sich auf fünf Kommandos: IN, PU, PD, PA und PR, als Unterprogramme realisiert, sowie drei Hilfsfunktionen.

Die Implementierung der Befehle IN, PU und PD kann direkt gemäß der HP-GL-Spezifikation erfolgen. Für das Plotten einer Linie (PA, PR) benötigt man wegen der hohen Auflösung des Plotters hingegen einen cleveren Algorithmus, damit der Plotter auch komplexere Grafiken in vertretbarer Zeit erzeugen kann.

IN – Initialize

Der Befehl *Initialize* (IN) bewegt den Schreibkopf mit angehobenem Stift an den Ausgangspunkt (0, 0) und setzt die Variablen *X* und *Y* (CAP) auf den Wert 0. Hier

[2] Die Plotter-Software inklusive zahlreicher Beispiel-HP-GL-Dateien findet sich im Download-Bereich der ft-Community: http://www.ftcommunity.de/data/downloads/robopro/steuerprogramm_hpglplotter_v1.1.zip

kommen die Endlagentaster zum Einsatz: Wenn die Taster gedrückt sind, werden die zugehörigen Encodermotoren gestoppt.

PU, PD – Pen Up, Pen Down

Die Befehle *PenUp* (PU) und *PenDown* (PD) heben bzw. senken den Schreibstift auf das Blatt. Die Stiftposition wird in der Variablen *PenUp* vermerkt.

PA, PR – Plot Absolute, Plot Relative

Die beiden HP-GL-Befehle *Plot Absolute* (PA) und *Plot Relative* (PR) bewegen den Plotterstift mit angehobenem oder gesenktem Stift zum Zielpunkt (*ZielX*, *ZielY*). An das Kommando PA werden die absoluten Koordinaten, an PR der X- und Y-Abstand von der Position des Schreibkopfs (*DeltaY*, *DeltaY*) übergeben. Es gilt also: $ZielX = X + DeltaX$ und $ZielY = Y + DeltaY$.

Das Unterprogramm PA berechnet den X- und Y-Abstand (*DeltaX*, *DeltaY*) des Zielpunkts von der aktuellen Position und ruft mit diesen Parametern PR auf. Die Ansteuerung des Zielpunkts übernimmt also immer das Unterprogramm PR.

Das effiziente Zeichnen von schräg verlaufenden Verbindungslinien ist allerdings keineswegs so einfach, wie man auf den ersten Blick meinen könnte. Denn der intuitive Ansatz, zu jedem X-Wert den zugehörigen Y-Wert über den Steigungsfaktor der Verbindungslinie zu berechnen, erfordert für jeden Punkt eine Fließkomma-Multiplikation. Bei unserem Plotter summiert sich der Aufwand wegen der hohen Auflösung auf fast 500 Multiplikationen pro Zentimeter – ein nicht unerheblicher Rechenaufwand.

Zunächst können die Einzelpunktberechnungen in drei Sonderfällen entfallen:

- Soll der Schreibkopf mit angehobenem Stift ($Pen\ Up = 1$) zum Zielpunkt bewegt werden, kann man beide Encodermotoren gleichzeitig starten. Das übernimmt die Hilfsfunktion *Directly*.
- Ist einer der beiden Abstände *DeltaX* oder *DeltaY* gleich Null, muss nur ein Encodermotor gestartet werden. Das ist Aufgabe der Hilfsfunktionen *Upward* bzw. *Rightward*.
- Ist $DeltaX = DeltaY$, muss der Schreibkopf auf einer diagonalen Linie bewegt werden. Dazu werden die beiden Encodermotoren von der Hilfsfunktion *Diagonal* synchron mit der Impulszahl *DeltaX* gestartet.

Auch in allen anderen Fällen lässt sich die Fließkommamultiplikation mit einem cleveren Algorithmus vermeiden: Bereits 1962 entwickelte der IBM-Programmierer *Jack E. Bresenham* für die Darstellung von Grafiken auf einem Computer-Bildschirm ein heute als *Bresenham-Algorithmus* bekanntes Verfahren [5, 6].

Bresenham-Algorithmus

Die Idee von Bresenham war, die Multiplikation durch schnelle Additionen und Subtraktionen zu ersetzen, indem man den Schreibkopf immer einen Schritt in Richtung der „längeren“ Koordinate bewegt und dabei die Abweichung von der exakten Schräglinie in einem Fehlerwert mitführt. Wird dieser Fehler zu groß, erfolgt ein Schritt in die orthogonale Richtung.

Etwas „algorithmischer“ ausgedrückt: Soll eine Linie zu einem Punkt mit den relativen Koordinaten (*DeltaX*, *DeltaY*) gezeichnet werden, und sei $|DeltaX| > |DeltaY|$, wird zunächst der Fehlerwert $Fehler := |DeltaX|$ gesetzt.

Dann wird der Schreibkopf *DeltaX* Schritte in X-Richtung bewegt, und nach jedem Schritt der Fehler korrigiert: $Fehler := Fehler - |DeltaY|$. Ist $Fehler < 0$, erfolgt ein Schritt in Y-Richtung und zum Fehlerwert wird $|DeltaX|$ hinzu addiert: $Fehler := Fehler + |DeltaX|$.

Für jeden Punkt fallen statt einer Fließkommamultiplikation nur ein bis zwei Ganzzahladditionen bzw. -subtraktionen und Vergleiche an. Hier der Pseudo-Code:

```
Fehler := DeltaX/2;
Zähler := DeltaX;
Solange Zähler > 0
{
  Zähler := Zähler - 1;
  X := X + 1;
  Fehler := Fehler - DeltaY;
  Wenn Fehler < 0
  {
    Y := Y + 1;
    Fehler := Fehler + DeltaX;
  }
}
```

Abbildung 5 zeigt meine Robo Pro-Implementierung des Bresenham-Algorithmus'.

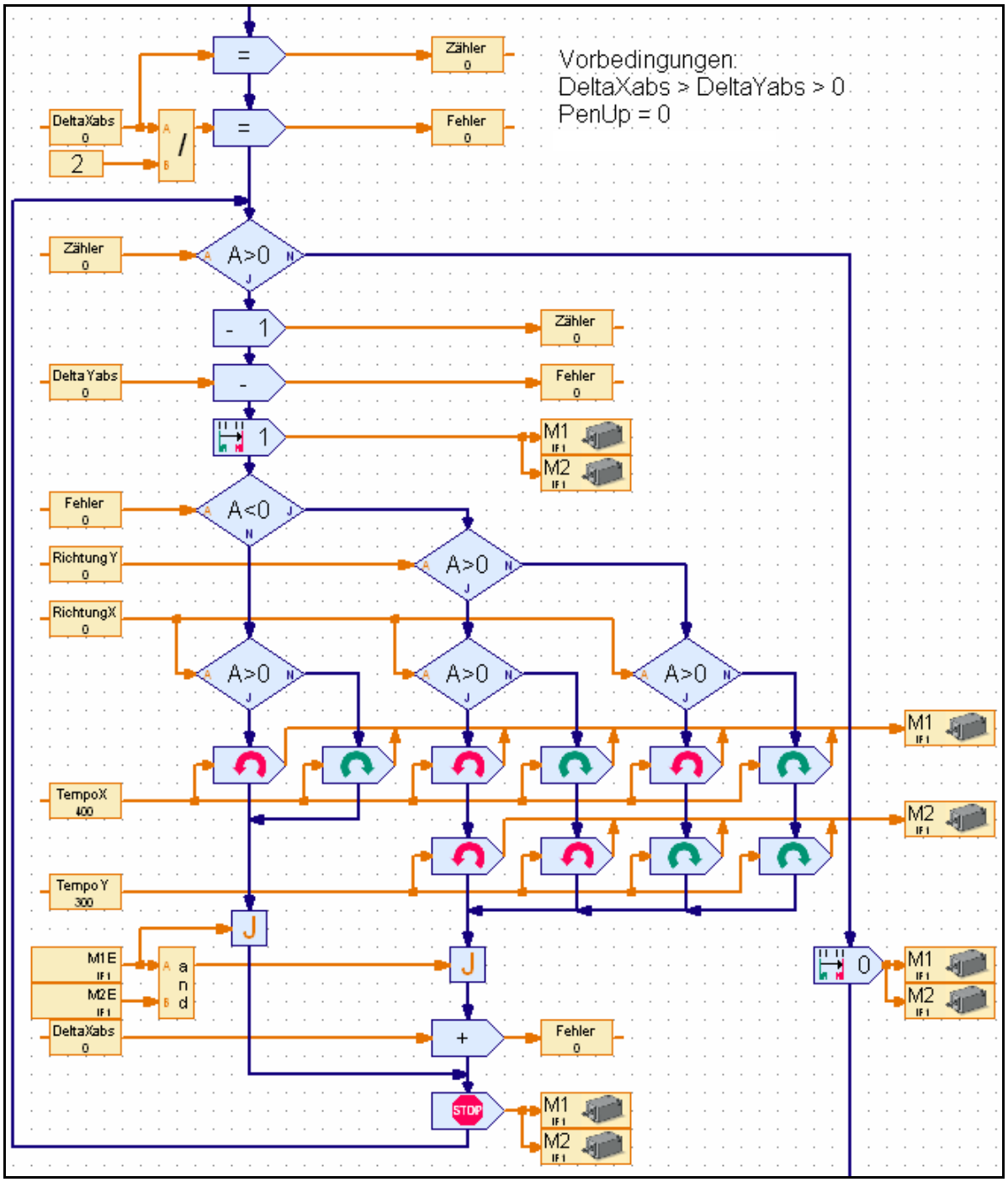


Abb. 5: Bresenham-Algorithmus in Robo Pro mit Encodermotoren

Beispiel-Plots

Schließen möchte ich mit ein paar Kostproben. Die zugehörigen HP-GL- und csv-Dateien finden sich zusammen mit dem Robo Pro-Programm im Download-Bereich der ft-Community.

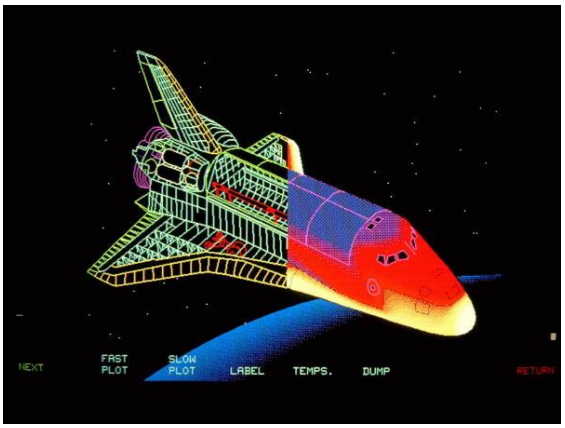


Abb. 6: HP-Demo-Software 9845C

Zunächst ein Klassiker: Die „Columbia“. Diese Grafik war in den 80er Jahren das Standard-Testbild der HP-Plotter und –PCs (Abb. 6). Inzwischen gibt es die Grafik auch als freie HP-GL-Datei:

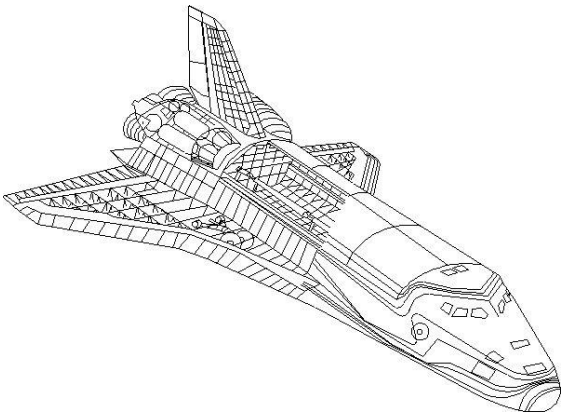


Abb. 7: Columbia in HP-GL

Die Datei umfasst 2.260 Plot-Befehle. Die Encodermotoren meines HP-GL-Plotters mussten insgesamt ca. 145.000 Impulse in X- und 115.000 Impulse in Y-Richtung abarbeiten; der gesamte Plot dauerte ca. 3 Stunden. Bei diesem Plot habe ich eine Ball-Pen-Mine mit einer sehr feinen Spitze verwendet. Leider kleckst sie wegen der niedrigen Zeichengeschwindigkeit etwas.

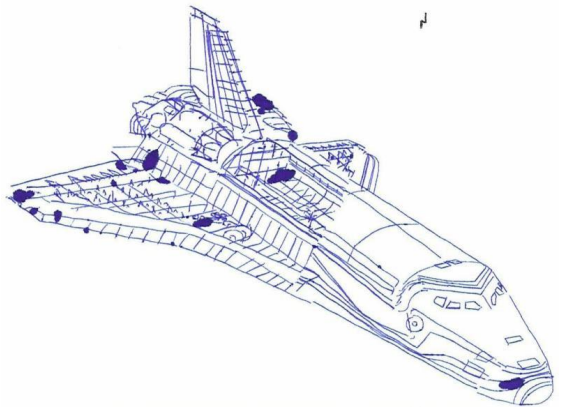


Abb. 8: Plot der Columbia

Der zweite Plot zeigt das Große Landeswappen des Landes Baden-Württemberg. Die Plot-Datei enthält 2.200 Plot-Befehle, die ca. 137.000 Impulse in X- und 116.000 Impulse in Y-Richtung an die Encodermotoren schicken. Als Stift verwendete ich eine (klecksfreie) Kugelschreibermine.

