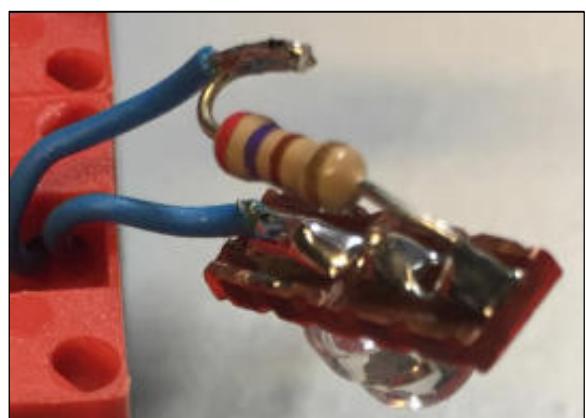
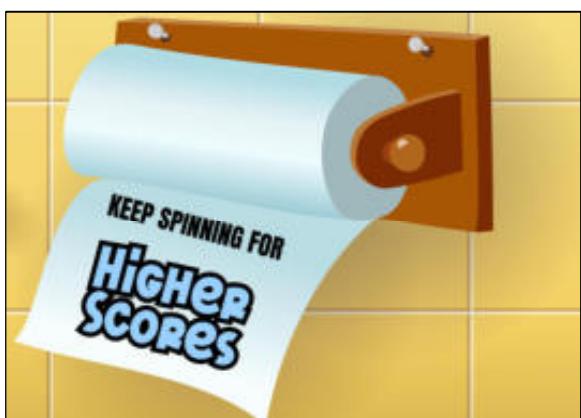
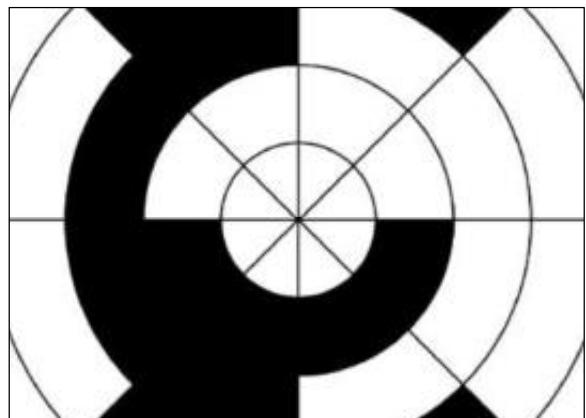


# ft:pedia

Heft 1/2022



MR. WESLER - Sonnenstand am 07. Mai  
Position: 51.2222° N / 7.9541° E  
Tag Nr. des lfd. Jahres: 221  
Mitteleurop. Zeit (MEZ): 10:58  
Wahre Ortszeit (WOZ): 10:24  
Mittlere Ortszeit (MOZ): 10:30  
Zeitgleichung: -5.58  
Sonnenaufgang: 5:03  
Sonnenuntergang: 20:04  
Tageslänge: 15:00  
Deklination: 16.15  
Azimut: 142.68  
Elevation: 50.09  
Refraktion: 0.01



Herausgegeben von  
Dirk Fox und Stefan Falk

ISSN 2192-5879

## Editorial

## Es wird ernst

Die Klagen über den drohenden Mangel an MINT-Fachkräften sind nicht neu. Auch die Ursachen sind bekannt: Sinkende Geburtenzahlen, die Streichung des Technik-Unterrichts in den Grundschulen, die Auflösung des dreigliedrigen Schulsystems und das geradezu phlegmatische Verharren der Curricula für die gymnasiale Mittel- und Oberstufe auf einem von Technik- und Informatik-Themen unbelasteten Fächerkanon ist heftiger Gegenwind für jeden Versuch, den Kern des wirtschaftlichen Erfolgs unseres Landes zu bewahren.

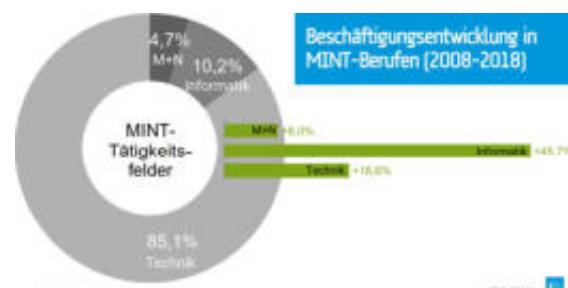
Daher sind die zahlreichen MINT-Initiativen eher ein Tropfen auf den heißen Stein, wie Dr. Christina Anger und Prof. Dr. Axel Plünnecke in dem seit 2011 alle sechs Monate erscheinenden [MINT-Report](#) des Kölner Instituts der Deutschen Wirtschaft dokumentieren. Allen politischen Lippenbekenntnissen zum Trotz fehlt der große bildungspolitische Wurf – jährliche „Girls Days“ oder die Auszeichnung „MINT-freundlicher Schulen“ kann den Wissensverlust eines auf acht Jahre verkürzten Gymnasiums sowie die ausbleibenden Auszubildenden in Technik-Berufen (als Folge der Schließung von Haupt- und Realschulen) nicht kompensieren.

Dabei müsste ein Aufschrei durch dieses Land gehen. Nach den Zahlen der ideologisch unverdächtigen Bundesagentur für Arbeit (BfA) waren 2018 bereits 29% aller Arbeitsplätze MINT-Tätigkeiten – Tendenz steigend. Bald wird jeder dritte Schüler später in einem MINT-Bereich arbeiten, ohne dass ihn die Schule darauf ausreichend vorbereitet hätte.

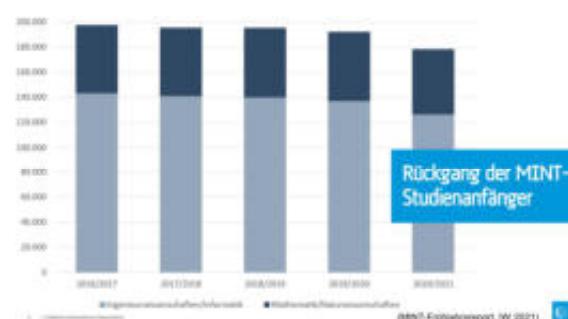
Dirk Fox, Stefan Falk



Mit MINT sind vor allem Informatik und Technik gemeint – die Bereiche mit dem größten Anteil und stärksten Wachstum...



... und sinkenden Studienanfängerzahlen.



Gäbe es kein fischertechnik, wäre unser Schicksal vorbestimmt – ein Volk technischer Dilettanten mit (welt-)wirtschaftlicher Bedeutungslosigkeit.

Herzliche Grüße,  
Euer ft:pedia-Team

## Inhalt

Es wird ernst.....	2
Eierbecher .....	4
Fortbewegung einmal anders.....	15
Chassis für 8x4-Truck.....	21
Eine Kabelklemme mit beweglichen Klemmbacken .....	27
Gabelstapler – ferngesteuert (1) .....	31
Großprojekt Seilbahn (Teil 4): Fundamentale Arbeiten.....	37
fischertechnik-Roboter mit Arduino (Teil 5): Elegante Leuchtsteine und gepimpte LKW .....	44
„Dirty Dishes“ Pinball machine .....	48
Silberlinge: Original oder Nachbau (Teil 6).....	60
Der Zauberling (Teil 4): Die Weiterentwicklung .....	71
Farbsortierer mit dem TCS3472.....	80
Scratch mit fischertechnik – Update 2022 .....	86
Einführung in ftScratch (1): Die Schranke .....	93
Solartracker .....	98
Single Track Gray Encoder mit fischertechnik.....	103
Sensoren am TXT: Die Kamera .....	112
Tuning der Mecanum-Roboter .....	117

## Termine

Was?	Wann?	Wo?
Clubtag	02.07.2022	<a href="#">Montfoort</a>

## Hinweise

[Tilo Rust](#) sucht weitere Mitstreiter für sein grandioses Seilbahn-Projekt – die Bundesgartenschau 2023 rückt näher!

## Impressum

<http://www.ftpedia.de>

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

**Autoren:** Florian Bauer, Arnoud van Delden, Stefan Falk, Dirk Fox, Ralf Geerken, Fabian Haas, Peter Krijnen, Claus Ludwig, Lutz Mönche, Rubem Pechansky, Rüdiger Riedel, Tilo Rust, Michael Schulte.

**Copyright:** Jede unentgeltliche Verbreitung der unveränderten und vollständigen Ausgabe sowie einzelner Beiträge (mit vollständiger Quellenangabe: Autor, Ausgabe, Seitenangabe ft:pedia) ist nicht nur zulässig, sondern ausdrücklich erwünscht. Die Verwertungsrechte aller in ft:pedia veröffentlichten Beiträge liegen bei den jeweiligen Autoren.

Modell

## Eierbecher

Ralf Geerken, Rüdiger Riedel

*Von Anpassungsfähigen über gestürzte Kronen bis hin zu Zweiteilern...*

In vielen Haushalten stellt sich die streitbare Frage, ob Frühstücks-Eier vor dem Verzehr eher weich oder hart gekocht, ob sie danach eher am breiten oder am schmalen Ende geöffnet, oder ob sie dort dann eher geköpft oder gepellt werden sollen. Da hat sicherlich jeder seine eigene Philosophie.

Unstrittig ist aber, dass es sinnvoll ist, nach dem Öffnen einen Eierbecher zu verwenden. Dieses Problems wurde ich mir erst so richtig bewusst, als ich vor ein paar Monaten bei einem Freund zum Frühstück eingeladen war. Er fragte mich was ich denn gerne zum Frühstück hätte. Die Frage konnte ich leicht beantworten: Ein weich gekochtes Ei.

Nicht so leicht zu beantworten war aber die Frage nach Eierbechern.



Abb. 1: Der Anpassungsfähige  
(hier klappt einfach alles)

Als fischertechniker braucht man dann nur kurze Zeit, um für dieses Problem eine Lösung zu finden.

Viel Spaß beim Heraussuchen aus der Teileliste, beim Nachbauen, beim Selber-Erfinden und beim Verwenden, wünschen euch jedenfalls Ralf und Rudi.



Abb. 2: Der Anschmiegsame  
(geeignet für alle Eiergrößen)



Abb. 3: Der Auflehner



Abb. 5: Der edle Dodekaeder  
(für wagemutige Vielwinkelbesitzer)

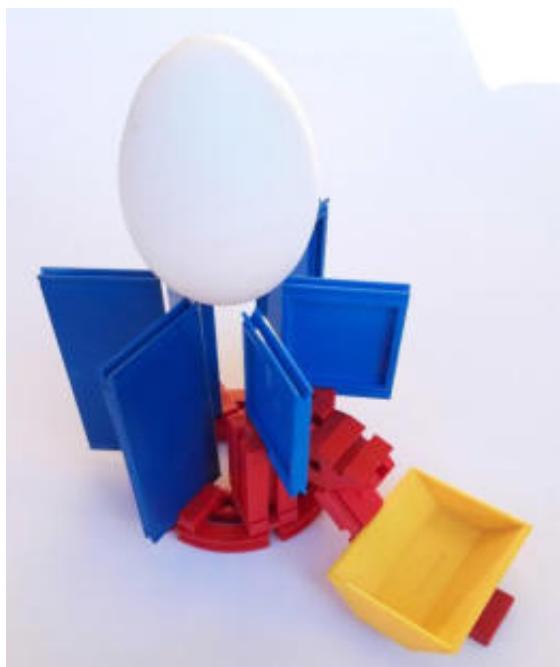


Abb. 4: Der bodenständig Blaue  
(mit Eierschalenablagebehälter)

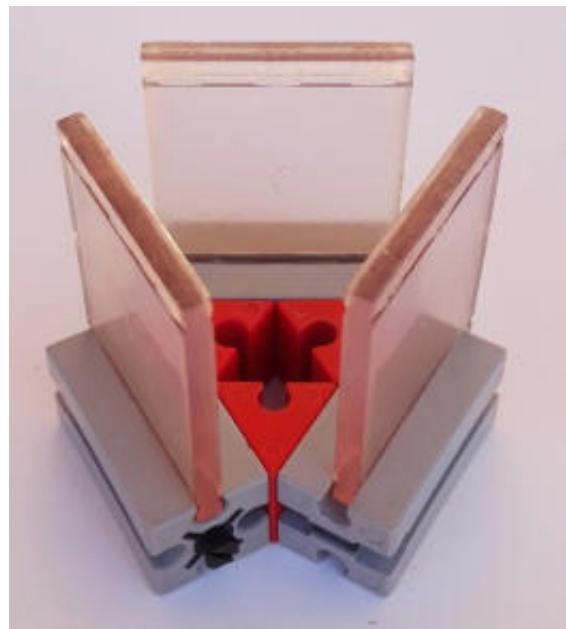


Abb. 6: Der Durchblicker  
(nix für Unix'ler oder Mac'ler,  
sondern nur was für Windows-User)



Abb. 7: Der Einfachste



Abb. 8: Der Familien-Eierbecher  
(oder was für Viereieresser)



Abb. 9: Der grünflammige Fackelträger  
(auf jeden Fall was für Olympioniken)



*Abb. 10: Der Geburtstagstortenhalter  
(nix für über 24-Jährige bzw. Wachteleier)*



*Abb. 12: Der Haltbare (nix für XXXL-Eier)*



*Abb. 11: Der Genügsame  
(auf jeden Fall schnell gebaut)*



*Abb. 13: Der Kragenbecher  
(nix für Halskrausengegner,  
aber anpassbar für alle Größen)*



Abb. 14: Der Hochbeinige  
(nix für echtgoldene Eier)



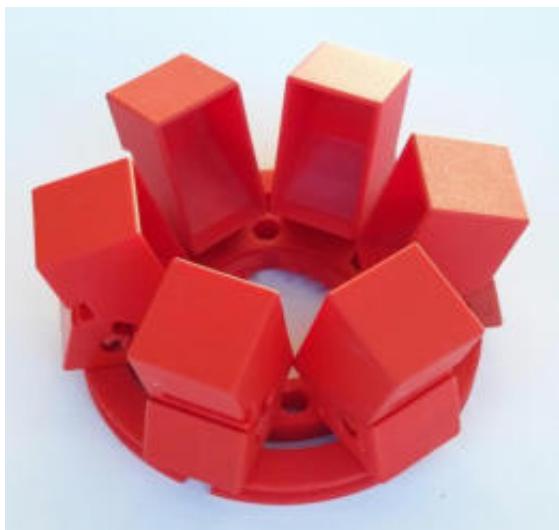
Abb. 15: Der einzige wahre Warmhalte-  
Eierbecher (mit wählbarer Temperatur,  
Trafoverwendung und alten Birnchen, leider  
auf gar keinen Fall spülmaschinenfest)



Abb. 16: Die gestürzte Krone  
(mit Eierschalenablagebehälter)



Abb. 17: Der Lenkbare  
(beidseitig verwendbar)



*Abb. 18: Der Bodenständige für Löffelbaggerer*



*Abb. 21: Der Rutschfeste (für Kunstliebhaber)*



*Abb. 19: Die Michelinvariante*



*Abb. 22: Der bodenständig Ostereierbunte (auf jeden Fall spülmaschinenfest)*



*Abb. 20: Der Raketenartige (auf jeden Fall was für Jules Verne- oder Tim und Struppi-Fans)*



Abb. 23: Der farbenfrohe Plattenbau (nix für West'ler aber was für farbenblinde Ost'ler)



Abb. 24: Der Sanfte  
(auf jeden Fall was für Erdbebenphobiker,  
der taugt auch für Fabergé-Eier)

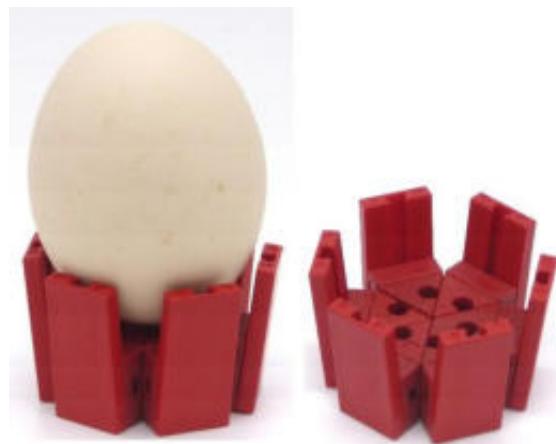


Abb. 25: Der bodenständig Schöne



Abb. 26: Der Verdrehbare  
(Eierscheiben sind nur für Hartgesottene oder  
bei fischertechnikern auch gut für den  
Eigenbau von Eierbechern.  
Leider nicht spülmaschinenfest)



Abb. 28: Der bodenständige Vielwinkler  
(als Armreif war er dem Enkel zu klein)



Abb. 27: Der Bodenständige für Verdrehte



Abb. 29: Der filigrane Weichenaufsteller  
(Eichen sollst du Weichen ... äh, oder?)



Abb. 30: Der Vielzahner (nix für Zahnradlose)



Abb. 32: Der schlichte 58-Zähnige  
(schnell aufzutischen, aber in der  
Spülmaschine sicherlich ein Wasserfänger)



Abb. 31: Der Wackelige  
(nix für dünnwandige Weicheier)



Abb. 33: Der Zweiteiler (Neu trifft auf Alt)

## Bauteile

Art.-Nr.	Bezeichnung	Anzahl
<a href="#">19317</a>	Speichenrad 90x15 Classic Line (rot)	1
<a href="#">31003</a>	Baustein 30 (grau)	4
<a href="#">31005</a>	Baustein 15 (grau)	6
<a href="#">31010</a>	Winkelstein 60° (rot)	47
<a href="#">31011</a>	Winkelstein 30° (rot)	9
<a href="#">31018</a>	Reifen 45x12 (schwarz)	1
<a href="#">31019</a>	Drehscheibe 60x5,5 (rot)	6
<a href="#">31027</a>	Seil 2000x0,5 (blau)	1
<a href="#">31036</a>	Achskupplung 40 Z22 m0,5 komplett (rot)	1
<a href="#">31038</a>	Nockenscheibe KR20 35 (grau)	2
<a href="#">31336</a>	Flachstecker (grün)	6
<a href="#">31337</a>	Flachstecker (rot)	6
<a href="#">31390</a>	Drehkranz-Oberteil Z58 m1,5 (schwarz)	2
<a href="#">31391</a>	Drehkranz-Unterteil (rot)	1
<a href="#">31426</a>	Gelenkwürfel-Zunge 7,5 (rot)	6
<a href="#">31436</a>	Gelenkwürfel-Klaue 7,5 (rot)	12
<a href="#">31556</a>	Flachstein 30x30 (transparent)	3
<a href="#">31577</a>	S-Eckverbinder 15 (grau)	2
<a href="#">31673</a>	S-Sternlasche (rot)	2
<a href="#">31674</a>	S-Adapterlasche (rot)	12
<a href="#">31766</a>	Figur-Sitz (schwarz)	3
<a href="#">31777</a>	Raupenbelag 29,5 Langnut	3

Art.-Nr.	Bezeichnung	Anzahl
<a href="#">31891</a>	Kunststoff-Federboden 5 (schwarz)	4
<a href="#">31892</a>	Kunststoff-Feder 26 (blau)	4
<a href="#">31916</a>	Lenkrad2 Inbus (schwarz)	3
<a href="#">31918</a>	Winkelstein 60° 3N (rot)	6
<a href="#">31981</a>	Winkelstein 15° (rot)	123
<a href="#">31982</a>	Federnocken 25x (rot)	27
<a href="#">32064</a>	Baustein 15 Bohrung (rot)	6
<a href="#">32071</a>	Winkelstein 7,5° (rot)	6
<a href="#">32228</a>	V-Achsschenkel R30 (rot)	6
<a href="#">32330</a>	Bauplatte 15x30x3,75 1N (rot)	7
<a href="#">32861</a>	Bodenplatte 30x90x7,5 (gelb)	4
<a href="#">35052</a>	Winkelträger 7,5° (gelb)	2
<a href="#">35066</a>	Rastachse 90 (schwarz)	6
<a href="#">35072</a>	Rastschnecke 20 m1,5 (schwarz)	6
<a href="#">35384</a>	V-Rohrhülse 30x28 (gelb)	1
<a href="#">35408</a>	V-Achse 28 (gelb)	24
<a href="#">35694</a>	Innenzahnrad Z30 m1,5 69 (rot)	3
<a href="#">35695</a>	Zahnrad Z15 m1,5 Z22 m0,5 anreichbar (rot)	3
<a href="#">35738</a>	S-Winkellasche (1970) (grau)	6
<a href="#">35799</a>	S-Riegelschlüssel (blau)	3

<b>Art.-Nr.</b>	<b>Bezeichnung</b>	<b>Anzahl</b>
<a href="#">35800</a>	S-Riegelschlüssel (grau)	6
<a href="#">36299</a>	S-Winkelträger 30 (gelb)	5
<a href="#">36323</a>	S-Riegel 4 50x (rot)	31
<a href="#">36324</a>	S-Riegel 6 (rot)	17
<a href="#">36329</a>	S-Scharnier (grau)	17
<a href="#">36334</a>	S-Riegelscheibe Z20 m0,5 (rot)	9
<a href="#">36341</a>	Flachstein 30x30 (blau)	2
<a href="#">36342</a>	Flachstein 30x60 (blau)	4
<a href="#">36458</a>	S-Prüfriegel 4 (transparent)	3
<a href="#">36593</a>	V-Grundplatte 45x45x5,5 (rot)	2
<a href="#">36912</a>	S-Strebe 30 L (schwarz)	2
<a href="#">36913</a>	S-Strebe 45 L (schwarz)	6
<a href="#">37209</a>	Förderbecher 15x15x24 (rot)	6
<a href="#">37236</a>	Reifen 60 (schwarz)	1
<a href="#">37237</a>	Baustein 5 (rot)	13
<a href="#">37414</a>	Lenkrad1 sw rt komplett	1
<a href="#">37468</a>	Baustein 7,5 (rot)	18
<a href="#">37869</a>	Kugel-Stecklampe 9V 0,1A (weiß)	6
<a href="#">38216</a>	Leuchtstein-Sockel (schwarz)	6
<a href="#">38413</a>	Kunststoffachse 30 (grau)	1
<a href="#">38423</a>	Winkelstein 10x15x15 (rot)	1

<b>Art.-Nr.</b>	<b>Bezeichnung</b>	<b>Anzahl</b>
<a href="#">38428</a>	Baustein 5 15x30 3N (rot)	2
<a href="#">38448</a>	V-Schaufel 45 (gelb)	1
<a href="#">38535</a>	S-Strebe 60 L (grau)	3
<a href="#">38537</a>	S-Strebe 30 L (grau)	2
<a href="#">38538</a>	S-Strebe 30 L (gelb)	7
<a href="#">38579</a>	S-Strebe 30 l (blau)	2
<a href="#">116251</a>	Baustein 30 (rot)	2
<a href="#">116252</a>	Baustein 15 (rot)	4
<a href="#">130593</a>	Rastaufnahmearm 22,5 (schwarz)	6
<a href="#">139646</a>	S-Strebe 60 L (rot)	3
<a href="#">143235</a>	S-Strebe 30 L (grün)	6
<a href="#">146529</a>	S-Strebe 15 l (grün)	4
<a href="#">156511</a>	Litze 500 (grün-rot)	1
<a href="#">160541</a>	S-Strebe 15 l (chrom-farben (silber))	2
<a href="#">160545</a>	S-Strebe 30 L (chrom-farben (silber))	2
<a href="#">163202</a>	Baustein 15x30 rund (grün)	12
<a href="#">165792</a>	Kugelbahn- Wechselweiche (blau)	3
<a href="#">167368</a>	Dynamics Flexschlauch Befestigung (grün)	1
<a href="#">172541</a>	Radachse (schwarz)	6
<a href="#">172544</a>	Baustein 15x30 rund (gelb)	4
<a href="#">172545</a>	Zahnrad Z15 (gelb)	3

## Modell

## Fortbewegung einmal anders

Rüdiger Riedel, Stefan Falk

*Es gibt ein seltsames Kinderspielzeug, Fahrzeug kann man es kaum nennen. Das Gefährt hat einen Sitz, hinten zwei Räder, einen Drehschemel mit zwei Rädern vorne (manchmal auch nur ein Rad) und daran befestigt ein Lenkrad. Wird das Lenkrad heftig nach rechts und links geschwenkt, bewegt sich das Ding vorwärts. Das probieren wir aus.*

### Der Dreh- und Wackelantrieb

Gesehen hatte ich es vor 35 Jahren im Freizeitpark Schloss Dankern. Meine Kinder, damals 4 und 6 Jahre alt, waren von den Spielmöglichkeiten im Park und in der Halle hellau begeistert. Das Wetter war schlecht, aber in der großen Spielhalle konnten sie sich austoben. Da gab es so skurrile Dinge wie eine Wellenbahn und die hier vorgestellten Wackelfahrzeuge.



Abb. 1: Der Trockenschwimmer-Prototyp

Alle hier vorgestellten Modelle sind sehr einfach aufgebaut. Das erste Modell hat eine weit ausladende Vorderachse, die es wild nach rechts und links schwenkt. Das erinnert ein wenig an heftige Schwimmbewegungen.



Abb. 2: Der Prototyp von unten

Für das elektrisch betriebene Fahrzeug in Abb. 1 und 2 benötigen wir ein nicht mehr hergestelltes Spezialteil, den Polwendeschalter ([31331](#)), den es schon 1968 gab. Zum Glück gibt es gebrauchte, funktionsstüchtige Schalter z. B. noch bei [2] und [3].

Die Drehrichtung des Minimotors ist so eingestellt, dass sich die Drehscheibe 60 ([31019](#)) in Abb. 2 im Uhrzeigersinn bewegt, wodurch das obere Scharnier ([36329](#)) den Polwendeschalter betätigt. Die Drehrichtung des Motors wechselt und die Drehscheibe mitsamt den Vorderrädern dreht sich entgegen dem Uhrzeigersinn. Dieser Vorgang wiederholt sich ständig, die Vorderachse schwenkt heftig hin und her,

das Gefährt bewegt sich tatsächlich vorwärts.

Alle Räder sind frei beweglich. Die Vorderräder laufen in „V-Radachsen“ ([36586](#)), die Hinterräder bestehen aus der Mini-Seilrolle 12 ([38258](#)) mit O-Ring ([35677](#)) in „V-Radhaltern 10“ ([35668](#)).

Die heftigen Umschaltungen sind sicher nicht gut für die Lebensdauer des Mini-Motors. Mit einem Kurbeltrieb lässt sich das ändern.



*Abb. 3: Trockenschwimmer Nummer 2*

Wichtig für die Vorwärtsbewegung ist der Versatz der Vorderachse zur senkrechten Schwenkachse.



*Abb. 4: Unterseite von Trockenschwimmer Nummer 2*

In Abb. 1 und 2 geschieht das durch die V-Stellung, in Abb. 3 und 4 durch einen Versatz nach hinten von der Vorderachse zur Schwenkachse. Dafür brauchen wir einen Baustein 5 15×30 3N ([38428](#)), zwei S-Winkelträger 15 ([35053](#)) und zwei V-Bausteine 15 Eck ([38240](#)).



*Abb. 5: Der dritte Trockenschwimmer*

Der dritte Trockenschwimmer (Abb. 5 und 6) hat eine gekröpfte Vorderachse mittels Winkelsteinen 60°.



*Abb. 6: Das dritte Modell von unten*

In Abb. 6 sieht man meine modifizierte Batteriebox. Ich habe ein Loch hineingeschnitten, so dass ich den speziellen Akku von außen an ein USB-Ladegerät anschließen kann.

Der Versatz der Vorderräder lässt sich gedanklich so weit treiben, dass sie aufeinandertreffen; sie werden zu einem Einzelrad wie in Abb. 7 und 8.



Abb. 7: Der Dreirad-Trockenschwimmer

Es ist jetzt ein V-Rad  $23 \times 10$  (z. B. rot ([36581](#)) oder schwarz ([36574](#))) mit Vollgummireifen ([34995](#)).

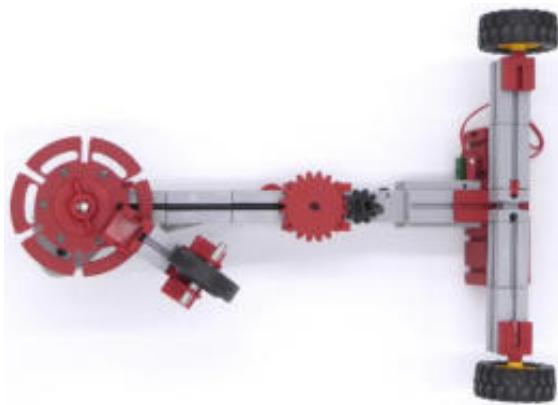


Abb. 8: Das Dreirad von unten

Zur Entlastung des Motors habe ich dem Modell 3 und dem Dreirad jeweils ein Getriebe 2:1 verpasst.

## Miniaturausführung



Abb. 9: Wackelgänger im Miniformat

Die kleine Ausführung des Gefährts erinnert weniger an einen Schwimmer. Vielleicht zeigt sie einen Entengang [1]?

Alle Räder bestehen aus Mini-Seilrollen 12 ([38258](#)) mit O-Ringen ([35677](#)). Es sind keine Rast-Seilrollen ([136775](#))! Die Räder drehen sich frei in den Rastachsen.

Die Gelenkwürfel-Klaue 7,5 ([31436](#)) mit der Lagerhülse 15 ([36819](#)) gleitet leicht über die Rastachse 60 ([35065](#)).



Abb. 10: Wackelgänger schräg seitlich



Abb. 11: Der Wackelgänger von vorne

Im Tandem neigt der kleine Wackelgänger nach kurzer Zeit zum Synchronisieren der beiden Antriebsachsen.



Abb. 12: Das Tandem

Die fischertechnik-Batterien oder Akkus sind für die kleinen Wackelgänger viel zu schwer. Mit fünf Knopfzellen LR44, eingespannt zwischen zwei Stabmagnete 15 mm lang, 4 mm Durchmesser und daran zwei Klemmkontakte ([31338](#), wahlweise blanke Drähte unter die Klemmhülsen ([35980](#)) klemmen) kommen sie gut in Fahrt.

## Was ist das Geheimnis des Wackelvortriebs?

Experimente zeigen, dass für die Vorwärtsbewegung auf glattem Untergrund folgende Voraussetzungen erfüllt sein müssen:

- Alle Räder sind frei beweglich.
- Die Vorderachse muss gegenüber der senkrechten Schwenkachse nach hinten versetzt sein.
- Die Räder sind gummibereifte.

Warum nun genau? Dazu folgende Überlegungen:

### Zwei gelenkte Räder wirken wie eines

Zunächst eine Vereinfachung: Die Modelle mit zwei Rädern an der geschwenkten Vorderachse sind funktional identisch mit solchen mit nur einem Rad an der Vorderachse (dem Dreirad). Das folgt daraus, dass die Vorderachse dieselben Freiheitsgrade hat, egal ob ein mittiges oder zwei äußere Räder darauf sitzen: Die Achse kann in Laufrichtung der Räder rollen, aber nicht quer dazu:

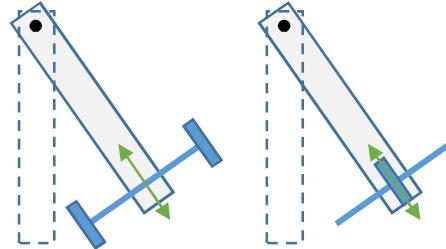


Abb. 13: Zwei Räder bieten dieselben Freiheitsgrade wie nur eines

Deshalb können wir im Folgenden von nur einem Rad ausgehen. Das erleichtert die Erklärung.

### Schwung holen

Gehen wir als Startposition davon aus, dass sich das schwenkbare Rad in der Extremposition – 90° ausgelenkt – befindet und der Schwenkmotor beginnt, das Schwenkrad zurück in die Mittelposition zu drehen:

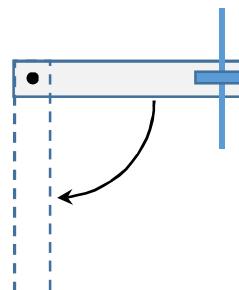


Abb. 14: Beginn des Zurückschwenkens

Das Rad zeigt hier also quer zur zu erklärenden Fahrtrichtung (die in der Zeichnung nach oben verläuft). Deshalb bleibt das Rad auf der Stelle stehen, und das Fahrzeug wird über den Schwenkhebel nach vorne – in Fahrtrichtung – bewegt.

Je näher das Rad seiner Mittelposition kommt, desto weniger stark wird diese Beschleunigung des Fahrzeugs sein. Gleichzeitig steht das Fahrzeug aber auch immer mehr in Laufrichtung des Rades, weil das Rad ja in die Fahrtrichtung hineingeschwenkt wird.

Jetzt kommt der springende Punkt: Um die Mittellage herum hat das Fahrzeug also Schwung (Impuls) nach vorne bekommen – und es kann wegen des gerade passend stehenden Schwenkrades diesen Schwung auch „ausleben“ und sich nach vorne bewegen!

Wenn das Schwenkrad also seine Mittellage erreicht und überschritten hat, hat sich das Fahrzeug ein kleines Stückchen nach vorne bewegt. Es hat Schwung geholt (als das Schwenkrad außen und mehr quer als längs zur Fahrtrichtung stand) und kann durch diesen Schwung etwas fahren (weil mittlerweile das Schwenkrad längs zur Fahrtrichtung steht und sich also drehen kann).

### **Bremsen, aber nicht zurückrollen**

Bei der Bewegung des Schwenkrades über seine Mittelstellung hinaus geschieht zunächst das Umkehrte wie bei der Bewegung von außen in Richtung Mittellage: Das Rad wird relativ zum Fahrzeug gesehen nach vorne bewegt und stellt sich quer zur Fahrtrichtung auf. Das Fahrzeug wird also abgebremst und sogar etwas zurückbewegt. Aber: Es kann diesen Schwung in Rückwärtsrichtung nicht „ausnutzen“, um sich stärker nach hinten zu bewegen, als es durch die Schwenkbewegung selbst bewirkt wird – denn das Schwenkrad steht ja nun wieder quer zur Fahrtrichtung!

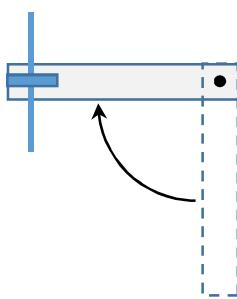


Abb. 15: Situation bei der Bewegung über die Mitte hinaus in die andere Endlage

Wir haben also mindestens zwei Komponenten im gesamten Ablauf:

- Das Bewegen des Fahrzeugs vor und zurück einfach dadurch, dass das Rad auf seine Bahn um die Schwenkachse gezwungen wird, und
- das Schwung Holen nach vorne (!) und das Ausnutzen dieses Schwungs für die Vorwärtsbewegung (während das Rad in Mittelstellung und also in Fahrtrichtung liegt). Hingegen kann während des Schwenks in die andere Richtung (von der Mitte nach außen) eben diese schwungbedingte Bewegung nicht stattfinden, weil das Schwenkrad dann quer zur Fahrtrichtung steht.

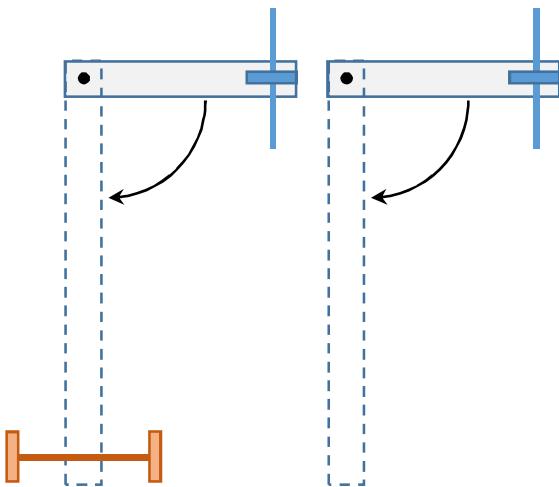
Die Bewegungen nach a) heben sich also gegenseitig auf (vor und zurück), aber die Bewegung laut b) wirkt nur nach vorne, in Fahrtrichtung. Voilà! Das Fahrzeug wird sich nach vorne bewegen, sofern die Lenkbewegung hinreichend schnell erfolgt.

Diese Überlegungen gelten auch für den Fall, dass das Rad nicht bis zur 90°-Auslenkung bewegt wird, nur ist der Effekt dann weniger stark.

### **Warum dürfen die Räder nicht zu glatt sein?**

Wir benötigen gummibereifte Räder. Vorne ist das leicht einzusehen, denn wir brauchen ja bei ausgelenktem Schwenkrad eine ordentliche Kraftübertragung vom Rad auf den Boden, um das Fahrzeug zu beschleunigen. Das Schwenkrad darf also nicht einfach in seiner Querrichtung durchrutschen.

Für die Hinterräder bedarf es aber einer genaueren Beschreibung: Angenommen, die Hinterräder und der Untergrund wären so glatt, dass sie auch in Querrichtung leicht schleifen könnten. Dann würde sich folgende Situation ergeben:



*Abb. 16: (Ideal) rutschende Hinterräder (rötlich, links im Bild) wirken beim Drehen der Lenkung, als ob sie gar nicht da wären (rechts im Bild)*

Die Räder haben also (idealisiert angenommen) überhaupt keine Reibung mit dem Untergrund, unabhängig von der Richtung. Dann wirkt das, als ob der lange Fahrzeugeenträger nur an der Schwenkkachse befestigt

wäre und ansonsten frei schweben würde. Damit aber ist leicht einzusehen, dass das Fahrzeugheck einfach nur wild herumwackeln würde – längs und quer herumgezogen und -geschleudert. Dabei kann keine gerichtete Fahrzeuggbewegung herauskommen. So würde das Modell also nicht funktionieren – wir benötigen rutschfeste Räder.

Ansonsten gilt wie immer: Alles reiner Spaß!

## Quellen

- [1] Rüdiger Riedel: *Der fischertechnik-Trockenschwimmer*. Auf [YouTube](#), 2022.
- [2] fitec Modellbau: <https://fi-tec-shop.de>
- [3] fischertechnik-Teile: <https://fischertechnik-teile.de>

## Modell

## Chassis für 8x4-Truck

Fabian Haas

*8×4-Trucks üben aufgrund ihrer schieren Länge und Leistung eine ganz besondere Faszination aus. Hier ein Modellvorschlag, der wesentliche Teile der Originale in ein fischertechnik-Modell umsetzt: die Doppeltenkung und frei bewegliche Hinterachsen. In einer weiteren Ausbaustufe kommen zwei Motoren und eine von den Aluprofilen getrennte Lenkung hinzu.*

Lange Trucks, die vier Achsen in einem Rahmen vereinigen, üben eine ganz besondere Faszination aus. Hohe Nutzlast bei hoher Motorleistung und Geländegängigkeit vereinen sich hier. Besonders militärische LKW nutzen diese Bauform, aber auch zivile Transporter oder Kipper. Dabei sind besonders bei militärisch genutzten LKW alle Räder angetrieben und man spricht daher, gemäß der gültigen Antriebsformel, von einem 8×8-Lkw (Antriebsformel: Zahl der Räder × Zahl der angetriebenen Räder). Im zivilen Bereich sind es meistens 8×4-Trucks; nur die nicht gelenkten Achsen werden angetrieben. Das verringert zwar die Geländegängigkeit, spart aber Gewicht und Mechanik, was wiederum der Nutzlast zugutekommt. Von passablen Straßen ist im Zivilleben ja auszugehen.

Mich packte die Begeisterung für diese langen Trucks irgendwann und ich musste einfach die Konstruktion in fischertechnik umsetzen. 8×8 erschien mir illusorisch, 8×4 allerdings gut machbar.

### Zuerst der Rahmen

Es standen zwei Aluprofile mit 210 mm Länge zur Verfügung. Als Bereifung entschied ich mich für die Gummireifen 50 (weich) und Felge 30 in Weiß und Gelb. Die Lenkung sollte über die Zahnspurstangen und Lenkhebel realisiert werden; damit war der Abstand zwischen den Aluprofilen auf 30 mm festgelegt. Die beiden notwendigen

Ritzel Z10 mit Spannzangen wurden mit je einem Baustein 30 mit Bohrung zwischen den Aluprofilen in die richtige, mittlere Position gebracht.

Wie üblich wollte ich die Radachsen über Bausteine 15 mit rotem rundem Zapfen beweglich mit dem Aluprofil verbinden. Allerdings war der Baustein 15 nur noch sehr schwer drehbar, darüber hinaus sind die linke und rechte Seite der Lenkung dann nur schwer exakt auf gleiche Höhe zu bringen. Die Bauplatte 15×30×5 mit drei Nuten schaffte Abhilfe. Je zwei Federnocken verbinden die Bauplatte mit dem Aluprofil, die einzelnstehende Nut ist nach innen offen und in dieser kann der Baustein 15 mit rundem Zapfen verankert werden.

An den drehbaren Baustein 15 wurden ein Baustein 15 nach außen abstehend angefügt, um Spurweite und Schwenkbereich zu erhöhen. Die Metallachse 60 wird innen mit einer Klemmbuchse 10 gesichert, außen schließt die Felge 30 mit der Nabennutter ab.

Damit ist die Lenkung fast fertig. Die Zahnspurstangen und Lenkhebel wurden aus Platzgründen nach vorne und hinten gelegt, damit sich Zahnspurstangen und Räder beim maximalen Lenkeinschlag nicht blockieren. Allerdings schlagen bei den Vorbildern die Räder nicht um denselben Winkel ein: Da sich die Räder auf unterschiedlichen Kurvenradien bewegen,

müssen sie auch unterschiedlich stark eingelenkt werden – die erste Achse stärker als die zweite. Wie konnte das in dem Modell realisiert werden?

## Die Vorstudie



Abb. 1: Vorstudie von unten



Abb. 2: Vorstudie von oben

Dazu gab es eine Vorstudie mit Grundplatte  $120 \times 60$ , auf der drei Achsen mit sechs Rädern montiert wurden (Abb. 1 und 2). Für eine vierte Achse war kein Platz und sie war für das untersuchte Problem auch nicht relevant. Zudem wurde ein derartiges Fahrzeug  $6 \times 6$  mit zwei gelenkten Achsen z. B. als Transportpanzer Fuchs realisiert. Es wurden zwei Zahnpurstangen montiert, auf der Unterseite Ritzel Z10 und auf der Oberseite standen die Achsen heraus. Sehr oft werden die Zahnpurstangen durch eine Schubstange/Querlenker verbunden, wie ich im Nachhinein herausfand. Ich nutze jedoch die beiden Achsen, um auf der Oberseite der Grundplatte zwei Zahnräder anzubringen. Da die Zahnpurstangen unterschiedlich einschlagen sollten, mussten die beiden oberen Zahnräder unterschiedlich sein, allerdings nicht zu unterschiedlich – die Winkel der Räder sind ja ähnlich.

## Das Fahrwerk

Die Wahl im endgültigen Modell fiel auf eine Kombination von Z20 und Z30, die Verbindung mit Kette erschien mir am einfachsten. Ausgiebige Fahrtests auf dem Wohnzimmerboden ergaben, dass ich das richtige, ein 2:3-Verhältnis, getroffen hatte – alle Räder bewegen sich zwangslässig in ihrer Spur und werden nicht über den Boden geschoben.

Diese Konstruktion wurde nun am großen Modell realisiert. Sie bietet mit den Gummireifen einen guten, spurtreuen Lauf und schafft zudem eine Möglichkeit zum Anschluss eines Steuermotors. An den vorderen Stirnzapfen der Aluprofile wurden zwei Bausteine 30 angebracht, da die Zahnpurstangen nach vorne überstehen. Außerdem lassen sich die Aluprofile bei der Justierung sicher z. B. gegen eine Tischplatte drücken und die Bestandteile und Lenkgruppen ausrichten.

Damit war die Doppellenkung geschafft, allerdings fehlten die beiden hinteren Achsen, die im LKW zum Antrieb genutzt

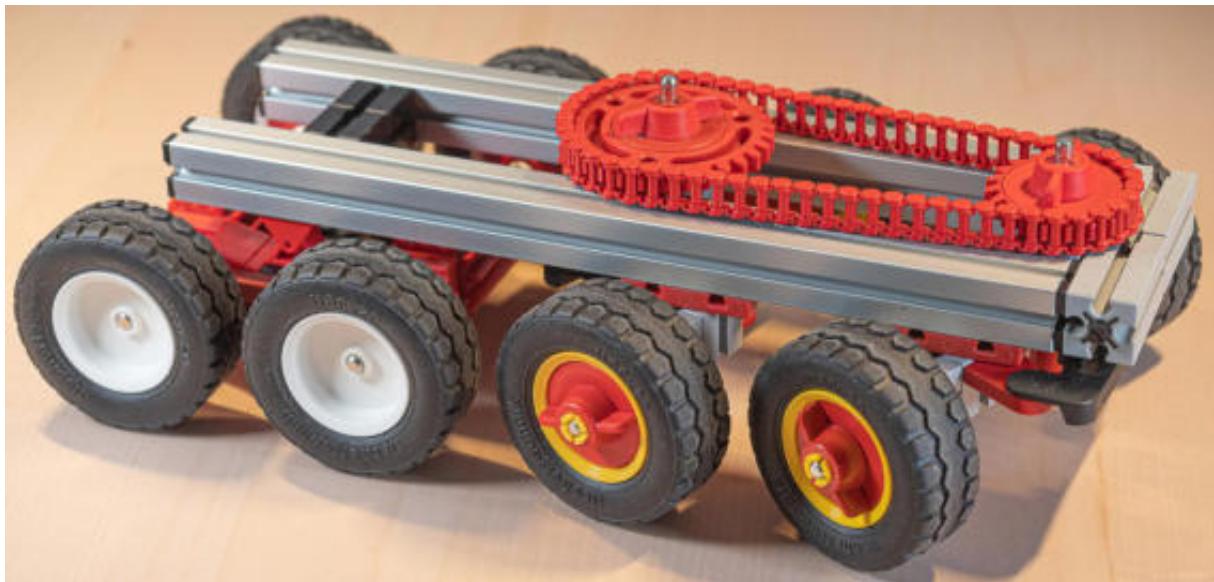


Abb. 3: Das fertige unmotorisierte Modell

werden. Sie sollten im Modell zu gegebener Zeit den Antrieb aufnehmen.

Die Achsen sollten einen eigenen, beweglichen Rahmen haben und natürlich sollten die einzelnen Räder bei der Kurvenfahrt geschmeidig mitrollen. Daher wurden – auch ohne Antrieb – die Differentialgetriebe Z15 und Achsen 110/120 als Achsen gewählt. Diese werden durch

Bausteine 15 mit Bohrung geführt und über eine Kombination von Bausteinen 5, Winkelsteinen 15 und einem zentralen Baustein 15 zusammengeführt.

Mit dieser schwanenhalsartigen Konstruktion konnten die Aluprofile horizontal ausgerichtet werden. Stimmte diese Höhe nicht, würden die Profile nach oben oder unten zeigen.