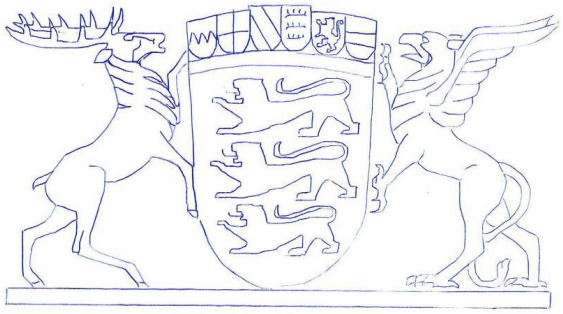


Abb. 9: Plot des Landeswappens

Weitere HP-GL-Dateien finden sich zusammen mit dem Steuerprogramm zum Download in der ft:c oder lassen sich aus frei verfügbaren dxf- und dwg-Dateien erzeugen. Sehr einfach geht das mit dem freien Tool BOcnc von Werner Stratmann, das einen HP-GL-Export (.plt) anbietet.

Abb. 10: Großes Landeswappen im Original

Grenzen von RoboPro

Tatsächlich bringt die Entscheidung für Robo Pro als Programmierumgebung einige Einschränkungen mit sich.

Auf die Beschränkung des csv-Imports auf ganze Zahlen und Gleitkommazahlen (mit Kommata als Trennzeichen) habe ich bereits hingewiesen. Auch die Listengröße ist limitiert: Aufgrund der 16 bit-Arithmetik stehen für den Index in Robo Pro (wie bei den ersten HP-Plottern) nur 15 Bit zur Verfügung, die Liste kann daher maximal 32.768 (2^{15}) Werte enthalten. Damit ist die Komplexität einer Zeichnung auf 32.768 Schreibkopf-Bewegungen begrenzt. Aus demselben Grund können *X* und *Y* auch nie größer als 32.767 werden – die Zeichenfläche darf damit maximal 32.767 x 32.767 Impulse (Motorschritte) groß sein.

Eine weitere Restriktion – wahrscheinlich verursacht durch einen Bug im Compiler – ist eine Fehlermeldung beim Übersetzen des Programms für den Download zum TX-Controller (der Fehler wird derzeit von fischertechnik analysiert). Daher funktioniert das Programm bei mir nur im Onlnie-Modus.

Schließlich „verschluckten“ sich meine Encodermotoren im Online-Modus: Keiner meiner insgesamt fünf Encodermotoren arbeitete 100% zuverlässig; jeder erzeugt bei größeren Plots einen reproduzierbaren Versatz, der weder durch eine fehlerhafte Datenübergabe an den TX noch durch mechanisches Spiel im Plottermodell zu erklären ist. Bei der Zuverlässigkeit der Encodermotoren gibt es also offenbar Mängel, von denen auch im Zusammenhang mit anderen Modellen berichtet wurde.

Es könnte allerdings sein, dass das Problem im Download-Modus nicht auftritt – das konnte ich jedoch aus den genannten Gründen nicht testen. Immerhin konnte ich – wie schon erwähnt – den Umfang der Impulsfehler senken, indem ich die Geschwindigkeit der Motoren verringerte – ein weiteres Indiz, dass die Encodermotoren den Versatz verursachen.

Trotz dieser Einschränkungen ist das Ergebnis dennoch beachtlich: Mit weniger als 170 (unveränderten) fischertechnik-Bausteinen plus einem TX Controller lässt sich ein äußerst präziser HP-GL-Plotter konstruieren und über Robo Pro ansteuern.

Wer den HP-GL-Plotter nachbauen möchte, findet außer den Modellfotos in der ft-Community inzwischen auch eine mit dem fischertechnik Designer erstellte Konstruktionsdatei, unterteilt in sieben Bauphasen, einschließlich einer Bauteilliste.

Aber noch ist nicht Schluss: In Teil 3 des Beitrags wird die HP-GL-Steuersoftware unter anderem um leistungsfähige Vektorgrafik-Befehle erweitert.

Quellen

[1] Fox, Dirk: *HP-GL-Plotter (Teil 1)*, ft:pedia 4/2011, S. 26-34.

[2] Hewlett-Packard: *The HP-GL/2 and HP RTL Reference Guide. A Handbook for Programm Developers*. Hewlett Packard, Second Edition, September 1996.

[3] Hewlett-Packard: *HP-GL Programmer's Reference Manual*, 1984 .

[4] PCL 5 Printer Language Technical Reference Manual, Part II: *An Introduction to HP-GL/2 Vector Graphics*. Hewlett Packard, 1999.

[5] Bresenham, Jack E.: *Algorithm for computer control of a digital plotter*. IBM Systems Journal 4, 1 (1965), S. 25-30.

[6] Wikipedia: *Bresenham-Algorithmus*.

Getriebe

Zahnräder und Übersetzungen (Teil 3)

Thomas Püttmann

In diesem Teil unserer Serie geht es um die genaue Form von Zahnrädern. Die meisten Zahnräder besitzen eine Evolventenverzahnung. Es wird mit Versuchen und Modellen erklärt, was eine Kreisevolvente ist und warum sie zur Verzahnung geeignet ist. Abschließend wird ein kurzes Postscript-Programm vorgestellt, mit dem fischertechnik-kompatible Zahnräder mit beliebigen Zahnzahlen gezeichnet und aus Karton angefertigt werden können.

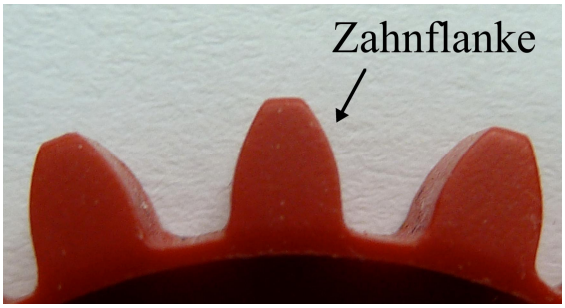


Abb. 1: Nahaufnahme eines Zahns

Hast du dir schon einmal die Zähne der fischertechnik-Zahnräder genauer angesehen? Ihre Flanken sind nicht gerade, aber auch keine Kreisbögen. Warum haben die Zähne genau diese Gestalt? Diese Frage wird dir in diesem Beitrag mit Versuchen und Modellen beantwortet. Außerdem wird dir erklärt, was sich hinter Begriffen wie Modul, Teilung und Eingriffswinkel verbirgt und welche Werte diese Größen bei den fischertechnik-Zahnrädern haben. Das ist besonders nützlich, wenn du das Programm benutzen willst, mit dem du eigene fischertechnik-Zahnräder entwerfen kannst. Die Daten der fischertechnik-Zahnräder genauer kennenzulernen kann dir bisweilen auch bei der Konstruktion von Modellen hilfreich sein.

Antike Zahnräder

Zahnräder gab es schon in der Antike. Hauptsächlich verwendet wurden damals Formen der Triebstockverzahnung, wie man sie von Mühlen her kennt. Es gab aber auch schon richtige Stirnräder, zum Beispiel beim Mechanismus von Antikythera (ca. 100 v. Chr.). Bekannt war natürlich schon, dass die Zähne zweier Stirnrädern gleich groß sein müssen und im gleichen Abstand auf den Rädern folgen müssen, damit die Übertragung funktioniert. Allerdings wusste man wenig darüber, wie man den Zahnflanken eine möglichst gute Form geben konnte. Beim Mechanismus von Antikythera sahen die Zähne so ähnlich aus wie in Abbildung 1. Die Zähne in unserem Modell haben einen sehr großen Abstand und daher gibt es viel Zahnspiel (in der Antike hätte man das sicher besser gemacht), aber wenn man die Räder nur in eine Richtung dreht, spielt das keine Rolle.

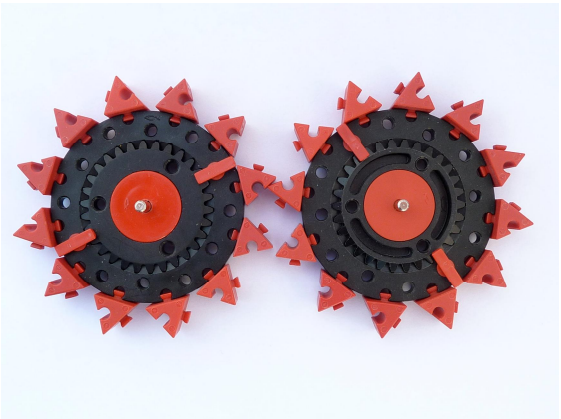


Abb. 2: Modell antiker Zahnräder

Falls du dieses Modell nachbaust, merkst du, dass es sehr gut funktioniert, solange beide Naben lose auf den Achsen sitzen und sich beide Räder ohne Widerstand drehen können. Aber zieh einmal die Nabe des rechten Zahnrads etwas fester auf eine unbewegliche Achse, so dass es sich nur mit etwas Widerstand bewegen lässt. Wenn du jetzt das linke Rad drehst, spürst du deutliche Stöße bei jedem Zahnkontakt. Außerdem knattert und rattert es. Drehst du das linke Rad langsam mit konstanter Geschwindigkeit, so kannst du beobachten, wie das rechte während jedes Zahnkontakts ganz kurz schneller und wieder langsamer wird. Das nennt man nichtkonstante Übersetzung.

Wenn du genau hinschaust, kannst du die Ursache dafür erkennen: Die Spitze des geschobenen Zahns gleitet zuerst an der Flanke des schiebenden Zahns entlang, dann sind beide Zahnflanken für einen kurzen Moment parallel, und schließlich gleitet die Spitze des schiebenden Zahns an der Flanke des geschobenen Zahns entlang. Dadurch entsteht nicht nur nichtkonstante Übersetzung, sondern auch Abrieb an den Zähnen.

Reibräder

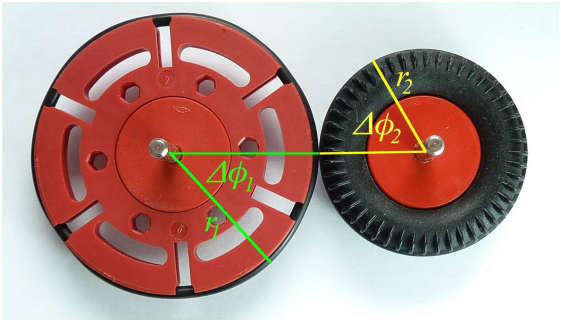


Abb. 3: Reibräder

Um zu verstehen, was konstante Übersetzung genau bedeutet, müssen wir uns kurz von den Zahnrädern entfernen. Schau dir zwei Räder ohne Zähne an, die aneinander abrollen (Abb. 3). Die Stelle, an der sie sich berühren, nennt man den *Wälzpunkt*. Wenn man das Rad mit Radius r_1 um den Winkel $\Delta\varphi_1$ dreht, dreht sich das andere um den Winkel $\Delta\varphi_2 = \Delta\varphi_1 \cdot r_1/r_2$ in die andere Richtung. Das liegt daran, dass zu dem Winkel $\Delta\varphi_1$ ein Kreisbogen mit der Länge $2\pi\, r_1 \cdot (\Delta\varphi_1/360°)$ auf dem Rand des ersten Rads gehört (denn $2\pi\, r_1$ ist der Umfang des gesamten Rades und $\Delta\varphi_1/360°$ gibt den Bruchteil des vollen Kreises an, den der Winkel $\Delta\varphi_1$ ausschneidet). Dieser Kreisbogen bewegt sich bei der Drehung durch den Wälzpunkt hindurch. Dabei wälzt er einen gleich langen Kreisbogen auf dem Rand des zweiten Rads in die entgegengesetzte Richtung. Das zweite Rad dreht sich dabei um den Winkel $\Delta\varphi_2$ weiter. Es gilt

$$2\pi\, r_1 \cdot (\Delta\varphi_1/360°) = 2\pi\, r_2 \cdot (\Delta\varphi_2/360°),$$

weil beide Kreisbögen gleich lang sind. Umstellen dieser Gleichung liefert

$$\Delta\varphi_2 = \Delta\varphi_1 \cdot r_1/r_2.$$

Wenn man von konstanter Übersetzung spricht, meint man, dass genau diese Gleichung $\Delta\varphi_2 = \Delta\varphi_1 \cdot r_1/r_2$ gilt. Ist das erste Rad größer als das zweite, so dreht sich das zweite um einen größeren Winkel weiter als das erste. Wenn sich das erste mit konstanter Winkelgeschwindigkeit (häufig Drehzahl genannt) dreht, dann dreht sich das zweite mit der r_1/r_2-fachen Winkelgeschwindigkeit in die andere Richtung.

Leider gibt es bei Reibrädern einen großen Nachteil, den *Schlupf*: Man muss die Räder ein bisschen aneinander pressen, damit sie aneinander abrollen. Aus diesem Grund bezeichnet man das Abrollen als *kraftschlüssige Übertragung*. Wenn nun das angetriebene Rad nur mit Widerstand zu bewegen ist, kann es sein, dass die Anpresskraft zwischen den Rädern nicht ausreicht und das eine Rad auf dem anderen durchrutscht. Dann sind die beiden oben angesprochenen Kreisbögen nicht genau gleich lang. Bei Zahnrädern ist die Übertragung dagegen *formschlüssig*: Weil Zahn auf Zahn ineinander greifen, kann es keinen Schlupf geben.

Evolventenverzahnung

Die antiken Zahnräder lieferten - wie oben erwähnt - keine konstante Übersetzung. Wenn ein Rad mit konstanter Geschwindigkeit angetrieben wurde, lief das andere mit nicht-konstanter Geschwindigkeit. Erst im 17. Jahrhundert erkannte Ole Rømer, dass man Zähne so formen kann, dass die Übersetzung konstant wird. Er erfand die Zykloidenverzahnung, die auch heute noch - vor allem in der Feinmechanik - eine Rolle spielt. Im 18. Jahrhundert schlug dann der berühmte schweizer Mathematiker Leonard Euler die Evolventenverzahnung vor, heute die bei weitem verbreiteste Verzahnungsform im Maschinenbau. Bei der Evolventenverzahnung ist die Übersetzung ebenfalls konstant.

Auch die Zähne der fischertechnik-Zahnräder haben Evolventenform. Dafür gibt es mehrere Gründe. Zum Beispiel funktioniert dadurch die Übertragung zwischen zwei Zahnrädern auch dann noch gut, wenn du den Achsabstand nicht auf ein Zehntel Millimeter genau einstellst. Außerdem gibt es viele Möglichkeiten, fischertechnik-Zahnräder untereinander zu kombinieren. Das funktioniert mit der Evolventenverzahnung fast ohne Probleme. Bei der Zykloidenverzahnung ist ein Zahnrad auf genau ein Gegenstück ausgelegt.

Kreisevolvente

Die Flanken eines Zahnrads sind Teile von so genannten Kreisevolventen. Im Folgenden wird erklärt, was eine Kreisevolvente ist und warum die Übersetzung bei der Evolventenverzahnung konstant ist. Am besten ist es natürlich, wenn du die folgenden Experimente selbst mit fischertechnik nachvollziehst. Es reicht aber auch aus, wenn du mitdenkst. An den wichtigsten Stellen werden dir einfache Fragen oder Aufgaben gestellt. Versuche sie nach Möglichkeit, selbst zu beantworten bzw. zu lösen. Zur Kontrolle findest du die Lösungen am Ende dieses Beitrags.

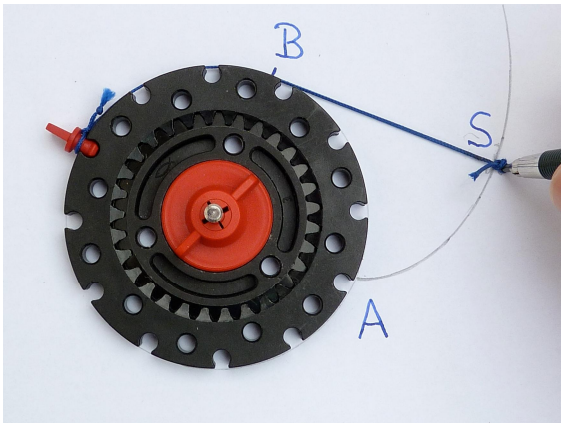


Abb. 4: Kreisevolvente

In Abbildung 4 siehst du, wie du eine Kreisevolvente zeichnen kannst: Du steckst die Achse durch einen Bogen Papier in eine Bauplatte und hältst das Innenzahnrad Z30 fest. Es darf sich nicht drehen. Das blaue Seil ist am roten S-Riegel befestigt und liegt zu Beginn ganz am äußeren Rand des Innenzahnrads Z30 an. Am Ende des Seils ist eine kleine Schlinge, in der ein Bleistift steckt. Der Punkt, an dem der Bleistift am Anfang ist, ist mit *A* bezeichnet. Wenn du nun bei gestrafftem Seil den Bleistift vom Innenzahnrad weg bewegst, zeichnet der Bleistift die Kreisevolvente. Evolvente ist übrigens lateinisch und heißt übersetzt *Abgewickelte* oder *Ausgerollte*. Ein ziemlich anschaulicher Begriff, findest du nicht? Der äußere Rand des Innenzahnrads heißt ab jetzt *Grundkreis*.

Lass' den Stift an einem Punkt *S* der Evolvente stehen. Das straffe Seil mündet an einem Punkt *B* in den Grundkreis. Der Seilabschnitt zwischen *S* und *B* ist Teil einer Geraden, die den Grundkreis in nur einem Punkt - nämlich *B* - berührt. Eine solche Gerade nennt man *Tangente*.

Frage 1: Wie verhalten sich die Länge der Strecke zwischen *S* und *B* und die Länge des Kreisbogens zwischen *A* und *B* zueinander?

Wo wir gerade bei Tangenten sind, schau dir einmal Abbildung 5 an: Die Gerade durch *S* und *B* ist - wie gesagt - eine Tan-

gente an den Grundkreis. Sie verläuft senkrecht zum Radius *MB*. Daran erinnerst du dich vielleicht noch aus der Schule. Auch die Evolvente selbst hat in jedem Punkt eine Tangente. Wie du in Abbildung 5 erkennen kannst, verläuft die Tangente durch *S* senkrecht zur Geraden durch *B* und *S* und damit parallel zum Radius *MB*. Das wird gleich noch wichtig werden.

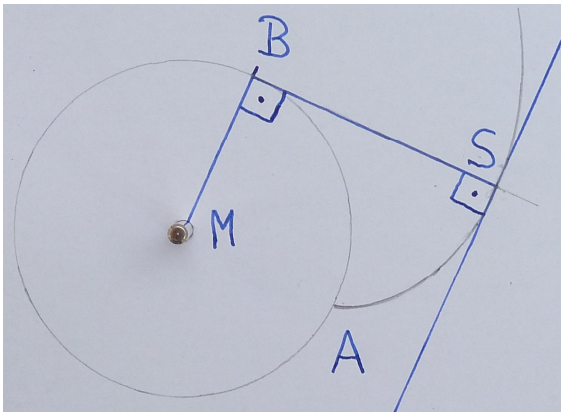


Abb. 5: Tangente an die Evolvente

Zwei übergroße Zähne

Was hat das Ganze mit Zahnrädern zu tun? Nun, stell dir einen übergroßen Zahn auf dem Grundkreis vor, dessen eine Flanke genau durch die Evolvente gegeben ist.

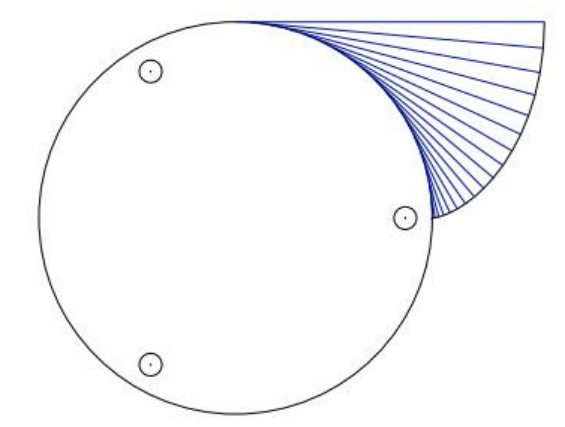


Abb. 6: Vorlage für einen übergroßer Zahn (Abbildung verkleinert)

Du kannst solch einen großen Zahn für das Innenzahnrad Z30 herstellen, indem du ihn - wie in Abbildung 4 gezeigt - zeichnest, die Zeichnung auf Karton klebst und dann den Zahn mit der Schere ausschneidest. Damit die Evolvente aber etwas genauer wird, stelle ich dir hier eine Vorlage mit Bohrungen bereit, die exakt auf das Innenzahnrad passt (Abb. 6). Dieser Zahn schiebt nun einen weiteren übergroßen Zahn auf einem zweiten Grundkreis.

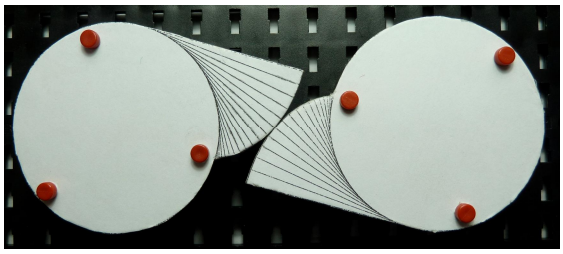


Abb. 7: Zwei übergroße Zähne schieben sich

Schau dir jetzt einmal an, wie sich die Zahnflanken im Punkt *S* berühren. Sie haben in *S* eine gemeinsame Tangente.

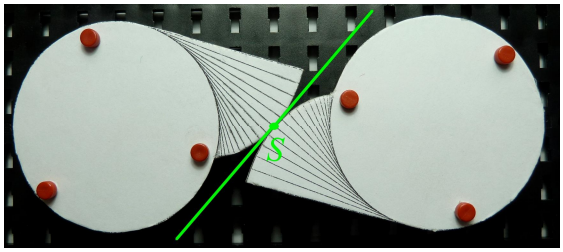


Abb. 8: Tangente (grün) an die Zahnflanken im Berührpunkt

Denk dir kurz die beiden Seilabschnitte von *S* zu den Grundkreisen wieder hinzu.

Frage 2: Warum müssen die beiden Seilabschnitte in *S* ohne Knick ineinander übergehen?

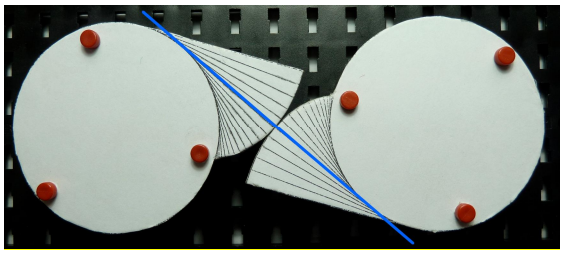


Abb. 9: Eingriffsgerade (blau)

Da die beiden Seilabschnitte ohne Knick ineinander übergehen, liegen sie auf einer gemeinsamen Geraden. Diese Gerade ist eine Tangente an *beide* Grundkreise (Abb. 5). Sie ist sehr wichtig und hat daher auch einen eigenen Namen. Man nennt sie die

Eingriffsgerade (Abb. 6). Der Berührpunkt zweier Zähne verläuft also während einer Drehung entlang dieser Eingriffsgeraden. Das kannst du im Modell wunderbar nachvollziehen.

Jetzt kommen wir zum eigentlichen Punkt, der Evolventen für die Verzahnung so interessant macht.

Konstante Übersetzung

Schau dir noch einmal Abbildung 9 an und stell dir vor, dass du das linke Zahnrad etwas weiter im Uhrzeigersinn drehst, oder probiere es am Modell aus: Der Berührpunkt zwischen den Zahnflanken wandert bei der Drehung entlang der Eingriffsgeraden. Du kannst die Situation zu zwei verschiedenen Zeitpunkten sehr gut mit Fischertechnik zeichnen, siehe Abbildung 10. Die Zahnflanken zum ersten Zeitpunkt sind grün gezeichnet, die Zahnflanken zum zweiten Zeitpunkt rot.

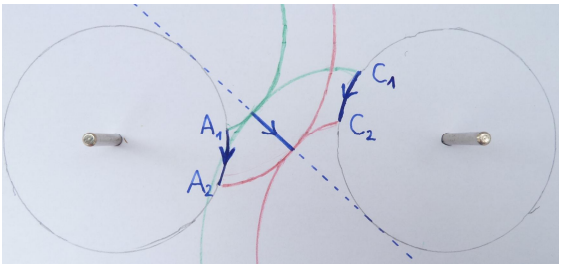


Abb. 10: Konstante Übersetzung

Frage 3: Wie lang ist der Grundkreisbogen zwischen A_1 und A_2 im Vergleich zu dem Grundkreisbogen zwischen C_1 und C_2?

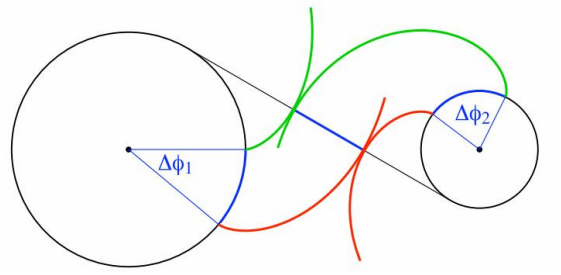


Abb. 11: Konstante Übersetzung bei verschiedene Radien

Diese Übertragung der Kreisbogenlänge von einem Zahnrad zum anderen funktioniert auch, wenn die Grundkreise verschiedene Radien r_1 und r_2 besitzen. Probiere es aus oder schau dir Abbildung 11 an!

Genau wie bei den Wälzrädern am Anfang folgt nun aus der Gleichheit der Kreisbogenlänge, dass die Übersetzung konstant ist.

Die Evolventenform der Zahnflanken bewirkt eine konstante Übersetzung:
$\Delta\varphi_2/\Delta\varphi_1 = r_1/r_2$.

Wälzkreis

Auf dem Grundkreis eines Zahnrads beginnen die Evolventen. Darum ist der Grundkreis für die Berechnung von Zahnrädern sehr wichtig. Am Ende wird aber nur ein sehr kleiner Abschnitt der Evolvente als Zahnflanke genutzt und dieser Abschnitt beginnt meist nicht am Grundkreis. Bei den meisten Zahnrädern kannst du daher den Grundkreis gar nicht direkt sehen!

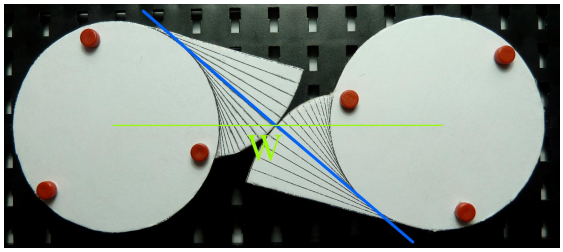


Abb. 12: Wälzpunkt

Ein weiterer, sehr wichtiger Kreis, den du in der Regel auch nicht direkt sehen kannst, ist der *Wälzkreis*. Um zu verstehen, was der Wälzkreis bei Zahnrädern ist, benötigst du zuerst den *Wälzpunkt*. Das ist der Schnittpunkt der Strecke zwischen den Achsmittelpunkten mit der Eingriffsgeraden. Die Wälzkreise der Zahnräder sind jetzt die Kreise um die Achsmittelpunkte, die durch den Wälzpunkte gehen. Zahnräder mit Evolventenverzahnung haben die gleiche konstante Übersetzung wie Reibräder, die so groß sind wie die Wälzkreise.