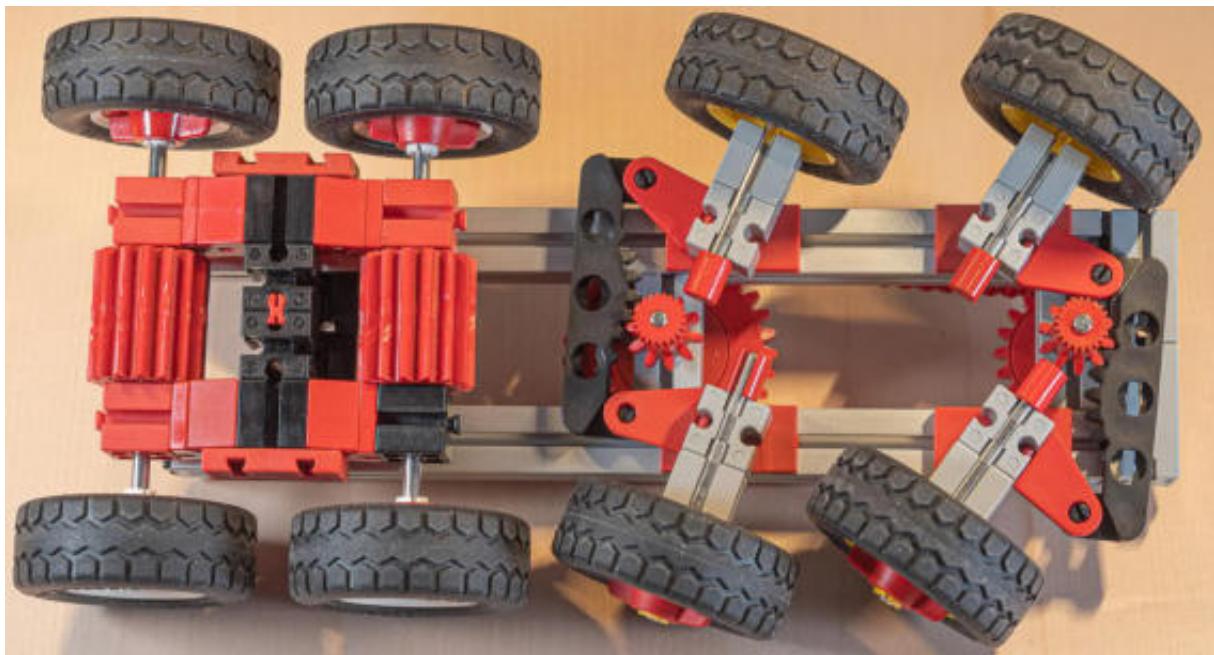
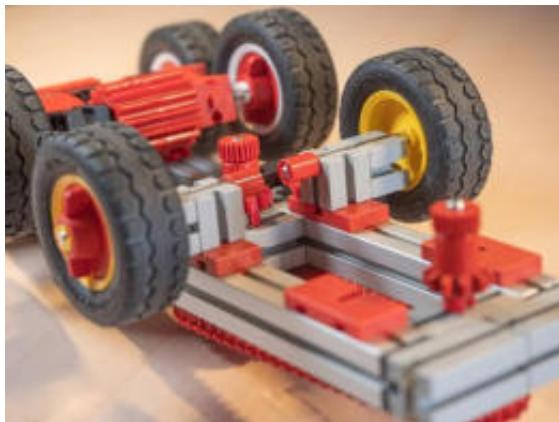


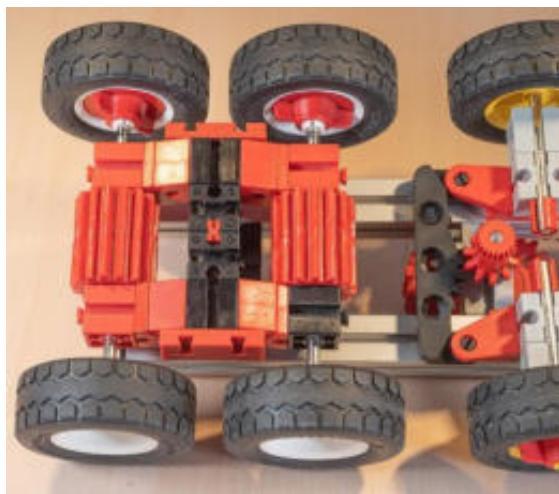
Abb. 4: Das unmotorisierte Modell von unten

Der Rahmen für die Hinterräder wurde am zentralen Baustein 15 über zwei Gelenksteine 15 mit den Aluprofilen verbunden. Obwohl die Differentialgetriebe Z15 den Abstand stabilisieren, wurden noch zwei Bausteine 15 mit Verbindungsstücken 15 links und rechts von den zentralen Bausteinen 15 eingefügt.



*Abb. 5: Detailblick auf die Lenkung*

Damit war das Modell fertig! Acht Räder, vier davon lenkbar, mit unterschiedlichem Einschlag und die Hinterräder an einem beweglichen Rahmen aufgehängt: Damit ist bei Bodenwellen und Stufen ein kontinuierlicher Bodenkontakt gewährleistet!

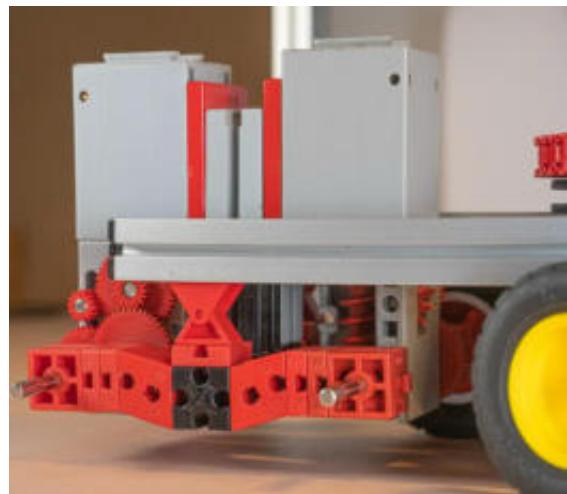


*Abb. 6: Der unmotorisierte Hinterrad-Rahmen*

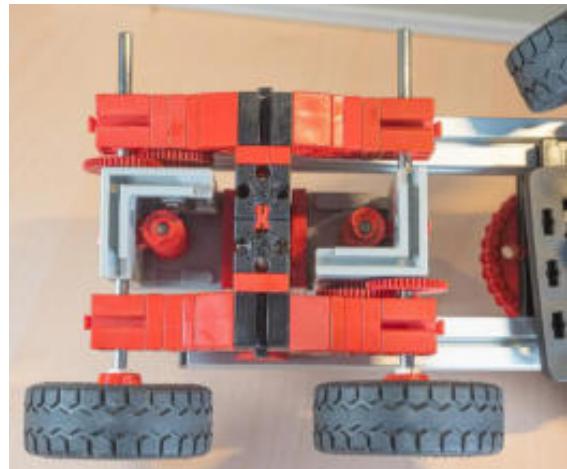
## Die Motorisierung

Die Motorisierung erwies sich allerdings als schwieriger als gedacht, obwohl schon Differentiale eingebaut waren. Im Kern

scheiterte die Motorisierung am zu geringen Abstand der beiden Profile. Die Motoren mussten mit dem Hinterradrahmen verbunden werden, um die Beweglichkeit bei gleichzeitigem Antrieb sicher zu stellen. Alle untersuchten Lösungen brauchen mehr Abstand zwischen den Aluprofilen.



*Abb. 7: Zwei Motoren geländegängig eingebaut*



*Abb. 8: Motoren von unten gesehen*

Der Rahmen wurde dann um einen Baustein 15 erweitert, so dass zwei Motoren mit Stufengetrieben (wegen der erzeugten Kraft, aber auch wegen ihrer kompakten Bauform) und Achsen 110 Platz fanden. Gehalten werden die Motoren durch eine gemeinsame Strebe aus vier Bausteinen 30, die zwischen den zentralen Bausteinen 15 mit zwei Bausteinen 5 mit zwei Zapfen angehängt ist. Die notwendige Anpassung

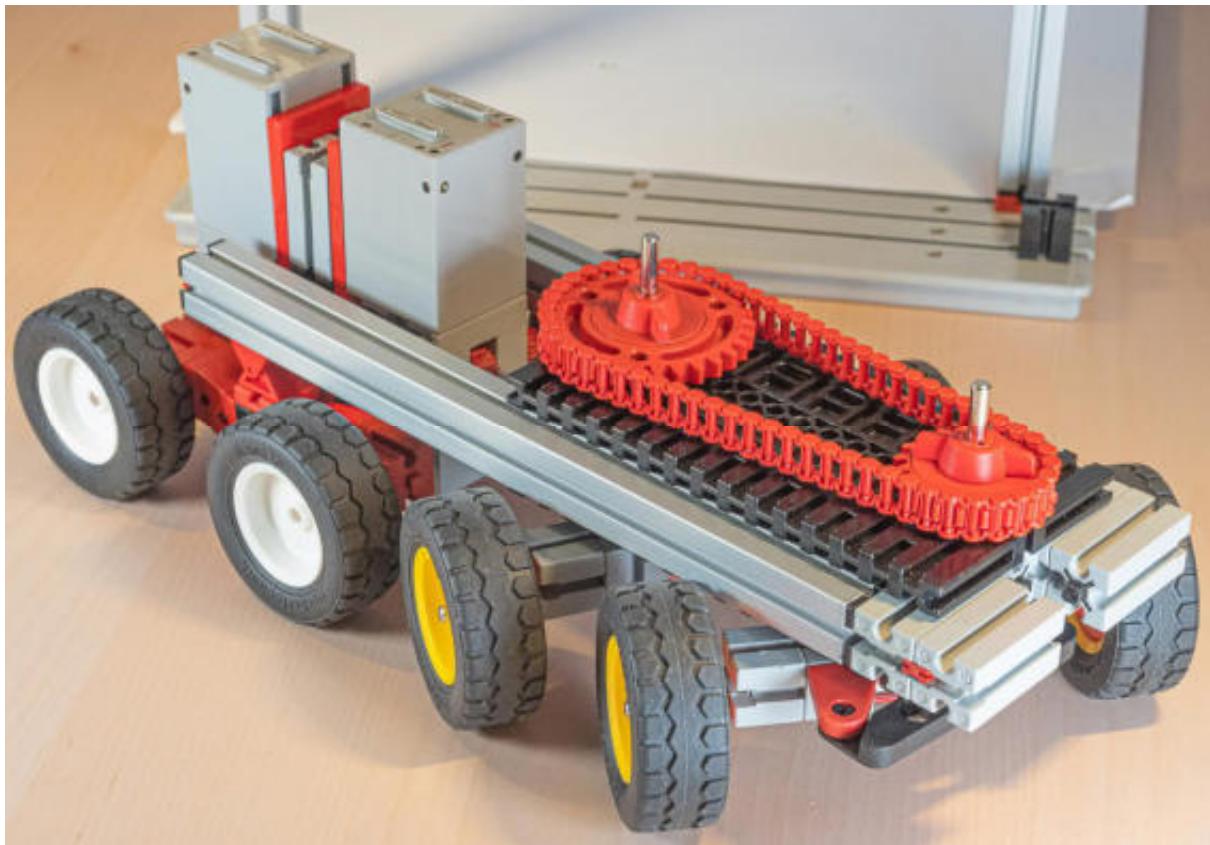


Abb. 9: Das motorisierte Modell

der Abstände zwischen dritter und vierter Achse wurde mit Bausteinen 5 realisiert. Die Verbindung mit den Aluprofilen erfolgt nach wie vor mit Gelenksteinen 15. Abb. 7 gibt darüber genauer Auskunft.



Abb. 10: Geländegängigkeit

Das Modell verfügt jetzt also über zwei Antriebsmotoren!

## Die Lenkung

Ein Problem gelöst, ein weiteres tat sich auf: Der Abstand der Zahnspurstangen ist notwendigerweise 30 mm zwischen den Bausteinen 15 mit rundem Zapfen. Wie konnte das gelöst werden? Eine vom Abstand der Aluprofile unabhängige Lösung musste her, aber mit exakt mittiger Ausrichtung! Hier konnte auf die Modellstudie zur Lenkung zurückgegriffen werden. Die Lenkung und die Grundplatte 120×60 waren sehr gut vereinbar, so dass ich die Bauplatte 15×30×5 mit drei Nuten verwarf (ebenso wie die Bausteine 30 mit Bohrung zwischen den Aluprofilen) und durch die Grundplatte 120×60 ersetze.

Diese musste nur noch zentriert werden. Der Abstand zwischen den Aluprofilen betrug nun 45 mm. 30 mm davon lassen sich mit den Bausteinen 15 abdecken. Bleiben noch 15 mm, gleichmäßig auf beiden Seiten, also  $2 \times 7,5$  mm. Dieser

Abstand lässt sich mit dem Baustein 7,5 passgenau abdecken. Mit Federnocken sind die Aluprofile, Bausteine 30 und 15 sowie Bausteine 7,5 miteinander verbunden, so dass die nun von den Aluprofilen unabhängige Lenkeinheit zentriert liegt.

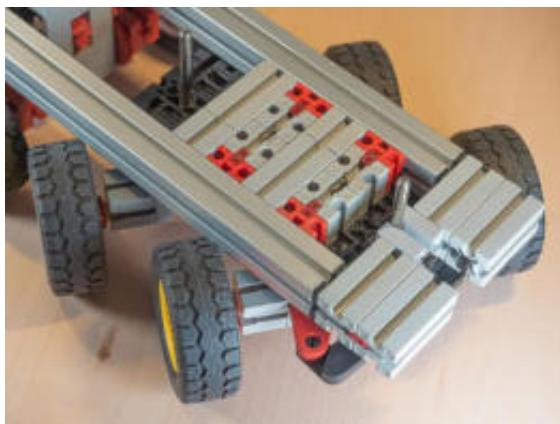


Abb. 11: Die breitere Lenkung von oben

Damit steht eine unabhängige Lenkeinheit für alle möglichen weiteren Modelle zur Verfügung. Eine zweite Grundplatte 120×60 wurde von oben aufgelegt und verbessert die Führung der beiden Achsen, auf denen die Ritzel Z10 und die Zahnräder

Z20 und Z30 liegen. Damit ist viel Platz für eine motorische Ansteuerung der Lenkung.

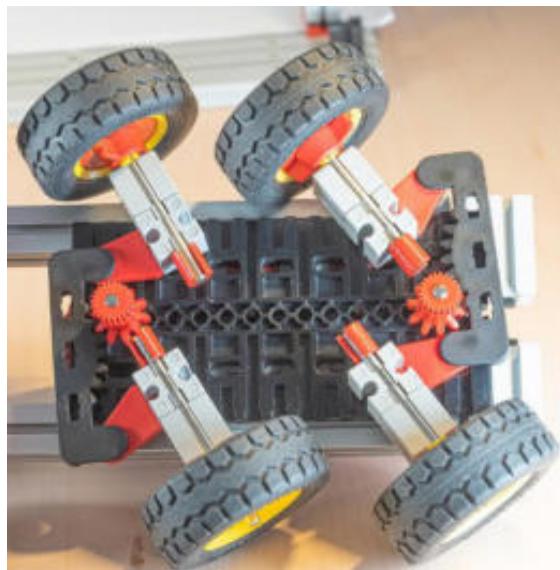


Abb. 12: Die breitere Lenkung von unten

So, nun viel Spaß bei den Gelände- und Zugtests des zweimotorigen 8×4-Chassis! Zwei Motoren haben nicht mal die Originale. Ich bin sicher, jemand realisiert interessante Aufbauten, genug Antriebskraft steht mit den beiden Motoren ja zur Verfügung.

Modell

## Eine Kabelklemme mit beweglichen Klemmbacken

Fabian Haas

*Immer wieder stehen wir vor der Aufgabe, Dinge, die keine fischertechnik-Verzahnung haben, in der fischertechnik-Welt zu verwenden und so auszurichten, dass wir sie verwenden können. Bei mir war es kürzlich eine Endoskopkamera mit einem recht widerspenstigen Kabel. Ich zähmte es durch eine Klemme, die entsprechend dem Sondenkabel recht schwer ausfiel. Damit klemmt es – sicher und zuverlässig!*

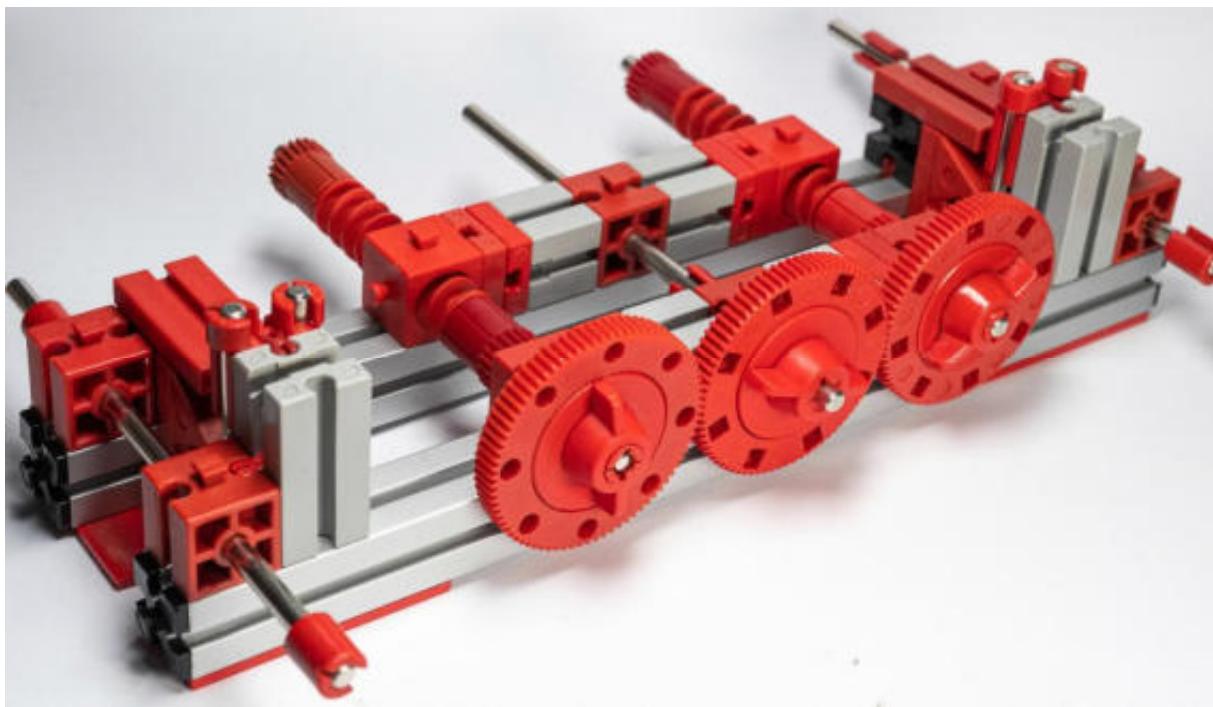


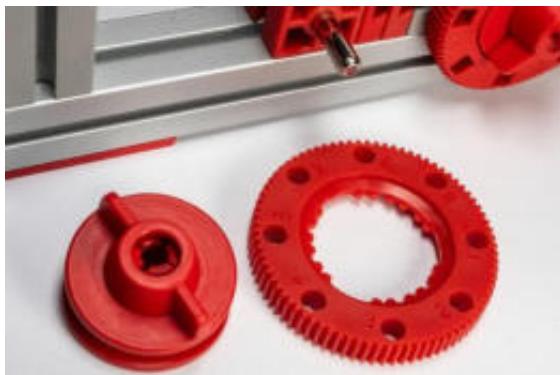
Abb. 1: Übersicht über die Klemmvorrichtung (Länge 210 mm) mit je einer Klemmbache am Ende der Aluprofile und den Drehkranzoberteilen als Handdrehräder

Nach der Beschaffung einer einfachen Endoskop- bzw. Sondenkamera, gedacht für die Inspektion von Rohren und Hohlräumen, war schnell klar, dass es vernünftige, unverwackelte Aufnahmen nur mit einer festen Halterung geben wird. Diese sollte dann gleichfalls in die fischertechnik-Welt mit Motoren, Profilen und Winkeln vermitteln. Schließlich sollte alles zu meinem

bereits vorgestellten „Fotostudio für Makrofotos“ passen, das in Heft 2021/2 vorgestellt wurde [1].

Eine einfache Lösung aus ein paar Bausteinen 30, einer Schneckenmutter, einer klemmbaren Schnecke ( $m = 1,5$ ) und einem Drehschalteroberteil ([32117](#), Z80, mit Verzahnung, Abb. 2) war schnell gebaut

(nicht gezeigt), erwies sich aber als insgesamt zu wackelig. Das Kabel des Endoskops ist recht steif, was für das Vordringen in Rohre sinnvoll ist; die Ausrichtung auf ein Fotomotiv wird dadurch aber nicht einfacher. Daher sollte das Kabel über weiter auseinander liegende Klemmbacken und in einer insgesamt stabileren Halterung gesichert werden.



*Abb. 2: Drehkranzoberteilen Z80 mit Verzahnung. In manchen ec-Kästen war ein Oberteil ohne Verzahnung beigelegt (was für den Drehschalter ja sinnvoll ist), hier wird aber eines mit Verzahnung benötigt, damit Kraft auf die Nabe übertragen werden kann.*

Daher musste eine schwerere, festere Lösung gefunden werden (Übersicht in Abb. 1). Es standen zwei Aluprofile mit 210 mm Länge zur Verfügung (wobei die Länge nicht entscheidend ist). Auf dem einen Aluprofil wurden fünf Bausteine 15 mit Bohrung platziert, auf dem anderen Profil nur drei, aber zusätzlich zwei Schneckenmuttern, die die klemmbaren Schnecken empfangen. Bei dem abzusehenden Arbeitsweg reicht eine Schnecke aus zwei klemmbaren und mit Zangenmuttern festgezogenen Schnecken – für größere Abständen einfach verlängern.

Verwendet werden fünf Metallachsen 110. Dabei dienen die beiden äußeren nur als Führung und Stabilisierung beim Einklemmen des Kabels. Die nächsten beiden (von außen nach innen gesehen) tragen die Schnecken und die mittlere Metallachse 110 ist wieder eine Führung. Aufgrund des gleichen Drehsinns der Schneckengewinde

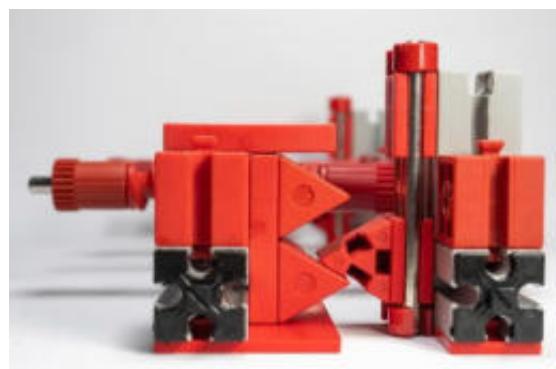
musste zwischen den beiden Drehschalteroberteilen (Z80) ein drittes Zahnrad eingefügt werden. Dadurch erfolgt eine zweimalige Umkehr der Drehrichtung und die Schnecken laufen wieder synchron.

Ich wählte – ohne besonderen Grund – wieder ein Drehschalteroberteil (Z80). Durch diese Wahl ergibt sich ein rasterkonformer Abstand zwischen den Bausteinen 15 mit Bohrung und den Schneckenmuttern, der jeweils mit einem Baustein 15 und zwei Bausteinen 5 realisiert wird. Etwas mehr Gewicht im Modell schadet nie, wenn es um eine stabile Halterung geht.

Bis jetzt liegt nur die Grundkonstruktion vor: Zwei Aluprofile, die sich über zwei Schnecken gegeneinander verschieben lassen (Abb. 1). Allerdings fehlen noch die Klemmbacken, in denen das Endoskopkabel tatsächlich eingeklemmt wird. Die Klemmbacken sollten selbstausrichtend und flexibel einsetzbar sein.



*Abb. 3a: Klemmbacken in Nahaufnahme, geschlossen*



*Abb. 3b: Klemmbacken in Nahaufnahme, geöffnet*

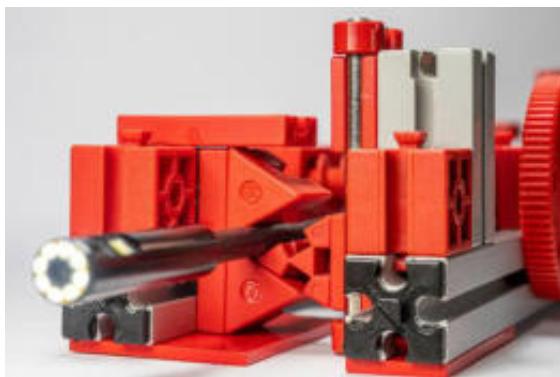


Abb. 4: Ein geklemmtes und ausgerichtetes Kabel der Endoskopkamera

Nach einigen Versuchen wird folgende Konstruktion verwendet, in der insgesamt drei Winkelsteine  $60^\circ$  mit drei Nuten verwendet werden (Abb. 3a, und 3b). Zwei davon liegen fest, sie wurden nebeneinander mit Federnocken auf einem Baustein  $15 \times 30 \times 5$  mit Nut und Zapfen angebracht.

Da beim Einklemmen signifikante Kräfte auf die Backen wirken, wurden sie oben mit einem Baustein  $15 \times 30 \times 5$  mit Nut und

Zapfen und unten mit einer Bauplatte  $30 \times 45$  gesichert. Die Bauplatte dient gleichzeitig als Auflage auf einem Tisch und ist deshalb größer gewählt als eigentlich notwendig. Am gegenüberliegenden Aluprofil sorgt die Bauplatte  $15 \times 45$  für die identische Höhe.

Auf dieser Seite wird die bewegliche Klemmbacke verwendet (Abb. 3a, b). Ein Winkelstein  $60^\circ$  mit 3 Nuten befindet sich auf einem Baustein 7,5, der auf zwei Metallachsen 50 beweglich ist. Die Metallachsen 50 werden von einem Baustein 7,5 gehalten, der an einem Baustein 30 mit Federnocken fixiert ist. Die Bausteine sind mit Federnocken oder Verbindungsstücken am Aluprofil und benachbarten Baustein 15 mit Bohrung gesichert und stabilisiert.

Beim Schließen wird also der Winkelstein  $60^\circ$  über den Baustein 7,5 direkt auf das Profil gedrückt und kann nicht ausweichen. Die Kräfte werden direkt ins Aluprofil

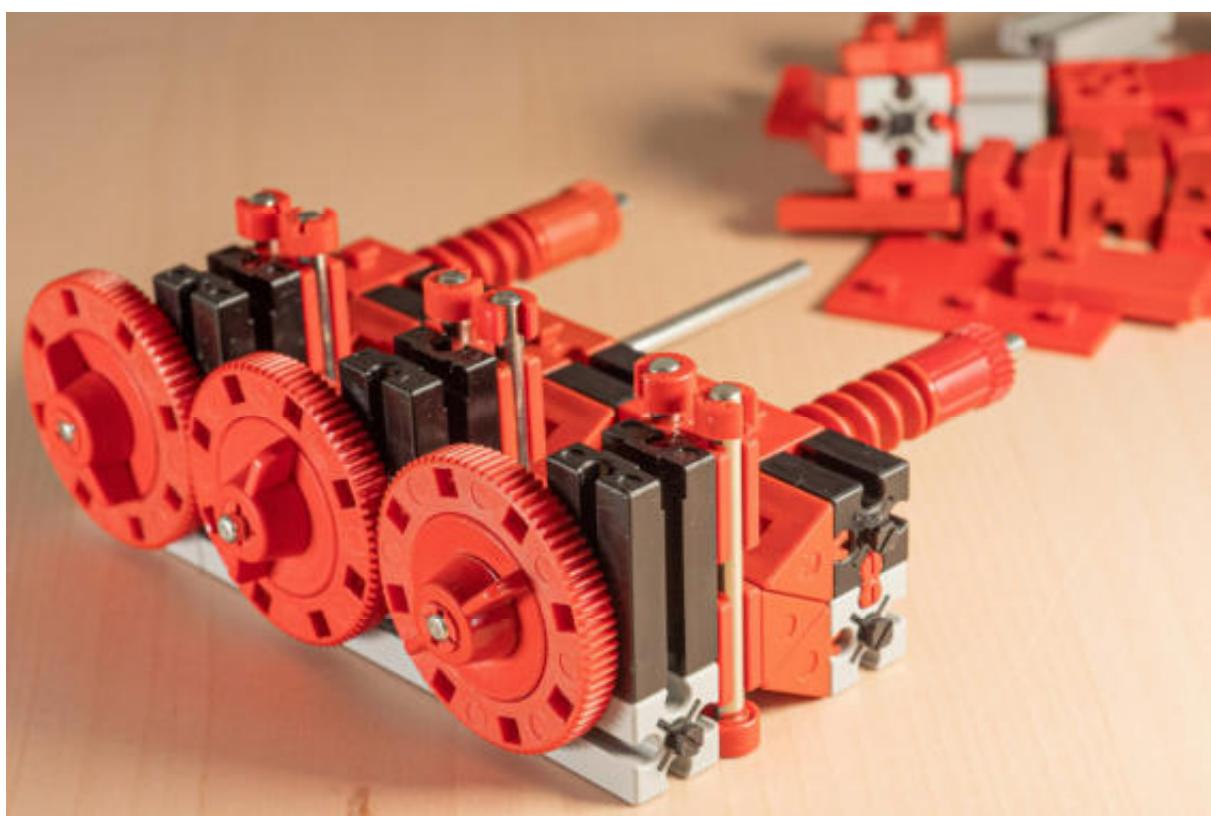


Abb. 5: Weitere, kompaktere Evolutionsstufe der Klemmvorrichtung für Kabel, ohne Aluprofile – und die dadurch eingesparten Bauteile

gelenkt, wodurch keine besonders schwere Halterung für die Achsen 50 notwendig ist.

Je eine komplette solche Klemmbacke wurde an beiden Enden der Aluprofile montiert, auf eine dritte oder vierte wurde bisher verzichtet; die Klemmung ist fest genug. Sie reicht aus, um das Kabel gegen Verdrehung in den Klemmbacken und gegen Verwindung zu sichern. Die Kräfte können erheblich sein, deshalb ist eine Sperrung der beiden Winkelsteine 60° der einen Seite gegen eine Verschiebung nach oben und unten notwendig ist. Die Gewinde der Schnecken erzeugen genug Reibung in den Schneckenmuttern, dass sich die Gewinde nicht von selbst lockern. Die Klemmbacken lassen sich wirklich festdrehen und es bedarf keiner weiteren Sicherung gegen die unbeabsichtigte Öffnung (Abb. 4).

Meine Bilder geben Aufschluss über weitere Details. Durch längere Schnecken lässt sich der Arbeitsbereich anpassen und

eine Motorisierung ließe sich über die Dreh-schalteroberteile, die hier einfach als Z80 verwendet werden, realisieren. Wer es etwas kleiner mag, kann kürzere Aluprofile verwenden oder ganz auf diese verzichten.

An den Schneckenmuttern und den Bausteinen dazwischen ist genug Raum, um Klemmbacken zu montieren. Die Klemmbacken und Aluprofile sind im Wesentlichen unabhängig voneinander; ein Widerlager für die Klemmbacken lässt sich ggf. auf andere Art und Weise montieren. Und wer den Mechanismus auf die Seite legt, bekommt einen Teil einer Hebebühne...

Die Endoskopkamera ist nun auf alle Fälle fest fixiert und in das fischertechnik-System wackelfrei integriert! Fotos und Filmen aus der Miniaturwelt steht also nun nichts mehr im Wege.

## Referenzen

- [1] Fabian Haas: *Fotostudio für Makrofotografie*. [ft:pedia 2/2021](#), S. 4–6.

Modell

## Gabelstapler – ferngesteuert (1)

Claus Ludwig

Ferngesteuerte Modelle sind schon immer fester Bestandteil meines fischertechnik-Programms. Anfangs nur mit Fernsteuerungen von fischertechnik, später auch mit Multifunktionsfernsteuerungen aus dem Modellbaubereich. In loser Folge möchte ich einige Modelle und deren Fernsteuerungen in der ft:pedia vorstellen. Den Anfang macht der Ende 2021 entstandene Gabelstapler.

### Hintergrund

Die Idee, einen Gabelstapler zu bauen, ist über zehn Jahre alt und hat ihren Ursprung in dem von Harald Steinhaus 2009 vorgestellten Modell [1]. „Gut Ding will Weile haben“ – nun war die Zeit reif. Die Hubvorrichtung habe ich von Harald übernommen und nach meinen Vorstellungen weiterentwickelt. Der Stapler selbst und der Gabelhalter mit Gabeln sind Neukonstruktionen.



Abb. 1: Der Gabelstapler

Zur Steuerung wird die aktuelle Fernsteuerung von fischertechnik verwendet. Neben den Fahrfunktionen kann der Gabelhalter mit den Gabeln gehoben und gesenkt sowie nach vorne und hinten geneigt werden.



Abb. 2, 3, 4: Ansicht von hinten, oben, unten

## Hubfunktion/-vorrichtung

In der Grundkonstruktion der Hubvorrichtung von Harald Steinhaus wurde im unteren Bereich der Winkelträger durch Grundbausteine ersetzt und das Ganze durch zusätzliche Bausteine verstellt. Für den Antrieb wird ein Powermotor 50:1 (rote Kappe) verwendet – wie auch von Harald empfohlen (Abb. 5).

Die Lage des Motors wurde so weit wie möglich nach unten verlegt, um einen tiefen Schwerpunkt zu erreichen. Um die Kraft des Motors auf die Schneckenwellen links und rechts zu übertragen, kann ein normales Anziehen des Z20 mit Hand – nach meinen Erfahrungen – nicht ausreichend sein. Bei größerer Belastung (zu hebenden Gewichten) kann das Z20 auf der Antriebswelle durchrutschen.

Es gibt mehrere Möglichkeiten damit umzugehen:

- Es reicht einem aus, was auf diese Weise gehoben werden kann, und man nutzt das Durchrutschen des Z20 als natürlichen Überlastungsschutz (Rutschkupplung);
- man zieht die Nabennutter mit dem Spezialschlüssel von Andreas Tacke an [2], was aber zu Verwerfungen des dünnen Z20-Rings führen kann und einen „eirigen“ Lauf des Z20 zur Folge hat; oder
- man verklebt die Flachnarbe und Steckachse mit Sekundenkleber, mit dem Nachteil, dass man die Teile später nicht anders verwenden kann.

Um die Stabilität weiter zu verbessern wurde der Abstand zwischen den beiden Schneckenmuttern auf der linken und rechten Spindel (Abb. 6) jeweils um 15 mm vergrößert. Zuletzt wurde die Hubvorrichtung am oberen Anschlag noch mit einem Endschalter versehen (Abb. 8).



Abb. 5: Hubvorrichtung

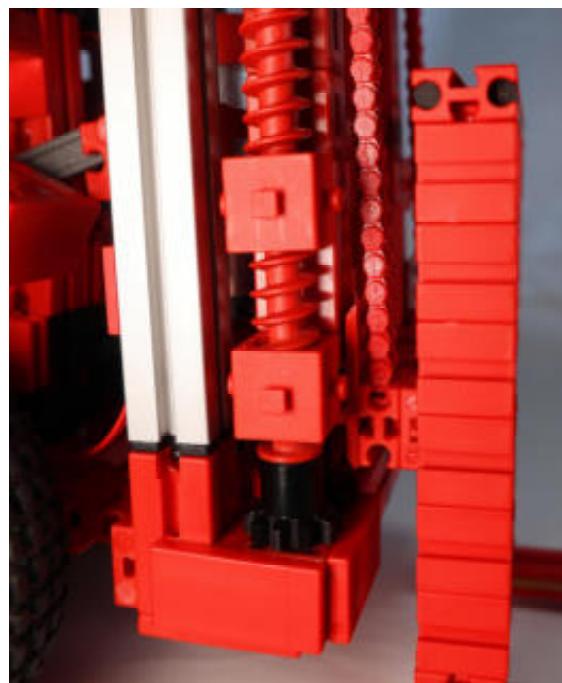


Abb. 6: Seitenansicht des Schneckenantriebs

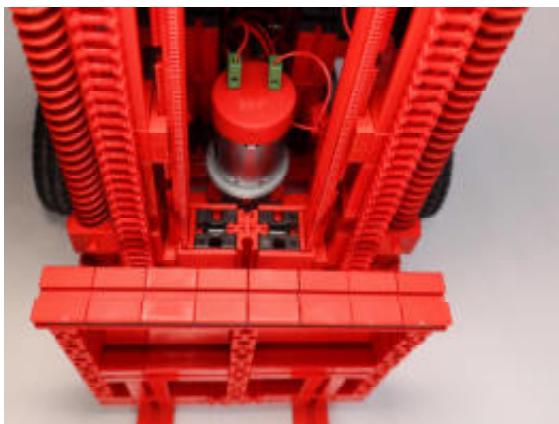


Abb. 7: Hubmotor

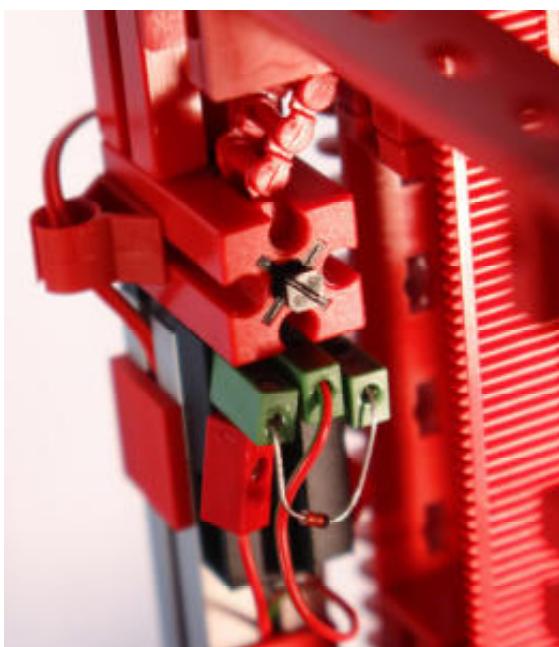


Abb. 8: Endlagentaster

## Gabelhalter und Gabeln

Der Gabelhalter mit den Gabeln wurde neu konstruiert. Um möglichst flache und stabile Gabeln zu erhalten, wurde eine Konstruktion aus Messingstangen und Bausteinen 7,5 gewählt (Abb. 10). Jede Gabel besteht aus zwei Messingstangen mit einem Durchmesser von 4 mm und einer Länge von 105 mm und sieben Bausteinen 7,5. Zur Befestigung der Teile wurden diese an den Enden mit Sekundenkleber verklebt.

Sekundenkleber wurde auch eingesetzt bei der oberen Verbindung vom Vorderteil des Gabelträgers und den Führungen (Abb. 11).



Abb. 9: Gesamtansicht ohne Gabel

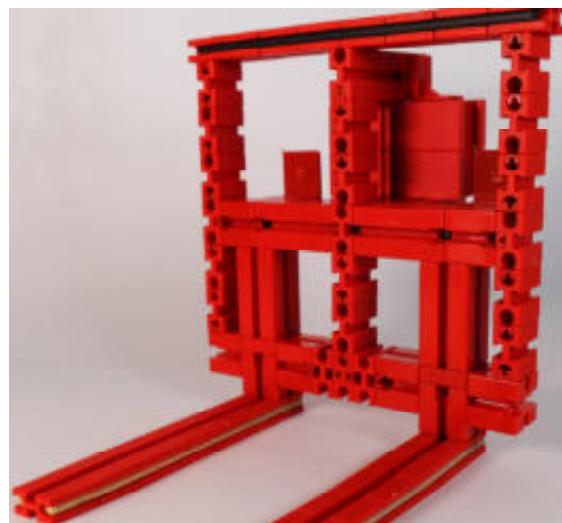


Abb. 10: Gabel mit Versteifungen

## Neigetechnik

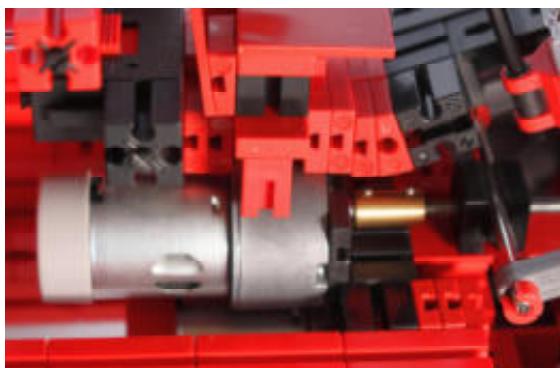
Wie beim Original wurde die Hubvorrichtung mit einer Neigetechnik ausgestattet. Dies erfolgt über einen Powermotor 20:1 (graue Kappe), eine 4 mm-Gewindestange,

einen Baustein 15 mit Gewinde und entsprechendem Gestänge. Motor und Gewindestange werden über einen Verbinder/ Adapter von Andreas Tacke verbunden.



*Abb. 11: Gabel von hinten*

Wer nicht über einen entsprechenden Baustein mit Gewinde verfügt und diesen auch nicht über [fischerfriendsman](#) (Stefan Roth) beziehen kann, der kann diesen auch selbst bauen. Benötigt werden ein Baustein 15 mit Ansenkung (32321), eine M4 Gewindemutter, etwas Gewindestange und eine Flügelmutter. Die Mutter wird auf die Gewindestange aufgeschraubt und erwärmt. Der Baustein mit Ansenkung wird Richtung Mutter auf die Gewindestange aufgeschoben und mittels der Flügelschraube auf die Gewindemutter gepresst.



*Abb. 12: Antrieb der Neigemechanik*



*Abb. 13: Einbau im Modell*

## Der Stapler

Der Stapler selbst ist eine komplette Neukonstruktion. Möglich wurde dies insbesondere durch die bei Aufräumarbeiten gefundenen Reifen von Conrad, die für die Vorderräder verwendet wurden und die den Maßstab vorgeben. Die Reifen werden seit ca. 2005 immer wieder gern von vielen Kollegen für fischertechnik-Modelle eingesetzt, sind aber seit einigen Jahren nicht mehr im Programm. Was man auf den Bildern nicht sehen kann, ist, dass ich, seitdem ich mit diesen Reifen arbeite, sie gerne mit den großen fischertechnik-Traktorreifen fülle. Von den Traktorreifen habe ich in der Regel mehr als ich brauche, und die Conrad-Reifen erhalten so eine deutlich bessere Festigkeit und Belastbarkeit.



*Abb. 14: Antriebsachse*

Der Antrieb erfolgt über einen Powermotor 50:1 und dem aktuellen Differenzial. Anstelle der Rastachsen aus Kunststoff wurden Metallachsen von Andreas Tacke verwendet (Abb. 14).

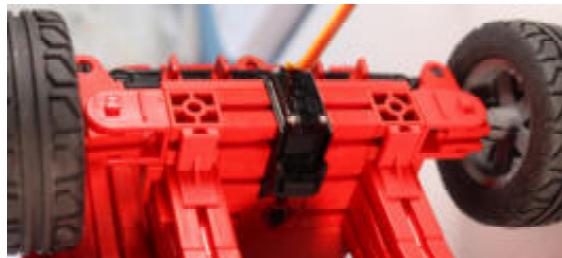


Abb. 15: Servo-Lenkung

Die Lenkung gestaltete sich anfangs etwas schwierig, da der Servo nicht so eingebaut werden konnte, wie es in den fischertechnik-Bauanleitungen beschrieben ist. Grund dafür ist die Hinterradlenkung (Abb. 15).

Hier kommt eine der Schwächen der fischertechnik-Fernsteuerung zum Tragen: Bei einer Fernsteuerung für den Modellbaubereich wäre es kein Problem, denn diese verfügen in der Regel über eine Servoumkehrfunktion. Mit der Servoumkehrfunktion kann die Ausschlagrichtung umprogrammiert werden, sodass es keine Rolle spielt, wie der Servo eingebaut wurde. Bei der fischertechnik-Fernsteuerung geht das nicht. Es kann nur durch einen entsprechend veränderten Einbau gelöst werden. Zusätzlich war bei diesem Modell auch eine Anpassung der Servolasche erforderlich. Diese musste in der Breite um 2 mm gekürzt werden, damit ein – dem Stapler – angemessener Lenkeinschlag möglich ist.



Abb. 16: Aufklappbare Heckhaube

Ein weiteres Feature des Staplers ist die aufklappbare Heckhaube (Abb. 16), unter der sich der Akku befindet. Erstmals habe ich hier auch die neuen Rund-Bausteine für das Design verwendet. Ebenfalls zum Design gehört das gläserne Oberteil der Haube. Das lässt sich allerdings nur realisieren, wenn man bereit ist, die gläsernen Bauplatten auf der Längsseite etwas zu kürzen. Neben dem Akku und vor dem Hauptschalter befindet sich noch ein Batteriegehäuse mit Metallkugeln, um mehr Gegengewicht für die zu hebenden Lasten zu haben (Abb. 17).

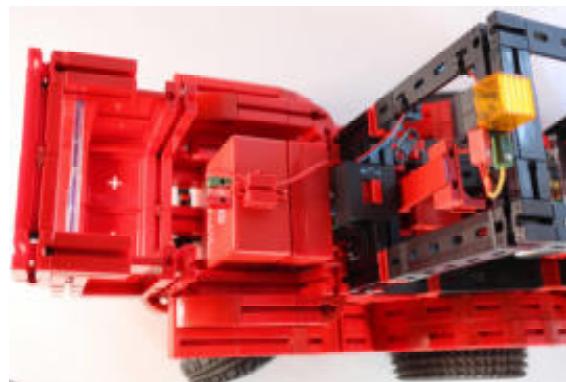


Abb. 17: Batteriegehäuse als Gegengewicht



Abb. 18: Geschlossene Heckhaube



Abb. 19: Führerhaus

Die Stromversorgung wird neben dem Hauptschalter noch über zwei Taster neben dem Fahrersitz gesteuert (Abb. 19). Dabei dient der eine Taster für die Stromversorgung des Empfängers, und der andere Taster schaltet die Lichtfunktionen ein.

## Referenzen

- [1] Harald Steinhaus: [\*Gabelstapler\*](#). ftc-Bildersammlung, 05.02.2009.
- [2] Andreas Tacke: *ft-Spezialteile made by TST (Teil 4)*. [\*ft:pedia 2/2013\*](#), S. 11–12.



Abb. 20: Gesamtansicht des Gabelstaplers

## Modell

# Großprojekt Seilbahn (Teil 4): Fundamentale Arbeiten

Tilo Rust

Diese Serie begleitet das Großprojekt „Kuppelbare Einseilumlaufbahn/Doppelmayr (10-MGD)“ im Fördertechnik-Museum Sinsheim von Anfang bis zur Fertigstellung und Ausstellung auf der BUGA 2023 in Mannheim.

Bei unserem Großprojekt werden wir in vielen Anlagenteilen große Kräfte auffangen müssen. Um gleichzeitig für ausdauernde Stabilität zu sorgen, müssen wir vor unserer Modellbauerarbeit mit Fischer-technik ein Fundament und Traggerüst bauen. Die Planung und Details sollen hier beschrieben werden.

## Kraftmeierei

Wie schon in den bisherigen Beiträgen dieser Reihe beschrieben werden wir eine Modellanlage bauen, die 30 m überspannt. Die Stationen werden mit über 3 m Länge nicht nur Aufgrund der Größe imposant werden. Die Stützen in der Strecke auf bis zu 4,60 m Höhe tragen nicht nur das Gewicht von Seil und Gondeln, sondern müssen stabil und schwingungsarm stehen.

Zur Spannung des Seils ist eine Zugkraft von 500 N (entspricht ca. 50 kg) vorgesehen. Wie schon in Teil 3 [1] und in unseren Videos auf YouTube [3] beschrieben, wird diese mit Hilfe eines Gewichtes im Unterbau der Talstation „Spinelli“ aufgebracht.

Da hier wenig Platz zur Verfügung steht, werden wir mit einem Flaschenzug 100 kg als Spanngewicht einbauen müssen. Damit wäre aber auch schon eine Abschätzung möglich: 100 kg Spanngewicht, geschätzte 100 kg Modellgewicht und eine horizontale Komponente von 50 kg (das Seil zieht an

der Station) verlangen, dass wir das Modell auf ein stabiles Fundament stellen.

Gleichzeitig muss das Gestell durch Doppelschwingtüren passen. Wir haben daher die Modulgröße auf max. 180 cm Höhe und 140 cm Breite festgelegt. Bei einer Modullänge von max. 250 cm (= maximale Verladebreite auf LKW) müssen beide Stationen in je zwei Module aufgeteilt werden.