Die Reibräder berühren sich, anders als die Zahnräder, immer nur im Wälzpunkt und die Reibung ist reine Rollreibung. Bei Zahnrädern mit Evolventenverzahnung ist das leider nicht ganz so gut: Wenn sie sich

gerade genau im Wälzpunkt berühren, gibt es auch nur Rollreibung. Je weiter der Berührpunkt auf der Eingriffsgeraden vom Wälzpunkt entfernt ist, desto größer wird die Reibung, denn es tritt anteilig immer mehr Gleitreibung auf.

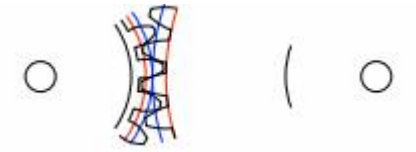


Abbildung 13: Grundkreise (rot) und Wälzkreise (blau) am Z20 und Z40

Den Unterschied zwischen Gleiten und Rollen kannst du in dem Modell sehr gut beobachten: Um die Gleitreibung möglichst klein zu halten, kannst du die Zahnflanken mit einem weichen Bleistift anmalen. Das Graphit wirkt als Schmiermittel.

Eingriffswinkel

Wie hängen nun Grundkreis und Wälzkreis an einem Zahnrad zusammen? Mit anderen Worten: Wie kann man den Grundkreisradius berechnen, wenn man den Wälzkreisradius vorgeben möchte? Um diese Frage zu beantworten, benötigst du eine weitere charakteristische Größe, nämlich den *Eingriffswinkel* (Abb. 14).

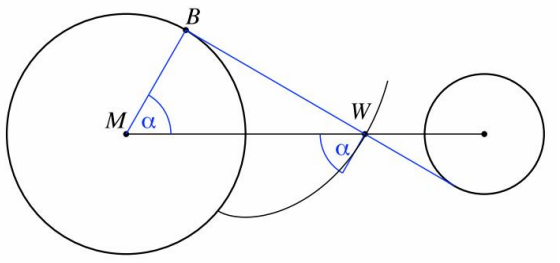


Abb. 14: Eingriffswinkel

Im Wälzpunkt *W* bildet die Tangente an die Zahnflanken einen Winkel α mit der Geraden durch die Mittelpunkte der Zahnräder. Dieser Winkel heißt Eingriffswinkel. Wegen der zwei eingezeichneten rechten Winkel findet man den Eingriffswinkel noch an anderer Stelle wieder, was für die Berechnung wichtig ist. Er ist nämlich genauso groß wie der Winkel zwischen *W* und *B* von *M* aus gesehen. Weil das Dreieck mit den Eckpunkten *M*, *B* und *W* rechtwinklig ist, bedeutet das:

$$\textit{Grundkreisradius} = \textit{Wälzkreisradius} \bullet \cos \alpha.$$

Die gleichmäßige Übertragungseigenschaften der Evolventenverzahnung erhält man nur, wenn man Zahnräder mit gleichem Eingriffswinkel miteinander kombiniert. Denn nur dann erhält man eine knickfreie Eingriffsgerade, längs der sich die Zahnflanken berühren. Alle fischertechnik-Zahnräder haben daher den gleichen Eingriffswinkel. Wie kannst du herausfinden, wie groß dieser ist? An einem Z40 zum Beispiel ist das sehr schwer zu bestimmen. Es geht anders viel besser: Stell dir dazu ft-Zahnrädern mit mehr und mehr Zähnen vor und bedenke, dass immer nur ein etwa gleich langer Abschnitt der Evolvente als Zahnflanke benutzt wird.

Frage 4: Was passiert mit gleich langen Evolventenabschnitten, wenn die Zahnzahlen größer und größer werden?

Ein Kreis mit unendlichem Radius ist eine Gerade. Gibt es ein Zahnrad mit unendlich vielen Zähnen und damit unendlichem Radius im fischertechnik-Programm? Natürlich nicht, aber einen Teil davon! Und dieser Teil hat gerade Zahnflanken.

Aufgabe 5: Finde dieses fischertechnik-Element und bestimme den Eingriffswinkel α mit einem Geometriedreieck.

Jetzt kannst du also aus dem Wälzkreisradius eines ft-Zahnrads mit der obigen Formel dessen Grundkreisradius nach der folgenden Formel berechnen:

$$\textit{Grundkreisradius} = \textit{Wälzkreisradius} \bullet \cos \alpha$$

Aufgabe 6: Ein Z40 hat einen Wälzkreisradius von 29,6 mm. Wie groß ist der Radius des Grundkreises?

Teilung und Modul

Wenn du zwei Zahnräder kombinieren möchtest, müssen nicht nur ihre Eingriffswinkel übereinstimmen, sondern auch ihre Moduln bzw. Teilungen. Modul und Teilung eines Zahnrads sind bis auf den Faktor π gleich. Genau definiert sind sie durch

$$p = \textit{Wälzkreisumfang / Zahnzahl},$$
$$m = \textit{Wälzkreisdurchmesser / Zahnzahl} = p/\pi.$$

Die Teilung gibt also an, in welchem Abstand gleichgerichtete Zahnflanken auf dem Wälzkreis aufeinander folgen.

Für die fischertechnik Zahnräder findet man z. B. in Einzelteillisten den Wert m = 1,5 mm für die Zahnräder mit den großen Zähnen und m = 0,5 mm für die Zahnräder mit den kleinen Zähnen. Stimmen diese Werte überhaupt? Wie kannst du sie überprüfen?

Zunächst einmal spricht alles dafür, dass diese Werte richtig sind. In der DIN 780 (DIN steht für Deutsche Industrienorm) ist eine Vorzugsreihe von Moduln angegeben, in der beide Werte enthalten sind. Der Wert 1,5 mm steht in dieser Reihe zwischen 1,25 mm und 2 mm, der Wert 0,5 mm steht zwischen 0,4 mm und 0,6 mm.

Wenn die ft-Zahnräder der DIN 780 entsprächen, müssten also 1,5 mm und 0,5 mm die korrekten Werte sein. Die DIN 780 gibt es aber andererseits hauptsächlich, damit man metallische Zahnräder mit standardisierten Werkzeugen herstellen kann. Da die fischertechnik-Zahnräder aus Kunststoff sind, ist es daher doch angebracht, den Modul einmal selbst zu überprüfen.

Da man an den Zahnrädern den Wälzkreis nicht direkt sehen kann, ist es schwierig, an ihnen selbst die Teilung und den Modul zu bestimmen. Es gibt jedoch Bauteile, die dafür sehr viel geeigneter sind, und das sind abermals die Zahnstangen!

Schon mit bloßem Auge kannst du erkennen, dass eine Zahnstange mit großen Zähnen länger ist als zwei BS 30 und eine Zahnstange mit kleinen Zähnen ein ganz kleines bisschen kürzer als zwei BS 30. Zum genauen Messen der Längen legst du am besten mehrere Bauteile hintereinander. Du wirst feststellen, dass 20 BS 30 nicht genau 600 mm lang sind, sondern nur ziemlich genau 597 mm. Damit sind zwei also 59,7 mm lang. Auf die gleiche Weise kannst du feststellen, dass die Zahnstange mit den großen Zähnen ziemlich genau 60,3 mm lang ist und die Zahnstange mit den kleinen Zähnen 59,5 mm.

Aufgabe 7: Zähle, wie viele Zähne sich an den Kanten der Zahnstangen befinden und bestimme die Teilung und den Modul.

Bei einem Modul von genau 1,5 mm wäre die Zahnstange mit den großen Zähnen noch etwa 1 mm länger als sie tatsächlich ist und damit 1,6 mm länger als zwei BS 30. Schon bei einer Kombination von drei Zahnstangen hätte das zu einer unschönen Längendiskrepanz von 5 mm geführt.

Mit Hilfe des genaueren Moduls kann man berechnen, dass der Wälzkreisradius eines Z40 ziemlich genau 29,6 mm beträgt. Diese Länge ist nur etwas kleiner als die Länge eines BS 30. Dadurch ist garantiert, dass die Zahnräder bei Lagerung im Raster fast optimal laufen, aber mit Sicherheit nicht klemmen.

Kopfkreis und Fußkreis

Der Modul bestimmt auch die Größe der Zähne, das heißt, welcher Abschnitt der Evolventen tatsächlich als Zahnflanke genutzt wird: In der Regel beträgt die *Kopfhöhe* 1 m, d. h. der volle Radius des Zahnrads – der so genannte *Kopfkreisradius* – ist gleich Wälzkreisradius plus Modul. Die *Fußhöhe* beträgt normalerweise zwischen 1,0 m und 1,3 m. Der *Fußkreisradius* entspricht Wälzkreisradius minus Fußhöhe.

Für die fischertechnik-Zahnräder Z20, Z30, Z40 und den Drehkranz lässt sich die Kopfhöhe von 1 *m* gut bestätigen. Beim Z10 ist die Kopfhöhe kleiner als 1 *m* – dazu gleich mehr. Die Fußhöhe ist etwas schwieriger zu messen, passt aber in den oben angegebenen Bereich.

Im fischertechnik-Heft [1] findet sich eine knappe Erklärung der Maße an Zahnrädern (ohne Erklärung der Flankenform). Dort werden die Werte 1 *m* und 1,2 *m* für die Kopf- und Fußhöhe angegeben.

Kombiniert man zwei Zahnräder so, dass der Abstand zwischen den Mittelpunkten der Achsen genau gleich der Summe der Wälzkreisradien ist, so gibt es etwas Kopfspiel, d. h. eine kleine Lücke zwischen dem Kopfkreis des einen und dem Fußkreis des anderen Zahnrads. Trotzdem sind die Übertragungseigenschaften in diesem Fall optimal.

Unterschnitt

Bei einem Eingriffswinkel von 20° ergibt sich bei Zahnrädern mit 17 oder weniger Zähnen *Unterschnitt*. Was das bedeutet, kannst du verstehen, wenn du ein Z10 mit etwas Druck auf der Kante der Zahnstange wälzt: Die Zahnspitzen der Zahnstange stoßen beim Wälzen merklich an die Zahnflanken des Z10 und versuchen, die Zähne des Z10 zu unterschneiden. Um dies zu vermeiden, verwendet man im Maschinenbau bei fest vorgegebenen Zahnradpaaren oft so genannte Profilverschiebungen, die hier nicht näher erklärt werden.

Bei den Z10 kannst du leicht eine reduzierte Kopfhöhe und eine vergrößerte Fußhöhe nachmessen. Für die Praxis bedeutet das: Wenn du eine Zahnstange oder ein anderes Zahnrad mit einem Z10 bewegen willst, solltest du jedenfalls den Abstand nicht so klein wie möglich machen.

Profilüberdeckung

Damit eine Verzahnung gut funktioniert, sollte am besten nicht nur ein Zahn einen anderen schieben, sondern mehrere gleichzeitig schieben. Die *Profilüberdeckung* ist ein Begriff, der das quantitativ macht. Wie die Profilüberdeckung genau definiert ist, lassen wir hier aus. Du solltest aber einmal beobachten, dass bei der Kombination zweier Z10 zu jedem Zeitpunkt nur eine Zahnflanke schiebt, während bei Kombination zweier Z40 oft zwei gleichzeitig schieben. Je höher die Profilüberdeckung ist, desto ruhiger ist der Lauf der Räder.

Zeichenprogramm

Ich habe ein einfaches Postscript-Programm geschrieben, das Zahnräder mit Evolventenverzahnung zeichnet. Du kannst damit ein bisschen experimentieren, die Profile der fischertechnik-Zahnräder überprüfen, deine Getriebe dokumentieren oder zum Beispiel ein ft-kompatibles Innenzahnrad mit 60 Zähnen entwerfen.

Die Datei `Zahnrad.eps` habe ich auf dem ft-community-Server abgelegt. Sie lässt sich mit jedem Texteditor öffnen. Besonders gut eignen sich Editoren, die gleich eine Syntax-Einfärbung vornehmen. Postscript ist eine ausgereifte, exzellent dokumentierte Grafikprogrammiersprache, die du aber nicht verstehen musst, um das Programm benutzen zu können. Am Ende der Datei findest du die Befehle, die die Zahnräder zeichnen. Aus den Kommentaren kannst du erkennen, wie du sie nach deinen Wünschen ändern kannst.

Was du benötigst, um Postscript-Dateien anzuzeigen, hängt von deinem Betriebssystem ab. Unter Mac OS X und Linux stehen dir mehrere Programme zum Anzeigen zur Verfügung. Unter Windows kann es gut sein, dass du bereits Software zum Anzeigen installiert hast. Mehrere Anzeigeprogramme (IrfanView, GSView) installieren den kostenlosen Postscript-

Interpreter Ghostscript selbstständig. Du kannst ihn natürlich auch selbst installieren. Dann steht dir insbesondere der Befehl `ps2pdf` zur Verfügung, der die Datei in eine pdf-Datei übersetzt. Mit Ghostscript bekommst du auch Fehlermeldungen, falls du beim Ändern Syntaxfehler erzeugst.

Ich habe mit Hilfe des Programms ein Z50 Außenzahnrad aus 2 mm starkem Karton angefertigt (Abb. 15). Mit Schere und Bastelmesser kann man natürlich den Flanken keine exakte Evolventenform geben, aber ein gut funktionierendes Zahnrad kann man trotzdem selbst basteln. Wichtig ist, dass man an den Wälzpunkten den Eingriffswinkel von 20° gut trifft und die Abstände zwischen den Flanken einhält.

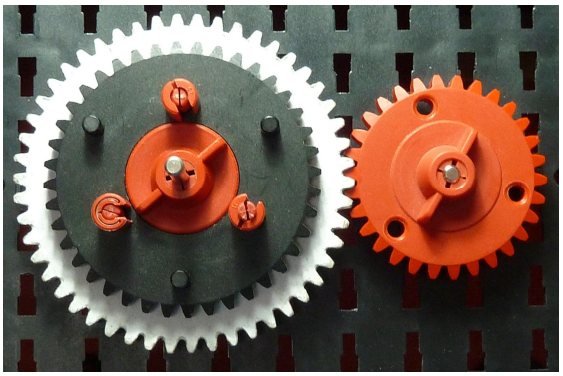


Abb. 15: Z50 aus Karton

Für Zahnräder mit kleineren Zahnzahlen ist das Programm nicht optimal. Wenn der Fußkreis nämlich innerhalb des Grundkreises liegt, wird der dazwischen liegende Teil der Zahnflanke einfach durch eine Strecke modelliert. Für die praktische Anfertigung von Zahnrädern aus Karton spielt das aber keine große Rolle.

Antworten und Lösungen

Antwort 1: Der Seilabschnitt zwischen *S* und *B* lag vorher straff am Innenzahnrad von *B* bis *A*, also ist die Strecke zwischen *S* und *B* genauso lang wie der Kreisbogen zwischen *A* und *B*.

Antwort 2: Hätten die beiden Seilabschnitte von *S* zu den Grundkreisen in *S* einen Knick, so wären nach Abb. 5 die Tangenten an die Evolventen in *S* verschieden und die Evolventen würden sich daher durchdringen und nicht berühren.

Antwort 3: Beide Grundkreisbögen sind gleich lang. Das erkennt man gut an Abbildung 5. Der Seilabschnitt, der zunächst entlang des Kreisbogens zwischen A_1 und A_2 liegt, wird durch Abwicklung auf die gemeinsame Tangente der beiden Grundkreise transportiert. Ein entsprechender Seilabschnitt des zweiten Grundkreises, der dort liegt, wird durch Aufwicklung auf den Kreisbogen zwischen C_1 und C_2 transportiert.

Antwort 4: Mit steigenden Zahnzahlen werden auch die Grundkreise und ihre Evolventen immer größer. Wenn du nun einen immer gleich langen Abschnitt der Evolvente betrachtest, ist das genauso, als würdest du ein Mikroskop auf eine feste Evolvente anwenden. Bei genügend hoher Vergrößerung siehst du die Krümmung kaum noch und der Evolventenabschnitt sieht aus wie eine gerade Linie.

Lösung 5: An der Zahnstange kannst du recht einfach bestimmen, dass der Eingriffswinkel α ungefähr 20° beträgt. Da 20° ein Normwert ist, liegt es nahe, dass auch fischertechnik Eingriffswinkel von 20° verwendet.

Lösung 6: Der Grundkreisradius des Z40 beträgt 29,6 mm • cos 20° ≈ 27,8 mm.

Lösung 7: An der Kante der Zahnstange mit den großen Zähnen befinden sich 13 Zähne. Die Teilung beträgt damit p = 60,3 mm / 13 = 4,64 mm und der Modul $m = p / \pi$ = 1,48 mm. An der Kante der Zahnstange mit den kleinen Zähnen befinden sich 38 Zähne. Die Teilung beträgt damit p = 59,5 mm / 38 = 1,57 mm und der Modul $m = p / \pi$ = 0,50 mm.

Literatur

[1] fischer Elemente der Technik, Heft 2, *Bewegungsübertragung*. Fischer Werke, 1981

Elektromechanik

Vom Zählen und Abzählen (1)

Stefan Falk

In der Motorsteuerungen-Artikelserie der letzten ft:pedia-Ausgaben hatten wir ja versprochen, Maschinen zu besprechen, die sich selbst steuern. Heute machen wir den Anfang und wenden die bisher dargestellten Schaltungen in zählenden Maschinen an.

„Zählen“ findet innerhalb von unzähligen Maschinen statt. Man denke nur an Münzzähler, Codeschlösser, Wiegeeinrichtungen, Stückgutförderung, oder auch das gute alte Wählscheiben-Telefon. Auch die zusammen mit den fischertechnik Computerinterfaces verwendeten Impulszähler dienen eben diesem Zweck.

Es geht letztlich immer um eine von zwei Anforderungen:

- Entweder wollen wir einfach wissen, wie viele zu zählende Ereignisse eintraten,
- oder wir wollen nach Erreichen einer bestimmten Anzahl dieser Ereignisse einen anderen Vorgang auslösen.

Überlegt einmal selbst, wie viele Anwendungsbeispiele euch dazu einfallen – es dürften eine ganze Menge werden. Wir fangen mal ganz einfach an:

Zählen nur mit Mechanik

Unser erstes Modell stellt das Problem „Zählen“ rein mechanisch dar: Wir bauen uns ein „Drehkreuz“, um zu zählen, wie viele Personen durch eine schmale Stelle gingen. Solche Drehkreuze kennt ihr bestimmt von Schwimmbädern oder dem Zoo: Es kann immer nur eine einzige Person zu einem Zeitpunkt hindurch. Ein einfaches Drehkreuz zeigt Abbildung 1:

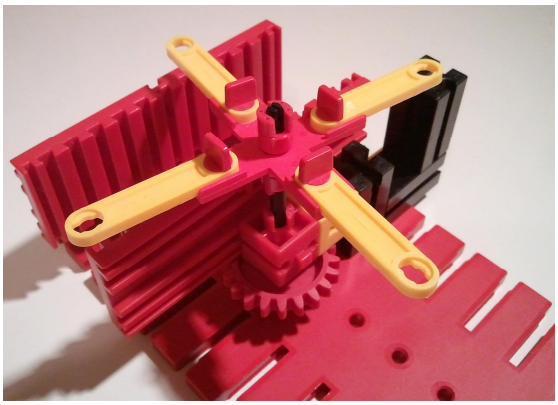


Abb. 1: Einfaches Drehkreuz

Diese Konstruktion bedient sich eines Statikmitnehmers (31712). Wer dieses Bauteil nicht zur Verfügung hat, kann alternativ auch ein Drehkreuz mit der Seiltrommel (31016), der Achsverschraubung (38844) oder einfach einer fischertechnik-Drehscheibe herstellen.

Damit ist zwar noch nichts gezählt, aber immerhin passt nur ein fischertechnik-Männchen zu einem Zeitpunkt hindurch. Damit ist der Anfang gemacht, die Personen zu zählen.

Rastsperre

Der nächste Schritt besteht darin, das Drehkreuz durch eine geeignete Mechanik zu ergänzen, sodass es sich nur in eine Richtung drehen kann. Denn sonst könnten ja Personen in beide Richtungen durchs Kreuz gehen und würden evtl. falsch gezählt.

Das geht ganz einfach mit einer „Rastsperre“. Eine Strebe wird als federndes Element verwendet und so angebracht, dass sie das Zahnrad am unteren Ende der Drehkreuzachse nur noch in eine Richtung drehen lässt, in die andere aber sperrt:

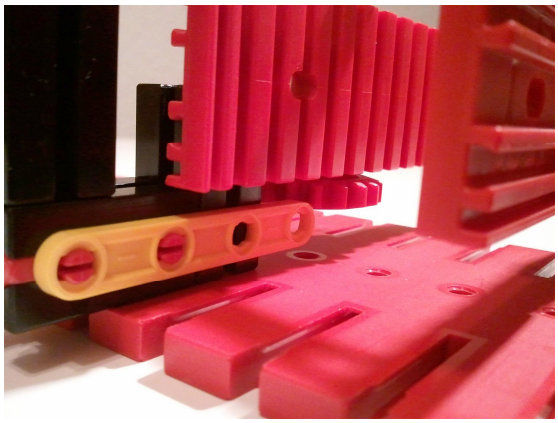


Abb. 2: Drehkreuz mit Rastsperre

Ein analoges Zählwerk

Durch einige wenige Getriebeteile kommen wir schon zu einer hübschen Anzeige der Anzahl hindurchgegangener Personen. Unser Modellvorschlag besteht aus einer Untersetzung, die in einem Zeiger endet, der die gezählten Personen auf einer Skala anzeigt:

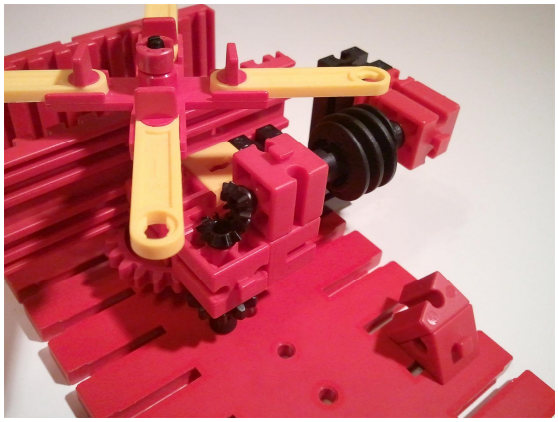


Abb. 3: Getriebe und Lagerung für die Anzeige

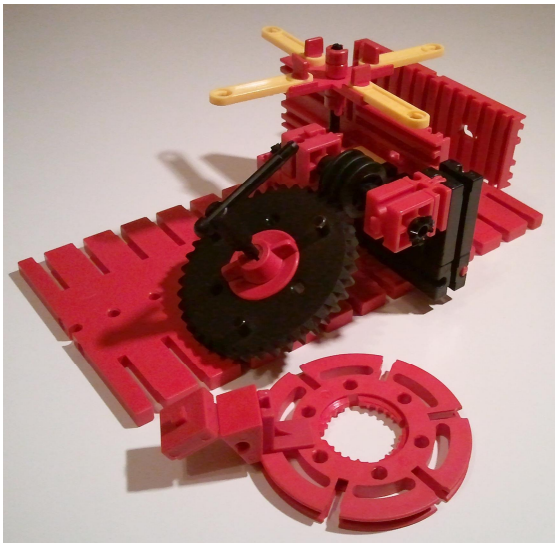


Abb. 4: Z40 eingesetzt

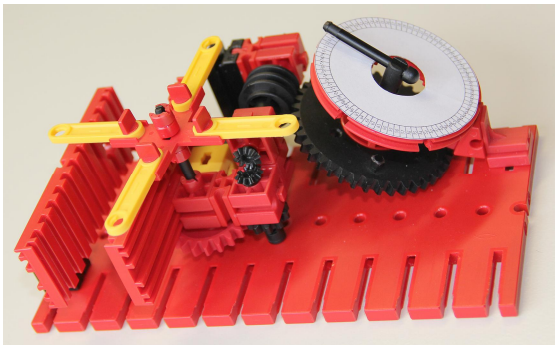


Abb. 5: Drehkreuz mit Anzeige

Rechnen wir einmal nach, wie weit sich der Zeiger bewegt, wenn eine Person durchs Drehkreuz geht: Pro Person dreht sich das Kreuz um eine Vierteldrehung, und mit ihm das Z20 auf seiner Achse. Das davon angetriebene Z10 dreht sich also ¼ · 2 = ½ Umdrehung pro Person. Die Schnecke bewegt das daran anliegende Z40 um einen Zahn pro Umdrehung. Aus unserer ½ Umdrehung pro Person wird also ½ / 40 = 1/80 Umdrehung. Wir können also mit einer Umdrehung des Zeigers von 0 bis 79 zählen – mit so wenigen Bauelementen!

Um uns eine gut ablesbare Skala zu basteln, teilen wir also einen Vollkreis in 80 gleiche Teile auf. Zwischen zwei Teilstrichen liegt dann ein Winkel von 360° / 80 = 4,5°. Wenn ihr Abb. 6 ausdruckt, ausschneidet und aufklebt, könnt ihr die Drehscheibe unterhalb des Zeigers mit einer Skala versehen.

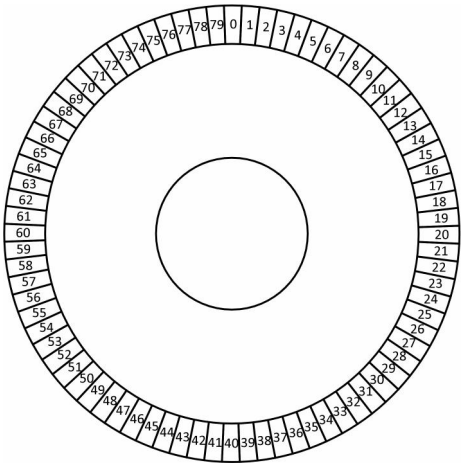


Abb. 6: Zählskala

Zwei Tipps: Das Loch in der Mitte bekommt ihr bequem ausgeschnitten, wenn ihr die Skalenscheibe in der Mitte faltet. Die Befestigung der Skala auf der Drehscheibe gelingt leicht mit zwei kleinen Stücken doppelseitigem Klebeband.

Wegen Überfüllung geschlossen

Eine weitere Funktion unseres Drehkreuzes könnte sein, nur eine bestimmte Anzahl von Personen hinein zu lassen. Zwei Dinge sind dafür erforderlich:

1. Wir müssen einstellen können, wie viele Personen durchgelassen werden sollen, und
2. das Drehkreuz muss nach Erreichen dieser eingestellten Anzahl automatisch sperren (es ist dann in beide Richtungen gesperrt).