Klar ist auch, dass diese Module mit Hubfahrzeugen („Ameise“ und Stapler) kompatibel sein müssen, jedoch auch ohne diese auf Rollen durch einen Raum geschoben werden – was wiederum der Standsicherheit widerspricht.

Dazu kommt zum Transport eine nötige Verschalung, die schnell, einfach, aber sicher angebracht beim Unterbau von vornherein berücksichtigt werden muss.

Auch die Stützen müssen auf einem sicheren Fundament stehen und darüber hinaus für den Betrieb gekippt werden! Der Grund: Sie müssen exakt in der Winkelhalbierenden des Seils stehen, da sie sonst abknicken würden (auch im Original stehen die Stützen so). Das bedeutet, dass die Stützen ein schweres Fundament benötigen sowie einen verlässlichen Kippmechanismus, der zudem auf unterschiedliche Winkel einstellbar ist (um unterschiedlichen

räumlichen Gegebenheiten bei Ausstellungen zu genügen). Alles muss zusammen kompakt und sicher verfrachtet werden.

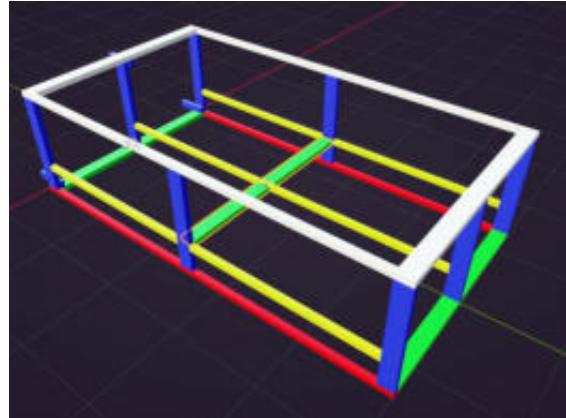
## Knochengerüst

Dass wir die auftretenden Kräfte der Stationen nicht mit fischertechnik abfangen können (oder korrekt gesagt: wir könnten es sicher schon, aber zu einem enormen Material- und Kostenaufwand), war uns von vornherein klar.

Glücklicherweise ist einer unserer Hauptspatoren das Fördertechnikmuseum in Sinsheim [4], welches wiederum beste Kontakte zum Hersteller der Axmann-Industrieprofile pflegt. So fiel die Entscheidung leicht, diese als tragenden Bau der Stationen zu verwenden. (siehe auch das Video auf unserem YouTube-Kanal [3]).

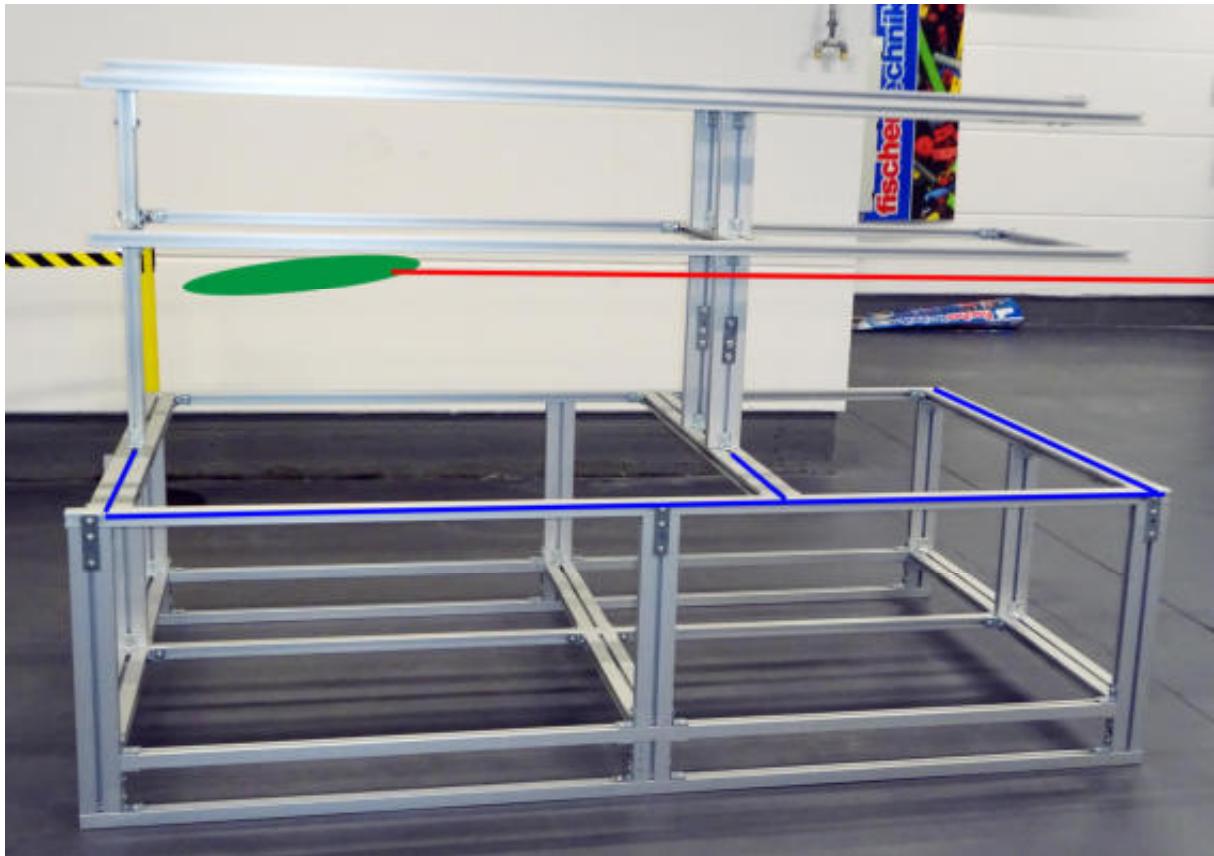
Gleichzeitig wollen wir unter das Modell eine Kastenkonstruktion bauen, die mit

Flurförderfahrzeugen kompatibel ist, Stauraum bietet und das Modell auf eine angenehme Betrachtungshöhe anhebt.



*Abb. 1: Untergestell*

Nach langer Planung und viel Tüfteln, auch am 3D-Modell, haben wir am Bautag im März 2022 endlich die Gestelle für die Talstation „Spinelli“ komplettiert. Die Konstruktion des Untergestells eines Moduls der Talstation erfordert viel Planung, denn



*Abb. 2: Das fertige Gestell für das vordere Modul der Talstation „Spinelli“ (Ansicht von rechts)*

die Materialien sind ein erheblicher Kostenfaktor im Projekt. In den Kästen werden später Holzplatten als Wände eingezogen, sodass ein Stauraum entsteht. In einem Modul hängt zusätzlich das Gestell für das Spanngewicht mit Seilführung. Kompatibilität zu Transportfahrzeugen, manuelle Positionierung des Modells und Einhalten der Maximalgröße sind zu beachten. Dazu soll alles optisch sauber aussehen und viele Jahre haltbar sein. Abb. 2 zeigt das 3D-Planungsmodell des vorderen Moduls von Spinelli. Es folgen das hintere Modul sowie zwei noch deutlich größere und komplexere für die Bergstation „Luisenpark“. Auf den weißen Profilen wird die Stützkonstruktion aufgesetzt.

Abb. 2 zeigt das fertige Gestell. Zum besseren Verständnis ist die Position des Seils in Rot angedeutet (die Strecke verläuft also rechts aus dem Bild heraus) und die Seilscheibe ungefähr dort, wo sich die grüne Ellipse befindet. Blau eingezeichnet ist das Bodenniveau des Modells, darunter der Kasten des Untergestells (in der obigen 3D Zeichnung sind das die weißen Profile). Das hintere Modul schließt sich dann links im Bild an.

In diesen Unterbau kommen nun Seitenwände, Schiebetüren und natürlich ein Boden, sowie die Aufhängung des Spanngewichts.

## Gewichtheben

Die Station muss zum Betrieb horizontal ausgerichtet sein und soll ohne zu wackeln sicher stehen. Das ist bei den Böden in unterschiedlichen Ausstellungsräumen nicht immer einfach. Deswegen haben wir für die Module passende Niederschraubfüße konstruiert, die bereits von unserem Sponsor Doppelmayr/Garaventa gefertigt werden (Abb. 3). Um die Niederschraubfüße sicher innerhalb der Transportmaße unterzubringen, sie dennoch gut zugänglich anzuordnen und die auftretenden Kräfte

sauber aufzunehmen, müssen diese präzise konstruiert werden.

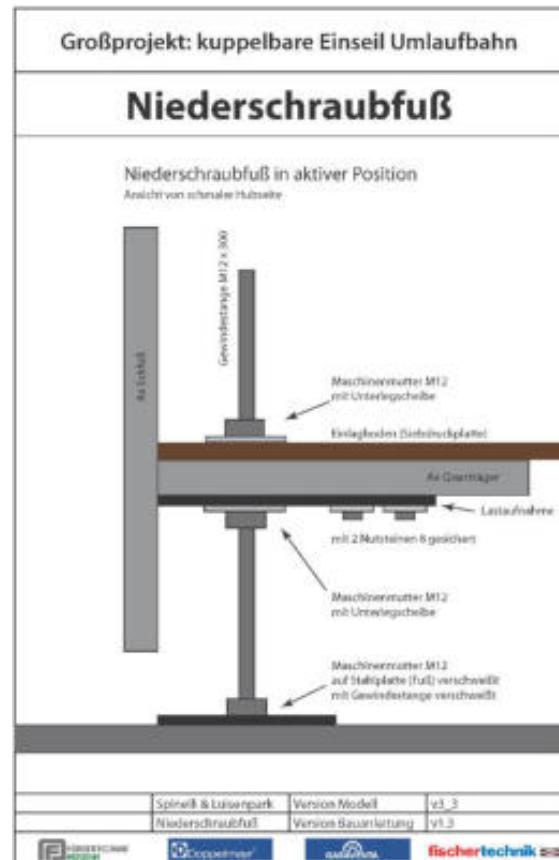


Abb. 3: Niederschraubfüße

## Passgenau

Ebenso bereits in Fertigung befindlich sind sogenannte Zentrierzapfen.

Da die Stationen in zwei Module aufgeteilt sind, müssen diese zum Betrieb genau zusammengesetzt werden. Hierbei reicht es nicht aus, dass die Module eng beieinander stehen. Sie müssen auf wenige zehntel Millimeter ausgerichtet und kraftschlüssig miteinander verbunden werden. Die ungefähre Ausrichtung kann mit den Niederschraubfüßen eingestellt werden. Eine Verbindung ist aber nur mit Hilfe von Passstücken möglich. Diese Zapfen werden zum Transport entfernt, um nicht über die Transportsicherung herauszuragen, dann aber vor Ort in eine Halterung eingesetzt. Sobald die Module mit Hilfe einer Gewindestange zueinander gezogen werden,

sorgen die konischen Zapfen (Durchmesser 60 mm) für die Passgenauigkeit (Abb. 4).

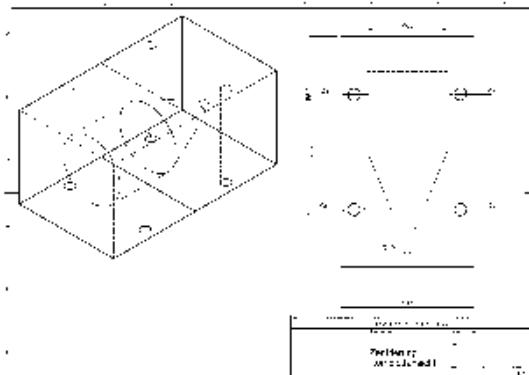


Abb. 4: Passstücke

Je vier solcher Zentrierzapfen werden pro Station für die Ausrichtung der Untergestelle sorgen. Die Zentrierung der Fahrbahnen erfolgt zusätzlich über weitere kleinere Zapfen.

## Aufhänger

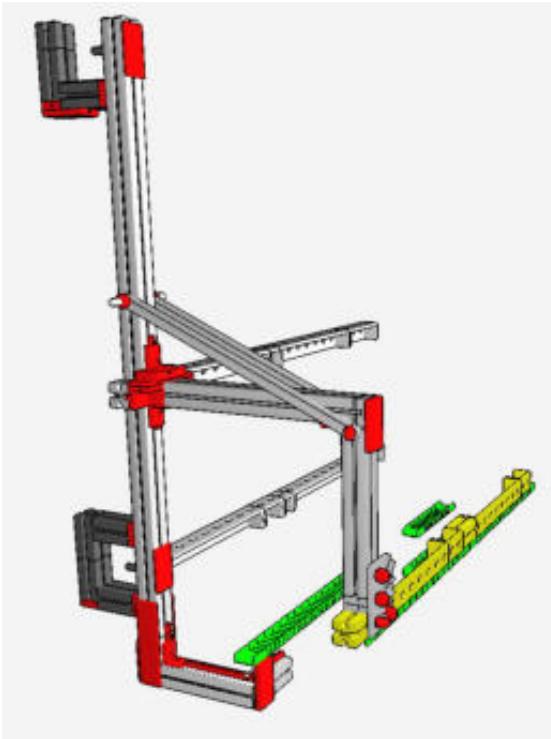


Abb. 5: C-Stück im fischertechnik-Designer

An die Stützgestelle aus Axmann-Profilen werden Halterungen (Fachbegriff „C-Stücke“) für die Schienen befestigt, auf denen die Gondeln durch die Stationen

fahren. Ebenso werden an den C-Stücken die Reifenförderer angebracht. Das sind technisch aufwändige Radaufhängungen, welche die Gondeln von oben antreiben, indem die Reifen auf einer Lauffläche der Klemme abrollen. Bei 30 Sektoren pro Station werden insgesamt 180 Reifen mit Ketten vollkommen synchron zum Antriebsseil laufen.

Daher sind diese C-Stücke ebenfalls stabil und in Serie zu produzieren. Um vielleicht einmal ein Gefühl für unser Projekt zu bekommen: Alleine für die Talstation beträgt der Materialwert der 30 C-Stücke über 1.200,00 € – und das bei guten Konditionen direkt ab fischertechnik-Werk.



Abb. 6: Fertiges C-Stück

Die Konstruktionszeichnung aus dem fischertechnik-Designer in Abb. 5 zeigt die Planung des C-Stücks. An den schwarzen Klammern wird dies an den Axmann-

Profilen angebracht. Die Fahrbahn der Gondeln wird unter den grün markierten Flexschienen mit Hilfe eines langen Aluprofils montiert. Gelb markiert ist die Niederhalteschiene, welche die Gondel gegen Kippen stabilisiert.

Fertiggestellt sehen die C-Stücke so aus wie in Abb. 6. Je Station werden 30 Stück benötigt. Die vielen Kleinteile sind dazu da, die C-Stücke so zu verschränken und verriegeln, dass sie trotz starker Vibrationen und Erschütterungen im Dauerbetrieb über viele Stunden pro Tag nicht auseinanderfallen können.

Zur Gesamtansicht der Talstation „Spinelli“ in Abb. 7: Die Position der Seilscheibe auf dem Spannwagen ist sichtbar. Dieser wird mit 500 N nach hinten gezogen, um das Seil zu spannen, und mit einem Motor angetrieben (ca. 0,25 Umdrehungen pro Sekunde). Auch die C-Stücke sind zu erkennen. Die massiven farbigen Elemente repräsentieren die Axmann-Profile, welche später optisch verkleidet werden und die Kräfte aufnehmen. Die untere Ebene (dunkelblau) symbolisiert das Bodenniveau und wird auf den Unterbau (Abb. 2) aufgesetzt. Im Foto des fertigen Gestells sind diese Profile dunkelblau markiert.

## Unter-Stützung

Am Bautag im März 2022 konnten wir im Museum auch die Fundamente der Stützen fertigen. Die vier Stützen, welche das Seil über die 30 m Distanz auf bis zu 4,60 m Höhe tragen müssen, sind baugleich. Sie bestehen aus einer Europalette, welche die Transportmaße vorgibt und die Kompatibilität zu Hubfahrzeugen herstellt. Darauf sind Grundplatten verschraubt und schließlich mit einem Kippmechanismus ein Kasten, der ein Betonfundament enthält.

Das Fundament ist sowohl Gegengewicht (ca. 80 kg) der gekippten Stützen als auch Montagepunkt der eingegossenen Kunststoffrohre (Durchmesser 110 mm). Im Innern verlaufen Stahlseile, welche die Rohre auf Zugspannung halten, um eine Kraftübertragung der Kopflast (ca. 10 kg) sicherzustellen.

Auch bei dieser Konstruktion berücksichtigten wir Haltbarkeit und sauberes Aussehen sowie praktische Bedienung und absolute Sicherheit dieses kritischen Elements.

Abb. 8 zeigt die Konstruktion. Basierend auf einer Europalette steht ein Holzkasten, der mit Beton vergossen ist. Dieser kann gekippt und verriegelt werden. Zum

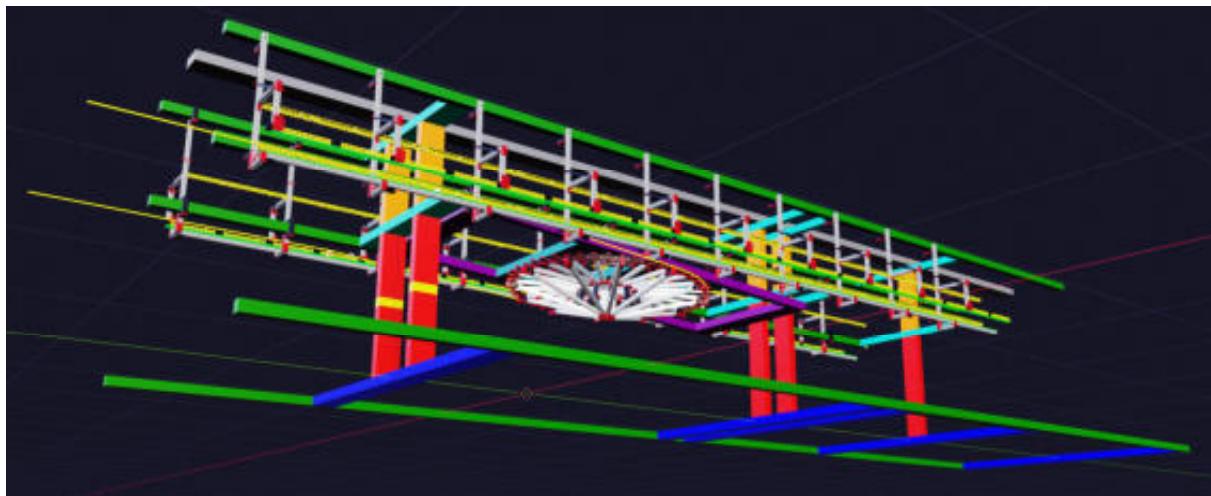


Abb. 7: Gesamtansicht der Talstation „Spinelli“ von unten vorne

Transport wird die Stütze auseinandergenommen und in der Box verstaut. Der restliche Platz kann zum Transport von Gondeln u. a. genutzt werden. Die Rohre sind azentrisch angeordnet, um den Schwerpunkt besser innerhalb der Palette zu halten und zusätzlichen Stauraum zu schaffen.

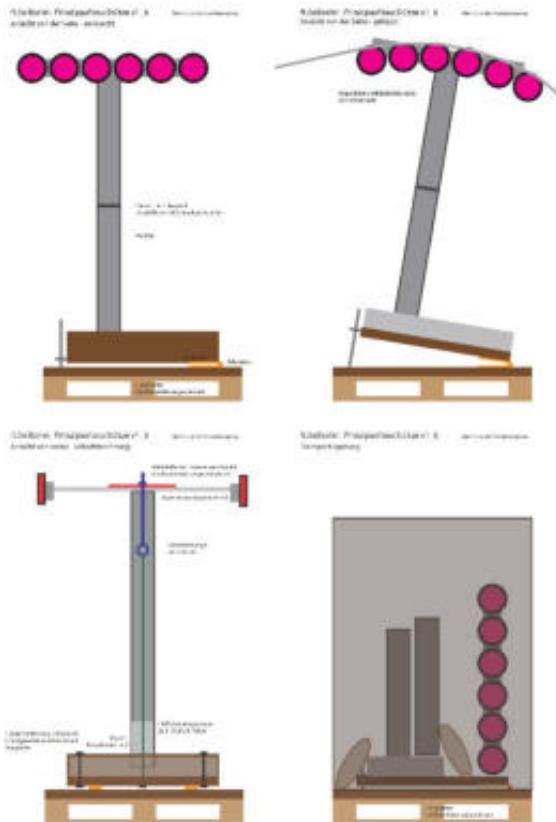


Abb. 8: Konstruktionsschema der Stütze

In Abb. 8 sieht man zwei Fundamente der Stützen auf den Europaletten. In die Holzkästen wurde Beton vergossen und dieser vierfach verspannt. Die Abwasserrohre stellen die unteren Abschnitte der Stützen dar und enthalten die Ösen für die Stahlseile zum Verspannen der Traversen am Kopf der Stütze.

## Unterstützen?

Auch wir brauchen Unterstützung und Mithelferinnen und -helfer in unserem Projekt: Tüftlerinnen und Tüftler, Konstruk-

teurinnen und Konstrukteure sowie helfende Hände beim Aufbau im Museum und später im Betrieb auf der BUGA23. Wer also mit dabei sein möchte bei diesem gigantischen Projekt, darf sich direkt melden [5].



Abb. 9: Fast fertig: Zwei von vier Fundamenten

## Quellen

- [1] Tilo Rust: *Großprojekt Seilbahn (Teil 3): Die Stationen.* [ft:pedia 3/2021](#), S. 28–37.
- [2] [Doppelmayr/Garaventa](#)
- [3] [YouTube-Kanal](#) des Projekts
- [4] [Fördertechnik-Museum Sinsheim](#)
- [5] Kontakt zur Projektgruppe „Seilbahn“: Projektleiter Tilo Rust, Schifferstadt. E-Mail: [ft.seilbahn@gmail.com](mailto:ft.seilbahn@gmail.com)



Abb. 10: Eifrigie Hände (im Bild Gertraud Rust und Thomas Magin) bei der Endmontage der C-Stücke. Im Hintergrund James D. Behra und Alexander Salameh beim Bau des Untergestells der Talstation. Rechts hinter der Absperrung ist die Stützkonstruktion bereits fertig.



Abb. 11: Teamwork: Jan Rust und Sven Petry arbeiten Hand in Hand bei der Montage der Stützenfundamente. Am Bautag im Fördertechnikmuseum sind viele Arbeiten gleichzeitig zu erledigen. Alle Teams legen ein unglaubliches Arbeitstempo vor, sodass die Seilbahn jetzt sichtbare Formen annimmt. Am Abend sind alle fertig – die Fundamente und auch die Helfer.

Modell

## fischertechnik-Roboter mit Arduino (Teil 5): Elegante Leuchtsteine und gepimpte LKW

Lutz Münche

*Die Roboter aus dem Buch „fischertechnik-Roboter mit Arduino“ [1] haben zahlreiche Leser zu weiteren Modellvarianten angeregt. In loser Folge werden in dieser Serie einige dieser Modelle vorgestellt. Diesmal wird der Flitzer gepimpt – und die Steuerung in LKW eingebaut...*

Wer das Buch von Dirk Fox und Thomas Püttmann [1] gelesen und den Flitzer nachgebaut hat, der bekommt neue Ideen. So zum Beispiel die, fischertechnik-LKW-Modelle mit dem Arduino zu steuern. Nur die Leuchtsteine mit Stecker passen irgendwie nicht ins Bild: Die aus den schwarzen Leuchtsteinen herausragenden roten und grünen Stecker wirken klobig. Das wollte ich ansprechender gestalten. Bei [fischergedanken](#) fand ich leere Leuchtsteine in rot ([132226](#)) – die waren 2008 beim Feuerwehrauto ([500884](#)) für die Blaulichter verbaut. Das ist es, dachte ich: Leuchtsteine selbst löten, ohne Stecker<sup>1</sup>.

Ab in den Keller also, Pucksäge und Feile ausgepackt und losgelegt. Die Lochrasterplatten gesägt und mit der Feile auf Maß gebracht (Abb. 1, 2). Wie vor 43 Jahren: U-Stahl feilen – in meiner Ausbildung als Energieanlagenelektroniker (was für ein Wort!).

Die Oberfläche mit einem edding 8750 weiß grundiert und nach dem Trocknen mit einem roten Marker bemalt (Abb. 3).

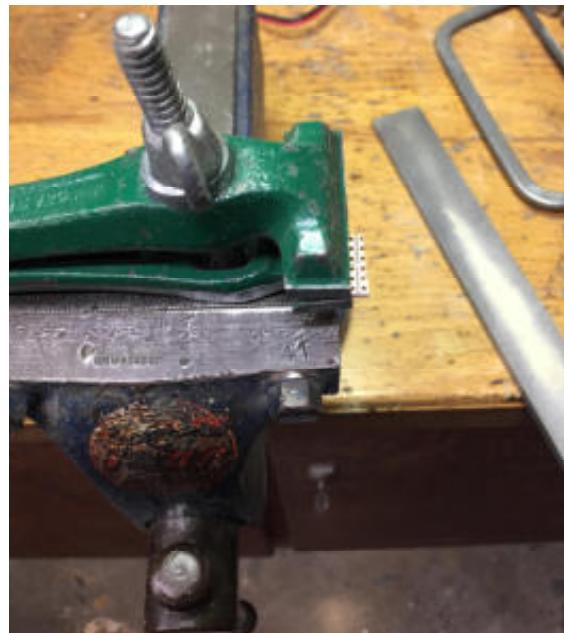


Abb. 1: Die Werkbank

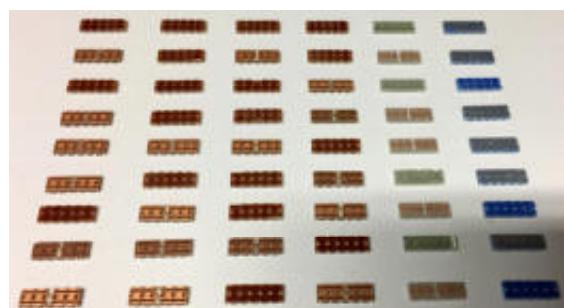


Abb. 2: Zugeschnittene Lochrasterplatten

<sup>1</sup> Thema neue Stecker: Die sind noch nicht ausgereift; dazu hatte ich bereits einen E-Mail-Austausch mit fischertechnik.

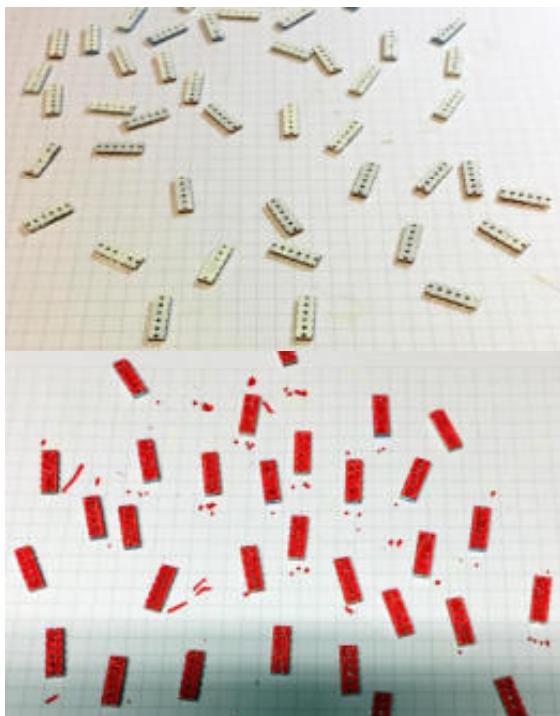


Abb. 3: Weisse Grundierung, rote Lackierung

Von einem Magnet-Reedkontakt mit 6 m Leitung habe ich die Leitung gekürzt, so dass man den Kontakt mit 20 cm Leitungslänge noch verbauen kann. Die Leuchtdioden und Widerstände eingelötet (Abb. 4) – dann kam der Hammer: Leitungen mal zu kurz, mal zu lang, dann genau 15 mm, fischertechnik-Maß (Abb. 5). Den Flitzer habe ich zuerst umgebaut: Der sieht gut aus ohne Stecker, vorn und hinten (Abb. 6).

Im LKW aus Advanced Trucks 2017 ([540582](#)) habe ich den Arduino im Führerhaus verbaut; mir war wichtig, das Originalmodell so wenig wie möglich in seiner Form zu verändern (Abb. 7).

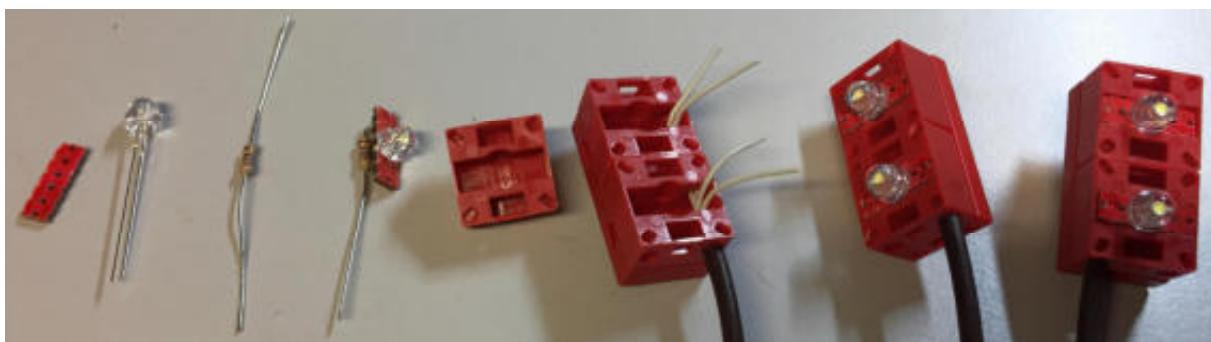


Abb. 4: LED, Vorwiderstand und Anschlusskabel in rotem Leuchtstein

Beim Super Trucks Hängerzug 2002 ([77790](#)) wird der Arduino unter der Plane versteckt – und bald verdrahtet (Abb. 8).

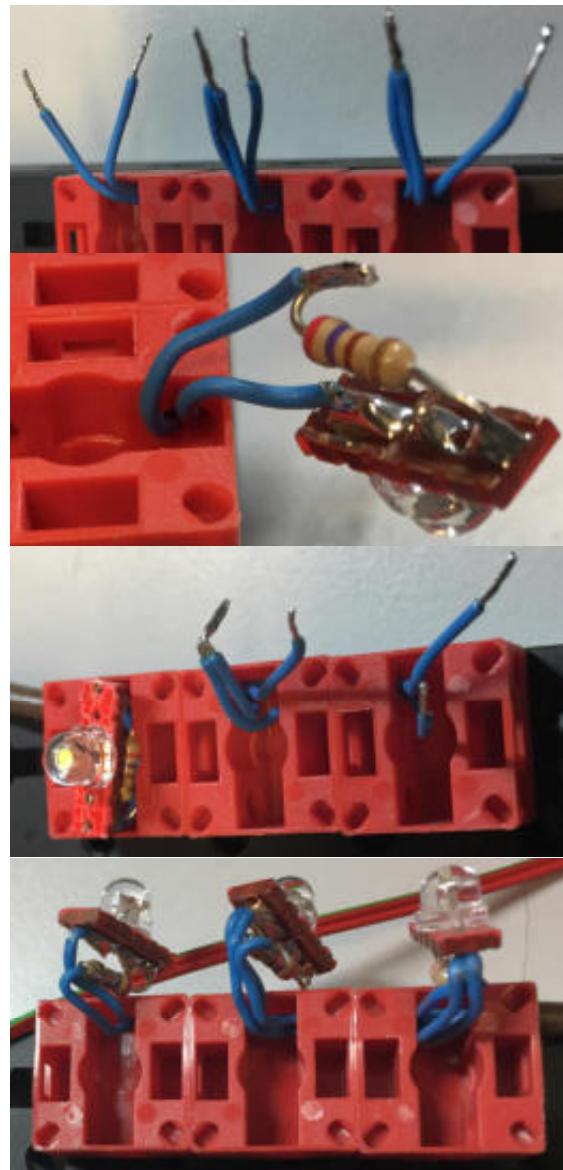


Abb. 5: Einlöten der LED mit Vorwiderstand



Abb. 6: Original-Flitzer (links) und steckerfreie Version (rechts) [1]



Abb. 7: „Gepimpter“ LKW aus Advanced Trucks, 2017 ([540582](#))

Die Modelle können mit dem Handy [2] oder einem Joystick [3] gesteuert werden. Alle Anschlussbilder und Programme sind im Buch von Dirk Fox und Thomas Püttmann zu finden [1].

Zu meiner Person: Ich bin Baujahr 1962, (alter Sack) fischertechnik-Fan und baue gerne. Mein größtes Modell ist der Nachbau der Lernfabrik 4.0 (9V). Lese ft:pedia und bin in Forum lesend unterwegs.

## Referenzen

- [1] Dirk Fox, Thomas Püttmann: *fischertechnik-Roboter mit Arduino*. dpunkt-Verlag, Heidelberg, 2020.
- [2] Dirk Fox: *fischertechnik-Roboter mit Arduino (Teil 1): Smartphone-Steuerung über BLE*. [ft:pedia 3/2020](#), S. 93–100.
- [3] Arnoud van Delden: *fischertechnik-Roboter mit Arduino (Teil 3): Steuerung mit dem Joystick Shield*. [ft:pedia 1/2021](#), S. 119–127.



Abb. 8: „Gepimpter“ Super Trucks Hängerzug 2002 ([77790](#))

Modell

## „Dirty Dishes“ Pinball machine

Rubem Pechansky

*2021 marked my 50-year anniversary with fischertechnik. It was about time I did my own pinball machine. It eventually turned out to be my most challenging hobby project ever.*



Fig. 1: Front view of the “Dirty Dishes” pinball table. The solenoids that activate the flippers are below the pink and blue covers.

### Introduction

This venture started as a shameless attempt to recreate the pinball table from the critically acclaimed fischertechnik TX ElectroPneumatic kit [1] to the best of my knowledge using spare parts, since I don't own the original kit. One of the main problems was that I didn't have the original pneumatic cylinders with springs ([133027](#)). Initially I retrofitted some old cylinders with springs taken from various sources, but the results were not great. The hacked cylinders were not strong enough for the flippers to propel the ball all the way up, and the result felt quite different from a real

pinball machine. Classic real world pinball tables do not even have an air compressor, and, albeit fascinating, pneumatics would imply several important limitations regarding design, mechanics and even gameplay. So, I decided to give up on pneumatics altogether [2] and the project evolved in a very different direction. After many months designing, building and coding, plus dozens of failed attempts, lots of tests, meters and meters of wire, solder and double-sided tape — plus a couple of smoked modules — I can say my pinball machine is finished. The end result doesn't



Fig. 2: Overall view of the finished table.

look much like the one from the fischertechnik kit anymore, as can be seen in Fig. 2.

This was a very complex undertaking so there is quite a lot to talk about. There is also some extra information in the respective forum thread [3]. Please feel free to ask any questions there, and don't forget to share your own pinball project if you ever want to create one.

## Construction

### Framework

The playfield is mounted on two baseplates 500. In order to accommodate all the electronics and for a sturdier assembly, however, I had to install a third baseplate 500 to the underside using screws and nuts with plastic washers as shown in Fig. 3.



Fig. 3: M3 screws, nuts and washers are used to secure the bottom baseplate.

In the process I confirmed my suspicions that, at  $258 \times 186$  mm, the most popular fischertechnik baseplate is simply not in the grid. This is usually not a problem for most models, but in this particular case I had to make precise size measurements and numerous adjustments because I didn't want any gaps between the two plates. In addition, there is no provision to attach anything on their underside, so the third board was essential (Fig. 3). There is also no standard way to attach parts to the sides

of the baseplates, so I had to figure out ways to secure the two U-girders, several BS 30 blocks, statics rails and more.

I also used many extra blocks and three aluminum profiles 210 below the play area for reinforcement. The model is moderately heavy at approximately 2.6 kg (without power supply), but this is actually a good thing because it contributes to the overall stability of the construction. I also used soft silicone feet to ensure the pinball table does not move during gameplay.



*Fig. 4: Silicone feet are used to ensure the table is kept steady during the game.*

## Flippers

Right from the start it became obvious that I needed a powerful replacement for the pneumatic actuators. The flipper action is solenoid-based as opposed to pneumatics. This approach is similar to a real pinball table, and the resulting movements are quick and precise. I first tried to put the solenoids under the table using spindles to transmit the rotation to the flippers, but the whole thing was wobbly and cumbersome. So I gave up on it and decided to put the solenoids right above the flippers (Fig. 5). This paid off quite nicely: The resulting assembly is very sturdy and still much more compact than with pneumatics. The solenoids were easily concealed below inkjet-printed covers (see Fig. 1).

The flippers themselves were made smaller than in the ElectroPneumatic kit, and not shaped like a rectangle. For each flipper I used two S-couplings [38260](#) with a hard rubber pad sandwiched between them and

shaped in such a way that the tip of the flipper becomes curved. This design gives much better control over the ball direction. Also, maximum and minimum flipper angles are kept similar to the real ones.



*Fig. 5: A solenoid as a flipper driver.*

Despite an audible PWM buzz when the flippers are held up, the machine is much quieter overall — there is no compressor.

## Plunger

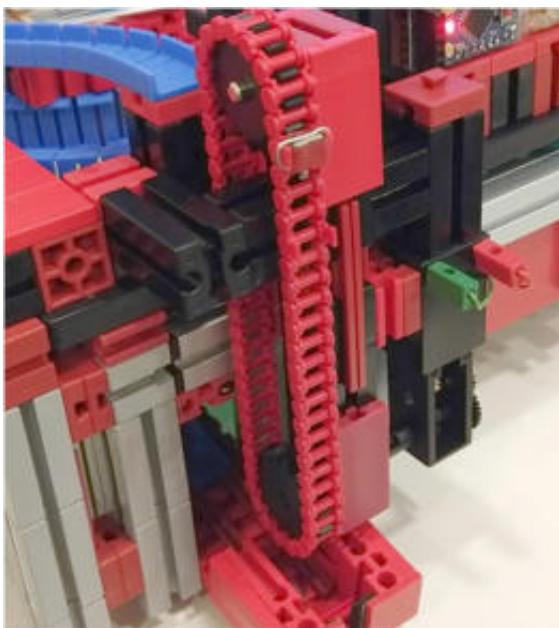
In the fischertechnik model, the plunger is located above the playfield and the balls fall in the upper baseplate. This is not ideal, so I used two U-girders [32854](#) to extend the playfield area to the right. This arrangement allows the ball to be launched on the same plane as the playfield, which gives much smoother action and is similar to the real thing. I was lucky to find good springs in my hardware stock and used plastic shoulder washers as diameter adapters.



*Fig. 6: Detail of plunger spring with washer.*

### **Ball return**

For the ball return mechanism, I first tried an approach similar to Dirk Wölffel's "Pirates of the Caribbean" table [4], which is a brilliant solution. Its main drawback in my case is that the added width to the right would conflict with the U-girders used to accommodate the plunger, so in the end I built a ramp with two flex rails 180 and mounted it underneath the table so it goes all the way from the ball drain to the rear (see Fig. 14), plus a chain elevator with a small magnet that brings the ball back to the playfield (Fig. 7).



*Fig. 7: The chain elevator used to return the ball to the playfield.*

The principle is simple enough, but making it work reliably was tricky and took me several attempts.

## **Design**

### **Tools**

Such a special project needs powerful tools. Figma [6] is a great application I am very familiar with, so I used it as my main design tool for most of the project (Fig. 8).

The original file consists of many vector layers that can be turned on and off at will.

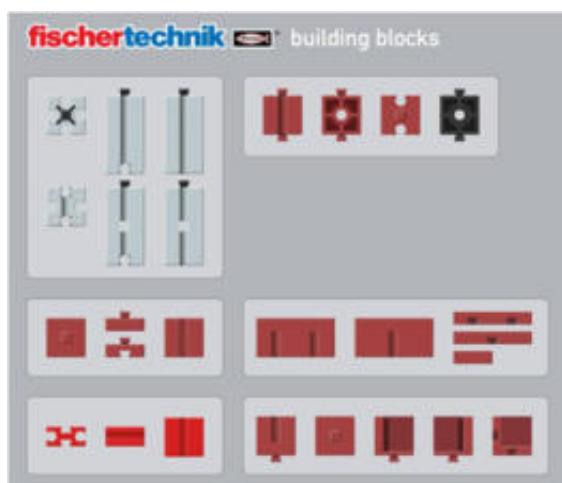


*Fig. 8: The project as it appears in Figma.*

For the mechanical project I used a vector library of fischertechnik parts that I've been working on for a while (Fig. 9).

### **Playfield**

In one of the most noticeable changes from the fischertechnik ElectroPneumatic pinball table, there are outlanes beside each flipper, so a player can lose the ball to the sides in addition to the center as in a real pinball table.



*Fig. 9: A page from my parts library.*

There are now four upper lane “rollovers”: the rightmost one is the skill shot lane. (The quotes are intentional: the sensors are actually located above the upper lanes, not below them. See Fig. 17.)

A target at the upper right includes a gray fischertechnik electromagnet [31324](#) which is used to hold the ball for a few seconds in certain conditions during gameplay.

There is a spinner target below the hold magnet. It consists of a red plate that revolves around a U-shaped steel wire. The spinner assembly is positioned diagonally on the playfield thanks to two old-style blocks 15 with a red peg.

A servo-driven door opens when there is a ball to be launched and closes as soon as the ball is past it. The closed door prevents the ball to go back to the plunger and also helps to direct it to the skill shot lane.

The upper left “orbit” sensor is kept is very similar to the one in the original fischertechnik table.

## Graphics

I've designed and illustrated a novel theme called “Dirty Dishes”. The analogy here is that the player is supposed to use the two faucet handles (also known as flippers...) to “wash” a pile of dishes on a kitchen sink (i.e., get points). I used some free images [8] and vector illustrations [9] as a reference, but ultimately everything was completely redrawn and/or heavily modified to go along with the overall look and color scheme (see Fig. 10).

The graphics feature several kitchen utensils, a greasy pizza slice on the upper left target, a kitchen roll (the spinner), a glass pitcher that can be “broken” when the spinner makes enough turns — I know, it doesn't make sense, but the sound effect is cool —, a rack where the player can put some plates to “dry” (i.e. the electromagnet holds the ball for a few seconds), and so on.

Some graphics are printed separately as shown in Fig. 11.



Fig. 10: The playfield illustration.

Apart from allowing the creation of a colorful and amusing theme, the printed playfield surface also has the very important function of smoothing out the table surface. Baseplates are inherently quite bumpy because of their many slots. This makes the ball slow down, change direction unexpectedly, and even halt to a stop, all of which can be quite annoying in the original fischertechnik model. A simple solution that worked quite well in the initial tests was to cut out just the slots needed to attach blocks and wiring from a plastic folder intended for office use. The measure is just right,  $240 \times 350$  mm. For the final model I printed the graphics in the back of two A4 transparent overhead projector sheets. This guarantees very bright colors, but it needs an opaque white layer below it. I used matte coated paper, which is very bright but got some wrinkles, most likely because of

humidity and temperature variations. I believe the ideal solution would be to have the graphics rendered on high quality white plastic film at an express print shop, but I decided to leave it as is. Fortunately, the wrinkles are almost completely gone.

The six top covers were also printed in the same way (Fig. 11). The bigger ones were glued to cardboard using double-sided tape and then cut together. The smaller ones — where light is supposed to shine through — were composed of several sheets of translucent paper. The covers were then fastened to fischertechnik cladding panels with double-sided tape. This arrangement allows easy insertion and removal.



*Fig. 11: Top cover graphics.*

Figma was also used for all graphics, which were subsequently exported to Affinity Designer [7] for printing.

## Gameplay

My main project goals were to achieve good playability and to simulate some classic pinball table features.

Several rules were copied from actual pinball machine tables, while others were adapted or created to somewhat overcome the limitations of a small table like this.

The most important gameplay features that were implemented are highlighted in the following paragraphs.

**Game start.** To begin, the player just presses one of the flipper buttons. Like in most pinball machines, a game consists of three balls and there is no time limit for each ball, so you may play as long as you like.

**Free replay.** If a ball is drained within 3 seconds or less, there is a free replay. Any points awarded are lost and the outlane LEDs are lit. The player then gets the ball back and tries again. This can happen up to two times per ball.

**Skill shot.** The red Skill Shot LED on the rightmost upper lane (the one with a corkscrew) flashes as each ball is launched. 1,500 points are awarded if the ball passes through it within two seconds after it crosses the launch lane door.

**Multiplier.** If the three leftmost upper lanes (“rollovers”) are triggered, the multiplier gets incremented, so scores are multiplied by 2, 3, etc. up to 8. The multiplier values are retained between balls. (My family asked me to do this. They are right — the game should be fun first and foremost.)

**Rotating upper lanes.** The upper lanes that are currently lit are rotated with each full revolution of the spinner — the resulting effect with the LEDs is nice to look at and allows a skilled player to increment the multiplier quickly.

**Spinner.** Points are scored every time the spinner rotates. Higher points are awarded when the plate is hit with sufficient force so that it spins a minimum number of times in a row.

**Hold target.** Another way to get high scores is to hit the Hold target. This makes the respective LED start flashing. Hitting it two more times lights the LED and activates the electromagnet for five seconds, which may (or not) hold the ball in position. In this

case, big points are awarded before the ball is released.

**“Greasy stuff” mode.** If the player gets 5,000 points or more during each ball, the “pizza” LED starts flashing, and the end-of-ball bonus is increased each time ball passes on the left “orbit” target — this is signaled to the player via a special sound.

As for the Extra Ball feature, it was not implemented despite it being conspicuously printed between the flippers in the playfield.

```
// -----
// Dirty Dishes pinball: Game constants
// Ruben Pechansky 2021

// -----
// General

#define BALLS_PER_GAME      3
#define MAX_FREE_REPLAYS    2
#define MAX_MULTIPLIER       8
#define HOLD_THRESHOLD       3           // No. of stop sensor hits to activate hold
#define BREAK_STREAK         14          // Spinner streak for higher scores

// Points awarded

#define LEFT_ORBIT_POINTS   50
#define HOLD_POINTS          1000
#define HOLD_ACTIVE_POINTS  1250
#define BALL_LOST_POINTS    250
#define OUTLANE_POINTS       800
#define SPINNER_POINTS      25
#define SPINNER_BREAK_POINTS 75
#define ROLLOVER_POINTS     50
#define SKILL_SHOT_POINTS   1500

#define GREASY_SCORE        5000
#define GREASY_BONUS         350

// Time constants

#define TABLE_START_DELAY    1000
#define BALL_SAVER_TIME      3000
#define SKILL_SHOT_TIME      2000
#define BALL_LOST_TIMEOUT    1700 // Time to reach the rear side
#define BALL_NEAR_HOME_TIME  250  // Time to place the ball into playfield
#define HOLD_TIME             5000
#define HOLD_COUNTER_TIME    800
#define RELEASE_TIME          500 // Must be enough to let the ball go
#define MULTIPLIER_RESET_TIME 1500
#define MULTIPLIER_WAIT_TIME  300
#define NORMAL_ONESHOT        800
#define NORMAL_FLASH_LEDS    200
#define LEDS_ANIMATION_TIME  800
#define MSG_END_GAME_TIME    1500
#define MSG_END_SCORE_TIME   1500
#define MSG_END_FLASH_TIME   250
#define DEFAULT_DISPLAY_TIME 500
#define LONG_DISPLAY_TIME    2000
#define SPINNER_STREAK_TIMER 200
```

Fig. 12: All game parameters are freely adjustable.

Neither was the Ball Save function, although the “Ball Save when lit” graphic now refers to the Free Replay.