Dazu denken wir ein wenig um: Unser Zeiger soll einen leichtgängigen Hebel betätigen, der bei Betätigung einen kräftigeren frei gibt, der schließlich das Drehkreuz sperrt. Abb. 7 und 8 zeigen eine Möglichkeit:

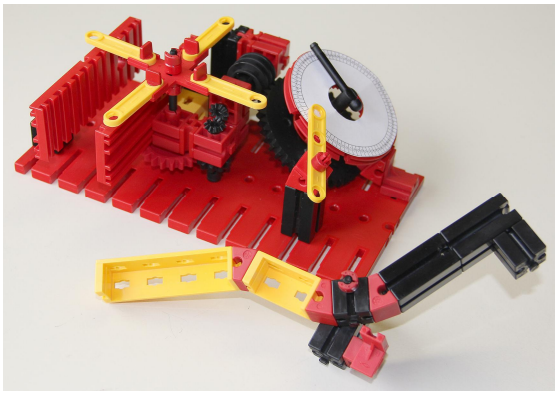


Abb. 7: Material für die Sperre

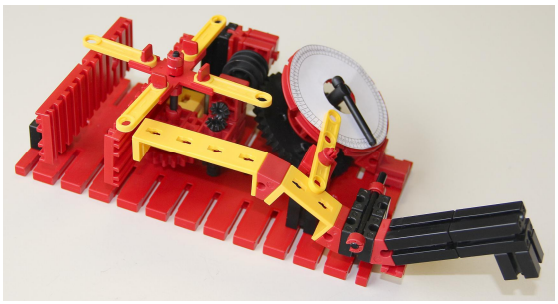


Abb. 8: Zählwerk mit Sperrhebeln

Zunächst stellen wir den Zeiger z. B. senkrecht nach unten. Beim Durchlaufen von Personen bewegt er sich im Uhrzeigersinn weiter, bis er die Statikstrebe so weit verdreht hat, dass der Kipphebel freigegeben wird. Dessen Gegengewicht sorgt dafür, dass er in den Weg des Drehkreuzes gelangt und es damit sperrt:

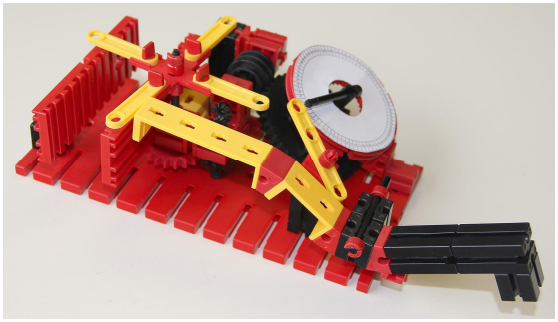


Abb. 9: Die Sperre ist ausgelöst

Für eine neue Personengruppe müssen wir den Zeiger wieder zurückstellen und die Hebel manuell in die Grundposition bringen.

Zur leichteren Einstellung des Zeigers könnt ihr diese andersherum beschriftete Skala ausdrucken und verwenden:

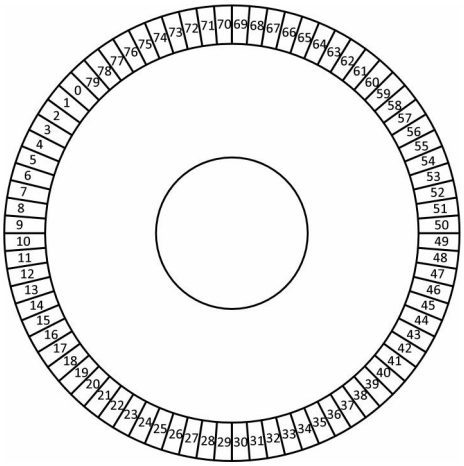


Abb. 10: Skala zum Rückwärts-Zählen

Damit haben wir rein mechanisch gezählt und bei Erreichen einer bestimmten Anzahl einen weiteren Vorgang ausgelöst – hier das Sperren des Drehkreuzes.

Zählwerk mit digitaler Anzeige

Unser erstes Zählwerk arbeitete *analog*. Der Zeiger bewegte sich nur deshalb schrittweise, weil unser Drehkreuz immer um eine Viertelumdrehung weiter gedreht wurde. Wenn ihr einfach am Drehkreuz gleichmäßig dreht, bewegt sich wegen des einfachen Getriebes auch der Zeiger gleichmäßig – er springt nicht von Ziffer zu Ziffer.

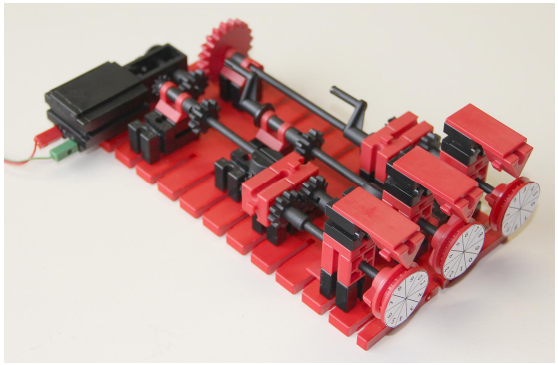
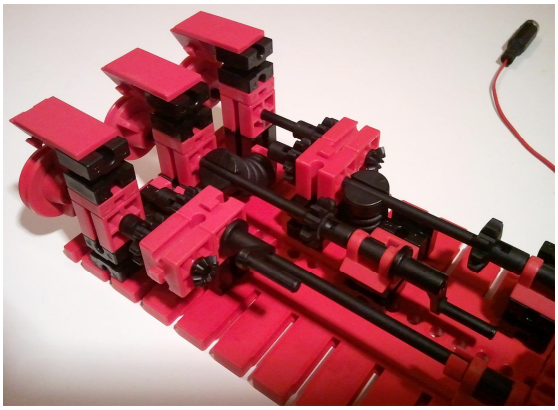


Abb. 11: Zählwerk mit digitaler Anzeige

Unser zweites Zählwerk macht es anders: Nur die vom Motor direkt angetriebene Welle bewegt den rechten Zeiger gleichmäßig. Die beiden anderen Anzeigeräder drehen sich schrittweise. Die rechte Anzeige dreht sich ungefähr ein Mal pro Sekunde und könnte, wenn wir Ziffern 0 bis 9 darauf anbringen, z. B. die Zehntelsekunden einer Stoppuhr anzeigen (Abb. 13 könnt ihr wieder zum Ausschneiden ausdrucken). Das mittlere Rad zeigt dann die Einerstelle der Sekunden an, und das linke Rad deren Zehnerstelle. Unsere Stoppuhr kann also Zeiten von 00,0 s bis 99,9 s anzeigen, wenngleich wir natürlich nicht erwarten dürfen, dass die angezeigte Zeit exakt ist – dazu ist unsere „Zeitbasis“ nicht genau genug.



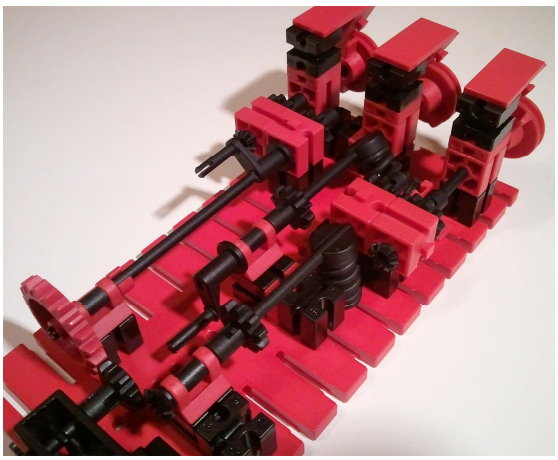


Abb. 12: Detailblicke auf die Übertragsmechanik

Das ziffernweise („digitale“) Zählen geschieht mit einer Übertragsmechanik: Ein Mal pro Umdrehung einer Welle wird das Z10 auf der jeweils nächsten Welle um genau einen Zahn weiter geschaltet. Die dafür notwendigen Abstände passen zwar

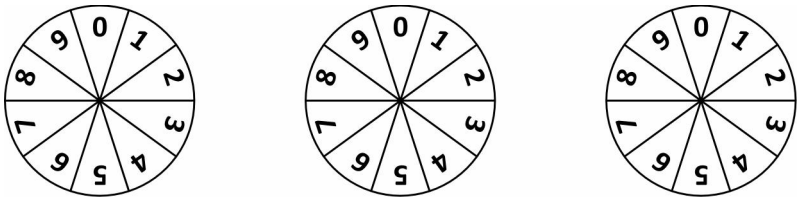


Abb. 13: Zum Aufkleben als Digitalanzeige

nicht genau ins fischertechnik-Raster, aber dafür gibt es ja die Nuten, in denen man Teile so wunderbar justieren kann.

Wichtig sind auch die zwei eingebauten Federn. Die sorgen nämlich dafür, dass die Achsen nicht etwa durch den Schwung beim Umschalten zu weit drehen, sondern etwas gebremst werden.

Beim Zusammenbau empfiehlt sich folgende Vorgehensweise:

1. Lasst den Motor zunächst weg und baut nur die erste Welle mit den Zehntelsekunden auf.
2. Dann baut die Einerstelle komplett dazu und justiert alles so, dass der Eingriff der Kurbel ins Z10 so weit geht, dass das Z10 leichtgängig, aber zuverlässig einen Zahn weiter geschaltet werden kann.
3. Danach baut die Zehnerstelle der Sekunden und justiert sie ebenfalls so, dass alles leichtgängig und zuverlässig „flutscht."
4. Baut erst zum Schluss den Motor ein. Während er läuft, müsst ihr vielleicht nochmal etwas nachjustieren.

Im Endeffekt muss eine perfekt funktionierende Zählmaschine heraus kommen. Da sie (wenigstens in etwa) Sekunden zählt, könnt ihr damit eine Zeitspanne messen und als aus mehreren Ziffern bestehende Zahl – *digital* eben – ablesen.

Für die Justage der Zifferblätter ist es hilfreich, die Flachnaben nur leicht anzuziehen. So könnt ihr sie leicht etwas verdrehen, um die Lage der Ziffern sauber einzustellen.

Je nachdem, wie schnell euer Motor läuft, könnte vielleicht eine etwas andere Untersetzung zwischen Motor und Zehntelsekundenwelle notwendig werden. Auch weniger „glatte" Verhältnisse sind z. B. durch die Kombination Z15/Z40 möglich, um die Zählgeschwindigkeit besser an euren Motor und eure Stromversorgung anzupassen.

Elektromechanisch zählen

Das Zählen und Abzählen elektrischer Impulse wollen wir an einem echten fischertechnik-Klassiker darstellen. Wer die alten Clubhefte aus den 1970er Jahren kennt, schlage mal im Clubheft 1971/1 in der Kategorie „Aktuelles zum Nachbauen" nach: Dort findet sich ein faszinierendes Modell unter der Bezeichnung „Lichtsignale öffnen ein Garagentor" [2].

Wie man in Abb. 14 sieht, sitzt direkt rechts neben dem Tor nämlich ein Fototransistor (oder ein Fotowiderstand) als Lichtaufnehmer. Dessen Signal wird durch das E-Tec-Modul so verstärkt, dass wir damit Motoren betreiben können.

Die Aufgabenstellung für das Tor lautet:

1. Das Auto kommt an und soll *drei Mal* die Scheinwerfer aufleuchten lassen. Daraufhin muss sich das Garagentor öffnen.
2. Wenn das Auto in die Garage gefahren wurde, kann man das Tor durch Drücken eines Tasters im Inneren der Garage schließen.
3. Will man wieder hinausfahren, drückt man denselben Taster erneut – das Tor muss sich wieder öffnen.

4. Nachdem das Auto herausgefahren wurde, lässt man wieder *drei Mal* die Scheinwerfer aufleuchten und bewirkt damit das Schließen des Tores.

Das Garagentor und seinen Antrieb können wir heute noch etwas naturgetreuer nachbauen, und anstelle des Lichtelektronik-Stabes von 1969 setzen wir ein E-Tec-Modul ein, um das Signal der Fotozelle bzw. des Fototransistors zu verstärken. Die elektrische Schaltung aber funktioniert praktisch genauso wie im 1971er Clubheft.

Zum Bau des Modells

Die Mechanik unseres Modellvorschlags ist einem tatsächlich existierenden Garagentoröffner nachempfunden: Eine Umlaufende Kette zieht über einen Mitnehmer das Tor auf oder zu. Die Kette enthält nämlich auch zwei Förderkettenglieder (37192), auf die je ein BS 7,5 aufgesteckt ist. Diese beiden betätigen den Endlagentaster beim Antriebsmotor (Abb. 15).

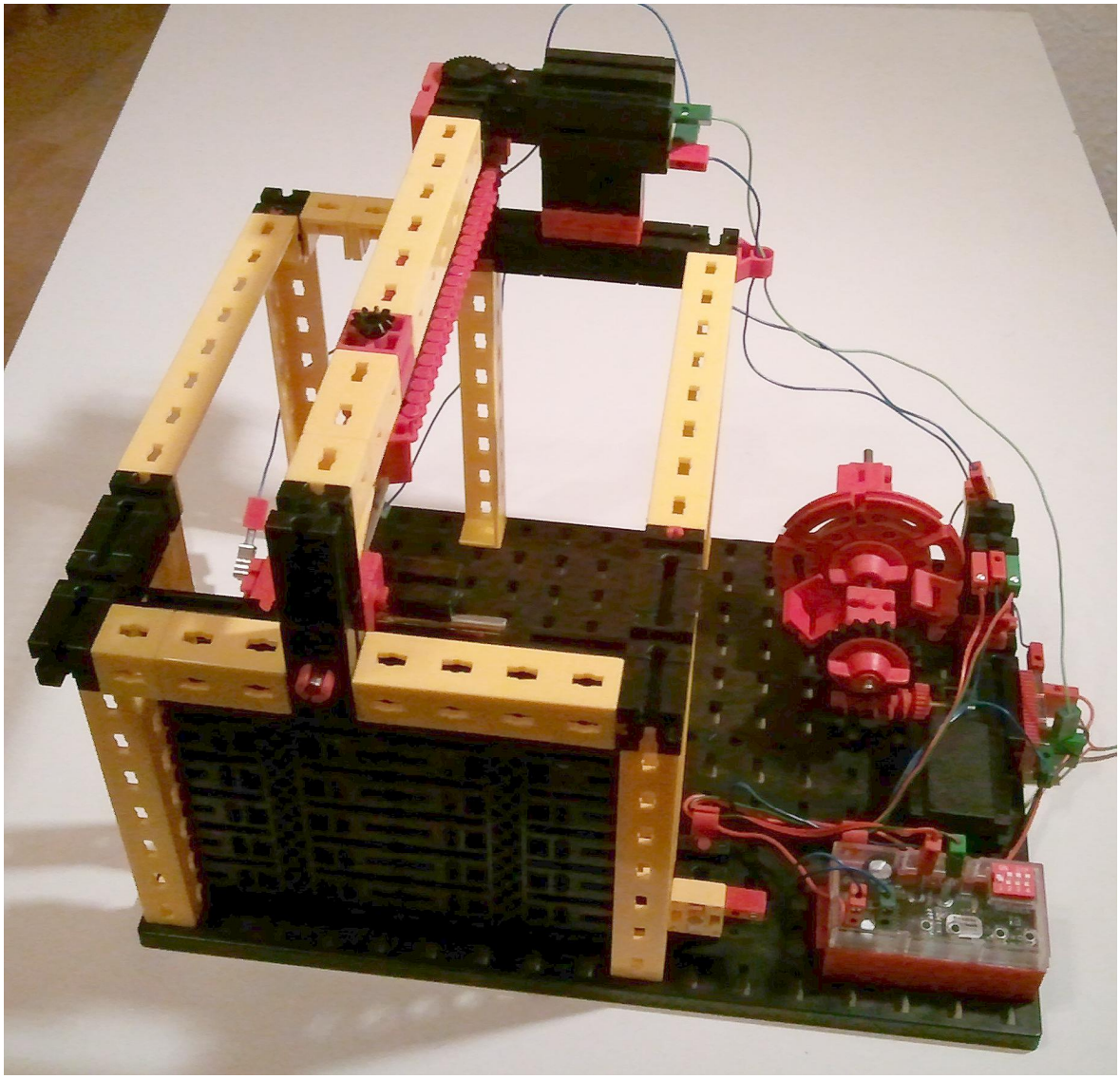


Abb. 14: Durch Lichtsignale gesteuertes Garagentor

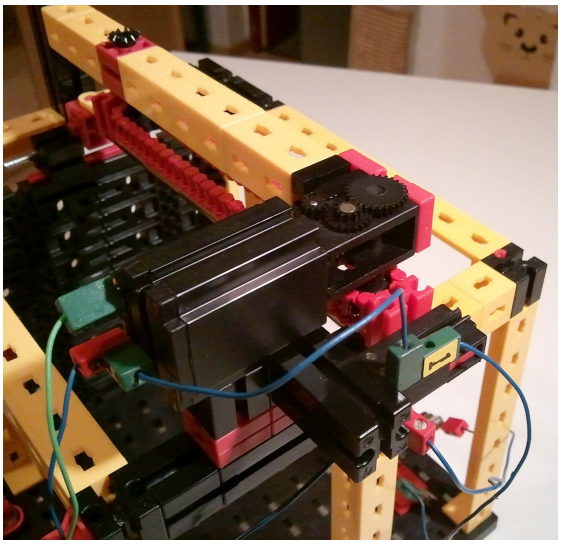


Abb. 15: Detailblick auf den Antrieb

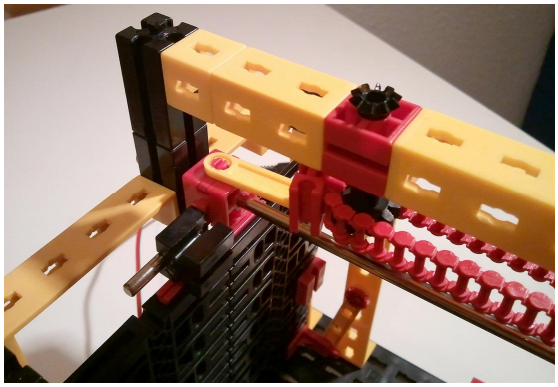


Abb. 16: Detailblick auf die Öffnungsmechanik

Die in Abb. 16 sichtbare Statikstrebe ist mit einer V-Achse 20 (31690) mit dem BS 7,5 verbunden. Die beiden schwarzen Grundplatten 120 x 60 (35129) sind oben durch einen Verbinder 30 und unten einfach durch eine in die unterste Nut eingesteckte Achse verbunden.

Abb. 17 zeigt die Aufhängung des Tores an seiner Unterseite. Die Gelenke werden oben durch einen S-Riegel 8 plus Riegelscheibe und unten durch einen Statikadapter (35975) gebildet. Der BS 5 rechts oben im Bild ist für den einwandfreien Öffnungsvorgang des Tores wichtig.

Abb. 17: Untere Toraufhängung

Die zählende Elektromechanik

Das Hauptproblem ist natürlich: Wie können wir die drei Lichtimpulse abzählen, und wie bewirken wir nach dem dritten das Öffnen bzw. Schließen des Tores? Dazu verwendet unser Modellvorschlag eine *Schaltwalze* – hier eine Drehscheibe 60 mit drei darauf angebrachten Bausteinen, die beim Lauf des Motors nacheinander einen Taster kurz drücken.

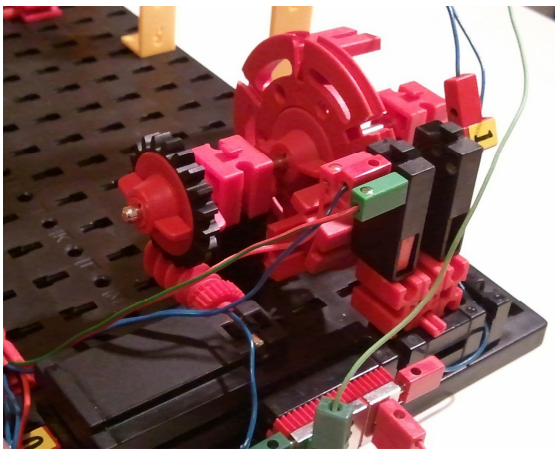


Abb. 18: Die Schaltwalze

Durch die geschickte Schaltung, die wir gleich noch ausführlich besprechen, wird der Steuerungsmotor die Walze so drehen, dass sie bei jedem Lichtimpuls um einen aufgesteckten Baustein weiterdreht, bis sie zum nächsten Baustein kommt. Die drei Bausteine sind auf der Drehscheibe direkt nebeneinander angebracht.

Auf der anderen Seite der Drehscheibe steckt genau gegenüberliegend ein einzelner Baustein. Nachdem der dritte Baustein passiert wurde – also der dritte Lichtimpuls ankam – dreht sich die Schaltwalze so lange, bis der erste der drei Bausteine wieder den in Abb. 18 linken Taster betätigt. Und während dieser Zeit kommt der gegenüber liegende einzelne Baustein am zweiten Taster vorbei – und löst damit das Öffnen bzw. Schließen des Tores aus.

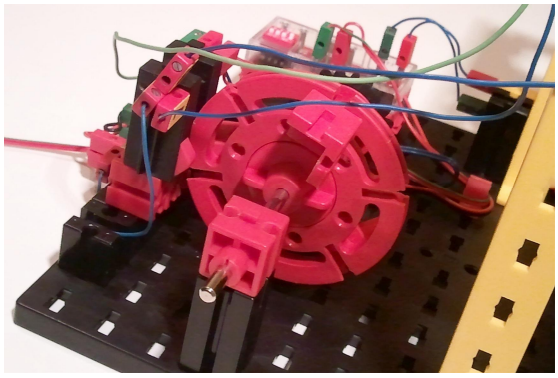


Abb. 19: Der einzelne Baustein zum Auslösen des Torantriebs

Außer der elektrischen Schaltung fehlt damit nur noch der im Garageninneren anzubringende Taster fürs Öffnen oder Schließen vom Innenraum aus. Abb. 20 zeigt einen Bauvorschlag:

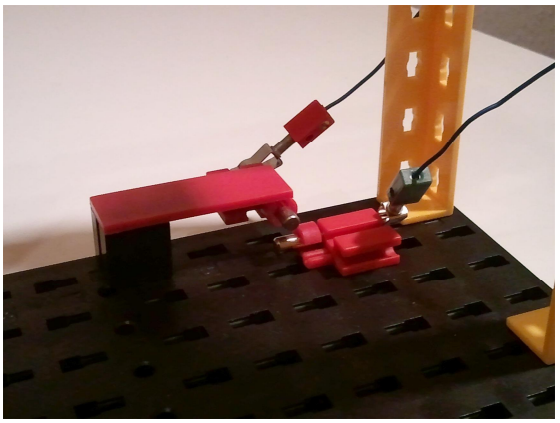


Abb. 20: Selbstbautaster im Garageninneren

Die Schaltung

Wie funktioniert unser zählendes Garagentor nun? Schauen wir uns dazu das Schaltbild an (Abb. 21):

Es gibt zwei Motoren: Einer treibt das Garagentor selbst an, der andere dreht unsere Steuerscheibe mit den Nocken, die die beiden Steuertaster T1 und T2 betätigen. Gehen wir zunächst von folgender Grundstellung unseres Modells aus:

- Das Tor ist geschlossen und der von der umlaufenden Kette betätigte Endlagentaster T3 ist gedrückt.
- Das Steuerrad steht so, dass T1 gerade vom ersten seiner drei Steuernocken gedrückt wird, während T2 unbetätigt ist.
- Die Lampe, die unseren Fototransistor beleuchtet, sei ausgeschaltet (das Auto ist noch gar nicht da).
- Der im Garageninneren angebrachte Selbstbautaster T3 ist auch gerade nicht gedrückt.

Im Schaltbild sind wie üblich die Ruhezustände der Taster eingezeichnet, also ihre unbetätigte Stellung. Da T3 aber tatsächlich gedrückt ist, bekommt der Antriebsmotor keinen Strom – das Tor bleibt geschlossen.

Die Elektronik kommt ins Spiel

Das E-Tec-Modul ist durch die Stellung seiner DIP-Schalter auf ein „Oder“-Logikglied eingestellt. Das ist eines der Zusatzprogramme, auf die man es einstellen kann [1] und bewirkt, dass die Ausgänge Q1 und Q2 des E-Tec-Moduls umschalten, sobald mindestens einer der Eingänge I1, I2 oder I3 mit „+“ verbunden wird. Wir verwenden es in dieser Betriebsart einfach als Verstärker für den Fototransistor – es ist für unsere Schaltung unerheblich, an welchen der drei Eingänge ihr den Fototransistor anschließt. Nur müsst ihr auf die richtige Polung achten: Verbindet den mit „+“ gekennzeichneten Pol des Eingangs mit dem rot gekennzeichneten Anschluss des Fototransistors. Falls ihr einen der älteren Foto*widerstände* verwendet (man erkennt sie an der geriffelten Lichtaufnahmefläche), ist die Polung egal.

Was passiert da nun genau? Solange der Eingang nicht angesteuert wird (der Fototransistor also hinreichend im Dunkeln liegt und deshalb den Strom kaum durchlässt), gibt das E-Tec-Modul am Ausgang Q1 „-“ aus. Der Ausgang Q2 liefert immer genau das gegenteilige Signal von Q1 (man sagt, Q2 ist gegenüber Q1 *invertiert*). Im unbeleuchteten Zustand des Fototransistors ist das also „+“.

Wenn der Taster T1 also wie vorausgesetzt vom ersten Schaltnocken betätigt ist und Q1 auf „-“ steht, wird durch unseren Steuerungsmotor kein Strom fließen, denn dessen zweiter Anschluss ist ebenfalls mit „-“ verbunden.

Sowohl der Torantrieb als auch der Steuerungsmotor bleiben also still – das Tor rührt sich nicht.