### Playability

It is possible to use several tricks akin to the ones used in real pinball machines [10]. The many hours spent on tweaking and testing the playfield really paid off here.

For example, as in a real pinball table, the player can actually pass the ball from one flipper to the other by pressing and releasing the flipper button very quickly.

Some difficult targets at the top, like the hold area and the skill shot lane, can be hit by letting the ball slowly roll down on a flipper, then shooting them when it is at the rounded tip. Precise timing is required to achieve this.

Because I placed soft rubber pads in some strategic spots, the resulting rebound action can be quite fast. A player has to be careful not to let the ball drain.

### Software

At the outset I decided not to use the TXT because it does not have enough inputs. I’m also much more at home with Arduinos and C++, so that was an obvious choice.

The software bit was quite challenging. It started to grow until the point I had to refactor everything. It now consists of many separate code files. Analyzing it is way beyond the scope of this article, but the source code is published in my public GitHub area under a MIT license and is freely available for those who want it [11].

One interesting point to note is that all game constants can be easily changed in the program if required, so some more tweaks in gameplay are never completely out of question (Fig. 12).

## Electronics

The fischertechnik controllers have limited I/O capabilities and were not considered for this project. Instead, the main logic is processed by two Arduino Nano clones. The main one is wired to the two buttons and the several sensors. It also controls the solenoids and the electromagnet. Since it did not have enough ports, I extended it using a “child” Nano connected to the main one via I<sup>2</sup>C. This second Arduino deals with LEDs, outputs and the MP3 module and does not take part in the game logic (Fig. 13).

It was clear from start that using fischertechnik plugs would be prohibitive for a compact design, so almost all connections are made using “tinned” wires or jumper cables attached directly to the three breadboards and the other electronic modules (Fig. 11). Using breadboards to mount electronic components mimics the modular nature of fischertechnik parts, so they are a natural match for each other (despite the adhesives on their backs having an irritating tendency to wear off).

A button below the table is wired to the reset pins of the two Nanos via a pair of diodes.

### Power circuit

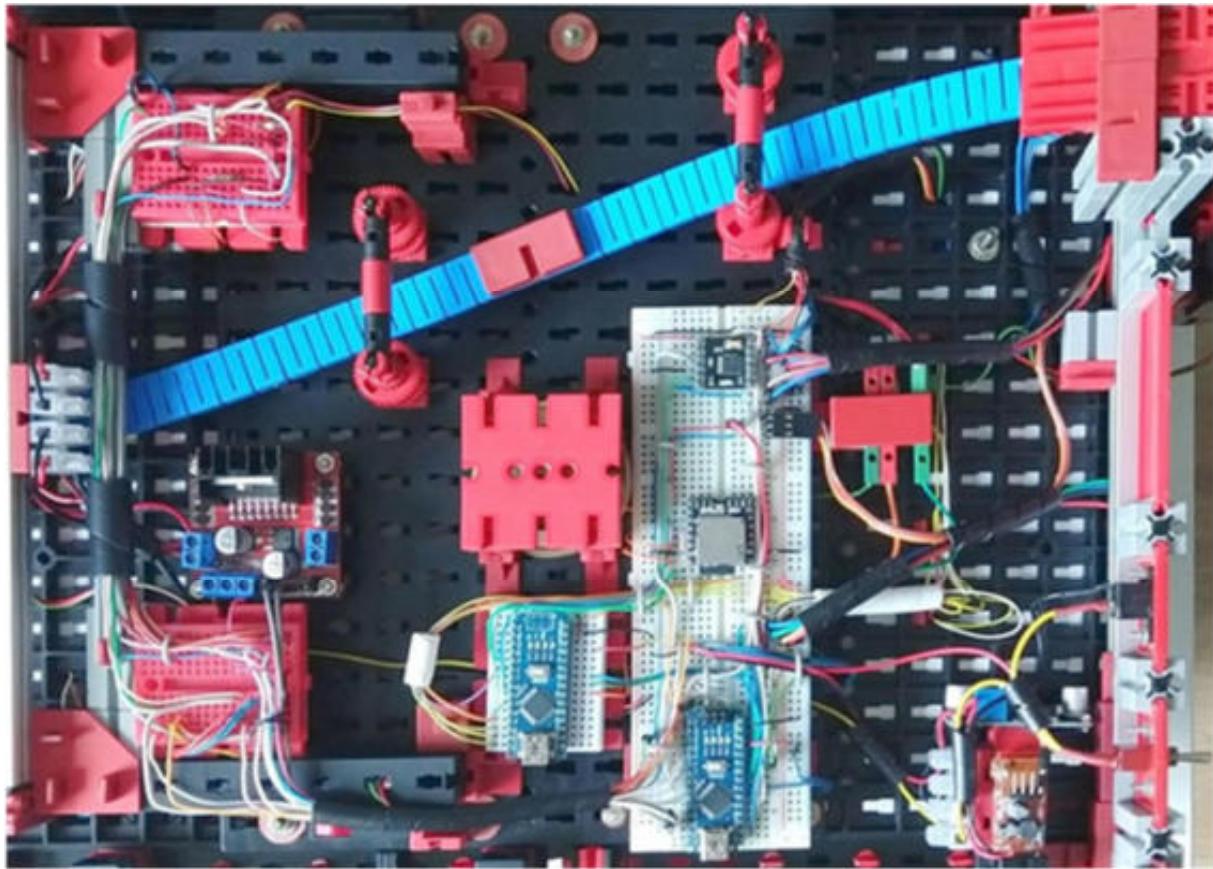
The power supply is taken from an old laptop computer and is rated at 19 V, 3.16 A. This relatively high voltage is necessary for the 6 V solenoids to have enough force to propel the ball.

The solenoid driver is a L298N module, which is admittedly outdated in terms of efficiency but very reliable and short-circuit protected. The solenoids get the full voltage — minus the drop in the L298N — for 50 ms and then the program drops the PWM duty cycle to about 15% (40 of 255) which is sufficient to hold the flipper up while the button is being pressed.

An adjustable step-down regulator provides 9 V for the motor and the fischertechnik electromagnet via a DRV8833. This voltage

"Master" Arduino	I/O	Driver	Pin	D/A	Port
leftButton	I			D	D2
rightButton	I			D	D3 ~
leftOutlaneSensor	I			D	D4
rightOutlaneSensor	I			D	D5 ~
rolloverSkillSensor	I			D	D6 ~
rollover3Sensor	I			D	D7
rollover2Sensor	I			D	D8
rollover1Sensor	I			D	D9 ~
leftFlipper	O	L298N (19V)	IN2	-	D10 ~
rightFlipper	O	L298N (19V)	IN1	-	D11 ~
ballLostSensor	I			D	D12
stopMagnet	O	DRV8833 (9V)	In4	-	D13
feederHomeSensor	I			D	A0
ballNearHomeSensor	I			A	A1
spinnerSensor	I			D	A2
leftOrbitSensor	I			D	A3
SDA	-	green		-	A4
SCL	-	blue		-	A5
holdSensor	I			A	A6
launchSensor	I			A	A7
reset	I			D	Reset

Fig. 13: Pinout map for the main Arduino. A similar one exists for the “child”.



*Fig. 14: Electronics are attached to an extra baseplate on the underside.*

is also used by the fischertechnik LEDs and the luminosity sensor. Another step-down provides 5 V to the Arduinos, servo, display, the LEDs and the L298N module logic.

### **Display and LEDs**

There is a 6-digit display module located above the playfield. I built it using an Arduino Pro Mini clone and two 3-digit seven-segment displays. The Pro Mini stores the character designs and provide basic functions like flashing and scrolling. These designs were adapted from various sources [12]. This module is controlled from the main Arduino via I<sup>2</sup>C. Since the displays are seven-segments only, I had to choose the wording carefully to avoid difficult letters like M, W and X although in some cases this was simply not possible, some legibility is kept nonetheless (see Fig. 15).



*Fig. 15: "Mult 3" is hopefully still readable.*

Of course, I could have used a ready-made pixel-based display so any wording would do, but that would be too easy... and the 7-segment displays have a retro look which is really cool.

There are also four LEDs positioned above the upper lanes, two LEDs installed near the outlanes, another one near the upper left sensor and another one near the hold electromagnet. Initially I wanted to use NeoPixels because they seemed to be the optimal solution: they are bright, colorful, programmable, and require only one Arduino pin. However, I found out that they are not reliable as control LEDs. They often

display the wrong color, light up when not asked to, or turn off at wrong times. (A problem with the specific clones I used is not ruled out.) I made several tests but could not figure out a reliable solution, so I ended up replacing them with regular LEDs: I was lucky to still have a few free ports in the child Nano. All LEDs on the playfield are miniature units taken from LED strips; fischertechnik lamp holders are simply too big.

### Sensors

My first approach was to use small reflective sensors placed in the peg holes below the baseplates. This idea was soon discarded because these sensors were very tricky to install and required holes in the playfield sheets, so the sensors are all placed above the baseplates. They perform several functions. Six of them are QRE1113 reflective sensors which are very small and can be fully embedded in the width of a PCB if a square hole is cut in the middle on the board, which is exactly what I did in the outlanes (Fig. 16). Similar sensors are also used on the upper lanes (Fig. 17).

The rotating wire attached to the spinner plate is bent in such a way that it interrupts the IR beam of a small fork sensor, so points are awarded with each complete revolution.

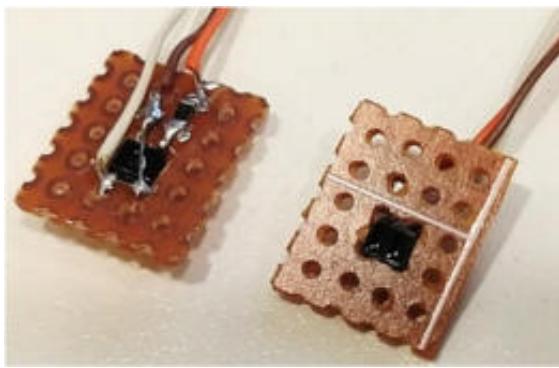


Fig. 16: QRE1113 sensors mounted on small PCBs act as outlane sensors.

A very important sensor detects when the ball has been launched. This particular sensor gave me headaches. The exit from the launch lane is almost 35 mm wide, and

even narrowing it with a rubber post, as shown in Fig. 18, was not enough to detect the ball passing under it with 100% reliability.

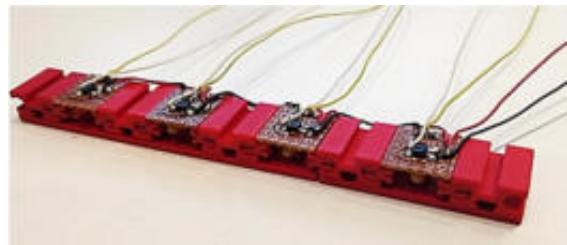


Fig. 17: Upper lane sensors. LEDs can be seen below (changed later to brighter ones).



Fig. 18: A cylindrical rubber post is used to narrow the exit from the launcher.

The solution was twofold. Now, in the case that this sensor does not fulfill its role, if any other sensor in the playfield is activated, the software concludes the ball has been launched anyway and changes state. Then I tested several arrangements (including a test with the fischertechnik IR tracking sensor — it didn't improve much) and even tried to place a small laser beam horizontally, but this would not be possible without making big changes to the design. In the end I simply moved it to be as close as possible to the launch door, and consequently the detection is satisfactory. Sometimes the most obvious solutions are the most effective.

There are also other types of sensors. A small magnet attached to a chain driven by a XS motor is used to place the ball back into the playing field (Fig. 7) and a reed switch tells the program to stop the motor at a "home" position. This arrangement works quite well.

A fork sensor speed module was retrofitted with a reflective sensor and placed near the end of the ball return ramp, below the playfield, to detect the moment the ball gets near its destination. This should then activate the motor. One problem with this type of sensor, however, is that it is too susceptible to variations in ambient light. Its vertical position proved to be very critical as well. Again, the solution was in software: if the sensor does not detect the ball, a simple time counter activates the motor. The sensor can actually be removed.

A lightness sensor [128599](#) (sorry, fischertechnik, that's not a color sensor) is triggered when the ball is near the electromagnet to activate the hold feature.

A fischertechnik phototransistor is used on the upper left target (the same as in the fischertechnik table), and another one is placed below the drain to detect when the ball is lost. These are fairly conventional arrangements for fischertechnik hobbyists. Incidentally, I found out that wiring a fischertechnik phototransistor directly to an Arduino (with the input pullup enabled) allows it to operate easily with an LED, while fischertechnik controllers usually need old-style incandescent light bulbs to perform the same function.

### Sounds

An inexpensive MP3 module with an embedded micro-SD card and loudspeaker is used to generate several high-quality sounds which unfortunately is something that is not possible with the fischertechnik sound module. The module is wired to the “child” Arduino via two digital pins that act as Rx and Tx ports.

Sound effects were edited with Audacity [14] from originals downloaded from Freesound [13]. They are credited in the source code.

### Third-party materials

This pinball table uses a lot of third-party parts. Here are the most relevant:

- 19 V DC, 3.16 A power supply
- Adjustable LM2596 DC-DC stepdown voltage regulator module
- 5 V DC-DC step-down voltage regulator
- 2 × JF-0530B 6 V solenoids
- 2 × Arduino Nano clones
- DIY 6-digit 7-segment display module (Arduino Pro Mini + 2 × 3-digit LCD display)
- DRV8833 motor driver module
- L298N motor driver module
- 8 × QRE1113 IR reflective sensors
- Fork sensor speed module (retrofitted with reflective sensor)
- DFR0299 DFPlayer Mini MP3 Player module with micro-SD card
- 8 × miniature white and red LEDs
- Small rectangular neodymium magnet (5 × 7 × 2 mm)

There were used also 4 breadboards, several resistors, 2 diodes for the reset function, a mini toggle switch, a power jack, solid and flexible wire, double-sided tape, adhesive tape, electrical tape, steel wire for the spinner, two steel springs for the plunger, screws, nuts, nylon and cardboard washers, spiral cable wrap, rubber foam, rubber sheets, plastic sheets, overhead sheets for inkjet printer, white paper, cardboard, cable clips, etc.

Some old fischertechnik parts were important too. Two BS15 with round peg [31059](#) were essential to place the spinner diagonally. I also used several cladding plates and some old-style BS7.5 building blocks (with flat sides).

Surprisingly, just a few part mods were necessary. The front part of a servo holder ([132290](#)) was partially cut to allow full 180° rotation; an old wheel axle [36586](#) had its pin cut off and screwed to a servo arm; and a cladding plate 90 ([31512](#)) was cut to 75 mm to match the width of the display module.

## Conclusions

This project was multidisciplinary: I had to design, illustrate, code, build and test it. Its most important characteristic was its complexity — the most I ever had in a hobby project —, and at various points I must admit I was daunted by it. The software required to make the pinball logic was especially intricate because it involved multiple timers and taking care of many asynchronous events, but carefully refactoring the code in smaller modules allowed me to solve most issues adequately. I know there is always room for improvement — not only in the software area, but also in the mechanics and electronics —, but this project already took a lot of time and work and in the end, I became very much pleased with the results.

I will dismantle the whole thing at some point in the future to reuse the parts for new projects. I ever want to build it again, the best way to start would be using a large baseplate 1000 ([35602](#)) so to avoid the off-grid issues that are inevitable with adjoining baseplates 500. A more ambitious installment would include a coin insert slot, kick-back solenoids, stored high scores, and more.

Finally, I want to express my deep appreciation for the warm welcome I had from the ft community, the incredibly creative models and their photographs in the picture pool, and thank the members for their invaluable help.

## References

- [1] fischertechnik: [TXT](#)  
[ElectroPneumatic kit](#).  
ftdatenbank.de.
- [2] ft community: [Questions and experiments with pneumatics](#).  
ftcommunity.de forum, 2021.
- [3] ft community: [New Pinball “Dirty Dishes”](#). ftcommunity.de forum,  
2021.
- [4] Wölffel, Dirk: [fischertechnik Pinball](#).  
ftcommunity.de picture pool, 2017.
- [5] Wölffel, Dirk: *fischertechnik-Flipper*. [ft:pedia 2/2017](#), P. 74–81.
- [6] Figma, Inc.: [Figma](#). figma.com.
- [7] Serif Europe: [Affinity Designer](#).  
affinity.serif.com.
- [8] Getty Images: [iStock](#).  
istockphoto.com.
- [9] EQT: [Freepik](#). freepik.com.
- [10] Internet Pinball Database. [Skills for the Pinball Player](#). ipdb.org.
- [11] Pechansky, Rubem: [Public repositories](#). github.com, 2022.
- [12] Google images. [7 segment display alphabet](#).
- [13] FreeSound team: [Freesound](#).  
freesound.org.
- [14] The Audacity Team: [Audacity](#).  
audacityteam.org.

## Elektronik

## Silberlinge: Original oder Nachbau (Teil 6)

Peter Krijnen

Wie in Teil 5 angekündigt, werde ich mich in Teil 6 mit dem Mono-Flop ([36480](#)) beschäftigen.

Während das Kapitel zum Flip-Flop im Buch „Experimente + Modelle hobby 4 Band 3“ [1] noch 12 Seiten umfasste, musste das Kapitel zum Mono-Flop ab Seite 75 mit nur fünf Seiten auskommen. Das Mono-Flop wird nur verwendet, um die Länge der am Eingang SP anliegenden Impulse zu ändern. Je nach verwendeter Verbindung zwischen den Anschlüssen „Zeit“ an „kurz“, „Zeit“ an „lang“ oder „Zeit“ an „ext“ können breite Impulse sehr schmal oder umgekehrt schmale Impulse sehr viel breiter gemacht werden. Der SV-Eingang kann verwendet werden, um den SP-Eingang zu blockieren.

Um das Mono-Flop (Abb. 152) als Pulsformer zu beschreiben, heißt es auf Seite 77 [1] das Modell 67.1 auf Seite 67 [1] aufzubauen. Es handelt sich um einen langsam laufenden alten grauen Motor mit einer Drehscheibe ([31019](#)) auf der Abtriebswelle. In diese Drehscheibe wurde eine Welle eingesetzt, die wiederum den Lichtstrahl von der Lampe zum LDR unterbrechen muss. Ich habe dieses Modell mit der gezeigten Schaltung gebaut. Anstelle von einer habe ich jedoch drei Achsen verwendet. Der Aufbau ist auf Abb. 153 zu sehen. Anstelle des grauen Motors habe ich jedoch einen Power Motor verwendet.

Das Messergebnis auf meinem Oszilloskop-Bildschirm ist auf Abb. 154 zu sehen. Auf Kanal 1 (gelb) sehen wir das Signal vom LDR, der an E1 des Grundbausteins angeschlossen ist. Auf Kanal 2 (hellblau) sehen wir A2 des Grundbausteins. Q des Flip-Flops ist mit Kanal 4 (dunkelblau) verbunden. Da der Motor trotz seiner eingebauten Untersetzung von 1:125 (Conrad 244031 oder 1711490) bei 9 V immer noch zu schnell dreht, ist die Schaltung zu langsam, um die Spannung am Motor umzupolen. Die Spannung wird zwar umgepolt, jedoch erst nachdem die Achse den Lichtstrahl bereits durchquert hat, so dass die Achse den Lichtstrahl erneut durchquert. Der Motor wird wieder umgepolt und die Welle durchläuft zum dritten Mal den Lichtstrahl, mit der Folge, dass der Motor

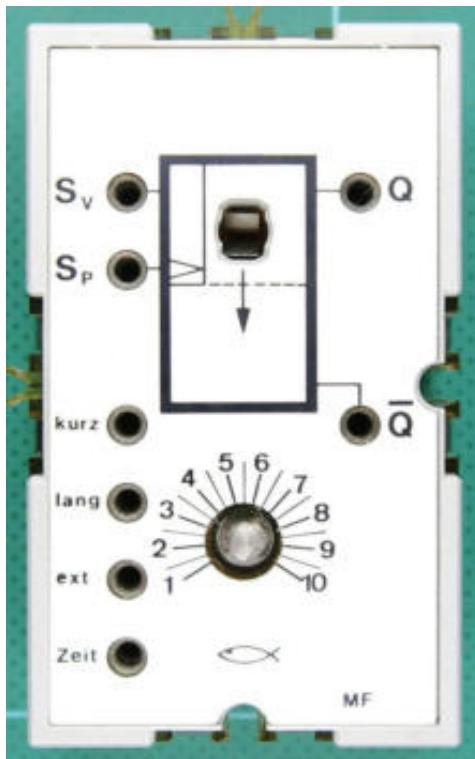


Abb. 152: Mono-Flop-Baustein [36480](#)

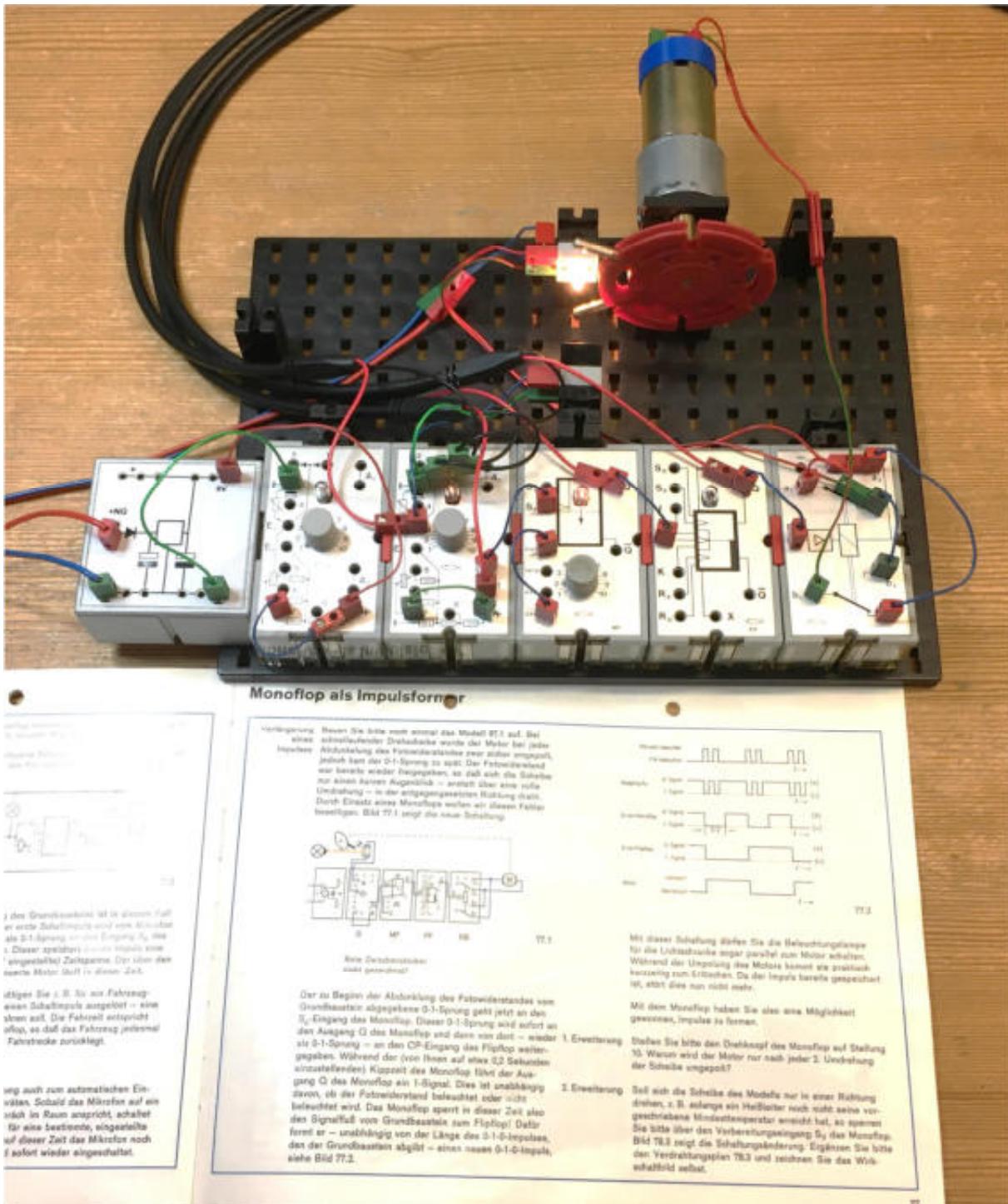


Abb. 153: Messaufbau

umgepolzt wird und unterricht den Lichtstrahl zum vierten Mal. Allerdings ist die Zeit zwischen dem vierten und dem darauf folgenden fünften Impuls nun so kurz, dass das Flip-Flop nicht mehr folgt und der Motor in die gleiche Richtung weiterdreht.

Um dieses Problem zu lösen, wird auf Seite 77 [1] die Schaltung 77.1 aufgebaut. Dies ist die gleiche Schaltung wie die von 67.1, jedoch um ein Mono-Flop ergänzt. Auf Abb. 154 sehen wir das gleiche wie auf Abb. 155, aber auf Kanal 3 (lila) sehen wir

jetzt auch Q des Mono-Flops. Das Potentiometer des Mono-Flops steht auf Position 1 und „Zeit“ ist mit „Kurz“ verbunden. Auch jetzt sehen wir, dass der Motor nach einem Zucken immer noch in die gleiche Richtung dreht.

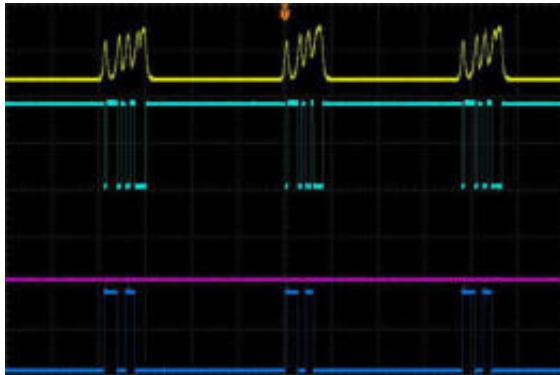


Abb. 154: Messergebnis ohne Mono-Flop

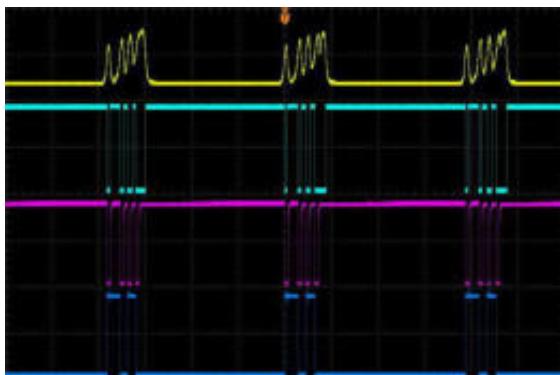


Abb. 155: Messergebnis mit Mono-Flop, Poti auf 1, Zeit an kurz

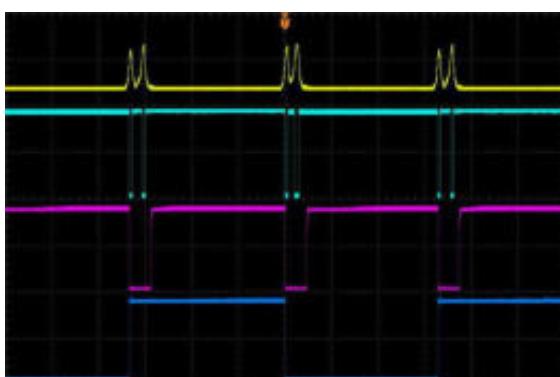


Abb. 156: Messergebnis mit Mono-Flop, Poti auf 2, Zeit an kurz

Durch Drehen des Potentiometers können wir den ersten Impuls so verlängern, dass der zweite Impuls (und ggf. mehrere Impulse) den CP-Eingang des Flip-Flops

nicht mehr erreichen kann. Dies sehen wir deutlich in Abb. 156. Das Potentiometer steht jetzt auf Position 2. Da der Motor noch zu schnell läuft, sehen wir weiterhin die zwei Impulse ab dem Moment, in dem die Welle den Lichtstrahl zweimal durchläuft. Deutlich zu erkennen ist auch, dass der Puls des Mono-Flop jetzt breiter ist als die Pulse des Grundbausteins. Drehen wir das Potentiometer weiter, geht es gut bis Position 4. Drehen wir noch weiter, sehen wir auf Abb. 157, dass der Puls des Mono-Flop so breit wird, dass er auch den nächsten Puls des LDR unterdrückt. Anstatt nach 1/3 Umdrehung umzupolen, geschieht dies jetzt erst nach 2/3 Umdrehungen. Ab Position 6.5 erfolgt eine komplette Umdrehung und ab Position 9.25 sind es sogar 4/3 Umdrehungen, bevor der Motor umgepolt wird.

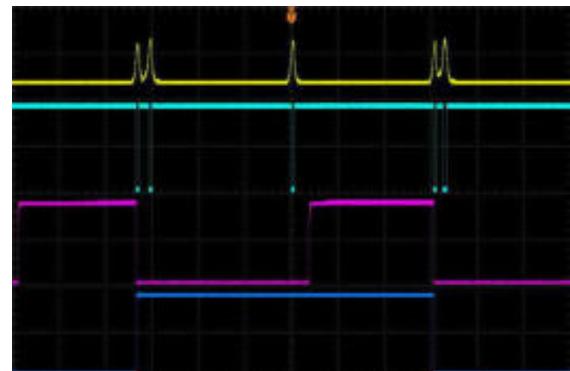


Abb. 157: Messergebnis mit Mono-Flop, Poti auf 4, Zeit an kurz



Abb. 158: Messergebnis mit Mono-Flop, Poti auf 1, Zeit an lang

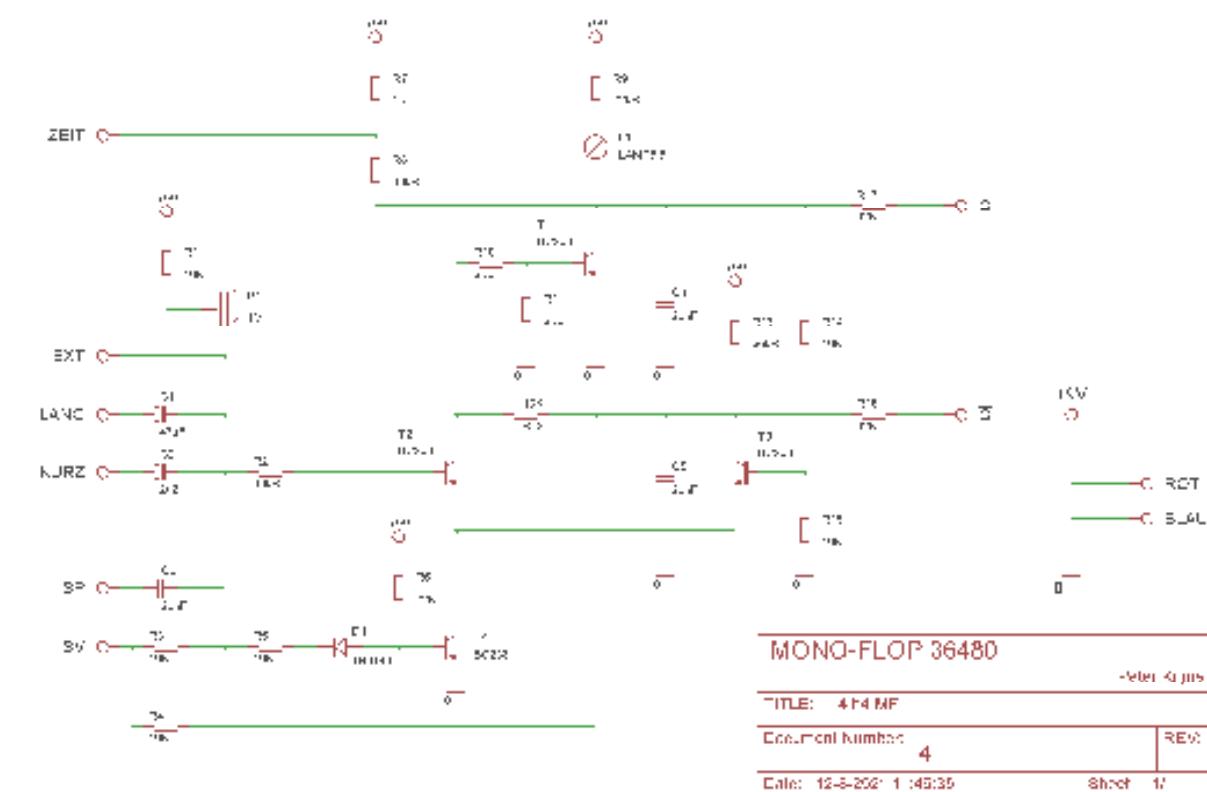
Verbinde ich nun „Zeit“ mit „lang“ statt „Zeit“ mit „kurz“, sehen wir auf Abb. 158, dass der Puls des Mono-Flops schon sehr breit ist. Und dann steht das Potentiometer nur auf Position 1. Auf Position 2 dreht der Motor bereits  $\frac{5}{3}$  Umdrehungen, also werden vier Lichtstrahlunterbrechungen durch den Mono-Flop unterdrückt.

Wenn wir uns das Schaltbild in Abb. 159 anschauen, sehen wir, dass am Anschluss „kurz“ ein Kondensator von  $2,2 \mu\text{F}$  und an am Anschluss „lang“ ein Kondensator von  $47 \mu\text{F}$  angeschlossen ist. Wir sehen auch, dass der „ext“-Anschluss mit „-“ beider Kondensatoren verbunden ist. Daran kann ein externer Kondensator angeschlossen werden. Die „+“-Seite der Kondensatoren muss dann mit „Zeit“ verbunden werden. Auf dem Schaltbild sehen wir auch, dass eines der Enden des  $1-\text{M}\Omega$ -Potentiometers mit „ext“ und damit auch mit „-“ der Kondensatoren verbunden ist. Der Schleifer

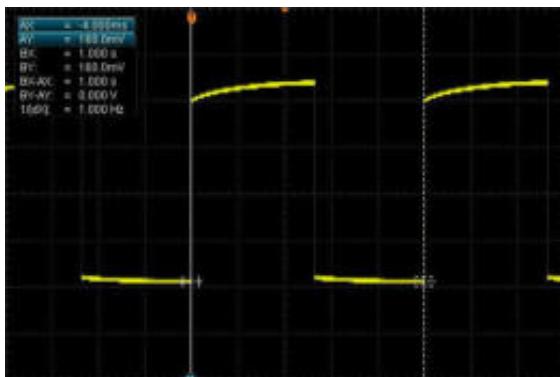
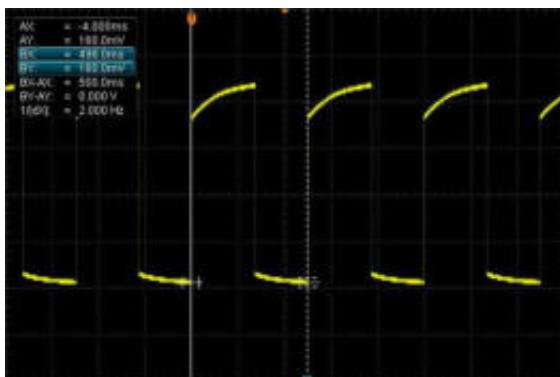
des Potentiometers ist mit einem Widerstand von  $10\text{ k}\Omega$  an „+“ angeschlossen.

In meinem Beitrag zum Flip-Flop erwähnte ich bereits das „RC-Netzwerk“ und dass die Kombination aus Kondensator und Widerstand die Frequenz bestimmt.

Wenn wir eine Frequenz von 1 Hz annehmen, sehen wir in Abb. 160, dass die Länge einer Periode zwischen den beiden Markierungen A und B eine Zeit von 1 s ergibt. In Abb. 161 sehen wir die Darstellung einer Frequenz von 2 Hz. Die Zeit zwischen den zwei Markern beträgt wieder eine Periode. Die Dauer dieser Periode beträgt 500 ms. Das ist die halbe Zeit einer Periode der Frequenz von 1 Hz. In der Tabelle, in der oberen linken Ecke beider Bilder, ist der unterste Eintrag „ $1/dX$ “ wobei  $dX$  für  $BX-AX$  steht, oder die Zeit zwischen Marker A und Marker B. Auf Abb. 160 steht „ $1/dX$ “ 1.000 Hz, Abb. 161 sagt 2.000 Hz.



*Abb. 159: Schaltbild des Mono-Flops*

Abb. 160:  $A_2$  des Grundbausteins: 1 HzAbb. 161:  $A_2$  des Grundbausteins: 2 Hz

Hoffentlich wird der Zusammenhang zwischen Frequenz und Periode jetzt etwas klarer: Ein Signal mit einer Frequenz von 1 Hz hat eine Periode von 1 s. Wenn das Signal eine Frequenz von 2 Hz hat, beträgt die Periode 0,5 s. „ $1/dX$ “ zeigt bereits, wie es funktioniert:  $1/\text{Zeit} = \text{Frequenz}$ .  $1/0,5\text{ s} = 2\text{ Hz}$ . Umgekehrt ist auch möglich:  $1/\text{Frequenz} = \text{Zeit}$ :  $1/2\text{ Hz} = 0,5\text{ s}$ . Die beiden Signale auf den Abb. 160 und 161 kommen von  $A_2$  eines Grundbausteins, von dem  $A_2$  auch an 6 und 7 verbunden ist, und 4 angeschlossen ist an E1.

Zurück zum Potentiometer und den Kondensatoren. Auf Seite 75 [1] ist angegeben, was die „Kippzeiten“ des Mono-Flops sind. Bei „Zeit“ an „kurz“ liegen die Zeiten zwischen 0,02 s und 2,5 s und bei „Zeit“ an „Lang“ zwischen 0,4 s und 60 s. Ich habe das überprüft und festgestellt, dass mein Mono-Flop etwas anders ist. Tabelle 1 (Abb. 162) zeigt die Ergebnisse meiner Messungen. Da ich sofort gesehen habe, dass etwas nicht stimmen muss, habe ich

alle Daten in Grafik 1 (Abb. 163) darstellt. Obwohl das Potentiometer ein linearer Typ ist, spiegelt sich dies nicht sofort in der Grafik wider. Diese Linearität fehlt an beiden Enden. Diesen Effekt sehe ich auch bei anderen Silberlingen, auch bei den Nachbau-Versionen. Der Gesamtwiderstand eines 1-MΩ-Potentiometers + 10-kΩ-Widerstands sollte mindestens 10 kΩ und maximal 1,01 MΩ betragen. Die Tabelle zeigt deutlich, dass dies bei diesem Mono-Flop nicht der Fall ist.

Gemessen am Mono-Flop

Zeit in Sekunden

poti	Widerstand Ohm	kurz $2\mu\text{F}$	lang $47\mu\text{F}$
10	1.140.200	1,97	64,80
9	1.124.330	1,94	63,60
8	1.023.500	1,78	58,60
7	850.100	1,47	48,00
6	680.300	1,16	36,40
5	527.780	0,89	29,40
4	369.700	0,61	19,00
3	190.700	0,30	9,56
2	59.360	0,20	2,77
1	10.200	0,15	0,44

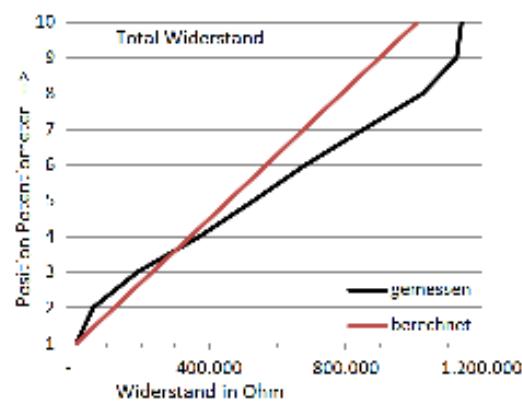
Abb. 162: Tabelle 1: Widerstand und Zeiten  
gemessen am Mono-Flop

Abb. 163: Grafik 1: Daten aus Tabelle 1

Die Zeit ist nicht mehr als die Multiplikation von Widerstand mit Kapazität:  $R \times C$ . Tabelle 2 (Abb. 164) zeigt die theoretisch

berechneten Zeiten. Diese sind für 2,2  $\mu\text{F}$  etwas länger und für 47  $\mu\text{F}$  viel kürzer als die tatsächlichen Werte.

Berechnet nach den Werten der Komponenten

Zeit in Sekunden

poti	Widerstand Ohm	kurz	lang
		2 $\mu\text{F}$	47 $\mu\text{F}$
10	1.010.000	2,22	47,47
9	898.888	1,98	42,25
8	787.777	1,73	37,03
7	676.666	1,49	31,80
6	565.555	1,24	26,58
5	454.444	1,00	21,36
4	343.333	0,76	16,14
3	232.222	0,51	10,91
2	121.111	0,27	5,69
1	10.000	0,02	0,47

Abb. 164: Tabelle 2: Zeiten berechnet

korrigiert um die Abweichung der Komponenten

Zeit in Sekunden

poti	Widerstand Ohm	kurz	lang
		1 $\mu\text{F}$	57 $\mu\text{F}$
10	1.140.200	1,94	64,99
9	1.124.330	1,91	64,09
8	1.023.500	1,74	58,34
7	850.100	1,45	48,46
6	680.300	1,16	38,78
5	527.780	0,90	30,08
4	369.700	0,63	21,07
3	190.700	0,32	10,87
2	59.360	0,10	3,38
1	10.200	0,02	0,58

Abb. 165: Tabelle 2: Zeiten berechnet nach Korrektur für Bauteilabweichungen

Um den Messwerten etwas nahe zu kommen, musste ich den Wert des kleinen Kondensators auf 1,7  $\mu\text{F}$  absenken und den großen auf 57  $\mu\text{F}$  erhöhen. Die Ergebnisse dazu findet sich in Tabelle 3 (Abb. 165). In Grafik 2 (Abb. 166) sind alle drei Messwerte für den 2,2- $\mu\text{F}$ -Kondensator und in

Grafik 3 (Abb. 167) für den 47- $\mu\text{F}$ -Kondensator zu finden.

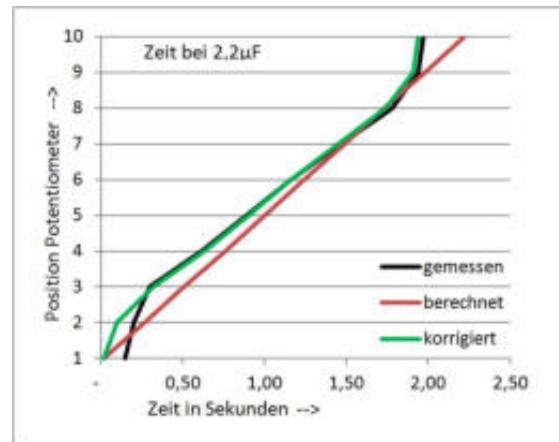


Abb. 166: Grafik 2: Zeiten für 2,2  $\mu\text{F}$

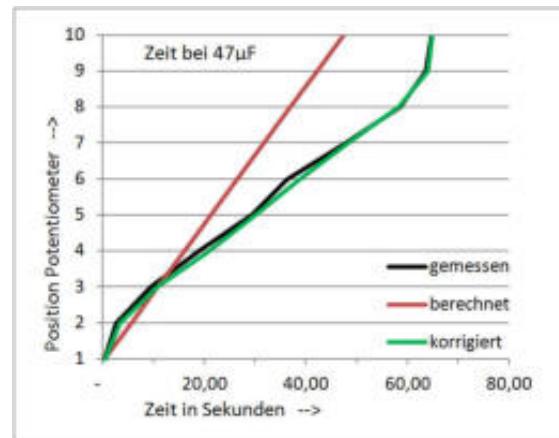


Abb. 167: Grafik 3: Zeiten für 47  $\mu\text{F}$

## Nachbau

Wie mittlerweile bekannt sein wird, zerlege ich alle Silberlinge. Das habe ich auch mit den Mono-Flop gemacht. Wie der Mono-Flop von innen aussieht, sehen wir auf den Abb. 168, 169 und 170. Abb. 171 zeigt das Layout. Abb. 172 zeigt die alternative Version, die als Ersatz im Originalgehäuse verwendet werden kann.



Abb. 168: Platine des Mono-Flops

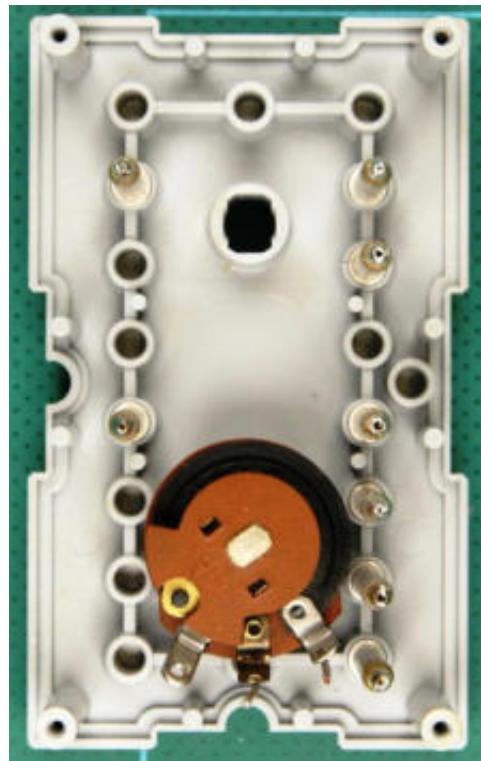
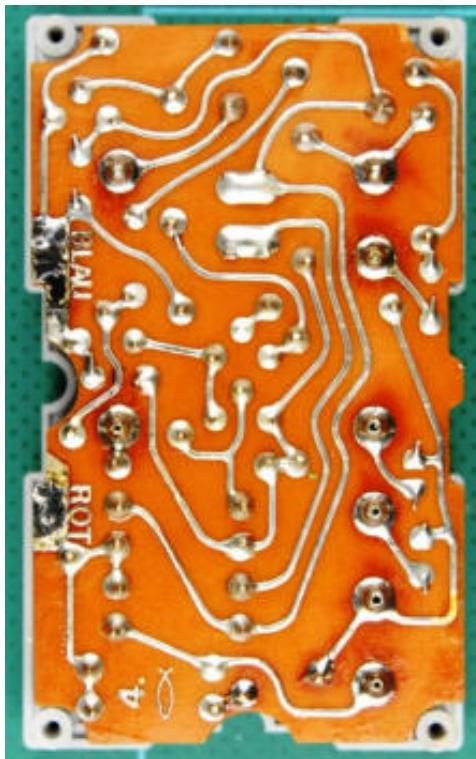
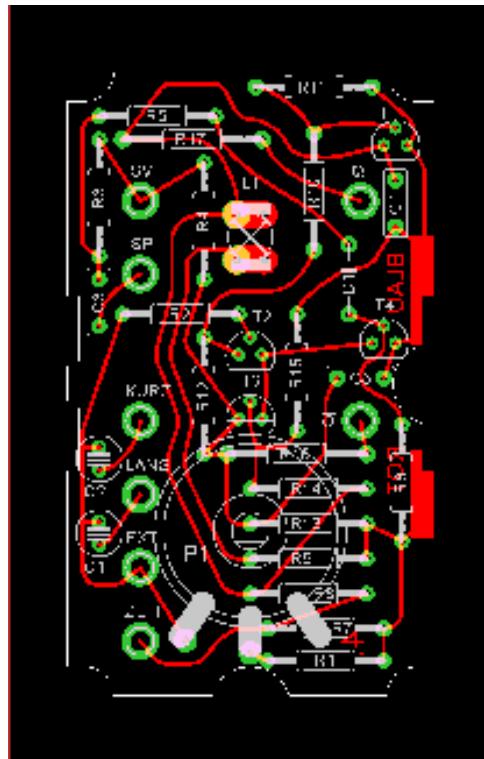


Abb. 170: Unterseite des Deckels



*Abb. 169: Leiterbahnseite des Mono-Flops*



*Abb. 171: Layout des Mono-Flops*

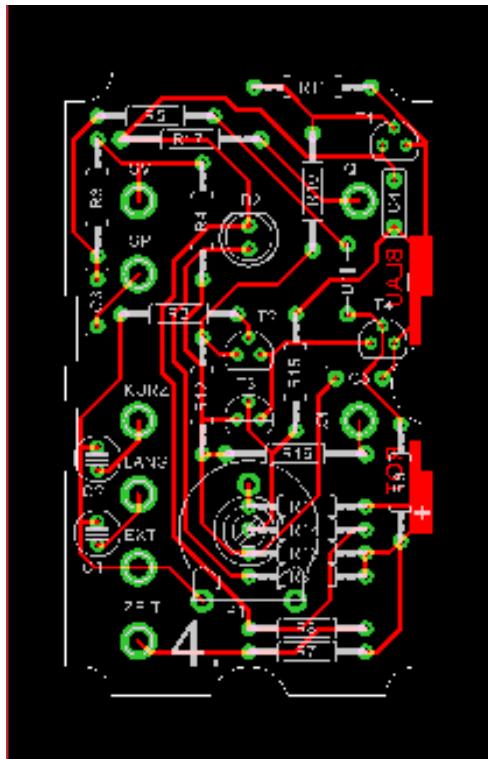


Abb. 172: Nachbau 1: Ersatz-Layout für das Original-Silberling-Gehäuse



Abb. 174: Nachbau 2: Platine in das 45×75-Gehäuse

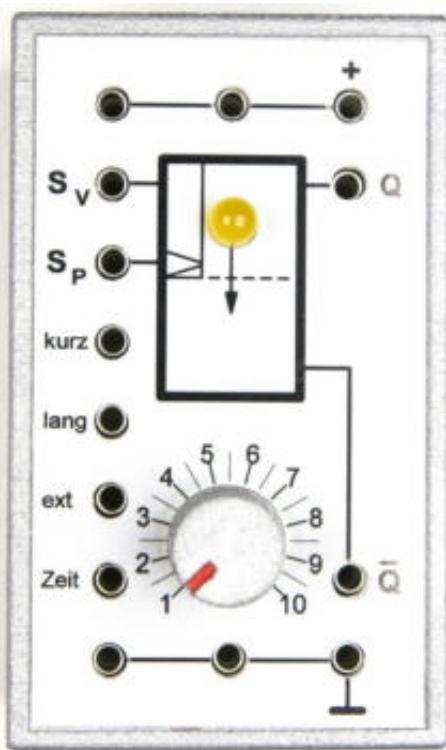


Abb. 173: Nachbau 2: Frontplatte für das 45×75-Gehäuse

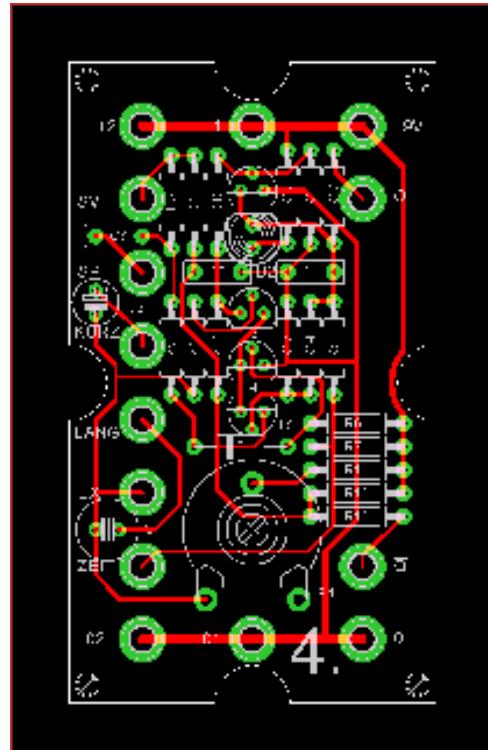


Abb. 175: Nachbau 2: Layout für mein 45×75-Gehäuse

Ganz neu ist die auf Abb. 173 zu sehende Version. Im Gegensatz zum Grundbaustein und dem Flip-Flop bietet das Mono-Flop ebenso wie der Relaisbaustein genügend Platz, um drei „+“- und drei „-“-Anschlüsse einzubauen. Abb. 174 zeigt, wie die Platine in mein 4575 3D-Gehäuse passt und Abb. 175 das Layout.

Die Platine (Abb. 177) für mein  $60 \times 60$  3D-Gehäuse (Abb. 176) passt auch wieder in die Kassette 32076. Den Aufbau findet sich auf Abb. 178.

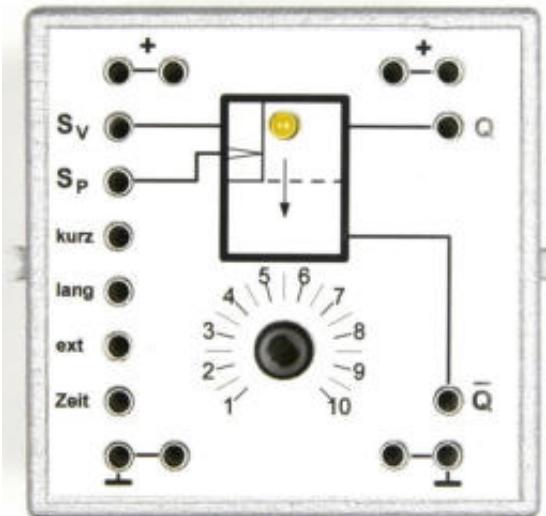


Abb. 176: Nachbau 3: Frontplatte für das  $60 \times 60$ -Gehäuse

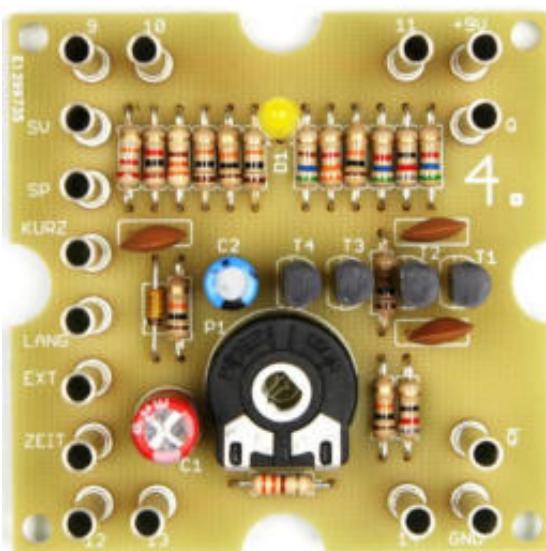


Abb. 177: Nachbau 3: Platine für das  $60 \times 60$ -Gehäuse

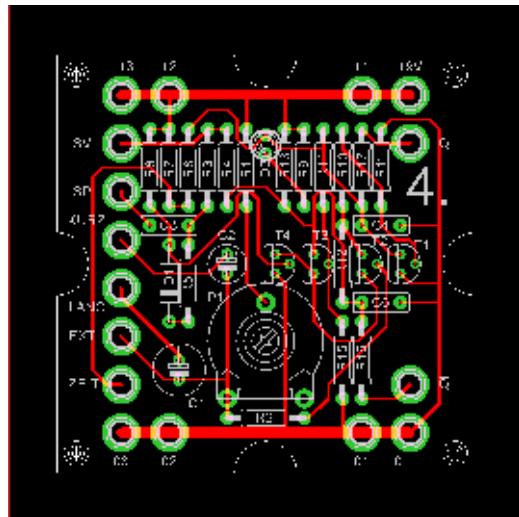


Abb. 178: Nachbau 3: Layout für die 60er-Kassette (32076)

Das Batteriegehäuse 32263 kommt auch nicht zu kurz. Auf Abb. 179 ist der 3D-Deckel mit der auf Fotopapier gedruckten Frontplatte zu sehen. Auf Abb. 180 sehen wir die Platine im Gehäuse. Auf der linken Seite sehen wir gerade noch zwei der vier Nocken, die benötigt werden, um die Platine auf die richtige Tiefe zu bringen. Das Layout gibt es in Abb. 181.

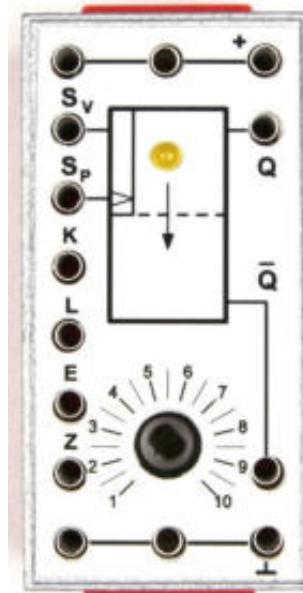


Abb. 179: Nachbau 4: Frontplatte für das Batteriegehäuse 32263



Abb. 180: Nachbau 4: Platine im 30×60-Gehäuse

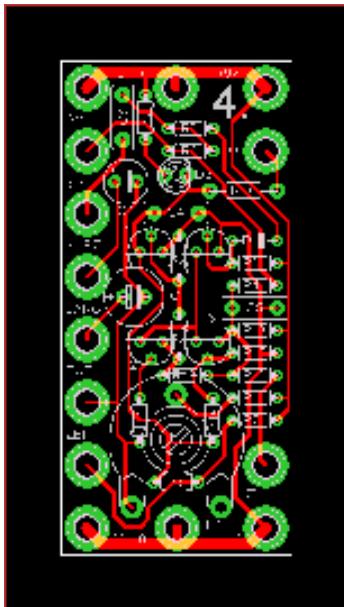


Abb. 181: Nachbau 4: Layout für das Batteriegehäuse 32263

Schließlich ist auf Abb. 182 der „Shield“ zu sehen. Das Layout (Abb. 183) zeigt, dass sich unter dem Potentiometer noch sechs Widerstände und zwei Kondensatoren befinden. Da die SMDs sehr niedrig sind, war dafür viel Platz.

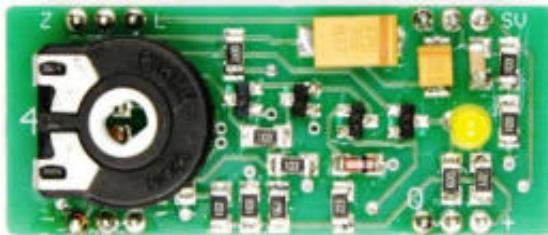
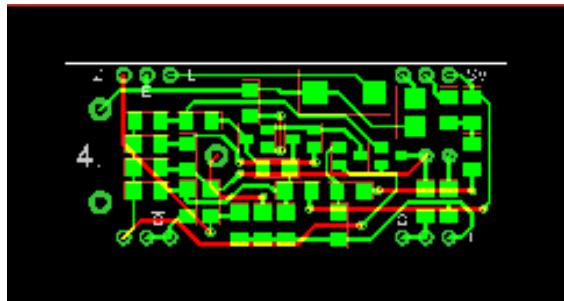


Abb. 182: Nachbau 5: für ein Breadboard:  
Mono-Flop als Shield in SMD-Technik



*Abb. 183: Nachbau 5: Doppelseitiges Layout des Shields*

Am Ende meines Beitrags möchte ich darauf hinweisen, dass es beim Mono-Flop keinen Sinn ergibt, die drei 22 nF-Kondensatoren gegen solche mit 10 nF Kapazität zu tauschen, wie ich es beim Flip-Flop gemacht habe. In meinem Beitrag zum Flip-Flop habe ich bereits angedeutet, dass die Grenzfrequenz des Mono-Flops bei etwa 100 Hz liegt. Wenn jedoch eine kleinere Kapazität an „ext“ angeschlossen würde, sollte es möglich sein, die Grenzfrequenz zu erhöhen. Da der Mono-Flop zum Einstellen eines einzelnen Pulses gedacht ist, leiden wir nicht unter einer niedrigen Grenzfrequenz.

Abschließend noch die Ankündigung, dass ich mich im nächsten Beitrag mit dem OR-NOR und dem AND-NAND beschäftigen werde. Beide sind auf der gleichen Platine aufgebaut. Merkwürdig ist aber, dass beide die gleiche Wirkung haben. Der Unterschied besteht jedoch darin, dass OR-NOR eine negative Wirkung hat und AND-NAND eine positive. Wie das dann? Das werden wir im nächsten Beitrag sehen.

## Referenzen

- [1] Peter Krijnen: *Silberlinge: Original oder Nachbau (Teil 1)*. [ft:pedia 1/2021](#), S. 80–93.
- [2] Peter Krijnen: *Silberlinge: Original oder Nachbau (Teil 2)*. [ft:pedia 2/2021](#), S. 80–89.
- [3] Peter Krijnen: *Silberlinge: Original oder Nachbau (Teil 3)*. [ft:pedia 2/2021](#), S. 90–100.
- [4] Peter Krijnen: *Silberlinge: Original oder Nachbau (Teil 4)*. [ft:pedia 3/2021](#), S. 38–48.
- [5] Peter Krijnen: *Silberlinge: Original oder Nachbau (Teil 5)*. [ft:pedia 4/2021](#), S. 35–44.

## Elektronik

## Der Zauberling (Teil 4): Die Weiterentwicklung

Arnoud van Delden

*Es war eine lehrreiche Herausforderung, den Prototypen des magischen Zauberlings mit einem Arduino Pro Mini in den relativ kleinen Innenmaßen des originalen Silberling-Gehäuses (40 x 70 mm und einer Tiefe von nur 25 mm) unterzubringen. Mit fortschreitender Erfahrung, Verbesserungsideen und anderen Zusatzwünschen fing ich bald an, über eine nächste Version mit diversen technischen Verbesserungen und einem schlankeren „Produktionsprozess“ für die Leiterplatten und Fronten zu phantasieren. Es war an der Zeit, Leiterplatten zu entwerfen und 3D-Drucker und Lötkolben einzuschalten.*

### Neue Wünsche tauchen auf

Die Programmierschnittstelle TTL FTDI hatte ich bereits auf der Rückseite des Prototyps des Zauberlings zugänglich gemacht. Um die Konnektivität weiter zu erhöhen und den Einsatz externer Sensoren und die Ansteuerung von Modulen mit bestimmten Aufgaben zu ermöglichen, sollte auch der serielle I<sup>2</sup>C-Bus nach außen geführt werden.

Manchmal wäre es auch nützlich, den Prozessor des Zauberlings einfach zurücksetzen zu können, zum Beispiel nach Programmänderungen mit den DIP-Schaltern. Ein großer Mangel des Prototyps war das Fehlen eines Druckknopfs, mit dem der interne Mikroprozessor Atmega328P zurückgesetzt werden kann.

Sollte Platz übrig bleiben und die technischen Möglichkeiten dies zulassen, wäre auch ein „Trigger“-Knopf auf der Vorderseite des Zauberlings sinnvoll. Seine Funktion könnte sich je nach Anwendung unterscheiden. Mit einem solchen Taster wäre es beispielsweise möglich, einen Zähler zurückzusetzen oder die Funktionalität zu

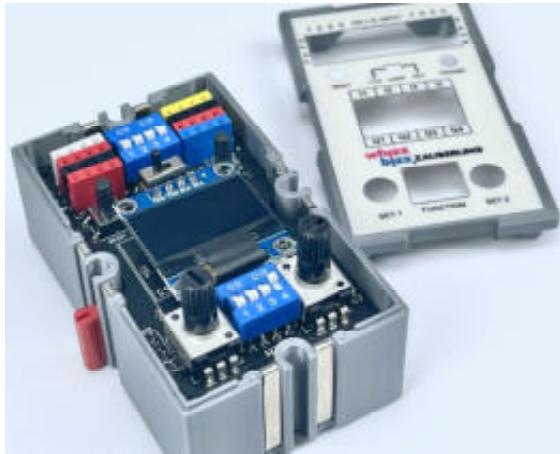
testen, ohne gleich einen externen kabelgebundenen Schalter anschließen zu müssen.

Für eine bessere Kompatibilität mit den Silberlingen wünschte ich mir zusätzlich zu den Motortreiberausgängen Logikausgänge mit dem traditionellen 9-Volt-Pegel. Für die beiden Arten von Ausgängen wäre es außerdem sinnvoll, wenn das Modul selbst automatisch zwischen der Spannungsversorgung über die Leisten an den Gehäuseseiten und der Versorgungsspannung eines optional rückseitig geschlossenen externen Netzteils wählen könnte. Auf diese Weise könnte der Zauberling mit beiden Anschlüssen verwendet werden, ohne dass man an das Umschalten der Versorgung denken muss.

Da der Platz auf der Frontseite begrenzt ist und ich unbedingt ein kleines Display einbauen wollte, habe ich beschlossen, auf die traditionellen fischertechnik-Anschlussbuchsen zu verzichten. Ich habe bereits für jedes Modell vorgefertigte oder maßgefertigte Experimentierkabel verwendet – bei

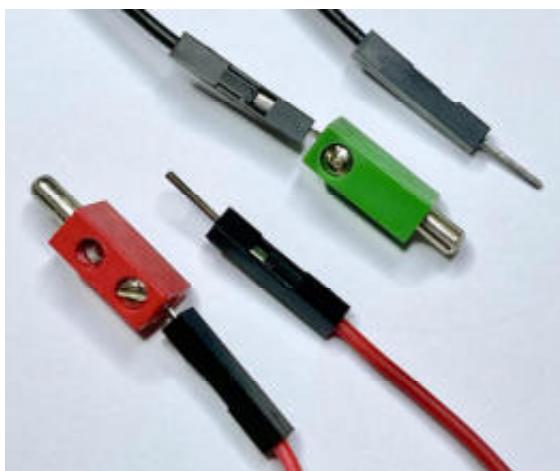


Bedarf mit fischertechnik-Steckern ausgestattet, um originale fischertechnik-Teile zu verbinden (Abb. 2).



*Abb. 1: Weiterentwickelter Zauberling mit geöffneter Frontseite*

Durch die Verwendung sogenannter „Male Header Connectors“ auf dem Zauberling wird viel Platz gespart und es bleibt Raum z. B. für nützliche zusätzliche Spannungsanschlüsse auf der Frontseite zur Versorgung aktiver Sensoren.



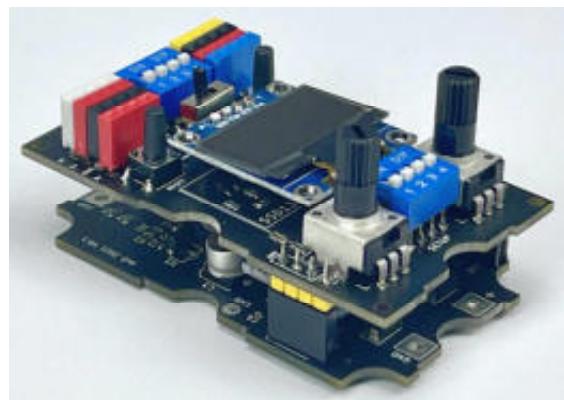
*Abb. 2: fischertechnik-Stecker an Male-Steckbrücken*

In der Praxis erwiesen sich schließlich die Servo-Anschlüsse am Prototypen als überflüssig. Schließlich könnte für spezielle Anwendungen, in denen Servos eine Rolle spielen, später ein separater Servotreiber wie der 16-Kanal-PCA9685 über I<sup>2</sup>C angeschlossen werden [1, 2]. Da dadurch E/A-

Pins auf dem Prozessor frei wurden, konnten am Zauberling ebenso viele Eingänge wie Ausgänge vorgesehen werden. Mit vier Ausgängen sind einige Funktionen (wie das SR-Flip-Flop) auch logischer verwendbar.

### Kleiner Raum optimal genutzt

Um den begrenzten Platz optimal zu nutzen, fiel die Wahl auf einen Aufbau mit zwei Leiterplatten, die wie eine Art „Sandwich“ aufeinander gesetzt werden (Abb. 3).



*Abb. 3: Sandwich-Platinen*

- Die Platinen an der Unterseite des Gehäuses stellt die verschiedenen Anschlüsse auf der Rückseite des Zauberlings, die Versorgungsspannungen und deren automatische Umschaltung bereit.
- Die obere Hauptplatine beherbergt den Mikroprozessor und das Motortreibermodul. Durch platzsparende SMD-Technik ist diese Platte beidseitig verwendbar, sodass die Bedienelemente auf der Frontplatte und das Display auf der gleichen Seite montiert werden können.

Die Netzteilplatine wurde so universell ausgelegt, dass sie auch später für ganz andere, beispielsweise über den I<sup>2</sup>C-Bus anzusteuernde Module in einem Silberling-Gehäuse eingesetzt werden könnte. Nicht benötigte Anschlüsse müssen in einem solchen Fall einfach nicht mit Steckern versehen werden.

Zweipolige JST-HX-Anschlüsse schienen mir geeignet, um eine optionale externe 12-

Volt-Stromversorgung anzuschließen und möglicherweise durchzuschleifen (Abb. 4). Diese können nicht falsch herum angeschlossen werden, und die Kabel sind relativ einfach selbst herzustellen.

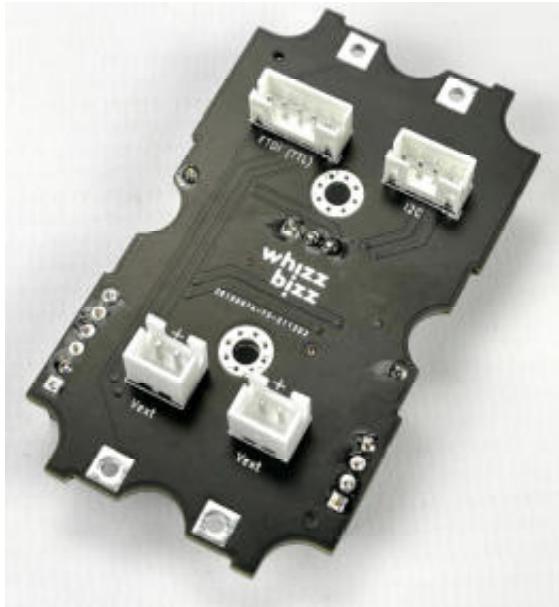


Abb. 4: Anschlüsse für I<sup>2</sup>C, TTL und die externe Stromversorgung

Für die Anschlüsse der Programmierschnittstelle und des I<sup>2</sup>C-Busses habe ich mich für SMH200/HY-Stecker entschieden. Diese ragen hinten mit ihren Rahmen ordentlich durch und haben den kleinen Vorteil, dass sie mit den Grove-Sensoren

und Breakout-Boards von Seeed kompatibel sind.

## Die Hauptplatine

Die Idee, dass sowohl Bausteine mit positiver als auch solche mit negativer Logik an den Eingängen angeschlossen werden könnten, war eine verlockende Idee. Leider erweist es sich als einschränkend für Sensoren, deren Ausgangssignal zu Beginn des Prozesses variieren kann. Daher wäre eine globale Einstellung des Moduls auf positive oder negative Logik noch geeigneter.

Eine Voraussetzung ist, dass sich sowohl Eingangs- als auch Ausgangspegel für die Verwendung mit den traditionellen Silberlingen (negative Logik) eignen. Daher sind neben den analogen Motorausgängen, die bis zum Spannungspiegel der externen Stromquelle liefern können, aus Kompatibilitätsgründen auch rein digitale Ausgänge auf dem festen „Silberlingen-Pegel“ (9 Volt) sinnvoll.

Da der Zauberling durch die DIP-Schalterstellungen so viele Funktionen annehmen kann, wäre ein kleines Display auf der Front das i-Tüpfelchen. Das Display könnte die aktuelle Funktion des Zauberlings und Zähler- oder Analogwerte anzeigen.

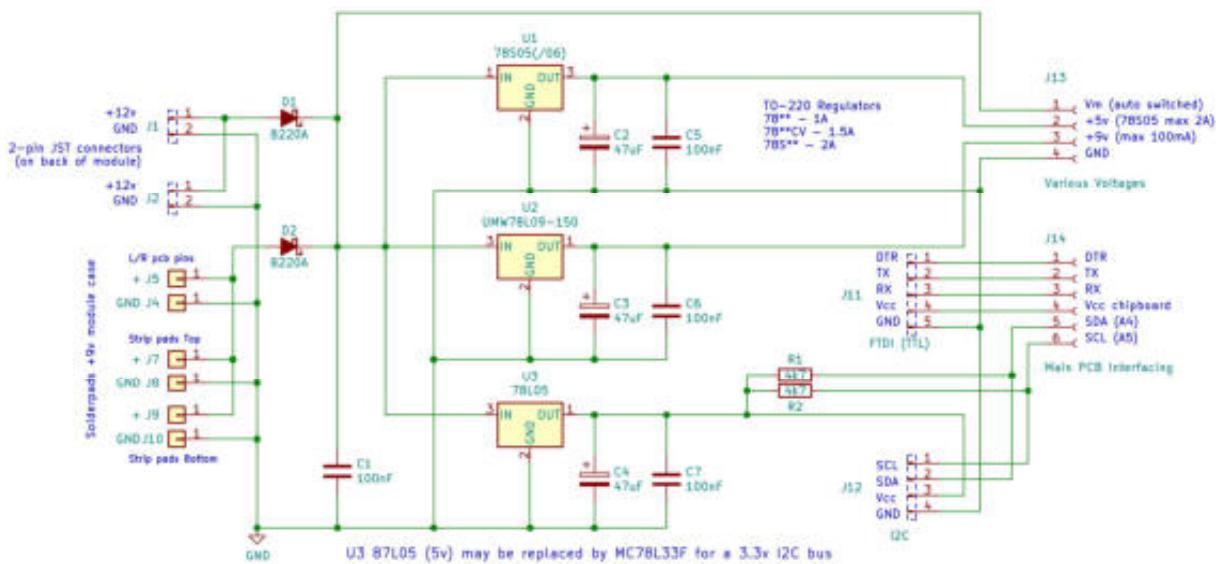


Abb. 5: Schaltplan der Schnittstellen- und der Versorgungsspannungsplatine

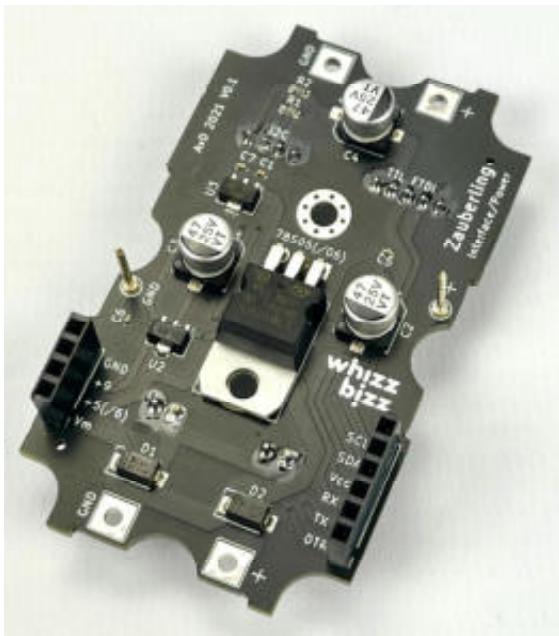


Abb. 6: Power-Interface (Front-Seite)



Abb. 7: Einbau in Silberling-Gehäuse

Da ein Display auch die an den Ein- und Ausgängen anliegenden Werte anzeigen könnte, würde dies einige (Kontroll-) LEDs auf der Vorderseite und der darunter liegenden Platine einsparen. Weil die Ports A4 (SDA) und A5 (SCL) bereits vom I<sup>2</sup>C-Bus belegt sind, lag die Wahl für ein I<sup>2</sup>C-Display nahe. Das Display SSD1306 mit

128X64 OLED-Pixeln auf ~2,5 cm (0,96 Zoll) Bildschirmdiagonale bietet bereits ausreichend Möglichkeiten für sinnvolle Anzeigen [3]. Dies ist ein billiges und leicht verfügbares Display, das über den I<sup>2</sup>C-Bus angesteuert wird (Abb. 8).



Abb. 8: Zauberling ohne Front-Platte

## Vor- und Nachteile eines Displays

Das Experimentieren mit dem SSD1306-Display zeigte einige Einschränkungen:

- Die Ausgabe auf dem Display ist relativ langsam und wirkt sich auf die Geschwindigkeit der Hauptschleife des Arduino-Skripts aus, in der die Eingänge abgefragt werden. In der Praxis reichen die 16 MHz des Arduino Pro Mini, damit dies kaum bemerkbar ist, allerdings sollte man sich darüber im Klaren sein, dass dadurch schnelle Impulse an den Eingängen übersehen werden können.

- Die Adresse auf dem I<sup>2</sup>C-Bus des Displays ist nicht (oder je nach Typ nur eingeschränkt) wählbar. Dies schränkt die gleichzeitige (und eindeutige) Verwendung des Displays ein, wenn mehrere Zauberlinge über den I<sup>2</sup>C-Bus verbunden werden.
- Die meisten Softwarebibliotheken für die SSD1306, wie z. B. die Adafruit-Bibliothek, benötigen die Hälfte des verfügbaren 2-KB-SRAM-Speichers des Arduino Pro Mini um das Display anzusteuern. Obwohl ich keine Probleme hatte, in den verbleibenden 1 KB alle Basisfunktionalitäten des Zauberlings unterzubringen, lässt dies nicht viel Raum für die Verwendung von möglicherweise zusätzlich benötigten Bibliotheken. Dies beschränkt die Verwendung von I<sup>2</sup>C-Sensoren mit relativ großen Softwarebibliotheken (wie der Time-of-Flight-Sensor VL53LXX, den ich ausprobiert habe).



Abb. 9: Geöffneter Zauberling mit Gleichrichter

Der enorme Vorteil einer Anzeige auf dem Zauberling wiegt diese Hürden in der Praxis jedoch locker auf. Darüber hinaus ist es immer möglich, für bestimmte Anwendungen eines Moduls auf die Verwendung des Displays zu verzichten. Und vielleicht ist es sinnvoll zu prüfen, ob das Display mit einer kleineren, einfacheren Bibliothek verwendet werden kann.

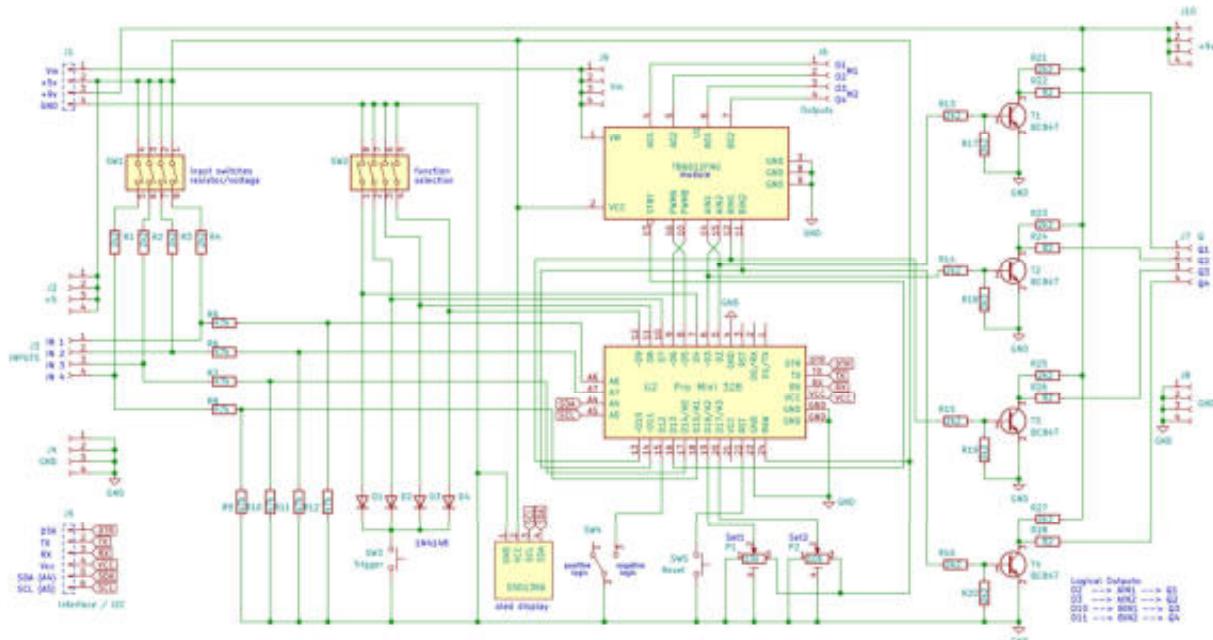


Abb. 10: Schaltplan des neuen Zauberlings

## Der modifizierte Schaltplan

Auf die Servo-Anschlüsse des Prototyps konnte verzichtet werden. Die vier analogen Eingänge können per DIP-Schalter zwischen Widerstands- oder Spannungsmesung umgeschaltet werden. Die Wahl zwischen positiver oder negativer Logik kann mit einem Schiebeschalter getroffen werden. Der Funktionsauswahl-DIP-Schalter wird während der Initialisierung des Zauberlings ausgelesen. Auf der Vorderseite stehen ein Reset-Taster und ein „Trigger“-Taster zur Verfügung. Außerdem gibt es zwei Potentiometer für frei programmierbare funktionsabhängige Einstellungen in Echtzeit.

Die Realisierung des begehrten „Trigger“-Tasters stellte noch eine kleine Herausforderung dar, da mittlerweile alle Ein- und Ausgänge des Mikroprozessors belegt waren. Der DIP-Schalter zur Einstellung der Funktionalität wird aber nur einmalig während der Initialisierungsphase des Zauberlings ausgelesen. Beim „Trigger“ wird über ein logisches ODER (mit vier Dioden) kurzzeitig die DIP-Schalterstellung 15 gedrückt, so dass die vier Eingänge während des normalen Programmablaufs wiederverwendet werden, um diesen Tastendruck zu signalisieren. Die einzige Einschränkung besteht natürlich darin, dass der „Trigger“-Druckknopf nicht mit der anfänglichen Funktionsauswahl 15 des DIP-Schalters verwendet werden kann. Da nicht für jede Funktion eine solche Extra-Taste benötigt wird, ist dies in der Praxis kein Problem.

Auf der Frontseite findet sich nun ein Reset-Taster. Für die Einstellung von Schwellenwerten oder anderen analogen Konfigurationen stehen nun zwei Potentiometer zur Verfügung. Die digitalen Ausgänge zur Steuerung der Motorausgänge liefern gepuffert 9-Volt-Silberlingen-Pegel. Obwohl diese Signale gekoppelt sind und nicht separat angesteuert werden können, kann dies bei manchen Experimenten sogar sinnvoll

sein, weil sie einen entsprechenden Ausgang mit unterschiedlichem Signalpegel und Polarität zur Verfügung stellen.

Auf dem Display sind die aktuelle Funktion des Zauberlings und weitere Konfigurationen (z. B. die Wahl positiver oder negativer Logik) sowie der Status der Ein- und Ausgänge ersichtlich. Auch spezifische Informationen pro Funktion (z. B. ein Zähler oder die Abstiegszeit eines Monoflops) können hier angezeigt werden. Da die Wahl für die frontseitigen Anschlüsse auf „Female Header Connectors“ gefallen war, war es einfach, auch hier die unterschiedlichen Versorgungsspannungen zur Verfügung zu stellen. Das erleichtert es, aktive Sensoren mit einem Stecker (wie der dreipolige eines Servos) mit Spannung zu versorgen.



Abb. 11: Montage des Mikroprozessors und der Motortreiber

## Die Leiterplatte und Montage

Schnell war klar, dass möglichst SMD-Bauteile verwendet werden sollten, um die beiden Seiten der Platinen optimal nutzen zu können. Sowohl der ATmega328 als auch der TB6612FNG-Motortreiber sind in SMT (*Surface Mount Technology*) erhältlich, aber für beide Komponenten habe ich mich schließlich für ein komplettes Board entschieden. Diese waren leichter verfügbar und seltsamerweise sogar billiger. Und da es sich in diesem Fall um eine kleine Serie von Halbprototypen handelte, schien mir der Vorteil von noch mehr PCB-Lagen für die Kupferbahnen die „Huckepack“-Montage dieser kleinen Platinen auf den ungebohrten SMT-Kupferinseln zu überwiegen.

Durch das Auflöten von „header strips“ auf die SMT-Inseln und das anschließende Abschneiden ist die Montage in der Praxis sehr einfach. Die Fotos zeigen, wie Komponenten auf beiden Seiten der Leiterplatte montiert werden. Die resultierende mehrlagige Hauptplatine wird auf die Stromversorgungs- und Schnittstellenplatine gesteckt, die auf der Unterseite des Gehäuses befestigt ist. Das dadurch entstehende „Sandwich“ nutzt den begrenzten Platz im Gehäuse auch in der Höhe optimal aus.

Am unteren Rand des Displays des Zauberlings können je nach gewählter Funktion entweder die vier digitalen oder die beiden Motorausgänge überwacht werden. Bei einer Motorfunktion werden Balken angezeigt, deren Position und Länge die Ausgangsspannung und Drehrichtung angeben. Für einige Anwendungen kann es sinnvoll sein, beide Ausgabearten zu verwenden, um beispielsweise ein digitales Signal zur Verfügung zu haben, das die Drehrichtung eines Motors anzeigt.

Für einen schnellen Überblick über die Ausgangssignale bietet ein Paar „Ausgangsanzeigen“ mit LEDs (inklusive Vorwiderstand) eine Lösung, die direkt auf die Ausgänge gesteckt werden können. Einer

bietet einen guten Überblick über die vier digitalen Ausgangssignale, der andere enthält zwei zweifarbiges LEDs, die das sofortige Ablesen der Drehzahl und Richtung jedes Motorausgangs ermöglichen (Abb. 12).

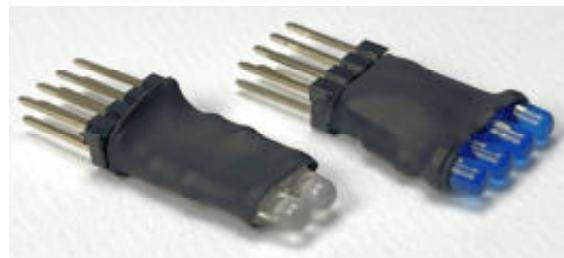


Abb. 12: Einfache Ausgangsanzeigen

## Die ersten Experimente

Der Basis-Sketch für den Zauberling kann von [GitHub](#) heruntergeladen werden und ist im Grunde derselbe wie in meinem vorherigen ft:pedia-Beitrag [4]. Mit dem DIP-Schalter „Function“ lassen sich die verschiedenen Funktionen im Zauberling einstellen. Nach Änderungen an der Konfiguration genügt ein kurzer Druck auf den Reset-Taster auf der Vorderseite, um die folgende Funktion nutzen zu können.

Standardmäßig sind zwölf nützliche Funktionen und eine Motor-Demo-Sequenz mit den DIP-Schaltern konfigurierbar. Hinzu kommen eine einstellbare Verzögerung des Ausgangssignals bei einigen Logikgattern und die Möglichkeit, spezielle Anwendungen unter den Programmoptionen 12 und 13 durch Auskommentieren im Quellcode zu aktivieren. Ich habe auch eine Version mit dem kleinstmöglichen Speicherbedarf erstellt, um mit Sensoren zu experimentieren, die eine größere Softwarebibliothek erfordern. Der Zauberling kann somit schnell auf eine bestimmte Aufgabe zugeschnitten werden, auch wenn die relativ große Belegung von SRAM durch das Display ein Problem darstellt.

In Abb. 13 ist ein Experiment mit Flip-Flops zu sehen, die als 3-Bit-Binärzähler arbeiten, und zwei Zauberlingen, die diese Binärzahl wieder entschlüsseln. Bei jedem Tastendruck leuchtet die nächste LED in

der Reihenfolge auf. Durch die Verwendung der oben beschriebenen „Ausgangsanzeigen“ direkt an den Ausgängen war keine weitere Verkabelung erforderlich. Ein schöner Proof-of-Concept für einige Grundfunktionen, aber rein digital nicht wirklich etwas, das den traditionellen Silberlingen viel hinzufügt. Tatsächlich werden diese mit ihrer blitzschnellen, parallel arbeitenden diskreten Elektronik die Zauberlinge deutlich an Geschwindigkeit übertreffen.

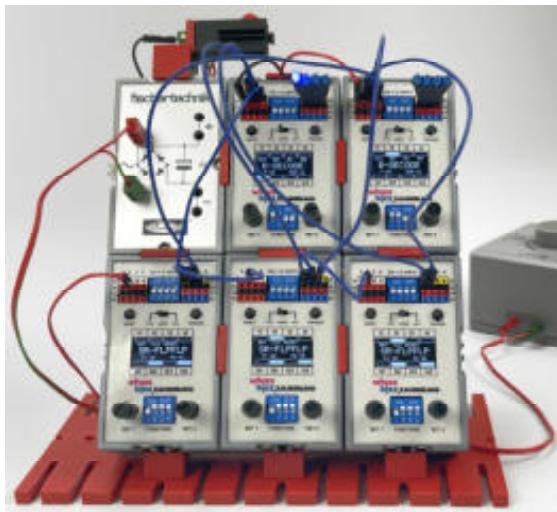


Abb. 13: 3-Bit-Binärzähler

Aber der Zauberling kommt natürlich als SPS (*Programmable Logic Controller*) für die Prozessautomatisierung oder zum Hinzufügen bestimmter Funktionen zur Geltung, die mit den Silberlingen nicht möglich sind. Seine Stärke liegt in seiner Programmierbarkeit und der I<sup>2</sup>C-Konnektivität.

## Spielen mit dem I<sup>2</sup>C-Bus

Die Hinzufügung des seriellen I<sup>2</sup>C-Busses ermöglicht es, den Zauberling mit allen Arten von externen Sensoren zu verwenden. Zum Test habe ich einige Anwendungen ausprobiert. Es stellte sich als recht einfach heraus, zwei Motoren mit einem sogenannten Nunchuk (Handcontroller) zu steuern (Abb. 14) [5]. Sowohl der Nunchuk als auch die Motoren können direkt an den Zauberling angeschlossen werden. Über denselben I<sup>2</sup>C-Bus, über den der Status der

Drucktasten oder die Beschleunigungs-/Positionsinformationen vom Nunchuk eingehen, könnten sie erneut zur Verarbeitung durch andere Zauberlinge oder spezifische I<sup>2</sup>C-Module kommuniziert werden.

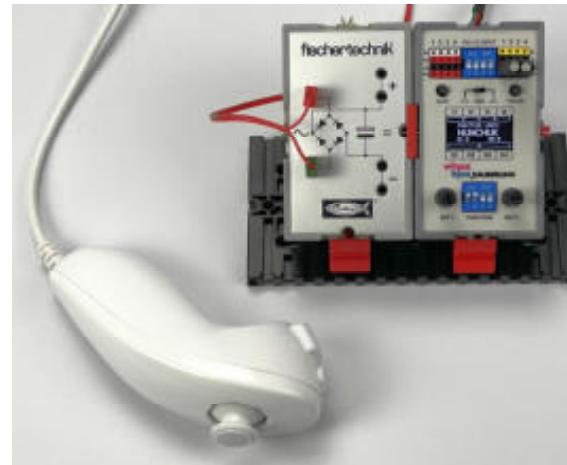


Abb. 14: Nunchuk-Steuerung

Ich habe ein zweites I<sup>2</sup>C-Sensor-Experiment mit dem Farbsensor TCS34725 durchgeführt. Die Ausgänge 1 bis 4 werden jeweils aktiviert, wenn ein roter, grüner, blauer oder gelber Dominostein erkannt wird. Dadurch ist es möglich, einen automatischen Farbsortierer zu erstellen (Abb. 15) [6, 7].



Abb. 15: Farbsortierer

## Fazit

Mit der Möglichkeit, über den seriellen I<sup>2</sup>C-Bus zu kommunizieren, ist der Zauberling bestens für die Zukunft gerüstet. Wenn beispielsweise in Zukunft mehr Ein- und Ausgänge für die Funktion eines einzelnen

Moduls benötigt werden, kann man ein externes Modul mit einer Ein-/Ausgangserweiterung (PCF8575 [8] oder MCP23017) anschließen, auf das über diesen Bus zugegriffen werden kann. Der I<sup>2</sup>C-Bus ermöglicht es auch, Schrittmotoren über einen externen Controller anzusteuern. Alles in allem gibt es viele Möglichkeiten, einige traditionelle Baubeispiele aus der [Hobby4-Buchreihe](#) mit moderner Technik umzusetzen.

Mehr über die Entwicklung der [universellen Leistungs- und Schnittstellenplatine](#) und die neue [Zauberling-Platine](#) findet ihr auf der [WhizzBizz-Website](#). Videos zum Design und der Entwicklung beider Leiterplatten findet ihr auch auf YouTube [9]. Die mit KiCad erstellten Gerber-Dateien sind auf [GitHub](#) zu finden. Die 3D-Dateien für den 3D-Druck des fischertechnik-Silberling-Gehäuses wurden in Blender erstellt und können bei [Thingiverse](#) heruntergeladen werden. Die Frontplatten, die ich mit dem Affinity Designer gezeichnet habe, wurden auf einem Laserdrucker gedruckt und dann zur Veredelung laminiert.



Abb. 16: Zauberling mit Front-Platte

Aufgrund seiner Vielseitigkeit und Konfigurierbarkeit ist der Zauberling ein so ge-

nannter „Drop-In“-Ersatz für jeden traditionellen Silberling. Aber dank seiner Programmierbarkeit, der Einstellmöglichkeiten und Leistungsausgänge geht er darüber hinaus. Da das Modul frei programmierbar ist, kann es auch die Funktionen der kleinen konfigurierbaren Elektronikmodule wie des E-Tec Module (108227), des Electronics Module (152063) oder des Robotics Mini-Bots Module (156499) in einem einzigen Modul zur Verfügung stellen.

## Quellen

- [1] Dirk Fox: *I<sup>2</sup>C mit dem TX(T) – Teil 16: Servo-Driver*. [ft:pedia 2/2017](#), S. 41–47.
- [2] Till Harbaum: *Der Mini-Servo-Adapter*. [ft:pedia 3/2019](#), S. 27–31.
- [3] Solomon Systech: [SSD1306](#). Datasheet. Rev. 1.1, April 2008.
- [4] Arnoud van Delden: *Der Zauberling (Teil 2): Das Zauberbuch*. [ft:pedia 4/2021](#), S. 45–51.
- [5] Dirk Fox: *I<sup>2</sup>C mit dem TX – Teil 4: Nunchuk-Fernsteuerung*. [ft:pedia 2/2013](#), S. 41–49.
- [6] Arnoud van Delden: *Farben sortieren mit dem TCS3472*. In dieser Ausgabe der ft:pedia.
- [7] Dirk Fox: *I<sup>2</sup>C mit dem TX(T) – Teil 13: Farbsensor*. [ft:pedia 1/2016](#), S. 79–89.
- [8] Dirk Fox: *I<sup>2</sup>C mit dem TX(T) – Teil 18: Keypads und GPIOPort-Erweiterung*. [ft:pedia 2/2019](#), S. 46–51.
- [9] Arnoud van Delden: [A printed circuit board providing various voltages and programming- and I<sup>2</sup>C ports](#) und [An Atmega328P controller: the ‘Zauberling’](#). Video, Youtube 2022.

Modell

## Farbsortierer mit dem TCS3472

Arnoud van Delden

Mit dem seriellen I<sup>2</sup>C-Bus-Anschluss auf der Rückseite des Moduls sind die Möglichkeiten des neuen Zauberlings stark gewachsen [1]. An den Bus können unter anderem zahlreiche externe Sensoren angeschlossen werden. Für mein erstes Experiment habe ich den Farbsensor TCS3472 gewählt. Die fischertechnik-Farbsortieranlage eignet sich dafür als Modell.