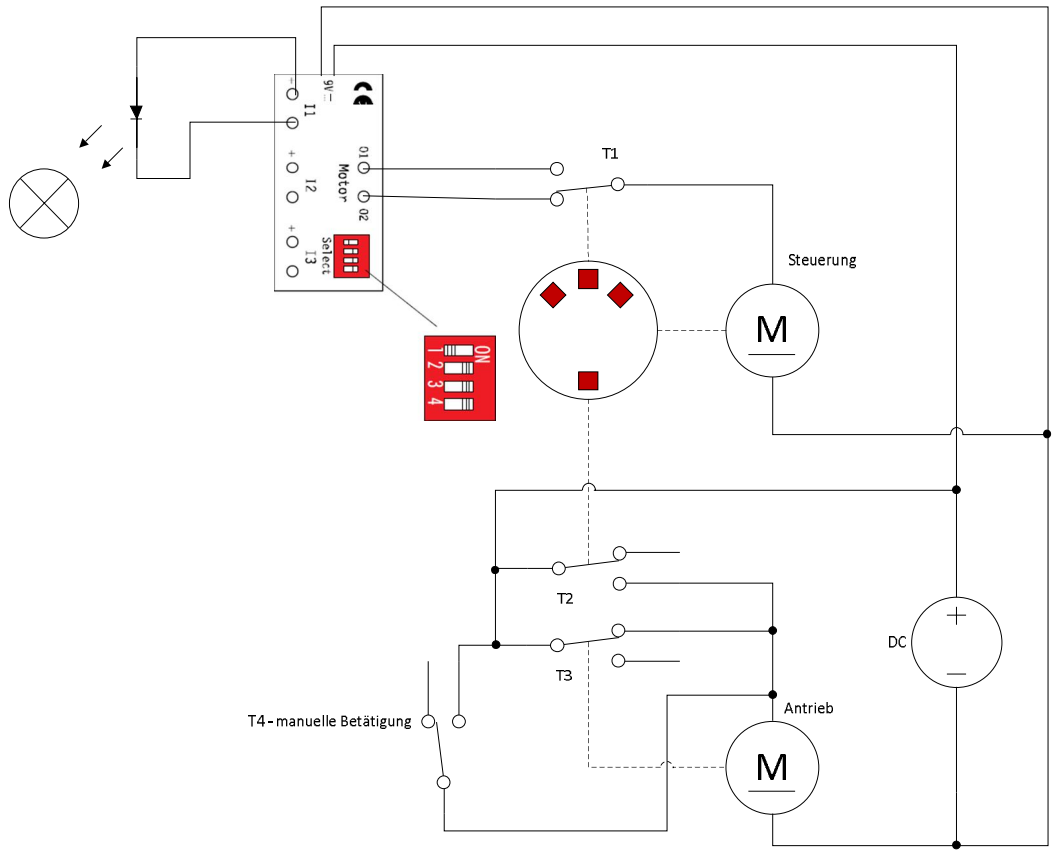


Abb. 21: Schaltbild des Garagentors

Das Auto blinkt ein Mal

Nun kommt das Auto (welches ihr nach Belieben dazu bauen könnt) an und sein Licht wird das erste Mal eingeschaltet. Was passiert?

Sofern das Licht stark genug auf den Fototransistor scheint, wird das E-Tec-Modul umschalten. Q1 liefert nun also „+“ und Q2 „-“. Nun kann von Q1 durch den betätigten Taster T1 hindurch Strom durch den Motor bis zurück zum Minuspol der Stromversorgung fließen. Der Steuerungsmotor dreht sich also.

Aber nur ein kleines Stück! Im Nu ist das Steuerrad nämlich so weit gedreht, dass der erste Steuernocken an T1 vorbei gedreht wurde und T1 deshalb losgelassen wird. Damit liegen beide Anschlüsse des Motors wieder auf „-“ (denn Q2 liefert ja bei beleuchtetem Fototransistor „-“). Der Motor bleibt also wieder stehen.

Wer immer im Auto sitzt, macht das Licht dann wieder aus (es war also nur ein kurzes Aufleuchten nötig). Und schon kehren sich die Verhältnisse wieder um: Q2 liegt jetzt an „+“, und wegen des immer noch unbetätigten Tasters T1 fließt wiederum Strom durch den Steuerungsmotor. Der wird sich also wieder drehen – bis Taster T1 von der (zweiten!) Steuernocke wieder gedrückt wird. Dann haben wir praktisch wieder den Grundzustand: Das Licht ist aus, T1 unbetätigt – alles bleibt still.

Erinnert euch das nicht an etwas? Das ist von der Wirkungsweise her genau die Wechselschaltung, die wir bereits in ft:pedia 4/2011 (Seite 6) kennengelernt haben, nur dass hier statt zwei Tastern nun je ein Taster und ein E-Tec-Modul zum Einsatz kommen.

Etwas hat sich aber doch gegenüber dem Urzustand geändert: Das Schaltrad ist durch das Ein- und wieder Ausschalten des Lichts am Auto um genau einen Steuernocken weiter gedreht worden. Wir haben „Eins“ gezählt! Durch die geschickte Wechselschaltung ist es auch völlig gleichgültig, wie lange das Licht genau eingeschaltet war. Unsere Zähleinrichtung wartet geduldig aufs Ausschalten, bevor sie wieder auf Licht reagiert.

Drei Mal geblinkt

Wenn das Auto ein zweites Mal das Licht ein und wieder aus schaltet, wird unsere Steuerscheibe also wiederum um genau einen Schaltnocken weiter gedreht. Schließlich passiert dasselbe noch ein drittes Mal.

Und plötzlich tut sich etwas: Nachdem der Taster vom dritten Schaltnocken wieder losgelassen wird (das ist bei eingeschaltetem Licht) und die Lampe wieder ausgeschaltet wird, dreht sich das Steuerrad ja wie beschrieben wiederum so lange, bis T1 wieder von einer Schaltnocke betätigt wird. Dieser Weg ist aber durch die Anordnung der Schaltnocken auf der Drehscheibe lang, und *während* dieses Laufs wird Taster T2 kurz gedrückt.

T2 überbrückt nun den die ganze Zeit über gedrückten Endtaster T3. Dadurch läuft der Torantrieb los. Der Endlagentaster T3 wird dann losgelassen (schneller als T2) und versorgt den Antriebsmotor nun alleine mit Strom – auch wenn T2 wieder betätigt wird.

Der Torantrieb läuft nun also, bis T3 das nächste Mal betätigt wird und die Stromzufuhr abschaltet. Durch die *zwei* BS 7,5 auf der umlaufenden Kette passiert das immer dann, wenn das Tor ganz geöffnet oder ganz geschlossen ist.

Voilà! Immer wenn man drei Mal das Licht am Auto hat aufleuchten lassen, wird das Tor sich also entweder öffnen (falls es geschlossen war) oder schließen (falls es geöffnet war). Damit können wir das Tor vom Auto aus bequem öffnen und schließen!

Nun bleibt uns nur noch der manuell zu betätigende Taster T4. Dessen Wirkungsweise ist nun einfach: Er überbrückt (wie T2) beim Drücken den Endlagentaster T3 des Torantriebs und startet somit ebenfalls einen Öffnungs- bzw. Schließvorgang. Vom Inneren der Garage aus kann man also ebenfalls das Tor öffnen und schließen.

Insgesamt ist der Ablauf damit wie folgt:

1. Das Tor ist geschlossen und wird von innen durch Druck auf T4 geöffnet.
2. Das Auto fährt heraus, blinkt drei Mal und bewirkt damit, dass sich das Tor schließt.
3. Wenn das Auto zurückkehrt, muss man wieder drei Mal blinken, um das Tor zu öffnen. Nun kann das Fahrzeug wieder in die Garage gefahren werden.
4. Vom Garageninneren aus wird das Tor durch Druck auf T4 wieder geschlossen.

Damit haben wir unseren „Klassiker“ (vgl. Abb. 22) mit aktueller Elektromechanik und Elektronik wieder aufleben lassen. Wie ihr seht, kann man mit ein wenig Schaltungskenntnis auch recht komplexe Steuerungsaufgaben ohne Computer meistern – aber das haben wir während der bisherigen ft:pedia-Ausgaben ja schon öfters erlebt.

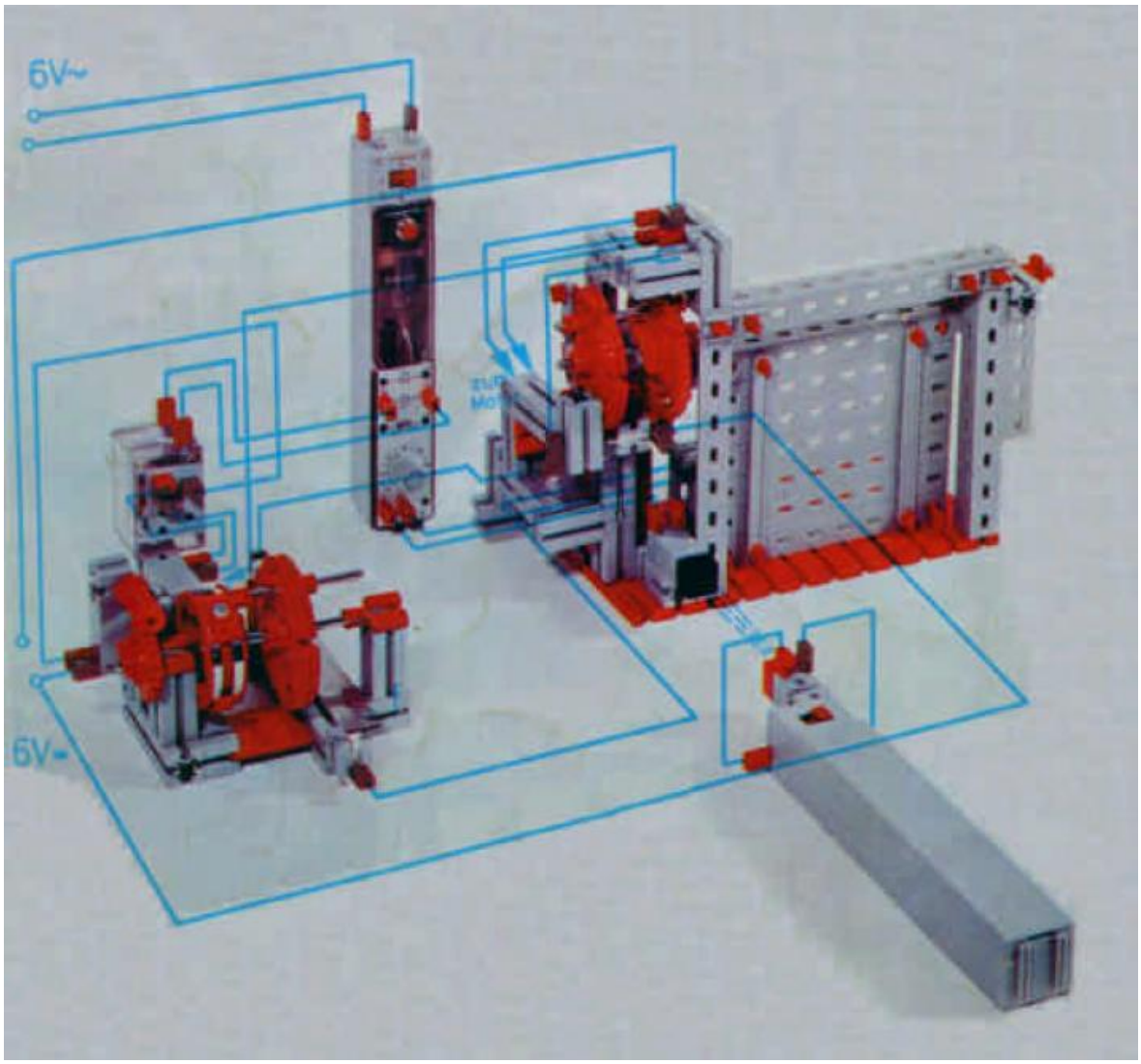


Abb. 22: Das Ur-Modell aus dem fischertechnik-Clubheft 1971-1 [2]

Wenn etwas nicht funktioniert

Wenn euer Garagentor nicht wie vorgesehen arbeiten will, kontrolliert bitte:

- Sind die Stromversorgungspole Plus und Minus genau so angeschlossen wie in unserem Schaltbild? „Plus“ ist üblicherweise immer ein rot gekennzeichneter Anschluss.
- Sind die Taster genau so angeschlossen? Im Schaltbild sind die Ruhestellungen (unbetätigt) der Taster dargestellt.
- Falls der Steuerungsmotor falsch herum drehen sollte, vertauscht seine Anschlüsse.
- Falls die Steuerungstaster zu fest sitzen oder nicht zuverlässig oder lange genug gedrückt werden, wenn sich die Steuerscheibe dreht, justiert sie ein wenig.
- Falls das E-Tec-Modul nicht schalten will, auch wenn ihr den Fototransistor beleuchtet, genügt vielleicht das Licht nicht. Geht dann mit der Lampe näher an den Fototransistor oder verwendet mehrere Lampen.

Jetzt aber viel Spaß beim Aufbau sowie beim ausführlichen Testen und Verstehen der Wirkungsweise eures „intelligenten Garagentors“!

Was kommt als nächstes?

In der nächsten Ausgabe der ft:pedia werden wir ein etwas komplexeres Modell ausführlich vorstellen und erklären, in dem fast alle bisher kennengelernten Schaltungstechniken zum Einsatz kommen: Wechselschaltungen, Relais mit Selbsthaltung und Selbstsperrung, Dioden, eine Lichtschranke und eine Zähleinrichtung.

Falls ihr vorsorgen mögt: Zum Aufbau werden wir vier Taster (mindestens einer davon kann aber ein Selbstbautaster sein), ein Relais (bzw. unser ft:pedia-Relais aus Ausgabe 4/2011), eine Lampe, einen Fototransistor, ein E-Tec-Modul, zwei Dioden und zwei Motoren benötigen. Auch mechanisch ziehen wir einige Register in diesem Modell: Exzenter- und Linearantriebe, Zahnstangen und eine Rutschkupplung werden zum Einsatz kommen.

Dennoch wird das Modell auf eine fischertechnik-Bauplatte 500 passen und mit relativ wenigen Bauteilen auskommen. Wenn ihr mit den Modellen dieser ft:pedia-Ausgabe „durch“ seid, könnt ihr ja schon mal raten, was das Modell der nächsten Ausgabe wohl leisten könnte. Bleibt uns also treu!

Quellennachweis

[1] fischertechnik: *Spezialprogramme für Digitaltechnik (Zusatzanleitung zum E-Tec-Modul),* ladbar von http://www.fischertechnik.de/home/downloads/profi-e-tech.aspx.

[2] fischertechnik: Clubheft 1971-1, http://www.ft-datenbank.de/search.php?keyword=Club+Nachrichten

Traktor mit Baumstammgreifer
(wahrscheinlich nach einem Entwurf von thomas004)
aus dem neuen Kasten Profi Pneumatic 3