### Trainings- und Simulationsmodelle

fischertechnik bietet seit vielen Jahren vorgefertigte [Trainings- und Simulationsmodelle](#) an. Diese Trainingsmodelle können im technischen Unterricht und in der Ausbildung im MINT-Bereich (Mathematik, Informatik, Naturwissenschaften und Technik) verwendet werden. Mit diesen, meist mit einem Förderband oder einem Greifarm ausgestatteten Modellen lassen sich verschiedene industrielle Automatisierungsprozesse simulieren.

Die Modelle sind meist in zwei verschiedenen Ausführungen erhältlich: Als 9-Volt-Version zur Ansteuerung mit den fischertechnik-eigenen Controllern oder als 24-Volt-Modell, bei dem eine Adapterplatine das Experimentieren mit einer externen SPS (Speicherprogrammierbare Steuerung) ermöglicht. Die Adapterplatine bietet dann unter anderem Schraubklemmen zum Anschluss der Pneumatikventile und Relais,

mit denen die Drehrichtung von Motoren überprüft werden kann.

Die Schulungsmodelle sind nur auf Bestellung erhältlich und werden fertig aufgebaut geliefert. Die technische Entwicklung kann jedoch schnell gehen: Die Werbefotos verschiedener Generationen der *Sorting Line with Color Detection* ([536633](#)) scheinen sogar fotobearbeitet worden zu sein, um zu zeigen, dass das Modell mit der 24-Volt-SPS-Platine ausgestattet ist (an der seltsamerweise die Relais für den Motor zu fehlen scheinen) oder mit dem 9-Volt-TXT-Controller gesteuert werden kann (Abb. 1).

Während die Grundfunktionen dieser Modelle über die Jahre weitgehend gleich geblieben sind (Fabriksimulation, Farb- oder „Produkt“-Sortierung), werden die Simulationsmodelle laufend auf den Stand der Technik gebracht und mit den aktuellen Komponenten der fischertechnik-Produktpalette ausgestattet.

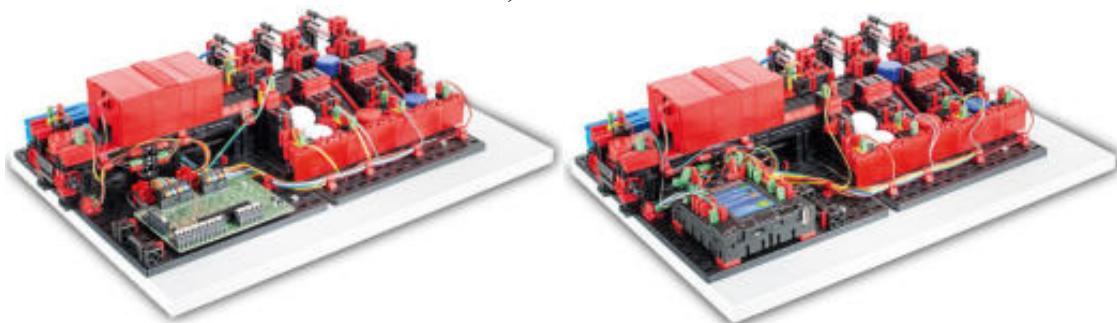


Abb. 1: Simulationsmodell [Farbsortieranlage](#) (Fotos: fischertechnik) – finde die Unterschiede...

Wo in früheren Modellen zur Farberkennung beispielsweise eine Kombination aus Kugellampe ([37869](#)) und Fototransistor ([36134](#)) als Reflexionslichtsensor verwendet wurde, kommt heute der Farbsensor ([128599](#)) zum Einsatz (Abb. 2), so wie die TX- und später die TXT-Controller die Rolle der früheren Schnittstellenmodule übernahmen.



Abb. 2: fischertechnik-Farbsensor ([128599](#))

Im Folgenden wird eine deutlich verbesserte Farbsortieranlage vorgestellt – mit drei Zauberlingen [1] als Steuereinheit.

### Die Originalfarbsortierlinie

Die Original-Sortierlinie simuliert die automatisierte Erkennung und Trennung von kleinen weißen, roten und blauen Scheiben mittels des fischertechnik-Farbsensors [2].

Sobald eine Scheibe auf dem Förderband die erste Lichtschranke passiert hat, beginnt eine Reihe von Reflexionslichtmessungen. Während sich das Objekt unter dem Sensor bewegt, beleuchtet der fischertechnik-Farbsensor das Objekt mit LED-Licht und misst die Menge des reflektierten Lichts. Sobald die Scheibe das zweite Detektionstor erreicht, enden diese Messungen und der niedrigste Reflexionslichtwert der passierten Scheibe ist ermittelt.

Jede der drei Farben ergibt einen anderen Mindestwert. Der Ausgangswert des Sensors ist eine analoge Spannung zwischen 0 und 2 Volt, die vom Umgebungslicht, der Messentfernung und der Farbe des Objekts abhängt. Mit diesem Wert kann man durch Vergleiche mit festen Schwellenwerten die

(wahrscheinliche) Farbe des passierenden Objekts bestimmen.

Danach muss sich das Förderband nur noch um die entsprechende Strecke drehen, dann kann der pneumatische Auswerfer das Objekt von Förderband in das richtige Magazin schieben. Die Anzahl der Impulse des Impulsgebers ist ein Maß für die erforderliche Wegstrecke, die das Förderband zurücklegen muss. Drei zusätzliche Lichtschranken erkennen, dass ein Farbbehälter voll ist. Das ursprüngliche Modell verwendet daher insgesamt fünf Lichtschranken.

### Farbsensor und Lasererkennung

Das Farberkennungsverfahren des fischertechnik-Farbsensors ist nicht optimal: Es wird nur ein Reflexionslichtwert gemessen. Heutzutage gibt es Farbsensoren für überraschend wenig Geld (etwa 1,50 € statt über 25 € für den fischertechnik-Sensor), die vier Reflexionskomponenten auslesen und auch über den I<sup>2</sup>C-Bus verbunden werden können. Ein gutes Beispiel ist der TCS3472 (Abb. 3) [3].

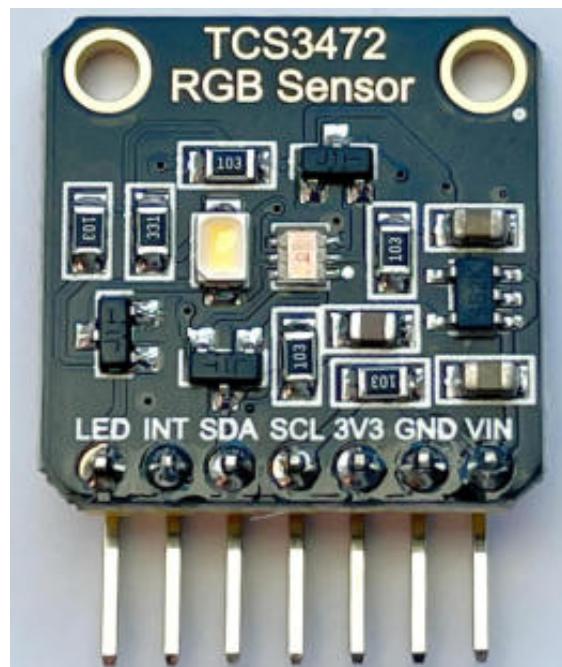


Abb. 3: TCS3472 (Adafruit)

Dieser Sensor verfügt außerdem über einen Infrarotfilter, um den störenden Einfluss von Umgebungslicht weiter einzuschränken. Dieser günstige, sehr genaue Sensor wurde bereits ausführlich in der ft:pedia beschrieben [4].

Da die Objekte in geräumige rote fischertechnik-Kassetten (130961) sortiert werden sollen (Abb. 4), habe ich die Erkennungssensoren in den Lagerplätzen weggelassen. Durch die kontinuierliche Farberkennung kann auch die Lichtschranke am Anfang entfallen. Um den Detektionsstrahl möglichst schmal zu machen, habe ich als Lichtquelle für die vier Lichtschranken entlang der Strecke Halbleiterlaser gewählt. Als Sensor reichen Fotowiderstände (LDRs) aus (Abb. 4).

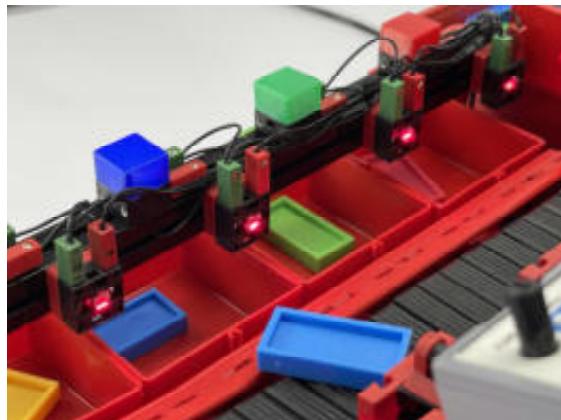


Abb. 4: Laser-Lichtschranken

Da die Detektionsschwelle bei Verwendung von Lasern sehr klein sein kann, hat das Umgebungslicht einen relativ geringen Einfluss und es müssen keine (empfindlicheren) Fototransistoren verwendet werden.

Halbleiterlaser sind nicht mehr teuer. Für wenige Euro kann man bereits ein 50-mW-Exemplar kaufen, mit dem man locker hunderte Meter weit strahlen kann. Für ein Projekt wie dieses, in dem der Abstand nur 10 cm beträgt, reicht ein kleinerer 5-mW-Laser aus. Ich habe das Modul KY-008 mit roter (650 nm) Laserdiode verwendet, das oft zusammen mit einem empfangenden Sensormodul verkauft wird (Abb. 5). Diese Module arbeiten mit 5 Volt und können

direkt vom Zauberling mit Strom versorgt werden. Als Sensor habe ich statt des mitgelieferten Laser Receiver Module ISO203 ein paar einfache LDRs verwendet, die ich in fischertechnik Leuchtsteine (38216) gelötet habe.

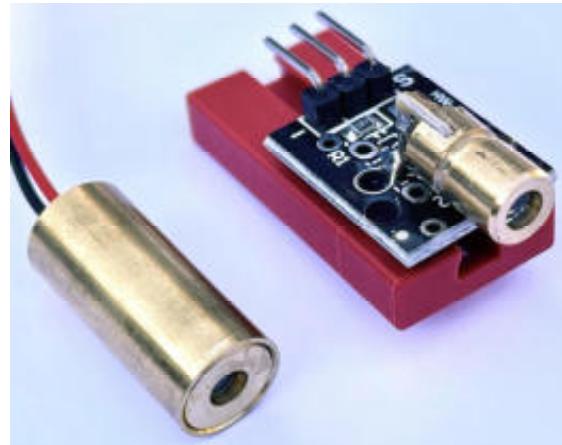


Abb. 5: Eine 50mW und eine 5mW Laserdiode

Auf Leuchtkappen zum Schutz vor Umgebungslicht habe ich verzichtet. In der Praxis waren sie nicht erforderlich, da die Empfängersensitivität des Zauberlings bei Beleuchtung mit einem so starken Laserlichtstrahl sehr niedrig eingestellt werden kann.

### Eine andere Art, Farben zu sortieren

Der Farbsensor ist über den I<sup>2</sup>C-Bus mit einem Zauberling-Modul verbunden, dessen einzige Aufgabe es ist, die Farbe zu dekodieren. Jeder der vier Ausgänge stellt eine erkannte Farbe dar und wird aktiv, wenn diese Farbe erkannt wurde. Die Ausgangssignale werden auch verwendet, um die Farbindikator-LEDs in der Nähe der Kassetten zum Leuchten zu bringen, sodass sofort nach dem Verlassen der Sensorschleuse erkennbar ist, in welche Kassette der Stein geschoben werden muss.

Jedes Ausgangssignal dient zugleich als Eingang eines AND-Gatters. Den anderen Eingang des AND-Gatters bildet immer das Signal einer Lichtschranke entlang des Förderbands.

Darüber wird das Pneumatikventil der entsprechenden Rutsche aktiviert, um das Objekt in den richtigen Behälter zu schieben. Der Erfassungszeitpunkt kurz vor einem Rutschen und die tatsächliche Rutschbewegung hängen leicht von der Geschwindigkeit des Förderbandes ab. Dazu ist eine einstellbare Verzögerung vorgesehen. Im (N)AND-Gatter des Zauberlings (Programmauswahl 2) ist dafür das rechte Potentiometer (SET 2) reserviert. Die Zeit, die der Ausgang des AND-Gatters den Eingängen nacheilt, kann zwischen 0 und 500 Millisekunden eingestellt werden.

Aufgrund der Programmierbarkeit des Zauberlings war die Steuerung relativ einfach. Ein Zauberling ist für die Farberkennung über den am I<sup>2</sup>C-Bus angeschlossenen Farbsensor zuständig. Diese Funktionalität kann durch Auskommentieren von `#define FUNC13_COLORENSOR` in der [Standard-Zauberling-Software](#) unter DIP-Schalterstellung 13 programmiert werden. Eingang 1 des Zauberlings kann ohne zusätzliche externe Beschaltung auf HIGH gesetzt werden, indem die Eingangskennlinie auf  $\Omega$  (DIP-Schalter nach oben) gestellt wird. Das Modul behält dann die zuletzt erkannte Farbe bei. Andernfalls erscheint die aktuell erkannte Farbe nur kurzzeitig am Ausgang. Jeder Ausgang repräsentiert eine andere Farbe: Q1 wird aktiv für Rot, Q2 für Grün, Q3 für Blau und Q4 für Gelb.

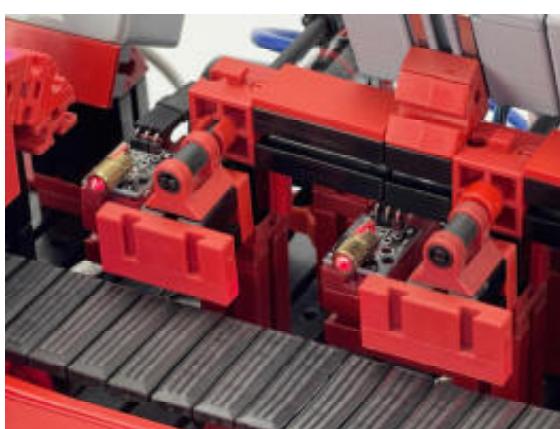


Abb. 6: Laser-Lichtschranken

Außerdem sind vier AND-Gatter mit zwei Eingängen erforderlich. Dazu werden zwei zusätzliche Zauberlinge verwendet, die auf Programm 2 eingestellt sind. Dafür könnten auch die normalen Silberlinge verwendet werden. Wie erwähnt ist diese Gate-Funktion aber beim Zauberling bereits mit einer einstellbaren „Verzögerung“ des Ausgangs ausgestattet. Dadurch entfällt die Notwendigkeit eines zusätzlichen Monoflops, um die Schieberbewegung zu verlangsamen, wenn sich das Band nicht schnell genug bewegt. Eventuell lässt sich das Timing auch mechanisch dadurch lösen, dass man die Laser noch näher an den Slidern platziert oder schräg über das Band strahlt.

### RGB-Farberkennung

Mit Infrarotfilter und echter RGB-Messung sind die Farbmessungen des TCS3472 viel zuverlässiger als die des Original-Farbsensors von fischertechnik. Die Messung ist viel unempfindlicher gegenüber Umgebungslicht und nicht auf die traditionellen weißen, roten und blauen Farbscheiben beschränkt. Mit dem originalen Reflexlightsensor wäre es wahrscheinlich schwierig gewesen, die Farbe Grün zuverlässig zu erkennen, während der TCS3472 sogar zwischen Gelb und Weiß gut unterscheiden kann.

Jede Messung erzeugt vier Werte, die man sich als RGB-Vektor einer bestimmten Länge (die „Clear Light“-Messung) vorstellen kann. Es ist leicht einzusehen, dass dies eine zuverlässigere Farbbestimmung ermöglicht, als anhand von Schwellenwerten eines einzelnen Reflexionslichtmesswerts.

Mit dem TCS3472 ist es möglich, mehr als die traditionellen drei (oder wie beim verbesserten Modell vier) Farben zu unterscheiden. Beispielsweise könnten wir einfach Objekte in einer fünften Farbe (z. B. Weiß oder Silber) in einem Behälter am Ende des Bandes ablegen.

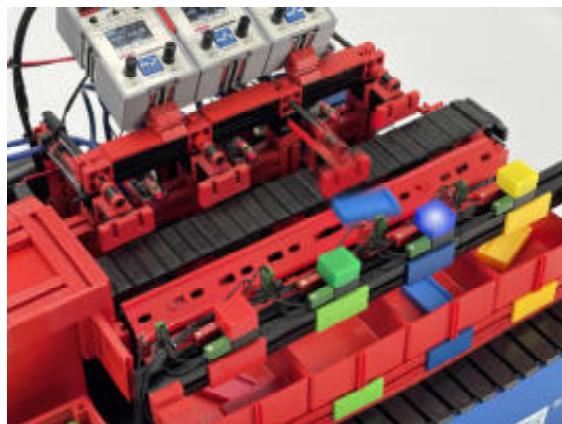


Abb. 7: Erkennung blauer Domino-Steine

In meinem Fall hätte dies eine Nachbearbeitung der von mir verwendeten Dominosteine erforderlich gemacht; diese haben nur vier verschiedene Farben. Um das Prinzip zu demonstrieren schienen mir vier Farben ausreichend.

## Kontinuierliche Messung

Ein weiterer großer Vorteil der Farbmessung mit dem TCS3472 ist, dass kein Ende des Messzyklus festgelegt werden muss. Allerdings wird dieser Vorteil im aktuellen Modell nicht optimal genutzt. Wie beim originalen fischertechnik-Modell geht die Information verloren, wenn mehrere Objekte dicht hintereinander auf das Förderband gelegt werden. Daher kann das Förderband immer nur ein farbiges Objekt auf einmal sortieren.

Die Geschwindigkeit des Systems könnte stark erhöht werden, indem die Farbe des nächsten Objekts bestimmt wird, bevor das letzte sein gewünschtes Farbsortierfach erreicht hat. Im aktuellen Aufbau ist das nicht möglich, da die Auswurfschieber auf die zuletzt erkannte Farbe reagieren und diese so lange unverändert bleiben muss, bis der entsprechende Laserdetektor entlang der Spur signalisiert, dass das Objekt von Förderband geschoben werden kann.

Dies könnte mit einem „Pufferspeicher“ für die Reihenfolge und den Abstand zwischen den Farbobjekten auf dem Förderband

gelöst werden. Voraussetzung wäre dafür eine genauere Messung des vom Förderband zurückgelegten Weges. Das originale fischertechnik-Modell nutzt bereits einen Impulsgeber, um den Abstand zum richtigen Auswurfschieber zu ermitteln; dies allein lässt jedoch keine Gleichzeitigkeit am Band zu. Es ist sicherlich eine spannende Herausforderung, diesen Prozess weiter zu optimieren, beispielsweise durch den Einsatz eines Encoders oder Schrittmotors. Wer weiß, vielleicht wird dieses Projekt in Zukunft noch fortgesetzt...

Das aktuelle Modell ist in einem [Film auf YouTube](#) in Aktion zu sehen; dort findet ihr auch einen [Film von der originalen fischertechnik-Sortieranlage](#).

## Warnung

Achtung: Im Modell werden rote 5-mW-Laser verwendet. Dies sind die Laserdioden, die z. B. in Laserpointern verwendet werden. Diese Laser sind relativ schwach und die rote Wellenlänge (650 nm) ist am ungefährlichsten, aber auch diese Laser können Augenschäden verursachen. Schauen Sie daher niemals direkt oder durch eine Optik in einen solchen Laser.

Aufgrund der Position im Modell ist ein direkter Blick in einen Laser unter normalen Umständen nicht möglich.

## Referenzen

- [1] Arnoud van Delden: *Der Zauberling (Teil 4): Die Weiterentwicklung*. In dieser Ausgabe der ft:pedia.
- [2] fischertechnik: [Farbsensor](#). FT-T-KN, Datenblatt, 02.08.2017.
- [3] ams: [TCS3472 – Color Light-to-Digital Converter with IR Filter](#). Datasheet, v1.04, 10.04.2020.
- [4] Dirk Fox: *I<sup>2</sup>C mit dem TX(T) – Teil 13: Farbsensor*. [ft:pedia 1/2016](#), S. 79–89.

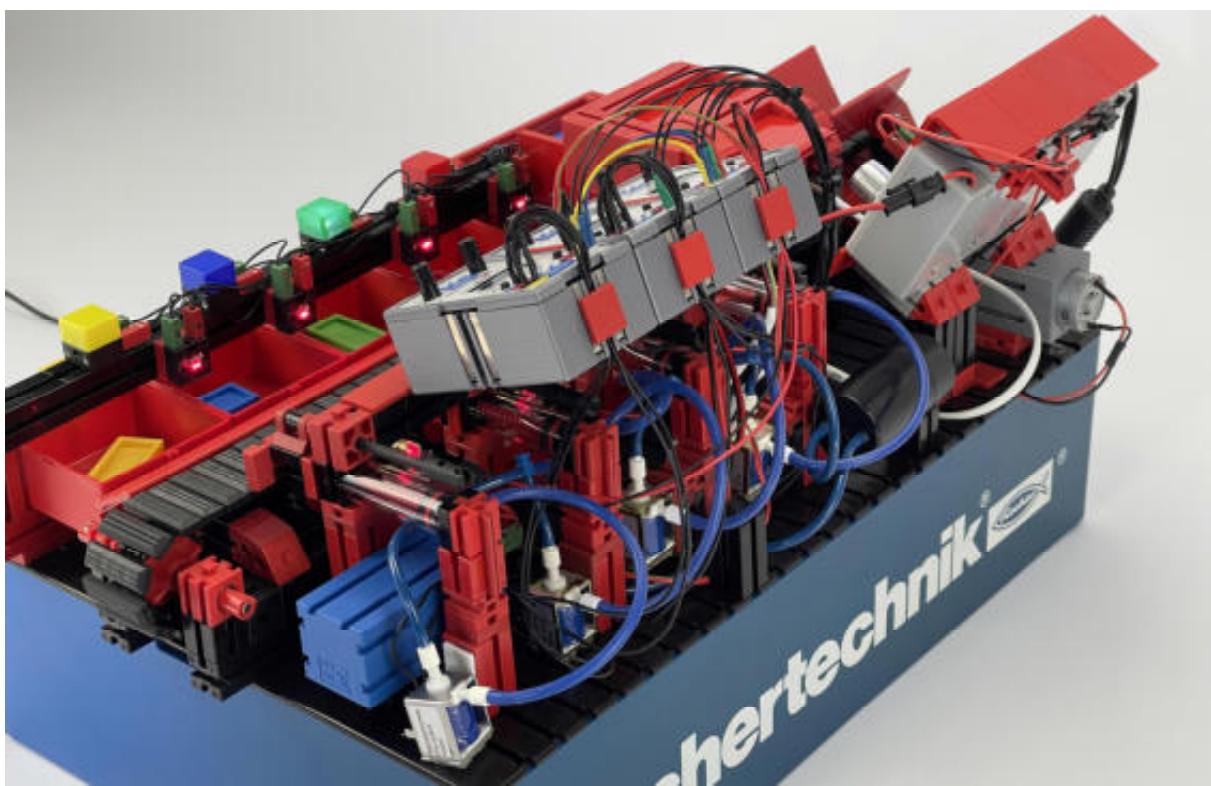
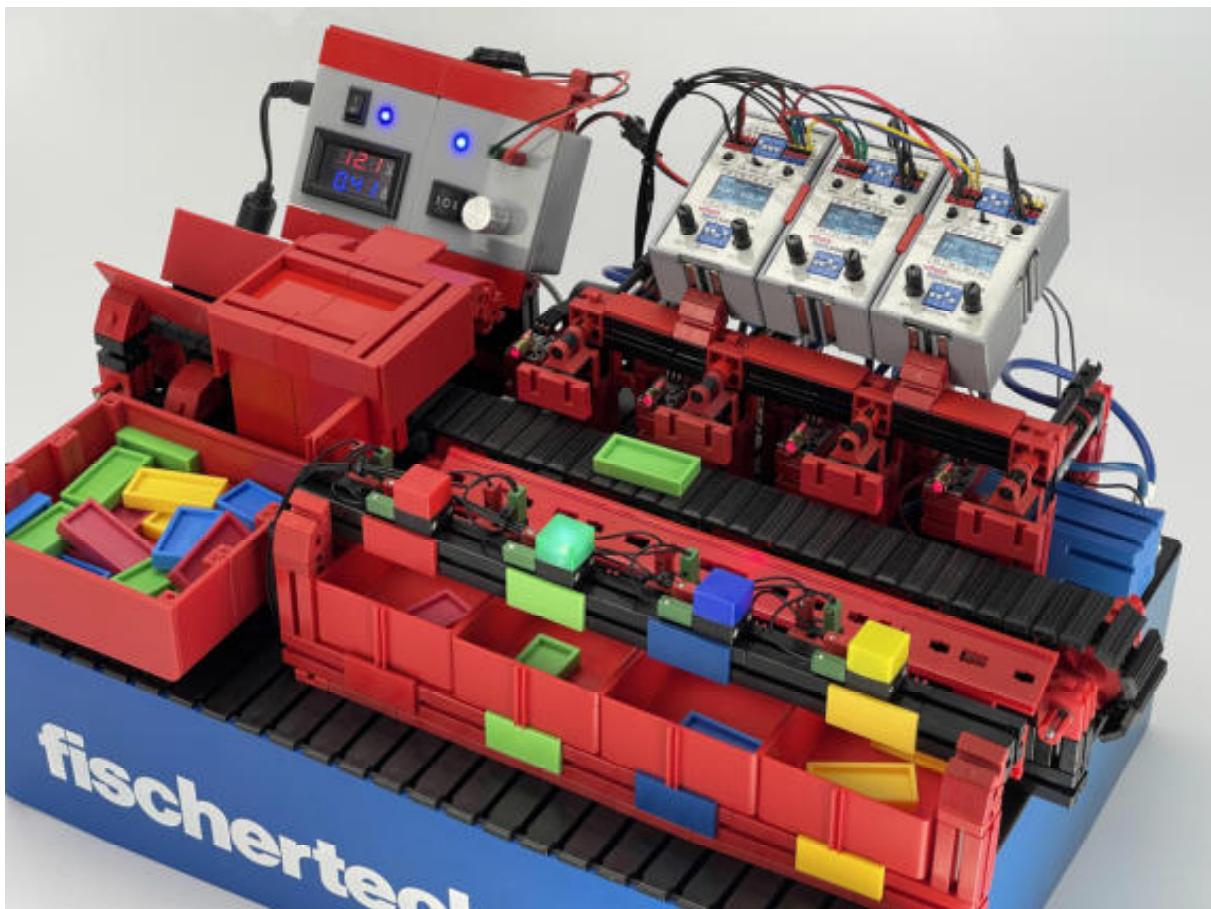


Abb. 8: Gesamtansicht der Farbsortieranlage

Computing

# Scratch mit fischertechnik – Update 2022

Dirk Fox

*Seit der Vorstellung der Scratch-Unterstützung durch fischertechnik in der ft:pedia 1/2018 [1] hat sich einiges in der „Scratch-Welt“ verändert und funktioniert heute anders als damals von mir beschrieben. Daher ist ein Update überfällig – denn Scratch ist nach wie vor eine verbreitete und für Einsteiger und Schulen sehr attraktive Entwicklungsumgebung.*

## Hintergrund

Visuelle oder grafische Programmiersprachen wie [Scratch](#) (MIT), [NEPO](#) (Fraunhofer), [Blockly](#) (Google) oder [Ardublock](#) (Arduino) erfreuen sich ungebrochener Beliebtheit. Daher entwickelte auch fischertechnik für die Programmierung des TXT 4.0 Controller mit [ROBO Pro Coding](#) eine (auf Blockly basierende) grafische Programmiersprache.

Grafische Programmierumgebungen erzeugen in der Regel Programmcode für eine andere Programmiersprache, wie beispielsweise Javascript oder Python, der dann von einem Interpreter bzw. dem Browser selbst ausgeführt wird.

**Vorteil:** Die Anpassung der grafischen Programmierumgebung an ein bestimmtes Betriebssystem oder einen Mikrocontroller verursacht (verglichen mit der Entwicklung einer kompletten IDE) relativ wenig Aufwand.

**Nachteil:** Mit grafischen Programmiersprachen lassen sich in der Regel keine zeitkritischen Anwendungen programmieren.

Dafür ist die Lernkurve bei Programmieransteigern, vor allem bei Kindern, sehr steil – schon nach einer kurzen Einführung gelingen erste funktionierende Anwendungen. Der Wechsel zu einer anderen Program-

miersprache mit Echtzeitfähigkeit und ausgereifter Entwicklungsumgebung ist später nur noch ein kleiner Schritt.

## Scratch

Warum aber gerade Scratch – und nicht eine der anderen grafischen Programmiersprachen?

Scratch wurde von einer Gruppe von Forschern (der „[Lifelong Kindergarten Group](#)“) um Seymour Papert (1928-2016) unter der Leitung von Mitchel Resnick am Media Lab des MIT entwickelt. Seymour Papert wird älteren fischertechnikern als Entwickler der Programmiersprache LOGO (aus dem Jahr 1968!) und durch sein Buch „Mindstorms“ (1980) bekannt sein. Hinter LOGO steckt das von Papert entwickelte didaktische Konzept des „Konstruktionismus“: In der Tradition des Kognitionswissenschaftlers Jean Piaget (1896-1980), mit dem er in den 50er Jahren in Genf zusammenarbeitet hatte, verstand er Lernen als „Rekonstruktion“ und nicht als Transfer von Wissen.

*“From constructivist theories of psychology we take a view of learning as a reconstruction rather than as a transmission of knowledge.”*

Daraus leitete er ab, dass Lernen am besten mit Lernmaterialien gelingt, die dieses Rekonstruieren unterstützen:

*“Then we extend the idea of manipulative materials to the idea that learning is most effective when part of an activity the learner experiences as constructing a meaningful product.”*

Dieses Konzept steckte bereits in LOGO. Die Programmierung der „Turtle“ erfolgte wie eine Bewegungsbeschreibung für einen Menschen und zielte auf die Veranschaulichung geometrischer Zusammenhänge.

LOGO steckt auch hinter den Bewegungsbefehlen für Objekte auf der ‚Bühne‘ in Scratch. Auch alle anderen Programmelemente wurden so gestaltet, dass sie möglichst intuitiv verstanden werden [2]: Zusammengehörige Befehle sind in Gruppen strukturiert, die an der Farbe erkannt werden können. Da sich nur zueinander passende Blöcke und Elemente kombinieren lassen, sind Syntaxfehler praktisch ausgeschlossen. Auch um Datentypen muss man sich nicht kümmern: Variablen werden erst zur Laufzeit dynamisch typisiert. Scratch ist zudem „always on“: Programme („Skripte“ genannt) können während der Laufzeit geändert werden; klickt man auf einen einzelnen Block (außerhalb eines Skripts),

wird dieser sofort ausgeführt. Daher erhält man ein unmittelbares Feedback: kein Compilerlauf oder Startprozess verzögert die Rückmeldung.

Skripte werden nach einem ausgeklügelten Multithreading-Konzept ausgeführt und ausnahmslos ereignisorientiert gestartet, was auch die Programmierung paralleler Abläufe äußerst intuitiv macht. Statt Unterprogrammen definiert man sich eigene Blöcke.

Daher ist Scratch vor allem als Einsteiger-Programmiersprache geeignet – und wurde seit der Veröffentlichung von Version 1.0 im Jahr 2007 viele Millionen Mal von der Webseite des MIT heruntergeladen. Inzwischen wird Scratch weltweit in vielen Schulen eingesetzt, wie Abb. 1 zeigt: Die Erfolgskurve von Scratch in Google Trends hat seit 2014 regelmäßig „Einbrüche“ im Juli/August – während der Sommerferien.

Andere grafische Programmierumgebungen (wie Blockly, NEPO oder Snap!) fristen im Vergleich mit Scratch ein nicht erwähnenswertes Nischendasein.

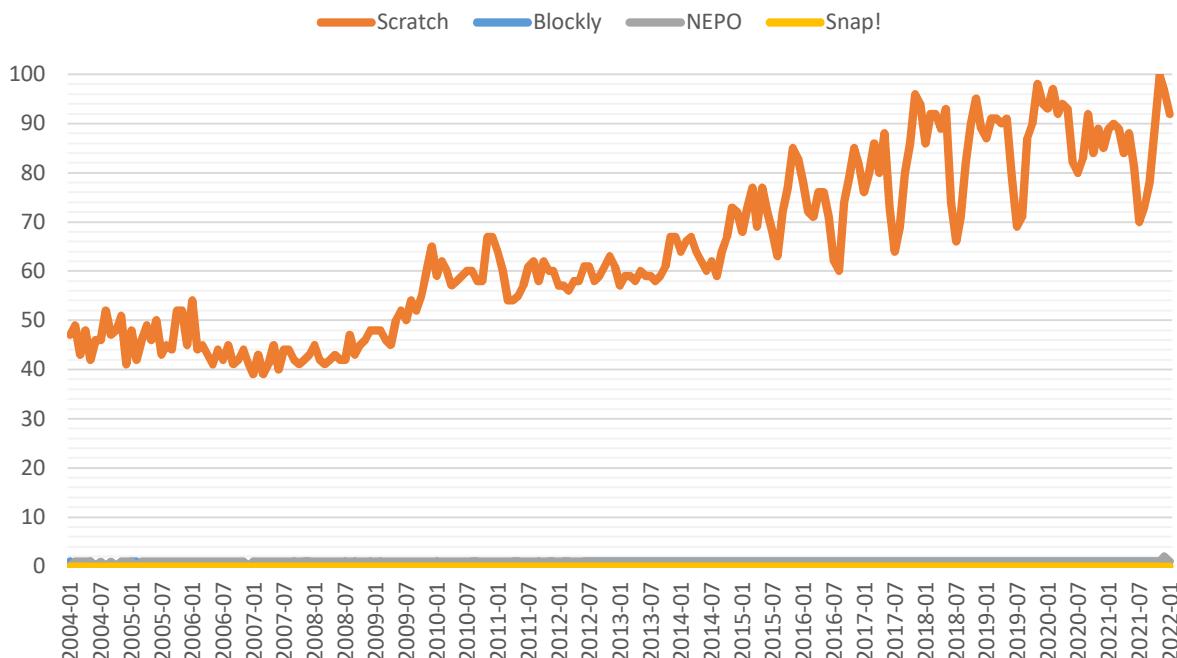


Abb. 1: Google Trends: Scratch im Vergleich mit anderen grafischen Programmiersprachen

Scratch kann sowohl [online im Browser genutzt](#) als auch [als App heruntergeladen](#) werden und stellt keine besonderen Anforderungen an die verwendete Hardware. Mit der am 2. Januar 2019 eingeführten Scratch-Version 3.0 (aktuell: v3.29) wurde Flash durch HTML 5 ersetzt: Die Programmelemente werden nun direkt in Javascript-Code umgewandelt, den der Browser ausführt.

## Scratch mit fischertechnik

fischertechnik unterstützt Scratch bereits seit Ende 2016 – zunächst für den TXT und den LT Controller [1, 3]. Wer heute für den TXT oder den BT Smart Controller einen Programmereinstieg mit einer grafischen Programmierumgebung sucht, für den führt ohnehin kein Weg an Scratch vorbei – andere grafische Programmierumgebungen werden nur unter der Community-Firmware unterstützt (Blockly/Brickly) [3, 4]. Die Nutzung des [LT Controllers](#) ist allerdings mit der aktuellen fischertechnik-Scratch-Version 3 nicht mehr möglich.

Die Ansprache des TXT und des BT Smart Controllers aus Scratch erfolgt jeweils über ein kleines „Gateway“-Programm, das die Auswahl der Schnittstelle zum Controller (USB, Bluetooth, WLAN) ermöglicht und (ähnlich dem Interface-Test in ROBO Pro) die an den Ein- und Ausgängen anliegenden Werte anzeigt. Die Gateways wurden, wie auch einige der fischertechnik-Apps, von der Informatik-Fakultät der Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt (FHWS) entwickelt.

### \$txt Controller

Das Gateway für den Robotics TXT Controller ([153513](#)), [FTScratchTXT](#), erhielt ein Update – von Version v1.23, die ich 2018 vorgestellt habe, auf v1.25 (Abb. 2). Es unterstützt vier verschiedene Verbindungsprotokolle: USB, Bluetooth und WLAN mit dem TXT als Client oder als Access Point. Zuvor müssen unter Windows die fischertechnik-USB-Treiber installiert werden.

Unter Linux und iOS läuft das Gateway unter [Mono](#) [5].

Im Ausgabefenster werden nach einem Connect mit einem TXT die Firmware-Version und die an den (acht) Eingängen anliegenden Spannungen bzw. Widerstandswerte angezeigt. Außerdem kann man den Stand der schnellen Zählereingänge C1 bis C4 ablesen (Abb. 2).

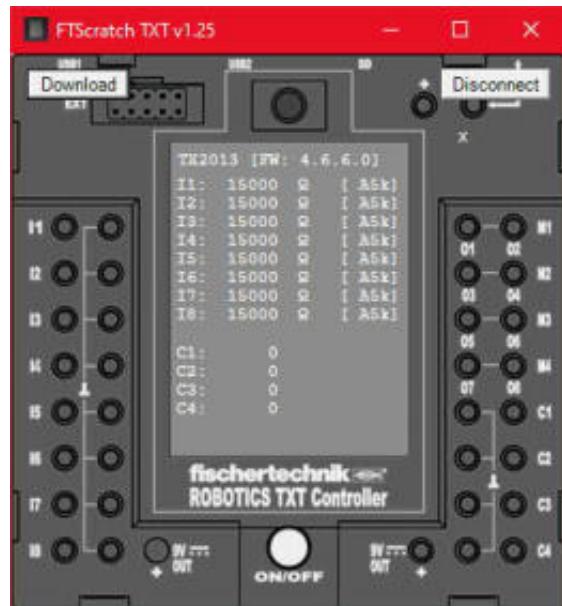


Abb. 2: Scratch-„Gateway“ zum TXT Controller (v1.25)

Für mobile Modelle ist die Verwendung einer WLAN-Verbindung zu empfehlen. Sie ist (insbesondere, wenn mehrere Controller in unmittelbarer Nähe betrieben werden) stabiler als ein Bluetooth-Connect, benötigt allerdings auch deutlich mehr Leistung, wodurch beim Akkubetrieb die Laufzeit spürbar sinkt.

### BT Smart Controller

Neben dem Robotics TXT Controller kann auch der 2017 eingeführte BT Smart Controller ([161944](#)) mit Scratch programmiert werden. Das Gateway [FTScratchBTSmart](#) wurde im April 2018, kurz nach der Veröffentlichung meines ft:pedia-Beitrags zu Scratch [1], auf [github.io](#) veröffentlicht (Abb. 3).

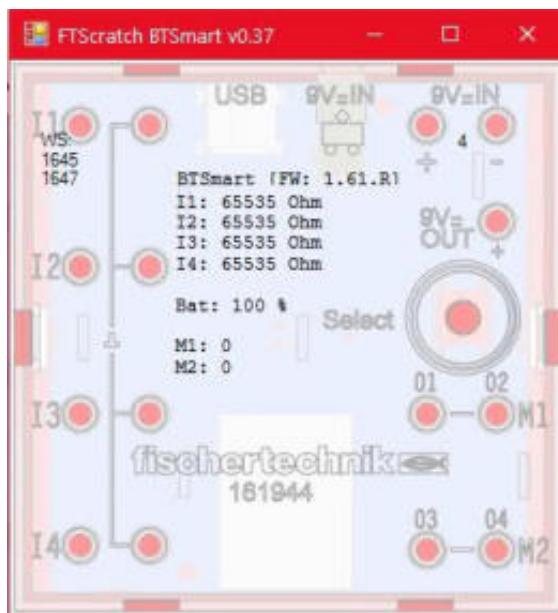


Abb. 3: Scratch-„Gateway“ zum BT Smart Controller (v0.37)

Es stellt unter Windows oder Linux die Verbindung zwischen Scratch und dem BT Smart Controller her, wahlweise über eine USB- oder eine Bluetooth-Verbindung. An der Monitor-Funktion, die – ähnlich dem Interface-Test in ROBO Pro – die an den Ein- und Ausgängen anliegenden Spannungen resp. Widerstandswerte anzeigt (Abb.

3), erkennt man bereits, dass die Eingänge am BT Smart Controller keineswegs nur digital (wie in ROBO Pro Smart oder ROBO Pro Light), sondern auch analog ausgewertet werden können – mehr dazu weiter unten.

Auch hier müssen die Windows-fischertechnik-USB-Treiber zuvor installiert sein. Unter Linux wird das Gateway ebenfalls unter [Mono](#) betrieben [6].

### **ftDuino**

Der ftDuino [7] lässt sich auch mit Scratch programmieren. Dafür ist kein Gateway erforderlich – es genügen die USB-Treiber des Arduino. Eine Bluetooth- oder WLAN-Verbindung bietet der ftDuino nicht.

### **ftScratch3**

Für Scratch 2 wurden die Scratch-Erweiterungen für die fischertechnik-Controller in der Entwicklerversion ScratchX zur Verfügung gestellt [1]. Der ftDuino ließ sich mit S4A (Scratch for Arduino) nutzen; dafür musste lediglich ein spezieller Arduino-Sketch auf dem ftDuino installiert werden [8].

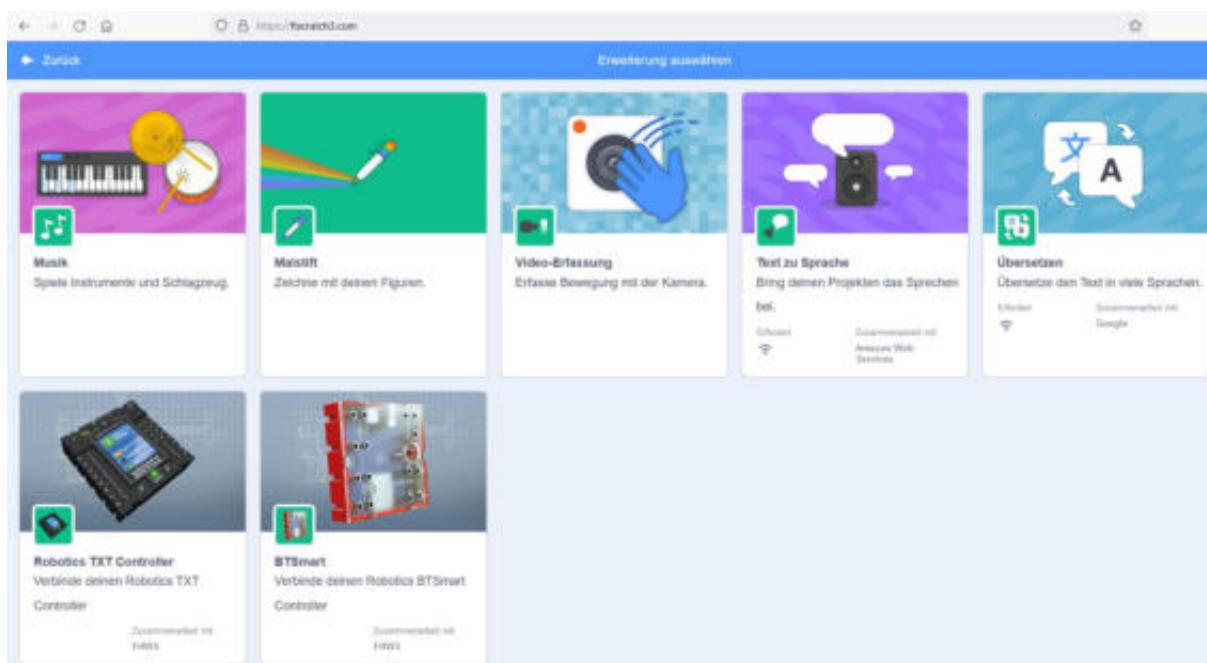


Abb. 4: Erweiterungen für TXT und BTSmart Controller in ftScratch3

Kurz bevor ScratchX aufgrund der Abkündigung des Adobe Flashplayers nicht mehr einsetzbar war, ließ fischertechnik einen „Fork“ von Scratch 3 zu „ftScratch3“ weiterentwickeln, der die erforderlichen Erweiterungen für den TXT und den BT Smart Controller enthält. Der LT Controller wird in ftScratch3 nicht mehr unterstützt.

ftScratch3 ist seit Anfang 2019 unter der Adresse [ftscratch3.com](http://ftscratch3.com) online verfügbar. Zwar sind einige Buttons auf der Startseite ausgegraut und zeigen seit der Veröffentlichung beim Mouseover Kommentare wie „Kommt bald“ oder „Wir arbeiten dran“, aber alle wesentlichen Funktionen werden von der IDE unterstützt.

## ftScratch3-Erweiterungen

ftScratch3 enthält die Befehlsblöcke für den TXT und den BT Smart Controller als Erweiterung (Abb. 4).

### **TXT Controller**

Die Blöcke der Extension für den TXT Controller bieten zur Auswertung der Sensor-Eingänge dieselben Möglichkeiten wie ROBO Pro: der Abstandssensor und alle digitalen und analogen (Widerstand und Spannung) fischertechnik-Sensoren werden darin unterstützt.



Abb. 5: Befehlsblöcke für den TXT Controller

Die Befehlsblöcke (Abb. 5) entsprechen bis auf die geänderte Farbe und Form denen der ScratchX-Erweiterung, die ich 2018 in [1]

vorgestellt habe. Die Befehlsblöcke für die Ausgänge (Aktoren) kennen nicht nur Motoren und Lampen, E-Magnet und E-Ventil, sondern auch Distanz-Kommandos für die Encoder-Motoren. Außerdem können auf dem TXT gespeicherte Sounds abgespielt werden.

Sogar die schnellen Zählereingänge C1 bis C4 können mit ftScratch ausgelesen werden – das funktioniert mit ROBO Pro nicht. Was fehlt ist eine Möglichkeit zur Nutzung angeschlossener I<sup>2</sup>C-Sensoren, und auch die sehr leistungsfähige Bildauswertung von ROBO Pro vermisst man schmerzlich.

### **BT Smart Controller**

Der BT Smart Controller bietet deutlich weniger Möglichkeiten als der TXT. Allerdings lassen sich mit ftScratch auch da sowohl analoge als auch digitale fischertechnik-Sensoren anschließen und auswerten (Abb. 6) – unter ROBO Pro Light/Smart sind die Eingänge nur digital nutzbar. Damit eröffnet sich eine deutlich größere Vielfalt an Programmiermöglichkeiten.

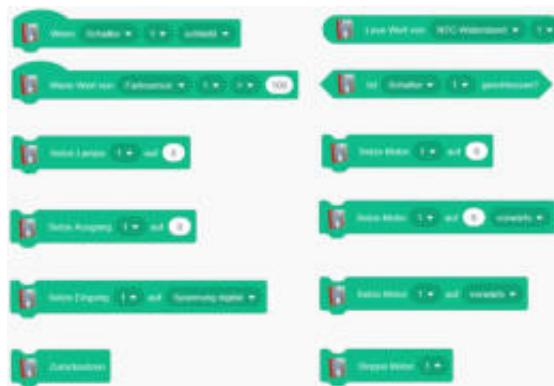


Abb. 6: Befehlsblöcke für den BT Smart Controller

Besonders der BT Smart Controller profitiert von den Standard-Erweiterungen in Scratch 3 (siehe nächster Abschnitt), die die Programmiermöglichkeiten z. B. durch die Nutzung einer am PC angeschlossenen (fischertechnik-) Kamera mit Mikrofon erheblich erweitern und weit über die Möglichkeiten von ROBO Pro Light/Smart hinausgehen.

## ftDuino

Für die Nutzung des ftDuino mit Scratch 3 hat Till Harbaum einen eigenen „Fork“ erzeugt. Er ist erreichbar über [Github](#). Darin gibt es eine ftDuino-Erweiterung (Abb. 7).



Abb. 7: Scratch-3-Extension „ftDuino“

Die neunen Scratch-Blöcke können die acht Eingänge auswerten, Motoren ansteuern und ebenfalls die schnellen Zähler auslesen (oder zurücksetzen), siehe Abb. 9.

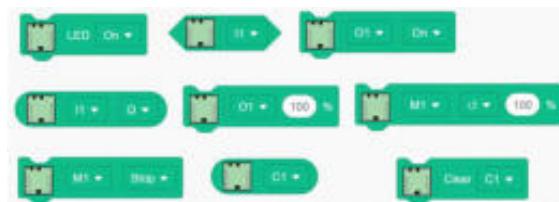


Abb. 9: ftDuino-Scratch-Blöcke

Damit der ftDuino aus dem Browser angesprochen werden kann, muss auf dem ftDuino der IoServer-Sketch (erreichbar aus der Arduino-IDE über „Datei > Beispiele > WebUSB > IoServer“) gestartet werden.

Der ftDuino kann dann via USB-Kabel mit einem PC oder einem Smartphone verbunden werden. Den Verbindungsstatus zeigt Scratch mit einem Icon oben rechts in der Statuszeile an [8].

Scratch 3 kann mit dem ftDuino auch offline genutzt werden. Dafür hat Till Harbaum in Github ein [Zip-File](#) bereitgestellt, mit dem Scratch und ein lokaler Webbrowser unter Windows gestartet werden kann. Im Browser ist Scratch 3 dann über [localhost](#) erreichbar.

## Scratch-Erweiterungen

Scratch 3 bringt zahlreiche eigene Funktionen und fünf Erweiterungen mit, die auch in ftScratch zur Verfügung stehen und mit denen sich Scatch-Skripte (und damit die zugehörigen fischertechnik-Modelle) gänzlich unabhängig von den Möglichkeiten des verwendeten Controllers erheblich aufwerten lassen.

### Klang

Natürlich kann Scratch Sounds ausgeben – und ist mit einer umfangreichen Sound-Bibliothek ausgestattet, die die des TXT weit übertrifft. Die Klang-Befehle von Scratch finden sich direkt im Scratch-Befehlsmenu. Sie erlauben nicht nur das Abspielen bereits existierender Sounddateien, sondern bieten außerdem eine sehr einfache, bedienungsfreundliche Möglichkeit, Sprache oder Geräusche über das Mikrofon aufzuzeichnen und später abzuspielen. Das geht deutlich über die Möglichkeiten von ROBO Pro hinaus.

### Musik

Scratch kann nicht nur Sound-Dateien abspielen, sondern erlaubt auch das „Komponieren“ von Musikstücken. So können Töne nach einem vorgegebenen Takt erzeugt und mit einem von 22 Instrumenten ausgegeben werden – oder von einem von 18 verschiedenen Schlaginstrumenten, von der Kuhglocke bis zur Triangel.

### Sprachausgabe

Mit der Erweiterung „Text zu Sprache“ lernt Scratch außerdem sprechen – in 23 verschiedenen Sprachen und vier unterschiedlichen Stimmlagen. Es genügt, dem Block einen Text zu übergeben und Scratch die Nutzung der Lautsprecher zu erlauben.

### Übersetzen

Bindet man außerdem die Erweiterung „Übersetzen“ ein, kann ein gegebener Text zuvor in eine vorgegebene Sprache übersetzt werden.

## Geräuschsteuerung

Ist der Zugriff auf das Mikrofon gestattet, kann die Lautstärke (in db) gemessen werden – auch mit einer am PC angeschlossenen (fischertechnik-) Kamera. Auf Wunsch zeigt Scratch die gemessene Lautstärke in einem kleinen Fenster auf der Bühne an.

## Zeit und Datum

Datums- und Zeitangaben können ebenfalls ausgelesen, genutzt und in einem Fenster auf der Bühne angezeigt werden. Damit kann auch eine Stoppuhr (oder ein Timer) gestartet werden, die die seit Programmbeginn oder einem Ereignis verflossene Zeit in Millisekunden misst.

## Maus und Tastatur

Schließlich stehen alle Tasten der Tastatur und die Bewegungen der Maus als Eingabe-einheit zur Verfügung und können in Scratch ausgewertet werden. Damit ist im Handumdrehen eine Fernsteuerung programmiert – mit den Pfeiltasten oder der Maus als Steuerkonsole für den via Bluetooth angebundenen BT Smart oder via WLAN gekoppelten TXT Controller.

## Kamera

Zwar wird eine an den TXT angeschlossene Kamera von Scratch nicht unterstützt, dafür bringt Scratch 3 eine eigene Kamera-Erweiterung mit. Wird sie eingebunden und gibt man den Zugriff auf die Kamera im Browser (oder in der App) frei, dann kann man das Kamerabild mit einstellbarer Transparenz als Bühnenhintergrund einblenden.

Die in Scratch integrierten Kamerafunktionen können Bewegungen erkennen und deren Richtung bestimmen. Damit lässt sich beispielsweise ein Bewegungsmelder oder auch eine Gestensteuerung programmieren – sowohl für den TXT als auch für den BT Smart Controller.

## Wünsche

Ein paar Dinge fehlen ftScratch3 noch, um Scratch zu einer perfekten IDE für die beiden fischertechnik-Controller zu machen. So können derzeit nur zwei Motoren synchron betrieben werden – für ein Fahrzeug mit Mecanum-Wheels benötigt man aber eine Synchronisation aller vier Motoren. Auch der Zugriff auf am TXT via I2C-Bus angeschlossene Aktoren und Sensoren würde die Möglichkeiten von ftScratch noch einmal erheblich erweitern.

Derzeit ist Torsten Stuehn dabei, mit dem [ftrobopy\\_server](#) eine Anbindung des TXT 4.0 an ftScratch3 zu ermöglichen – noch im Alpha-Stadium, aber bereits lauffähig.

## Referenzen

- [1] Dirk Fox: *Scratch mit fischertechnik*. [ft:pedia 1/2018](#), S. 69–78.
- [2] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, Evelyn Eastmond: [\*The Scratch Programming Language and Environment\*](#). ACM Transactions on Computing Education, Vol. 10, No. 4, November 2010.
- [3] Till Harbaum: *Von Lucky Logic zu RoboPro Coding*. [ft:pedia 1/2021](#), S. 103–109.
- [4] Till Harbaum: *Brickly auf dem TXT: Grafische Programmierung à la Google-Blockly*. [ft:pedia 1/2017](#), S. 92–98.
- [5] fischertechnik: [\*FTScratchTXT\*](#).
- [6] fischertechnik: [\*FTScratchBT\*](#).
- [7] Till Harbaum: *ftDuino – Open Source trifft Konstruktionsbaukasten*. [ft:pedia 1/2018](#), S. 85–91.
- [8] Till Harbaum: [\*ftDuino – ein fischertechnik-kompatibler Arduino. Bedienungsanleitung\*](#). 12.01.2021.

Computing

## Einführung in ftScratch (1): Die Schranke

Dirk Fox

Scratch ist eine für Einsteiger und Schulen besonders geeignete Programmiersprache. Die Entwicklungsumgebung ftScratch3 unterstützt mit entsprechenden Erweiterungen die fischertechnik-Controller TXT und BT Smart [1]. In dieser Serie führen wir in die Programmierung mit ftScratch ein – mit kleinen Modellen und Aufgaben.

### Die Knickschranke

Türen, Tore und Schranken finden sich in sehr vielen fischertechnik-Anleitungen bis zurück in die 70er Jahre. Sie eignen sich besonders gut für eine einführende Programmieraufgabe, denn

- sie gehören zu unserem Alltag,
- sie nehmen nur wenige definierte Zustände ein (offen, geschlossen),

- sie können mit verschiedenen Sensoren ausgestattet werden (Taster, Fotodiode, Kamera, Reed-Kontakt),
- sie können als ein Teilelement einer komplexeren Aufgabenstellung (z. B. einer Parkraumverwaltung) verstanden werden.

Für die folgenden Aufgaben eignet sich z. B. die in Abb. 1 gezeigte Knickschranke. Eine fischertechnik-Designer-Datei der Schranke findet sich im [Download-Bereich der ft:pedia](#) zu dieser Ausgabe.

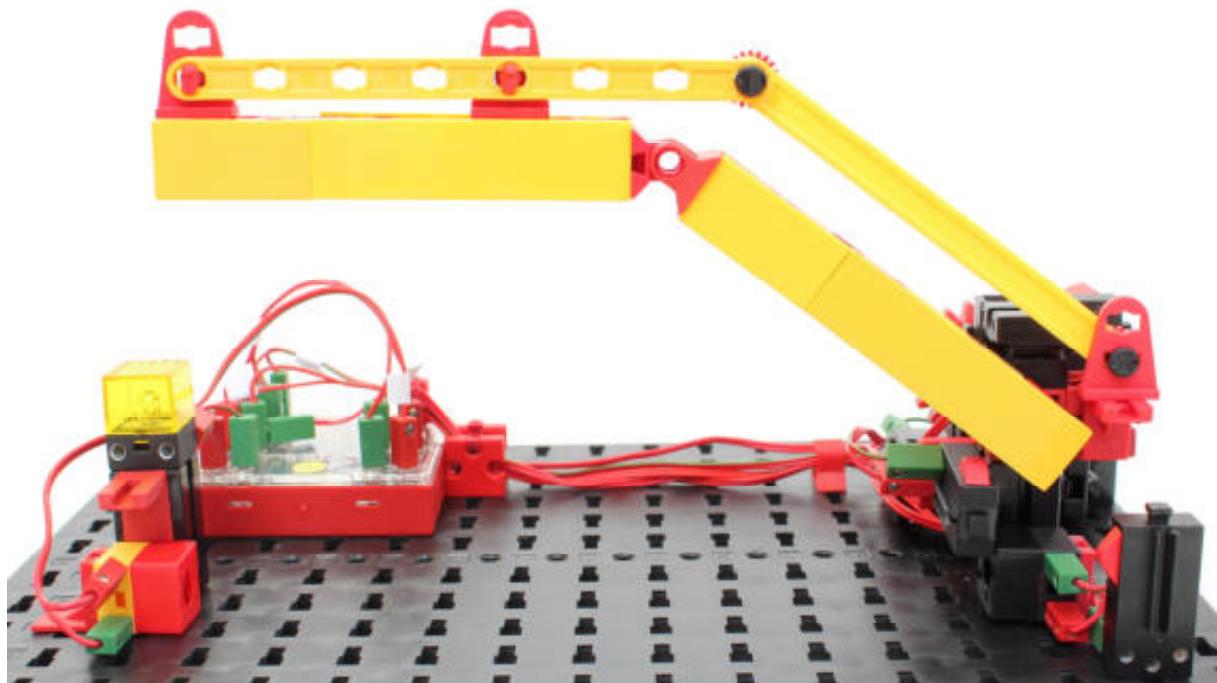


Abb. 1: Knickschranke mit Endlagentastern, Lichtschranke, Bedarfstaster und Blinklicht

## Sensoren und Aktoren

Die folgenden Aufgaben können sowohl mit dem TXT als auch mit dem BT Smart Controller gelöst werden.

Für die Aufgaben benötigen wir

- drei Taster
- eine Lichtschranke (Fototransistor mit Linsen-LED)
- ein Blinklicht und
- eine USB-Kamera.

Die Sensoren und Aktoren werden wie folgt angeschlossen (bei den Dioden Polarität beachten):

- I1: Endlagentaster (Schranke) oben; ist der Taster gedrückt, soll der Stromkreis unterbrochen sein
- I2: Endlagentaster (Schranke) unten
- I3: Bedarfstaster
- I4: Fotodiode (Lichtschranke)
- M1: Schrankenmotor
- M2: Blinklicht
- 9V/GND: Linsen-LED (Lichtschranke)

Die Anschlüsse des Motors an M1 sollten so gepolt sein, dass die Schranke öffnet, wenn sich der Motor „rückwärts“ dreht.

## Aufgaben

### Aufgabe 1

Schreibe zunächst ein einfaches ftScratch-Skript, das die Schranke öffnet, zwei Sekunden offen hält und dann wieder schließt.

Wähle eine geeignete Geschwindigkeit für den Motor.

### Aufgabe 2

Nun soll die Schranke jedes Mal öffnen, wenn der Bedarfstaster rechts vor der Schranke gedrückt wurde. Das Öffnen soll

eine Sekunde nach dem Drücken des Tasters erfolgen.

### Aufgabe 3

Damit die Schranke beim Schließen ein durchfahrendes Fahrzeug nicht beschädigt, soll sie erst schließen, wenn ein Fahrzeug (oder ein Fußgänger) die Schranke passiert hat und mindestens eine weitere Sekunde verstrichen ist.

Das Passieren der Schranke kannst du mit Hilfe der Lichtschranke feststellen.

### Aufgabe 4

Falls Schwierigkeiten mit dem Bedarfstaster auftreten oder die Lichtschranke nicht reagiert, soll ein Wartungseingriff von außen möglich sein. Ergänze dein Skript um die Möglichkeit, die Schranke über die Taste „a“ zu öffnen und mit der Taste „s“ zu schließen.

### Aufgabe 5

Um Passanten vor der Bewegung der Schranke zu warnen, soll das Blinklicht blinken, solange sich die Schranke nicht im geschlossenen Zustand befindet.

*Tipp:* Realisiere das Blinklicht als parallelen Thread.

### Aufgabe 6

Erweitere die Schranke nun um gesprochene Hinweise: Vor dem Öffnen soll der Hinweis „Achtung, Schranke öffnet“ gegeben, nach dem Öffnen soll „Bitte durchfahren“ durchgesagt werden, und kurz bevor sie schließt der Warnhinweis „Achtung, Schranke schließt“ folgen.

### Aufgabe 7

Zusätzlich zum Bedarfstaster kann auch die Kamera erkennen, dass ein Fahrzeug vorfährt. Ergänze den Bedarfstaster durch die Bewegungserkennung der Kamera.

## Lösungsbeispiele

Die folgenden Lösungsbeispiele wurden für den BT Smart Controller programmiert. Die Skripte finden sich im [Download-Bereich zu dieser Ausgabe der ft:pedia](#). Die Anpassung an den TXT (Ersetzung der entsprechenden Blöcke in ftScratch) ist sehr einfach.

### Aufgabe 1

Die Geschwindigkeit des Motors solltest du beim Schließen der Schranke etwas kleiner wählen als beim Öffnen. Zwar ist der Schneckenantrieb ein selbstsperrendes Getriebe, aber dennoch wirkt sich das Gewicht der Schranke auf das Drehmoment aus.

Beachte außerdem, dass das Impulsrad den Endlagentaster erst nach etwa einer Achtelumdrehung der Achse drückt. Daher musst du beim Öffnen und beim Schließen der Schranke zunächst einen Moment warten, bevor du den Status des Tasters auswertest.

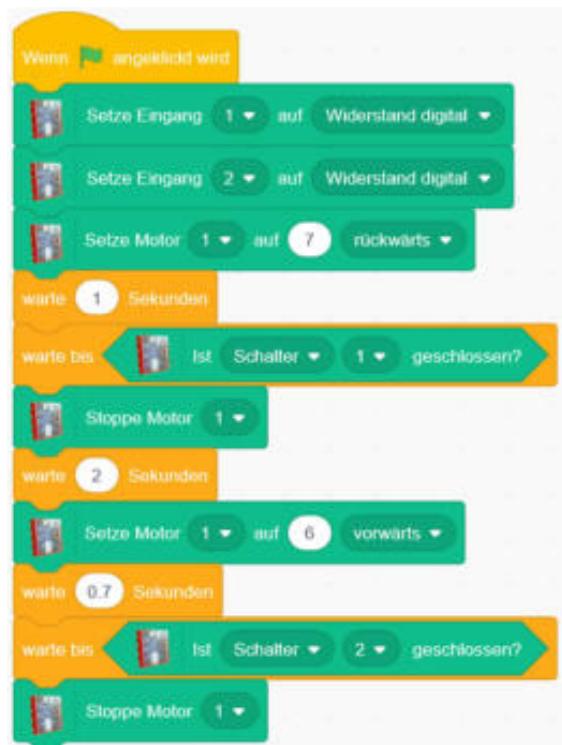


Abb. 2: Öffnen und Schließen der Schranke

### Aufgabe 2

Die Lösung der zweiten Aufgabe erfordert zunächst die Ergänzung einer Dauerschleife, damit die Schranke jedes Mal öffnet, wenn der Bedarfstaster gedrückt wird.

Außerdem muss zu Beginn der Bedarfstaster abgefragt werden. Das kannst du entweder durch einen „Warte“-Befehl oder durch eine Bedingung (wie im Lösungsbeispiel in Abb. 3) verwirklichen.

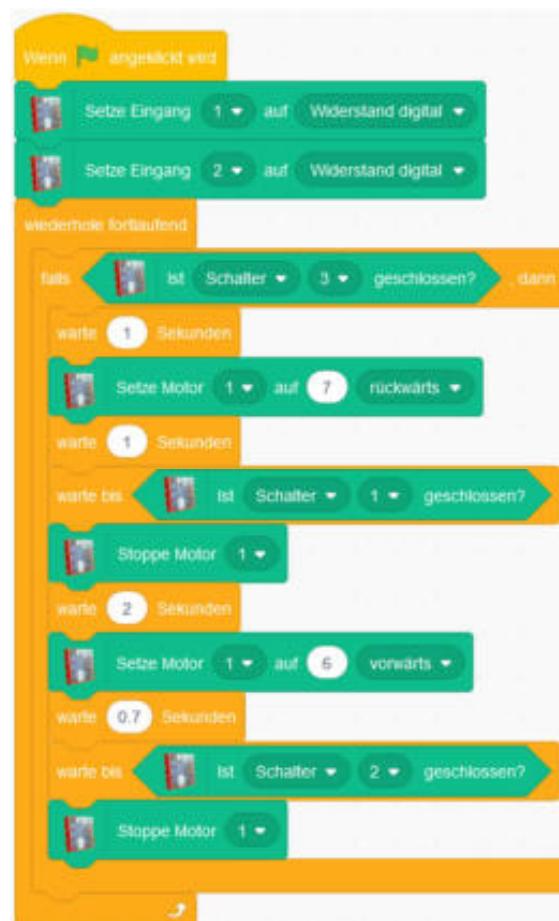


Abb. 3: Öffnen der Schranke auf Anforderung

### Aufgabe 3

Um sicher zu sein, dass sich niemand mehr unter der Schranke befindet, musst du die Lichtschranke zweimal abfragen: Sie wird zunächst unterbrochen, dann musst du warten, bis sie wieder geschlossen ist. Eine Sekunde später kannst du die Schranke schließen.

Zur Abfrage, ob die Lichtschranke unterbrochen ist, benötigen wir den Negationsoperator „nicht“ (Abb. 4).

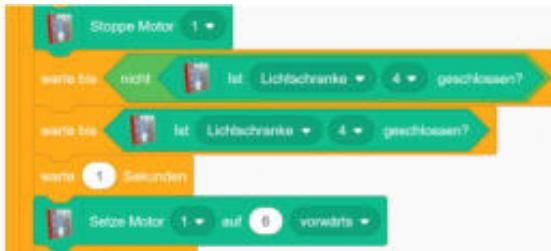


Abb. 4: Auswertung der Lichtschranke

#### Aufgabe 4

Soll die Schranke alternativ zum Bedarfssteller auch beim Drücken der Taste „a“ („auf“) geöffnet werden, muss die zweite Bedingung mit einem logischen „ODER“-Glied verknüpft werden.

Dasselbe gilt für das Schließen der Schranke über die Taste „s“. Hier musst du die Tastatur zugleich mit der Lichtschranke zweimal abfragen.



Abb. 5: Wartungseingriffe

#### Aufgabe 5

Ein paralleler Prozess, auch als „Thread“ bezeichnet, kann in Scratch an unterschiedliche Auslöser („Trigger“) geknüpft werden.

In unserer Aufgabe ist der Trigger schnell gefunden: Wir prüfen, ob der Endlagetaster für das Schließen der Schranke geöffnet wurde. Jedes Mal, wenn dieses Ereignis eintritt, soll das Blinklicht gestartet werden. Es darf erst erloschen, wenn der Endlagentaster wieder geschlossen wird. Abb. 6 zeigt den Thread.



Abb. 6: Paralleler Prozess: Blinklicht

#### Aufgabe 6

Sprache und Stimmlage können zu Beginn des Skripts eingestellt werden. Danach genügt es, den gewünschten Text an die Sprachausgabe zu übergeben.



Abb. 7: Ansagen

#### Aufgabe 7

Die Bewegungserkennung ist einfach zu aktivieren: Kamera anschließen (und freigeben), Video anschalten (erscheint als Hintergrund zur Bühne) und die Bewegungserkennung als weitere Bedingung neben dem Bedarfsschalter aufnehmen (Abb. 8). Dabei wird jede Bewegung vor der Kamera (also auf dem Bühnenhintergrund) erkannt.



Abb. 8: Bewegungserkennung

Der Schwellenwert für die Bewegungserkennung kann zwischen 0 (keine Bewegung) und 100 (sehr starke Bewegung) eingestellt werden.

## Weiterentwicklung des Modells

Mit einem TXT verfügt ihr nicht nur über vier, sondern über acht Eingänge – ihr könnt damit also zwei Schranken zugleich steuern. Damit lässt sich ein ganzes Parkhaus verwalten, indem ihr die freien Parkplätze zählt, den Wert bei jeder Ein- und Ausfahrt eines Fahrzeugs anpasst und anzeigt. Das Display des TXT könnt ihr zur Anzeige der gerade freien Parkplätze nutzen.

Auch eine Statistik über die Parkhausauslastung könnt ihr erstellen, indem ihr die Belegung bspw. alle 10 Minuten in eine Liste einträgt.

Wenn ihr euch die Liste auf der Bühne anzeigen lasst, könnt ihr die Ergebnisse jederzeit mit einem Klick auf die rechte Maustaste in eine csv-Datei exportieren und die Daten anschließend bspw. mit Excel auswerten.

Oder aber ihr verwendet statt der Knickschranke eine andere Schrankenkonstruktion, ein Klapp- oder Schiebetor [2, 3] oder

eine Zugbrücke, wie z. B. das Tor aus dem Baukasten Profi Electronics ([524326](#)), die Brücke aus dem Baukasten Advanced Universal 4 ([548885](#)) oder das Garagentor aus dem Clubmodell 11 [4].

In hobby 1 Band 3 finden sich großartige Klapp-, Hub- und Drehbrücken [5]. Bei diesen Modellen lassen sich sowohl eine Schranke auf jeder Seite als auch der Klapp-, Hub- oder Drehmechanismus der Brücke selbst steuern...

## Referenzen

- [1] Dirk Fox: *Scratch mit fischertechnik – Update 2022*. In dieser Ausgabe der ft:pedia.
- [2] Stefan Busch: *Schwenktüren*. [ft:pedia 3/2017](#), S. 23–33.
- [3] Stefan Busch: *Schiebetüren*. [ft:pedia 1/2018](#), S. 24–33.
- [4] fischerwerke: *Clubmodell 11: Garagentor mit Solarantrieb*. In: Fan-Club News 2/1997.
- [5] fischerwerke: *Hub-, Dreh- und Klappbrücken*. In: hobby 1 Band 3, S. 56–79.

Modell

## Solartracker

Michael Schulte

*Ein Solartracker optimiert die Energieausbeute einer Solarzelle durch die exakte Ausrichtung nach dem Sonnenstand. Viel präziser als mit dem einfachen „Solarzellenachführer“ aus der ft:pedia 1/2020 [2] gelingt das mit einem Solartracker, der zu jeder Zeit den exakten Sonnenstand bestimmt.*

### Warum ein Solartracker?

Ein zweiachsiger Solartracker ist ein elektromechanisches Gerät, das eine Ebene (z. B. Solarpanel, Kamera, Laser etc.) exakt im Jahres- und Tagesverlauf zur Sonneneinstrahlung ausrichtet. Dabei werden Elevation (Höhenwinkel, Neigung) und Azimut (Horizontalwinkel, Himmelsrichtung) dieser Ebene durch Stellmotoren mittels eines Steuerprogramms so verändert, dass die Sonnenstrahlen z. B. auf ein Photovoltaik-Modul stets senkrecht fallen und so eine maximale Strahlungs- und damit Energieausbeute erzielt werden kann.



Abb. 1: Solarpanel mit Nachführung (Foto: USFWS/Rachel Molenda)

Ein zweiachsiger nachgeführter Solartracker empfängt z. B. in Norddeutschland im langjährigen Mittel 26% mehr Einstrahlung als eine fest ausgerichtete Anlage (mit 30° Neigung, Südrichtung). In den Morgen- und Abendstunden ist die Einstrahlung im Jahresmittel bis zu 70% höher.

Wenn das Solarpanel bei bedecktem Himmel aus der astronomisch berechneten Position in die Horizontalposition gefahren wird, steigt die mittlere Jahresteinstrahlung um ca. 2% [8].

### Das Konzept

Das hier vorgestellte zweiachsige Solartracker-Modell berücksichtigt alle relevanten Einflussgrößen (Datum, Jahrestag, genaue Uhrzeit, Ortskoordinaten, Azimut, Elevation) und richtet ein kleines Solarpanel vollautomatisch so exakt zur Sonne aus, dass die Energieausbeute im Tages- und Jahresverlauf jederzeit maximiert ist.

Nach dem Sonnenuntergang im Westen wird das Solarpanel selbsttätig zur Ausgangsposition (Osten) zurückgefahren. Hier beginnt morgens bei Sonnenaufgang ein neuer Betriebszyklus.

Mein Steuerprogramm arbeitet mit Algorithmen, die auch Astronomen zur Berechnung der Sonnenposition im Tages- und Jahresverlauf verwenden [1].

Für meinen Standort (lat = 51.222 Grad Nord, lon = 7.954 Grad Ost) gilt: Der maximale Höhenwinkel der Sonne über dem Horizont (Elevation) bewegt sich von ca. 15° am 22. Dezember (Tag der Wintersonnenwende) um 12:00 Uhr WOZ (= Wahre Ortszeit) bis ca. 62° am 21. Juni um 12:00 Uhr WOZ (Sommersonnenwende).

Der entsprechende Horizontalwinkel (Azimut) liegt am 22. Dezember 12:00 Uhr WOZ zwischen etwa 128° (bei Sonnenaufgang) und 231° (bei Sonnenuntergang), und zwischen 50° (Sonnenaufgang) und 310° (bei Sonnenuntergang) am 21. Juni.

Entscheidend für die Berechnungen ist eine hochpräzise Uhr (z. B. hier eine DS3231 Real Time Clock (RTC), siehe auch [3], oder ein GPS-Modul [4]), weil genaue Sonnenpositionen hochgradig zeitabhängig sind. Weitere wichtige Einflussgrößen für die Berechnung der Sonnenposition sind neben der genauen Uhrzeit das Datum, die Tag-Nr. des laufenden Jahres (1-365), die Geographische Breite (Latitude) und die Geographische Länge (Longitude) des Standortes. Ein GPS-Modul hat den Vorteil, dass es den Solartracker ortsunabhängig macht, indem es die notwendigen Standortkoordinaten liefert. Außerdem stellt es dem Solartracker die geforderte hochpräzise Zeitinformation bereit.

Von allenfalls geringem Einfluss ist die atmosphärische Refraktion (Brechung eines Lichtstrahls in der untersten Erdatmosphäre), die insbesondere bei Sonnenauf- und -untergängen den Einstrahlwinkel um maximal 1-2° verändern kann. Bei diesem Solartracker-Modell bleibt die Refraktion daher unberücksichtigt, weil der Refraktionswert innerhalb der Fehlertoleranz des Servos liegt. Das spart außerdem Arbeitsspeicher, der beim Arduino UNO auf 64 kB begrenzt ist; der Sketch bewegt sich ohnehin am Speicherlimit. Für die nachfolgende Solartracker-Version ist deshalb ein Arduino-Mega vorgesehen, der 256 kB Arbeitsspeicher zur Verfügung stellt.

## Das Modell

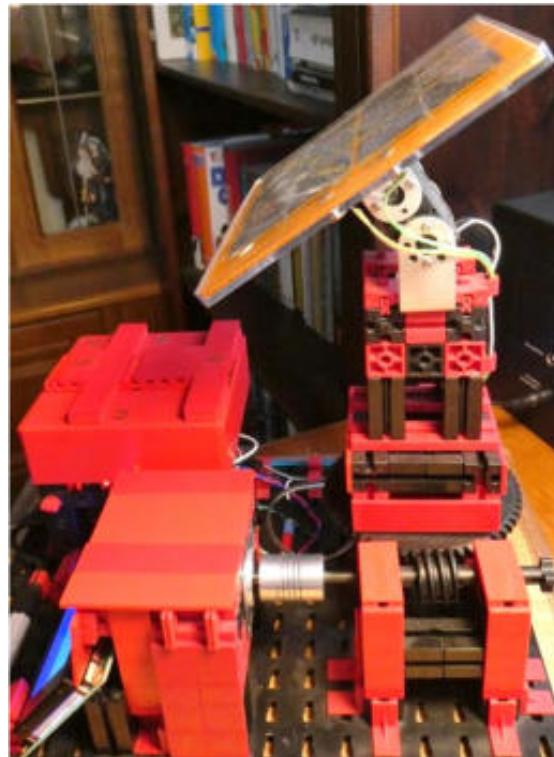


Abb. 2: Das Solartracker-Modell, gebaut u. a. mit fischertechnik-Teilen

Unter dem roten „Deckel“ in Abb. 2 (hinten links) befindet sich die Steuerzentrale mit einem Arduino-Uno-Mikrocontroller [5] und einem Motorshield.

Das C++-Programm wird von einem PC per USB auf den Microcontroller geladen und danach ohne PC selbstständig ausgeführt. Die Aktualisierung der Azimut- und Elevation-Positionen erfolgt kontinuierlich in 1°-Schritten.

Unter dem waagerecht liegenden Drehkranz in Abb. 2 befindet sich ein Hall-Drehwinkel-Sensor (0-360°), dessen Achse sich 1:1 mit dem Solarpanel bewegt. So kann jederzeit die aktuelle Azimut-Position (AzI, Istwert) des Solarpanels bestimmt und mit dem astronomisch berechneten Azimutwert (AzS, Sollwert) verglichen werden. Ein Schrittmotor (in Abb. 2 verdeckt) sorgt über einen Schneckentrieb für die Bewegung des Drehkränzes.



Abb. 3: Modellansicht von vorne



Abb. 4: Longrunner-Servo für die Höhenwinkel-Verstellung (0-180 Grad) mit aufmontiertem 3-Volt-ETM500-Solarpanel

Der astronomisch berechnete Elevation-Wert wird 1:1 direkt auf den Servo übertragen (Abb. 4) und bewirkt eine ständig aktualisierte, exakte Schrägausrichtung des Solarpanels zur Sonne. Die Spannung am Solarpanel wird gemessen und auf dem Display angezeigt.

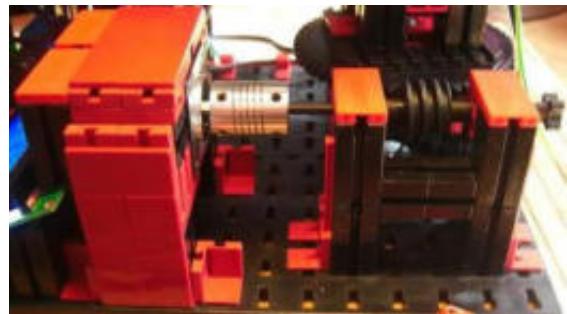


Abb. 5: Ein Pololu-Steppermotor mit Schneckentreiber steuert die Azimut-Verstellung

Falls der berechnete Azimut-Sollwert (AzS, s. Abb. 6) größer ist als der vom Hall-Drehwinkelsensor gelieferte Azimut-Istwert (AzI), so dreht der Steppermotor in Abb. 5 32 Schritte vorwärts; ist AzS kleiner als AzI, so dreht der Steppermotor 32 Schritte rückwärts (32 Schritte entsprechen ca. 1° des Drehwinkels). Auf diese Weise ist (mit einer Auflösung von ca. 1°) im Idealfall immer AzI = AzS, d. h. die Azimutposition des Solarpanels (AzI) stimmt mit dem berechneten Azimutwert (AzS) überein.



Abb. 6: Ein SunFounder-20x4-LC-Display (siehe auch [5]) zeigt die wichtigsten Daten an

Auf dem LC-Display (Abb. 6) werden angezeigt: Datum, Uhrzeit (MEZ), AzS = Azimut-Sollwert (astronomisch berechneter Wert in Grad), WZ = Wahre Ortszeit (um 12:00 Uhr steht die Sonne am Standort genau im Süden), AzI = aktueller Azimut-Sollwert des Solarpanels (durch einen Hall-Drehwinkel-Sensor gemessener Azimut-Ist-Wert in Grad), SO = aktuelle Himmelsrichtung (SüdOst), TL = Tageslänge in [h], Elv = Elevation (Sonnenhöhe, berechneter Wert in Grad), PV = aktuelle Spannung am Solarpanel [Volt].

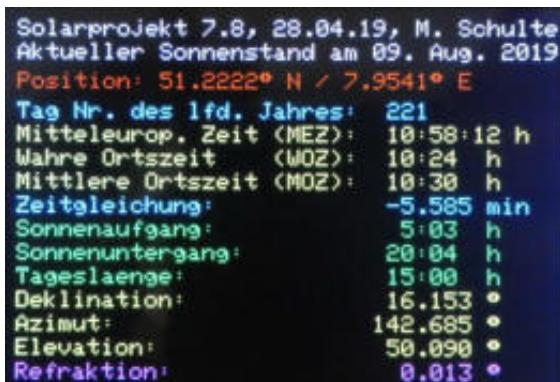


Abb. 7: Zusätzliches Display

Ein Zusatzdisplay (Kuman UNO R3 3,5" TFT Touchscreen, Abb. 7) zeigt die wichtigsten Zeit- und Sonnenstands-Werte im 10-Sekunden-Takt an. Das Zusatzdisplay wird von einem Arduino-Mega-2560-Mikrocontroller angesteuert.



Abb. 8: Hall-Drehwinkelsensor (0-360° entsprechend 0-5 Volt)

Der Hall-Drehwinkelsensor (Abb. 8) dient zur Bestimmung der aktuellen Azimut-Position des Drehtellers (Ist-Wert) und des aufmontierten Solarpanels.

Die Stromversorgung des Mikrocontrollers und des Motorshield erfolgt über die runde Steckbuchse vorne auf dem Mikrocontroller-Board. Die Eingangsspannung sollte mindestens 7,5 VDC betragen und das Maximum von 9,0 VDC nicht überschreiten. Das Motorshield versorgt den Schrittmotor mit 7,5 VDC (~ Nennspannung). Der Servo wird über den Pin Vin auf dem Motorshield mit seiner Nennspannung von 7,5 VDC versorgt.

**Achtung:** Keinesfalls dürfen Schrittmotor und Servo allein über die USB-Leitung (+ 5 VDC) betrieben werden. Es drohen Defekte bei Mikrocontroller und angeschlossenem Computer! Die Stromversorgung der LEDs,

des Hall-Sensors, der RTC-Uhr und des LCD-Displays erfolgt über die Anschlüsse auf dem Motorshield mit +5 VDC.

#### Status-LED:

- *Grüne LED:* Drehkranz dreht vorwärts im Uhrzeigersinn
- *Rote LED:* Drehkranz dreht rückwärts gegen den Uhrzeigersinn (siehe Tab. 1).
- *Blaue LED* (in Vorbereitung): Drehkranz bewegt sich nach Sonnenuntergang sehr schnell zurück zur Startposition am Folgetag im Osten.

Der Arduino-Code zur Ansteuerung des Modells findet sich auf meiner Webseite zum Download [1].

#### Berechnung des Getriebes

Die Zähnezahl des Drehkränzes beträgt 58. Die Schnecke dreht den Drehkranz bei jeder Umdrehung um einen Zahn. Also liegt das Untersetzungsverhältnis bei 58:1.

Der Steppermotor benötigt für eine Achsumdrehung 200 Schritte. Damit benötigt eine Umdrehung des Drehkränzes 200 Schritte x 58 = 11.600 Schritte.

Pro Winkelgrad dreht sich der Drehkranz um  $11.600 \text{ Schritte} / 360^\circ = 32.222 \text{ Schritte} / ^\circ$ .

Eine Winkelveränderung des Drehkränzes um  $1^\circ$  benötigt also mindestens 32 Schritte des Steppermotors.

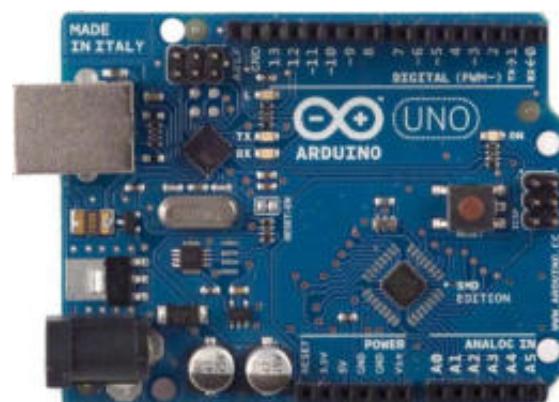


Abb. 9: Arduino Uno R3 (SMD Edition)

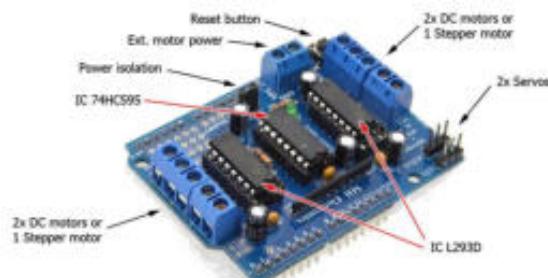


Abb. 10: Adafruit Motorshield v1.0

### Pinbelegung am Arduino UNO

Pin	Belegung
D9	(Servo 2, Pin S): Servo-Steuerleitung (Elevation)
Vin	Stromversorgung des Servos (für Elevation), ca. 7,5 VDC, am Motorshield mit 9 V beschriftet
D14 (A0)	grüne LED (Drehsteller vorwärts)
D15 (A1)	rote LED (Drehsteller rückwärts)
A2	Hall-Drehwinkelsensor (für Azimut), dreipolig
A3	3-Volt-Solarpanel (+), (GND)
A4	SDA: LCD-Display, DS3231 RTC (jeweils vierpolig)
A5	SCL: LCD-Display, DS3231 RTC (jeweils vierpolig)
M3/M4	Motorshield: Anschluss Schrittmotor (Azimut), vierpolig

Tab. 1: Pinbelegung

### Hardware-Liste

- Arduino-Uino-Mikrocontroller R3
- Arduino-Mega-2560-Mikrocontroller
- 2 x DS3231 RTC-Echtzeituhr (Zeitbasis Solatracker und für Zusatzdisplay)
- Adafruit-Motor-Shield V.1
- Pololu 1207-Stepper-Motor, 7,4 VDC
- Longrunner-17kg-Digital High-Torque Robot-Servo-Motor, 7,5 Volt DC

- SunFounder I2C 20x4 LC-Display-Modul, 5 VDC
- Hall-Drehwinkelsensor (0-360°/0-5 V Output), Versorgungsspannung 5 VDC (s. Abb. 7).
- diverse fischertechnik-Bauteile
- Kuman UNO R3 3,5" TFT Touchscreen (Zusatzdisplay)
- Solarpanel ETM500-3V (14 x 13 cm, 3 VDC; Kosmos-Solarbaukasten)

### Referenzen

Eine sehr hilfreiche Website zur Sonnenstands berechnung mit vielen astronomischen Formeln und Programmierbeispielen findet sich unter [astronomie.info](#).

Sehr hilfreich zum Verständnis der astronomischen Sonnenstands berechnung ist auch das Buch von Volker Quaschning [7].

- [1] Michael Schulte: [Sonnenstandsberechnung und Solartracking](#). Arduino-Sketch.
- [2] Dirk Fox: [Solarzellennachführer](#). [ft:pedia 1/2020](#), S. 79–84.
- [3] Dirk Fox: [I<sup>2</sup>C mit dem TX – Teil 7: Real Time Clock \(RTC\)](#). [ft:pedia 4/2013](#), S. 28–34.
- [4] Dirk Fox: [I<sup>2</sup>C mit dem TX – Teil 6: GPS-Sensor](#). [ft:pedia 3/2013](#), S. 54–62.
- [5] David Holtz: [Alternative Controller \(1\): Der Arduino](#). [ft:pedia 2/2016](#), S. 56–59.
- [6] Dirk Fox: [I<sup>2</sup>C mit dem TX – Teil 9: LC-Displays](#). [ft:pedia 1/2014](#), S. 47–57.
- [7] Volker Quaschning: [Regenerative Energiesysteme](#). Carl Hanser Verlag, München, 11. aktualisierte Auflage 2021.
- [8] Siegfried Kreußler, Manfred Bergmann: [Lohnen sich Solartracker?](#) [Sonnenenergie 1/2016](#).

Computing

# Single Track Gray Encoder mit fischertechnik

Florian Bauer

In diesem Beitrag wird ein prototypischer Aufbau eines Single Track Gray Encoders mit fischertechnik vorgestellt. Dies ist ein multidisziplinäres Projekt, das aufzeigt, wie man mit Hilfe von fischertechnik, etwas Elektronik und einem Mikrocontroller ein mathematisches Modell in die Realität umsetzen und begreifbar machen kann.

## Drehgeber

Für die Messung von Positionen und Winkel werden oft Drehgeber eingesetzt, die auf unterschiedlichen physikalischen Mess-Prinzipien beruhen wie zum Beispiel Widerstandsmessung, Zählung mechanischer, optischer oder magnetischer Impulse. Man unterscheidet zwei Arten von Drehgebern:

### Inkremental-Drehgeber

Hier wird die Änderung der Position gemessen. Bei den weit verbreiteten optischen Drehgebern wird eine Encoderscheibe mit einem äquidistanten Streifenmuster von zwei versetzt angeordneten Lichtschranken abgetastet. Durch Zählung und Auswertung der Phasenlage der Lichtschranken-Impulse kann die Position relativ zu einem Ausgangszustand bestimmt werden. Will man eine Absolut-Position messen, muss zuerst eine Referenzfahrt zu einem Endschalter vorgenommen werden.

Diese Art von Drehgebern ist sehr oft in Druckern zu finden. Ausrangierte Drucker sind daher eine ideale Quelle für hochauflösende Positions-Geber.

### Absolut-Drehgeber

Bei diesen Drehgebern liegt die Rotationsposition zu jedem Zeitpunkt als Messwert oder Digitalwort vor. Der Vorteil ist, dass keine Referenzfahrt nötig ist und auch nach

einem Stromausfall die Position ohne Referenzfahrt korrekt ausgegeben wird.

Im Folgenden wollen wir uns einen 3-Spur-Absolut-Drehgeber ansehen. Die acht Digitalwerte sind als Schwarz-Weiß-Muster auf einer Scheibe mit drei Spuren codiert (Abb. 1).

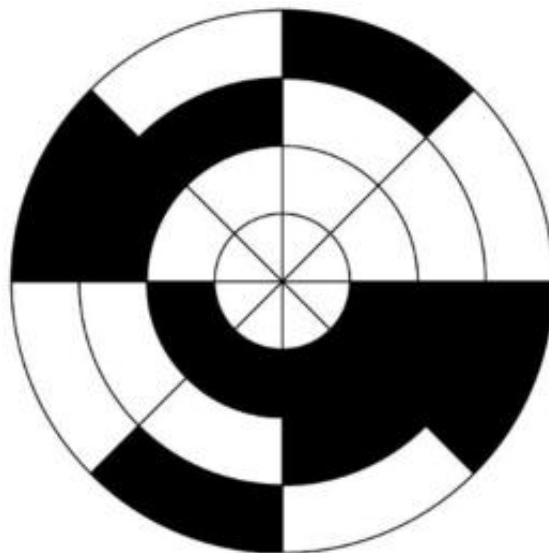


Abb. 1: 3-Binärer Code

Jede Spur entspricht einer Bit-Position, ein schwarzer Bereich kodiert eine 1 und ein transparenter Bereich eine 0. Die Spuren werden von einem Lichtschranken-Array mit radial angeordneten Fotozellen abgetastet. Es lassen sich acht verschiedene Winkelwerte auflösen; die Auflösung beträgt damit  $45^\circ$  ( $= 360/8$ ). Für höhere Winkelauflösungen benötigt man mehr Spuren.

Um das Muster abzutasten verwendet man eine Maske ( $45^\circ$ -Segment). Die Öffnungen für die weiter innen liegenden Sensoren sind kleiner als die für die äußeren, was bei Sensoren, die kleiner sind als die Maskenöffnung, zu Ablesefehlern führen kann. So müssten bei einem Übergang von 111 auf 000 alle drei Bitpositionen gleichzeitig springen, was aber i. a. nicht möglich ist, da die äußeren Sensoren erst verzögert schalten, was zu Pseudopositionen, also falschen Winkelwerten führt.

Abhilfe schaffen hier sogenannte Zyklische Gray-Codes, bei denen sich zwischen den Worten jeweils nur ein Bit ändert, und das auch bei Start- und End-Wort. Dies erreicht man beispielsweise durch sogenannte *Binary Reflected Gray Codes*.

*Beispiel:*

$C_3 = 000, 001, 011, 010, 110, 111, 101, 100$

Eine zugehörige Masken-Scheibe ist in Abb. 2 dargestellt.

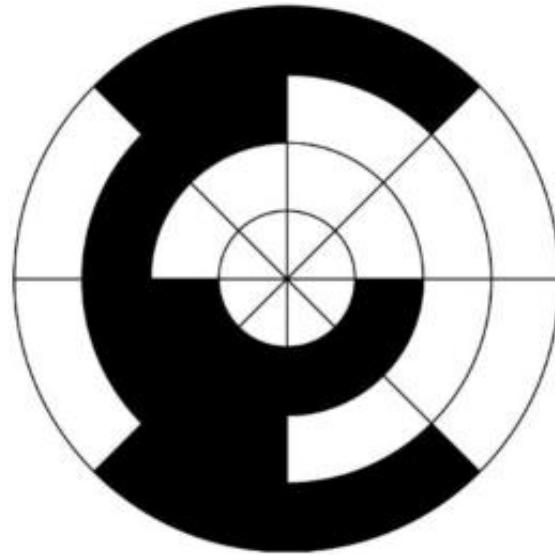


Abb. 2: 3-Bit Cyclic Binary Reflected Gray Code Encoder

Damit die Masken- und Sensorgröße kein Problem werden, muss man bei höheren Auflösungen hinreichend große Code-Scheiben verwenden oder entsprechend kleine Strukturen und Sensoren einsetzen.

Letzteres ist im Hobby-Bereich sehr schwierig zu realisieren.

Für 12-bit-Werte ist eine Winkelauflösung von  $0,0015^\circ$  nötig (=  $360/4096$ ). Würde die innerste Spur auf einem Kreis mit Radius 50 mm liegen, müsste die Maskenöffnung 0,08 mm klein sein. Für eine Maskenöffnung von 1 mm bräuchte man für die innerste Spur einen Radius von 667 mm. Um die restlichen Spuren unterzubringen braucht man noch mehr Platz oder entsprechend kleine Masken und Sensoren.

### Single Track Gray Codes

Um das Problem mit den spurabhängigen Masken-/Sensorgrößen zu vermeiden ist es wünschenswert, die Sensoren entlang einer einzigen Spur zu platzieren. Encoder mit nur einer Spur werden auch als *Single Track Encoder* bezeichnet.

Man braucht dazu einen sogenannten zyklischen *Single Track Gray Code* (STGC). Zu gegebener Bitlänge  $n$  lassen sich solche STGC mit Periode  $n \cdot t$  konstruieren (mit  $t$  gerade), sodass gilt [3]:

$$2 \leq t \leq 2^{n-\lceil \sqrt{2(n-3)} \rceil - 1}$$

Für  $n = 5$  gibt es beispielsweise einen Code für  $t = 6$  mit 30 Werten und damit einer Winkelauflösung von  $12^\circ$ . Durch die zusätzlich geforderte Eigenschaft kann nicht der volle Bereich von 32 Werten ( $2^5$ ) ausgeschöpft werden. Dafür hat man aber einen 30-stelligen zyklischen Single Track Gray Code:

00100, 00101, 01101, 01111, 01110,  
01100, 01000, 01010, 11010, 11110,  
11100, 11000, 10000, 10100, 10101,  
11101, 11001, 10001, 00001, 01001,  
01011, 11011, 10011, 00011, 00010,  
10010, 10110, 10111, 00111, 00110

Dieser kann durch die folgende 30-Bit-Sequenz mit Sensoren im Abstand von 6 Bits realisiert werden:

000000001111111111000110011100

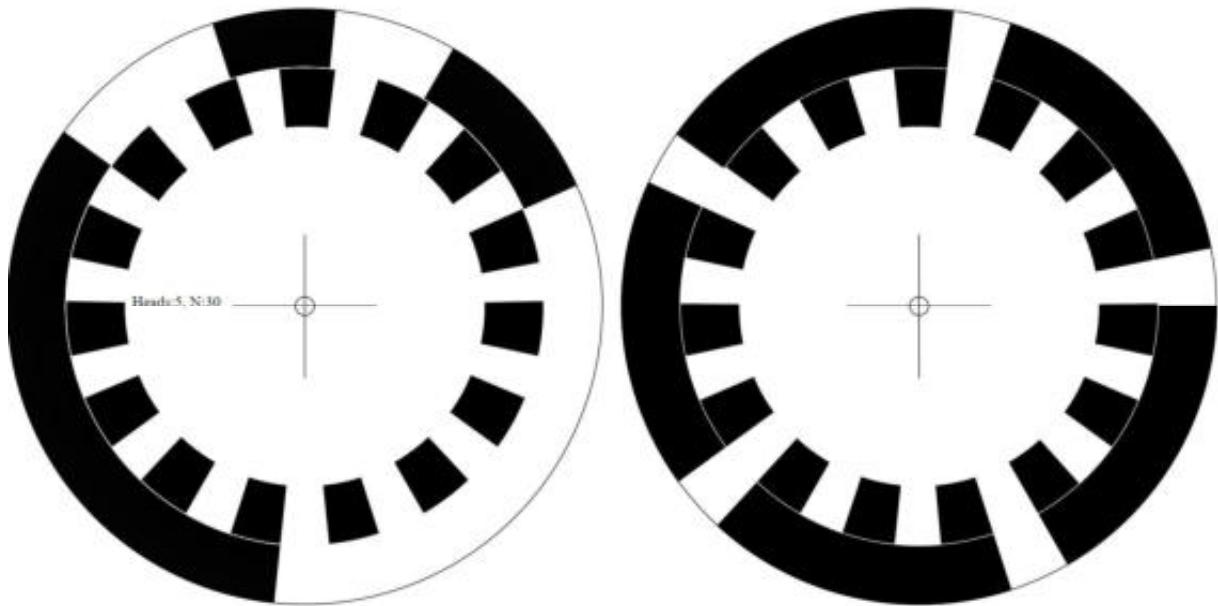


Abb. 3: Code- und Maskenscheibe eines STGC-Encoders mit fünf Sensoren und 30 Werten

(Zur Verdeutlichung sind für die ersten vier Positionen die Ableseposition im Bitstring und das ausgelesene Binärwort farblich markiert.)

Die zugehörige Encoder-Scheibe für fünf Sensoren und die Masken-Scheibe sind in Abb. 3 dargestellt. Das Bitmuster des STGC ist auf der äußeren Spur aufgebracht. Die innere Spur ist nur eine Hilfe zur leichteren Identifikation der Bit-Position:

Bei einem Absolut-Winkelencoder ist die Masken-Scheibe fest mit dem Rahmen des Encoders verbunden und die Code-Scheibe dreht sich koaxial über der Masken-Scheibe. An den Aussparungen befinden sich die Lichtschranken.

### **Generierung von Single Track Gray Codes**

Die Suche von STGC-Codes mit höheren Bitlängen der Worte wird schnell kombinatorisch komplex. Für die Erzeugung der STGC-Sequenzen gibt es verschiedene Verfahren. Eines soll im Folgenden kurz beschrieben werden, wobei nicht näher auf die Mathematik eingegangen wird, die in [1] genauer behandelt wird.

Ausgangspunkt ist eine sogenannte Seed-Code-Binärwort-Folge mit bestimmten Eigenschaften:

- Die Wortfolge bildet einen zyklischen Gray-Code,
- alle Worte sind sogenannte Lyndon-Worte und
- alle Worte sind sogenannte Necklaces.

Ein Wort ist ein Lyndon-Wort, wenn es (lexikographisch) kleiner ist als jede seiner „zyklischen Rotationen“.

*Beispiel:* 1010 ist kein Lyndon-Wort, da 0101 kleiner ist als 1010. 0011 hingegen ist ein Lyndon-Wort, da es keine Rotation gibt, die ein kleineres Wort erzeugt (0110, 1100 und 1001 sind alle größer).

Ein  $n$ -Bit-Wort ist ein Necklace, wenn es keine zyklische Rotation (mit  $1..n - 1$  Rotationen) gibt, die das Wort auf sich selber abbildet.

*Beispiel:* 0101 wird durch eine zweistufige zyklische Rotation auf sich selbst abgebildet und ist daher kein Necklace. 0011 ist ein Necklace, da es durch keine zyklische Rotation auf sich selbst abgebildet werden kann.

Als Ausgangsbasis für die Suche nach Seed-Code-Worten erzeugt man eine Menge von Lyndon/Necklace-Worten. Aus dieser Menge bildet man Graphen, deren Knoten Worte aus dieser Menge sind und deren Kanten Verbindungen zwischen Wörtern, die sich nur in einem Bit unterscheiden – die eine Hamming-Distanz von 1 haben, also „gray“ sind. Jede Seed-Code-Folge ist ein geschlossener Pfad in diesem Graphen, bei dem jeder Knoten mit genau zwei Nachbarn verbunden ist. Ein solcher geschlossener Pfad wird auch als „Coil in the Box“ bezeichnet (offene Pfade heißen „Snake in the Box“ und spielen für STGC keine Rolle).

Die Suche nach einer Seed-Code-Folge kann beispielsweise mit Hilfe einer rekursiven Suche (*Depth First Search*) erfolgen, die für  $n \geq 12$  und lange Pfade  $t = 300$  schon länger dauern kann.

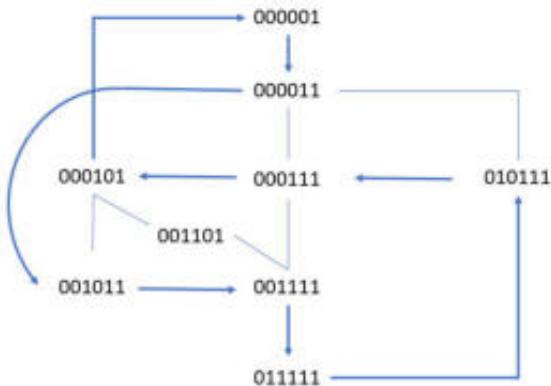


Abb. 4: Seed-Code-Folge ( $n=6$ )

In Abb. 4 ist ein Graph mit Seed-Code-Wörtern und einer zyklischen Seed-Code-Folge für  $n = 6$  dargestellt. Die Kanten des

Graphen sind Verbindungen zwischen Wörtern, die sich um ein Bit unterscheiden.

Hat man nun eine Seed-Code-Folge, kann daraus durch sukzessive Anwendung von Bit-Rotationen ein *Single Track Gray Code* aufgebaut werden.

Für  $n = 6$  findet man z. B. folgende acht-elementige Seed-Code-Folge:

000011, 001011, 001111, 011111,  
010111, 000111, 000101, 000001

Diese Folge kann man zu einem 48-stelligen STGC erweitern:

000011, 001011, 001111, 011111,  
010111, 000111, 000101, 000001

100001, 100101, 100111, 101111,  
101011, 100011, 100010, 100000

110000, 110010, 110011, 110111,  
110101, 110001, 010001, 010000

011000, 011001, 111001, 111011,  
111010, 111000, 101000, 001000

001100, 101100, 111100, 111101,  
011101, 011100, 010100, 000100

000110, 010110, 011110, 111110,  
101110, 001110, 001010, 000010

mit dem Bit-String für die Encoder-Scheibe:

000000000001100000001110011110000  
11111111111111

Die zugehörige Encoder-Scheibe für sechs Sensoren und die Masken-Scheibe sind in Abb. 5 dargestellt. Die Auflösung beträgt  $7.5^\circ$ .

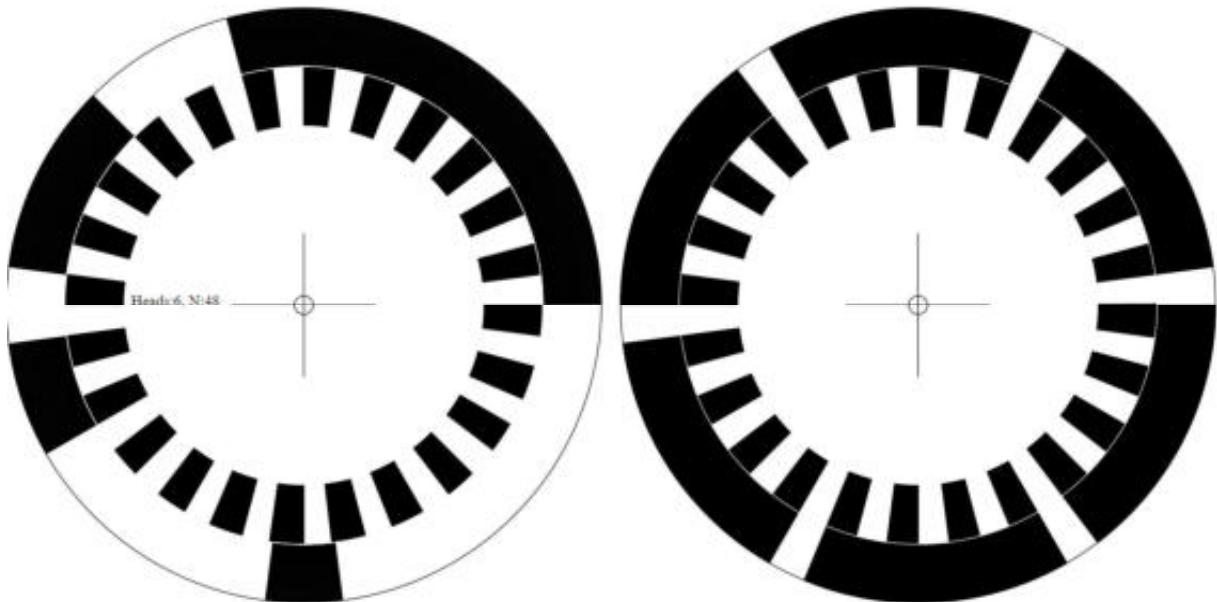


Abb. 5: Code- und Maskenscheibe eines STGC-Encoders mit sechs Sensoren und 48 Werten

## Realisierung eines 6-Bit-Encoders mit 48 Stufen

Nachdem das Prinzip erläutert wurde soll es nun in ein fischertechnik-Modell umgesetzt werden (Abb. 7). Für den Encoder benötigt man:

- Eine transparente bedruckte Encoder- und eine Masken-Scheibe:

Diese kann man herstellen, indem man das Muster mit einem Laser- oder Desk-Jet-Drucker auf eine Transparentfolie druckt, sodass der Durchmesser der Masken 75 mm beträgt.

Wenn die schwarzen Bereiche der Masken nicht vollkommen deckend sind, kann man sie mit einem schwarzen Edding-Stift oder schwarzer Acrylfarbe nachbehandeln, was bei Bitlängen  $\leq 9$  noch gut funktioniert.

- Einen Halter für die Masken-Scheibe

Als Halter wird eine Drehscheibe 60 verwendet. Die hier nötige sechszählige Symmetrie kann mit 60°-fischertechnik-Winkelsteinen leicht realisiert werden.

Die ausgeschnittene und in der Mitte gelochte Masken-Scheibe kann mit doppelseitigem Klebeband befestigt werden.

- Ein Rad, das die Encoder-Maske aufnimmt

Auch die Encoder-Scheibe wird mit doppelseitigem Klebeband auf einer Drehscheibe 60 aufgeklebt. Die Achse der Encoder-Scheibe wird in die Nabe der Maskenscheibe gesteckt.

- Lichtschranken bestehend aus LEDs und LDRs oder Fototransistoren

Für die Beleuchtung können 3-mm-LED, 5-mm-LED oder fischertechnik-Lampen eingesetzt werden. Als Lichtschranken können LDRs oder Fototransistoren verwendet werden.

- Halterungen für Lichtschranken

Die Halterungen hängen davon ab, welche Foto-Elemente man verwendet. 3-mm-LED und 3-mm-Fototransistoren können z. B. mit einer kleinen gedruckten Adapterhülse (Innendurchmesser 3 mm, Außendurchmesser 4 mm) in den Lagersteinen eingesetzt werden.

- Pick-Up-Elektronik für die Lichtschranken

Fototransistoren werden über 47-kΩ-Widerstände an +5V angeschlossen. Das

Signal leitet man am Kollektor des Transistors ab und führt es an einen Digital-Pin eines Mikrocontrollers. Ein abgedeckter Fototransistor sperrt und das Signal beträgt +5V, ein beleuchteter Fototransistor schaltet durch und zieht das Signal gegen Masse auf einen Wert < 1V.

- Ein Mikrocontroller für die Auswertung und eventuelle Anzeige der Positions- bzw. Winkelwerte

Für den hier vorgestellten Aufbau kommt ein Arduino-Uno zum Einsatz. Der Uno übernimmt das Auslesen der Lichtschranken und der Winkel-Dekodierung über eine Tabelle. Das Programm gibt die Winkelwerte auf der seriellen Schnittstelle aus.

- Optional eine 7-Segment-Anzeige um die Winkelwerte darzustellen

Darüber hinaus steuert der Arduino ein 7-Segment-Display an, das die Werte Interrupt-gesteuert anzeigt.

Eine Schaltung, die auch einen 9-Bit-Encoder unterstützt, zeigt Abb. 6.

Abb. 7 zeigt einen Blick auf die Empfängerseite des STGC-Encoders mit zwei installierten Fototransistoren. Auf der sichtbaren Drehscheibe 60 ist die Code-Folie aufgeklebt. Die Achse steckt in einer identischen Drehscheibe 60 dahinter, die am Rahmen fixiert ist und auf der die Masken-Folie aufgeklebt ist.

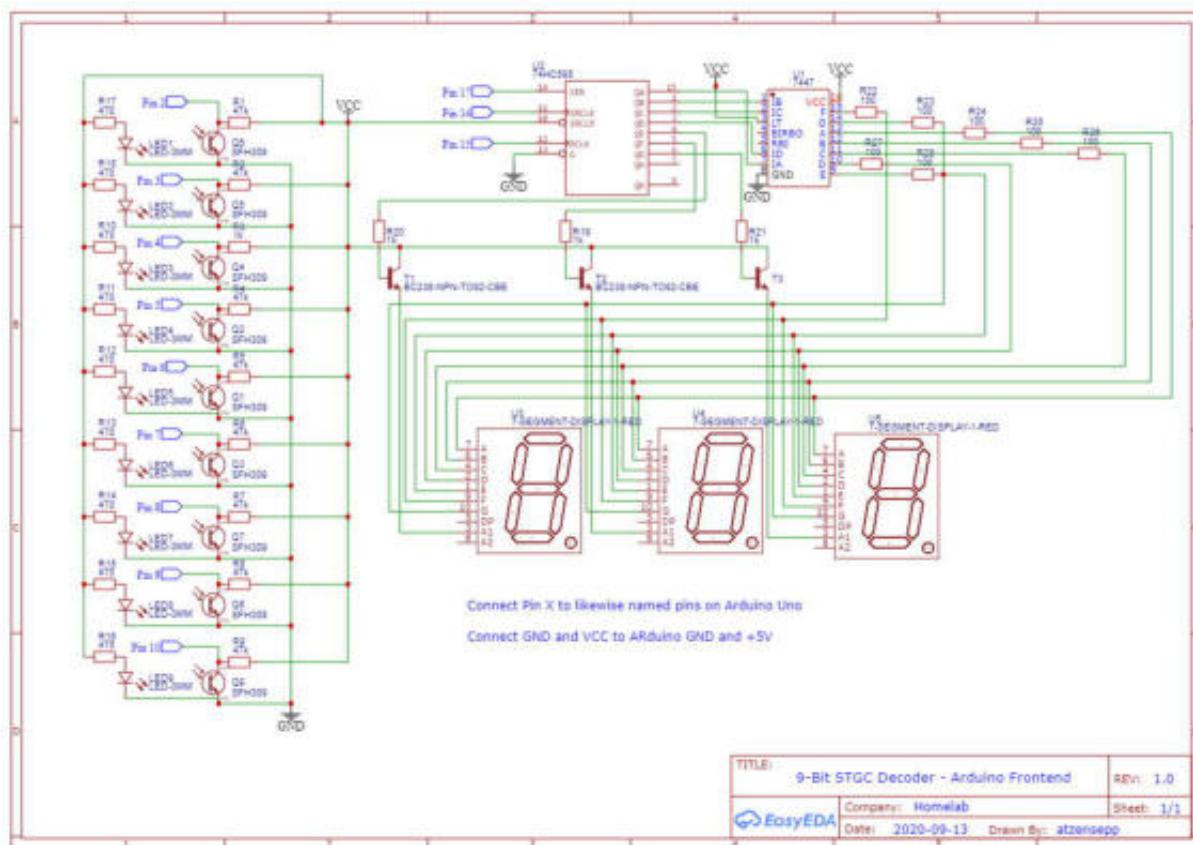


Abb. 6: Schaltplan für einen STGC-Encoder bis zu 9 Bit mit LED-Anzeige

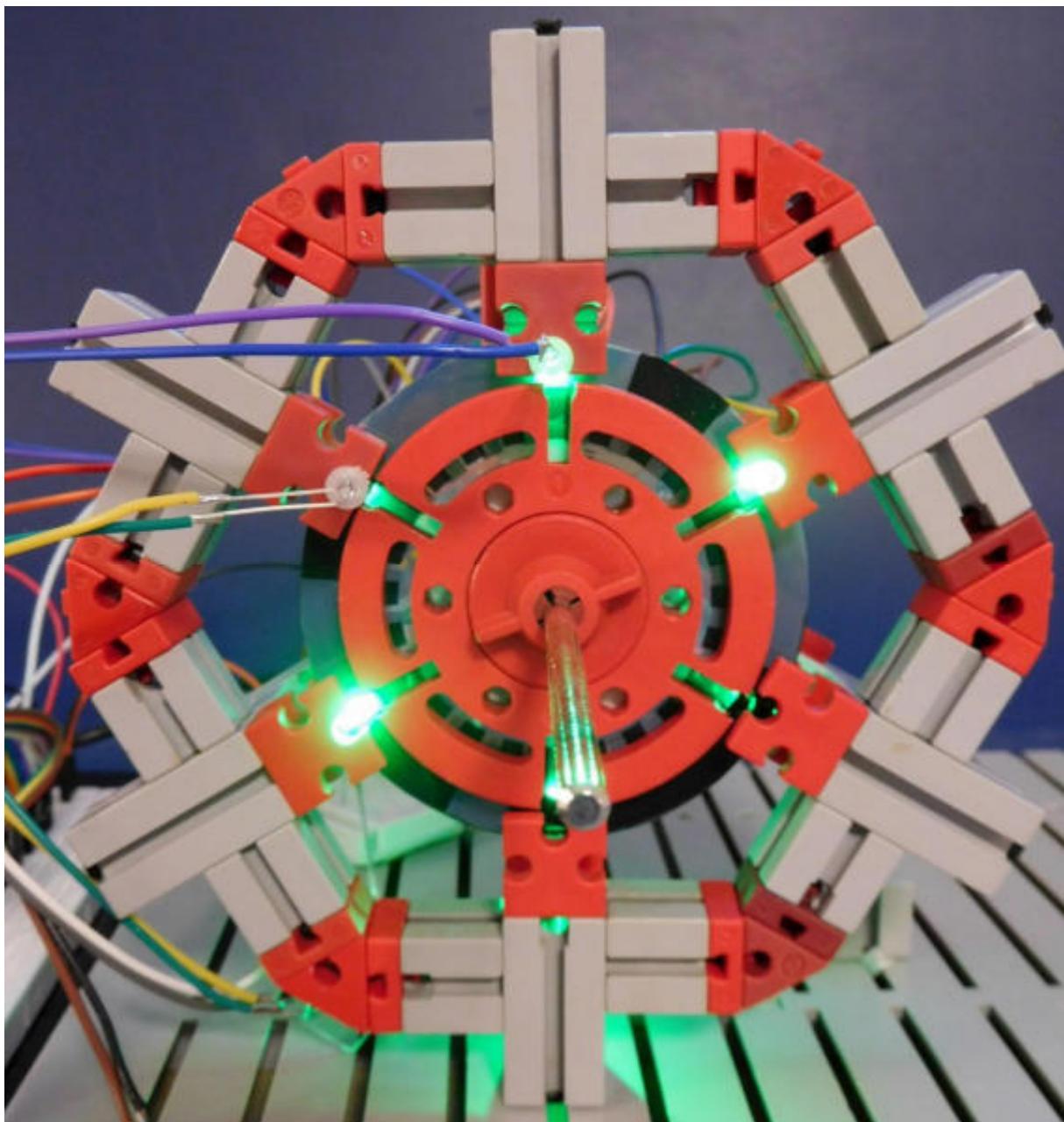


Abb. 7: 6-Bit-STGC-Encoder mit fischertechnik

## Realisierung eines 9-Bit STGC-Encoder mit 360 Stufen

In Abb. 8 ist ein 9-Bit-Encoder gezeigt, der 360 Stufen, also  $1^\circ$ -Schritte auflöst. Zugehörige Encoder- und Maskenscheiben sieht man in Abb. 10.

Um die neunzählige Symmetrie zu realisieren wurde für diesen Encoder ein  $40^\circ$ -Winkelstein in 3D-Druck angefertigt.<sup>2</sup>

<sup>2</sup> Herzlichen Dank an Jan (juh), der mir kurzfristig ein parametrisches Winkelstein-Modell in FreeCAD zur Verfügung gestellt hat [4]. Hier

muss ich mal meiner Begeisterung Ausdruck verleihen, wie hilfsbereit die Leute im fischertechnik-Forum sind.

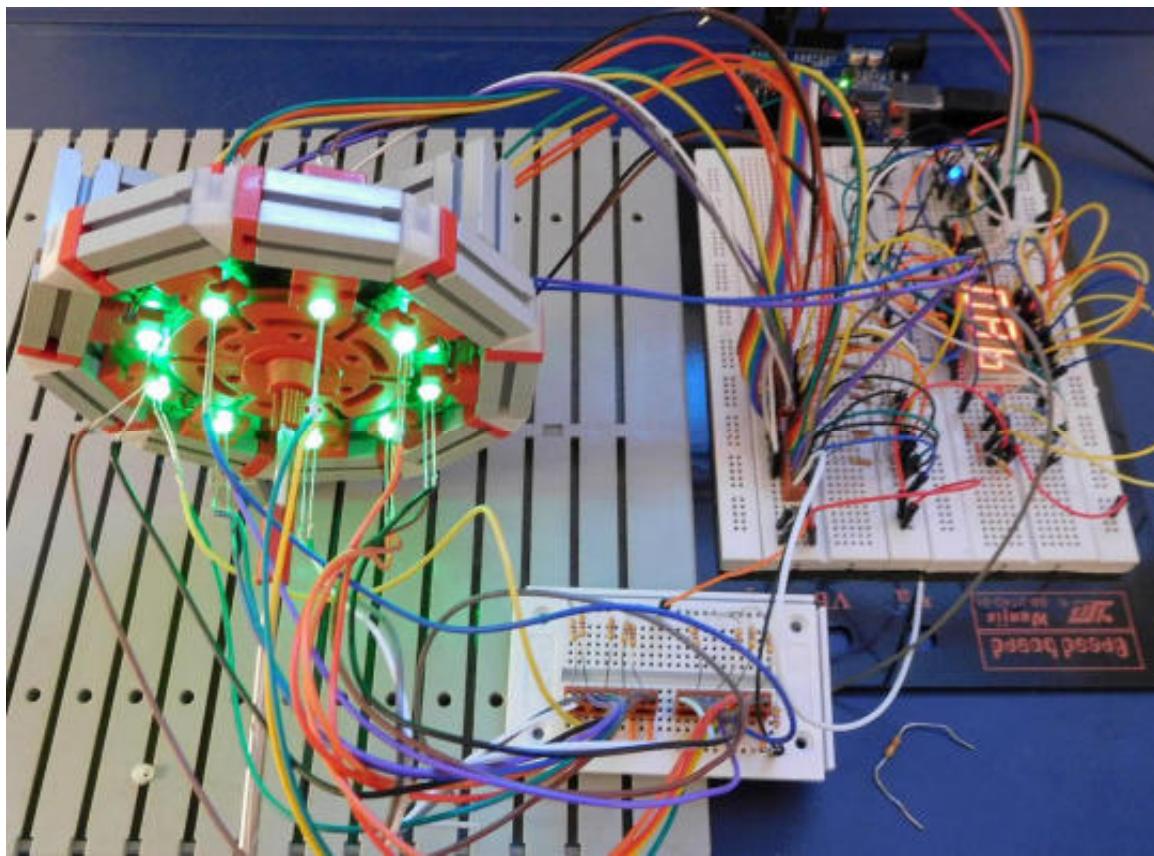


Abb. 8: 9-Bit-Encoder, Emitterseite

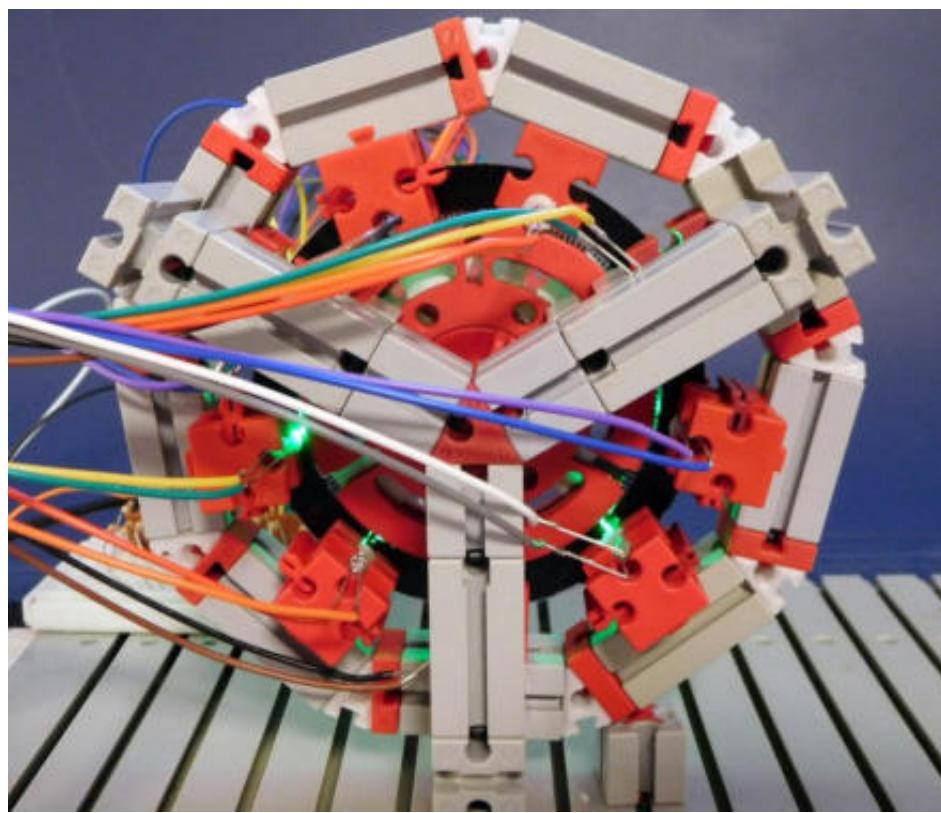


Abb. 9: 9-Bit-Encoder, Empfängerseite

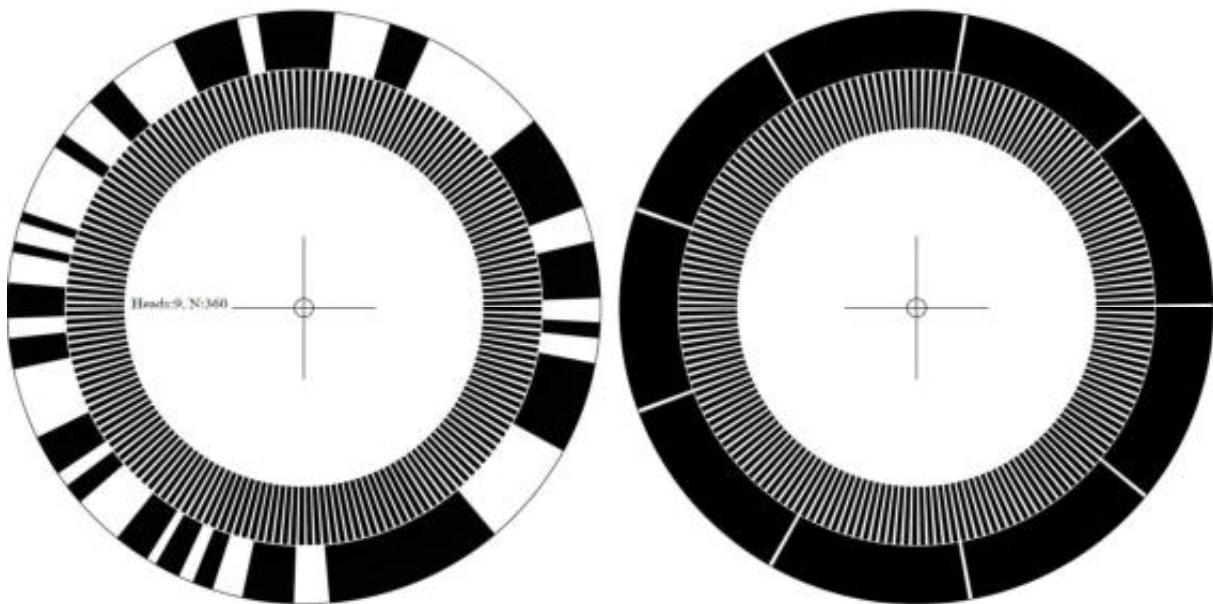


Abb. 10: Code- und Maskenscheibe für 9-Bit-STGC-Encoder mit 360 Schritten

Abb. 8 zeigt einen Blick von oben auf die LED-Emitterseite. Auf der rechten Seite sind die Steckboards mit der Auswerte-Elektronik und der Arduino (oben rechts) zu erkennen.

Da die Hülsen für die Fototransistoren nicht ganz fest sitzen, wurden sie zusätzlich mit roten Lagersteinen fixiert. Oben rechts in Abb. 9 ist einer dieser Steine entfernt und gibt den Blick auf eine Hülse mit einem Fototransistor frei.

## Ausblick

Höhere Auflösungen können mit größeren Bitlängen erreicht werden. Die Herausforderungen sind einerseits die Berechnung von Seed-Code-Sequenzen und andererseits die hohen Anforderungen an die kleinen Strukturen der Encoder-Scheiben und bzw. die sehr hohen Auflösungen der Lichtschranken oder alternativ großen Scheibendurchmesser. Zudem muss auch der mechanische Aufbau stabiler um das Lagerspiel zu verringern, das die effektiv erzielbare Auflösung schnell verringert.

Eine 12-Bit-Decoder-Scheibe mit einer Musterlänge von 3600 (300 Binärworten),

also einer Auflösung von  $0,1^\circ$ , kann berechnet werden. Die Scheibe müsste aber deutlich größer sein als die hier vorgestellten.

Von der [Webseite zu dieser Ausgabe der ft:pedia](#) können die Beispielcodescheiben und ein Programm für deren Erzeugung heruntergeladen werden.

## Referenzen

- [1] Moshe Schwartz, Tuvi Etzion: [The structure of single-track Gray codes](#). IEEE Transactions on Information Theory, 45, 1999, pp. 2383-2396. 10.1109/18.796379.
- [2] Darko Dimitrov, Tomás Dvorák, Petr Gregor, Riste Škrekovski: [Linear time construction of a compressed Gray code](#). European Journal of Combinatorics, 2013. 10.1016/j.ejc.2012.07.015.
- [3] Georgie Botev: [Single-Track Gray Codes: An Efficiently Decodable Maximal Period General Construction](#). Github.com.
- [4] Jan Hanson (juh): [Parametrischer Winkelstein](#).

Computing

## Sensoren am TXT: Die Kamera

Kurt Mexner

Mit verschiedenen Sensoren werden die Möglichkeiten des TXT Discovery Sets erheblich erweitert. Mit dieser Reihe möchte ich einige dieser Sensoren vorstellen. In diesem Beitrag beschäftigen wir uns mit der mitgelieferten USB-Kamera.

### Die Möglichkeiten der Kamera

Die Entwickler des TXT haben der Kamera in ROBO Pro einige erstaunliche sensorische Fähigkeiten beigebracht. Die Kamera erkennt:

- Bewegung
- Licht und Infrarotlicht
- Farbe
- Linien
- eine runde Form
- Geräusche (über das eingebaute Mikrofon)

Natürlich kann sie auch einfach nur Bilder erzeugen und via WLAN/USB zum PC übertragen. Im TXT Discovery Set spielt die Kamera beim Linien- und Ballverfolger eine große Rolle.

### Schalten mit Papier

Hättet ihr gedacht, dass man mit gemalten Schaltknöpfen (Abb. 1) richtig schalten kann?

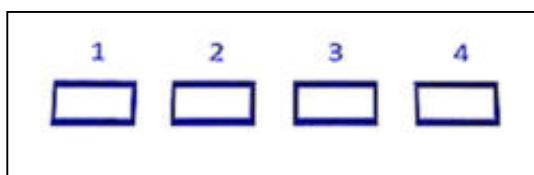


Abb. 1: Gemalte Schaltknöpfe

Des Rätsels Lösung offenbart sich im nächsten Bild: Es gelingt mit einer darüber montierten Kamera. Ein 7,5°-Winkelstein sorgt für eine leichte Schrägstellung:

Dadurch kann die Kamera das davor liegende Blatt besser erkennen (Abb. 2).



Abb. 2: Kamera-Erkennung der Schaltknöpfe

### Einstellung des Sensorfeldes

Nach dem Anklicken des Reiters „Kamera“ wird der Bildschirm der Kamera angezeigt. Hierfür müssen im linken oberen Bereich das Häkchen „Kamera einschalten“ gesetzt und die Kamera am TXT Controller angelassen sein.

Ganz links oben unter „Elementgruppen“ findet sich die Eintragung „Sensorfelder“. Beim Anklicken werden die verschiedenen Sensorfelder angezeigt. Wir klicken das Feld „Bewegung“ an und gehen mit der Maus in den Kamerabildschirm. Der Mauszeiger verwandelt sich in einen Bleistift, mit dem wir jetzt ein Viereck beliebiger Größe in dem Kamerabildschirm platzieren können. Mit einem Doppelklick wird es fertiggestellt.

Obwohl noch kein Programm vorhanden ist, kann man testen, ob die Kamera eine Bewegung erkennt. Dazu wird der Start-Knopf gedrückt. Wenn man nun mit der Hand unter der Kamera eine Bewegung durchführt, wechselt die Farbe des Vierecks von blau zu gelb, und bei Kontrast und Größe werden Werte angezeigt.

Das funktioniert aber nur, wenn eine relativ schnelle Bewegung vorgenommen wird. Eine langsame Bewegung wird nicht erkannt. Die Empfindlichkeit des Sensorfeldes lässt sich einstellen, in dem man das Feld mit der rechten Maustaste anklickt.

Im Programm „4 Lampen als Schalter“ werden auf diese Weise vier Sensorfelder angelegt, die sich genau über den aufgemalten Punkten befinden (Abb. 3). Dabei sollte die Kamera eingeschaltet sein, damit man die Sensorfelder richtig positionieren kann.

## Bedienung

Wenn ein Finger schnell in den aufgemalten Schaltknopf bewegt und schnell wieder herausgezogen wird, wird auf dem Monitor die entsprechende Lampe 1, 2, 3 oder 4 eingeschaltet. Wird der Finger sehr spät wieder entfernt, besteht die Gefahr, dass zusätzlich ein Ausschaltimpuls erzeugt wird. Man benötigt daher für die Bedienung etwas Fingerspitzengefühl.

Durch Einstellung der folgenden Variablen (Grundeinstellungen) kann die optimale persönliche Variante gefunden werden.

- Verzweigungselement A>10: eine Verkleinerung des Vergleichswerts steigert die Empfindlichkeit. Der Nachteil besteht darin, dass dann bereits der Schatten der Hand oder eines Fingers einen Schaltimpuls auslöst.
- Das Element Wartezeit wurde von mir auf 1,4 s eingestellt. Wenn die Wartezeit vergrößert wird, kann man sich mit dem „Rückzug“ des Fingers mehr Zeit lassen.

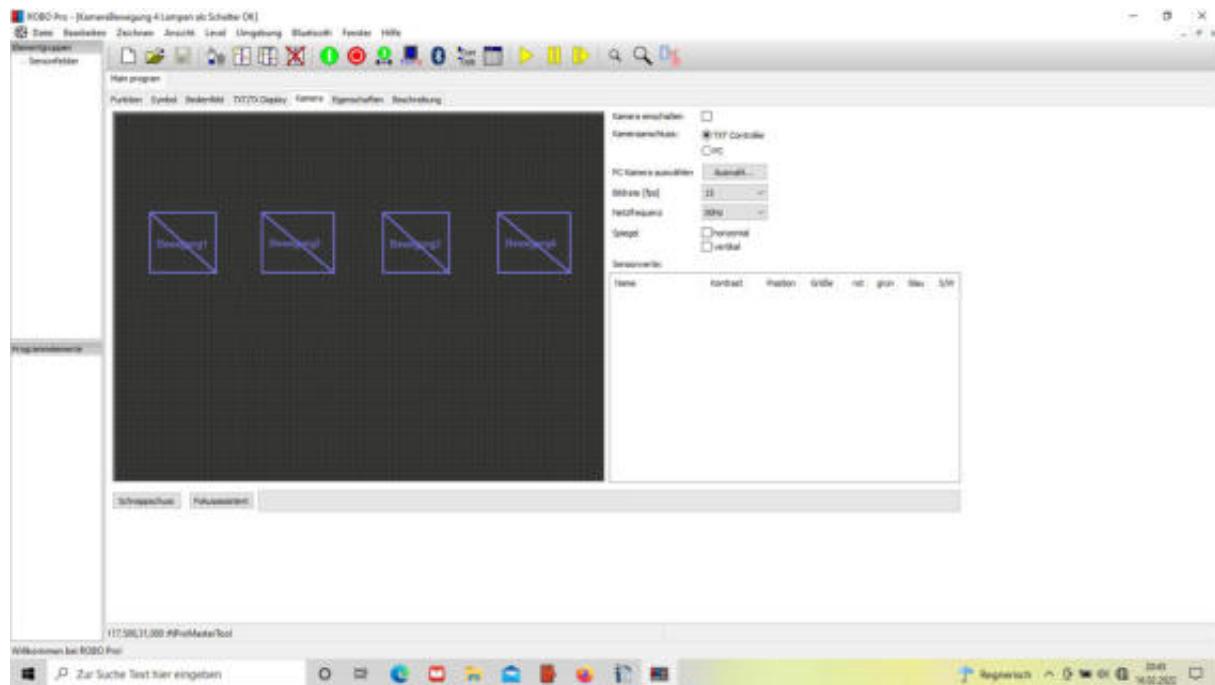


Abb. 3: Auswahl der Sensorfelder in ROBO Pro

- Anklicken des Sensorfeldes im Kamera-Reiter: Es öffnet sich ein Menü, in dem die Stärke und Größe der Bewegung eingestellt werden können.

### Das Programm

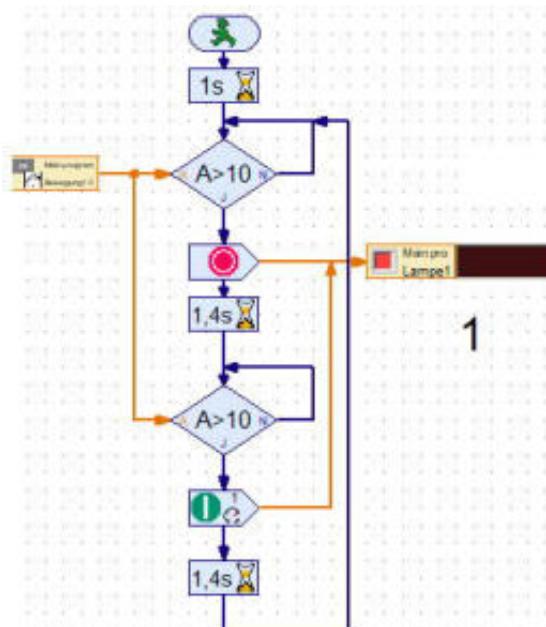


Abb. 4: ROBO Pro-Programm „4 Schalter“

Es werden vier Programme gleichzeitig gestartet. Jedes Programm überwacht ein Sensorfeld, das seinerseits den aufgemalten Schaltknopf im Kamera-Blick hat (Abb. 4).

Ein Verzweigungslement wartet auf einen Wert größer 10 und erzeugt dann den Befehl zum Einschalten eines Motors bzw. einer Lampe. Danach wird 1,4 Sekunden gewartet. Eine neue Bewegung schaltet danach Motor oder Lampe wieder aus.

### Gesten erkennen

Eine kleine Programmänderung und ein Aufstellen der Kamera auf dem Boden mit Blickrichtung zur Decke ermöglichen eine Steuerung des TXT mit Gesten. Wenn man vor der Kamera steht und die linke Hand nach vorne bewegt, wird eine Lampe oder ein Motor eingeschaltet. Mit der rechten Hand kann ein weiteres Gerät eingeschaltet werden. Eine nochmalige Bewegung nach vorne schaltet aus. Mit einer weiten Bewegung von rechts nach links (oder umgekehrt) können auch beide Sensorfelder angesteuert werden.

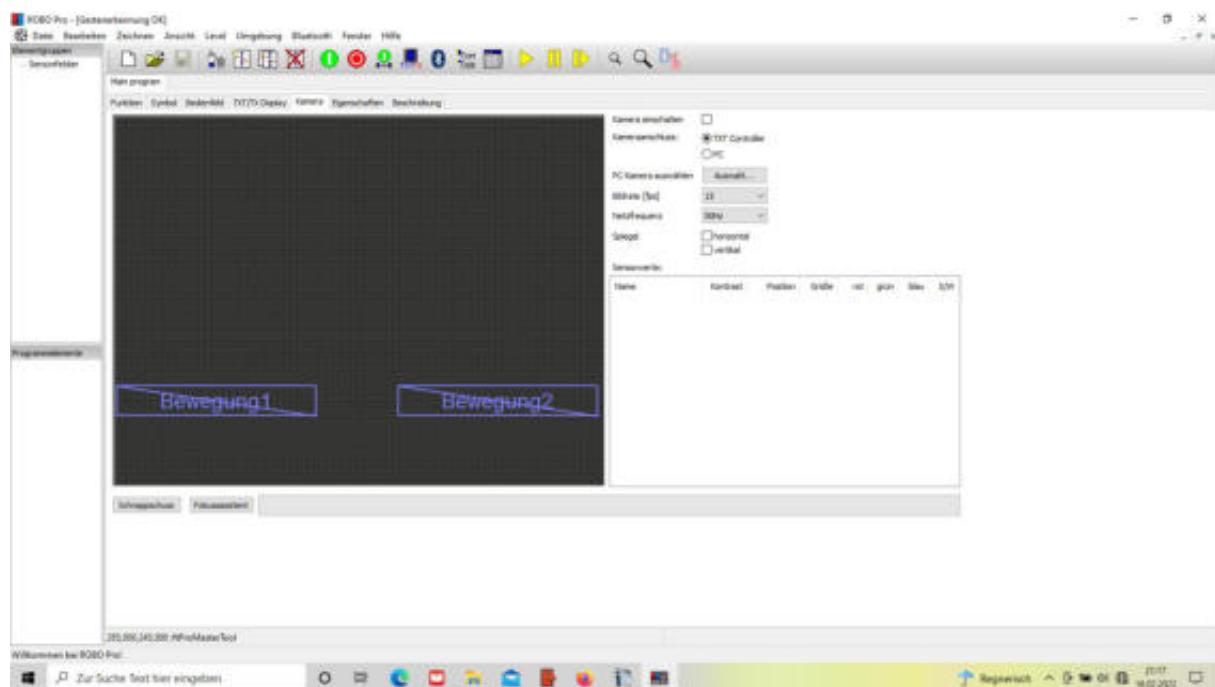


Abb. 5: Anlage der Sensorfelder in ROBO Pro

Im Reiter „Kamera“ werden zwei Sensorfelder angelegt (Abb. 5). Beide Felder sind flach und breit. Das Sensorfeld „Bewegung1“ beginnt ganz links und endet ein gutes Stück vor der Mitte. Das Sensorfeld „Bewegung2“ beginnt ganz recht und endet auch kurz vor der Mitte. Somit entsteht in der Mitte ein Bereich, in dem keine Bewegung wahrgenommen wird. Die folgende Abb. 6 verdeutlicht die Zusammenhänge.

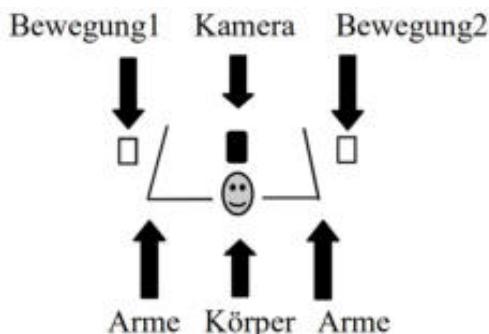


Abb. 6: Kamera und Sensorfelder

### Programmablauf

Es laufen zwei Programme gleichzeitig, jeweils eins für die rechte und eins für die linke Bewegung. Sie müssen beide mit dem Knopf „Ein“ gestartet werden. Das Element

„warten auf“ wartet darauf, dass es von Bewegungselement1 oder Bewegungselement2 einen Wert erhält, der größer 10 ist. Der Nutzer muss somit mit seinen Armen eine Bewegung nach vorne machen. Daraufhin wird eine Lampe eingeschaltet.

ROBO Pro wartet zwei Sekunden, damit es nicht mehrere Befehle erhält und der Nutzer aus Versehen die Lampe ausschaltet. Nach zwei Sekunden schaltet eine erneute Bewegung die Lampe wieder aus.

Das Programm „Gestenerkennung“ (Abb. 7) kann über die [Webseite zu dieser Ausgabe der ft:pedia](#) heruntergeladen werden.

### Anwendungen

Das Programm ermöglicht, durch eine Bewegung einen Motor, eine Lampe oder einen anderen Aktor zu steuern. Es wäre auch möglich, komplexere Programmabläufe zu kontrollieren: z. B. einen Greifarm, der nacheinander Objekte aufnimmt und an unterschiedlichen Stellen wieder ablegt. Das Weiterschalten zu einem nächsten Programmschritt könnte durch Gesten gesteuert werden.

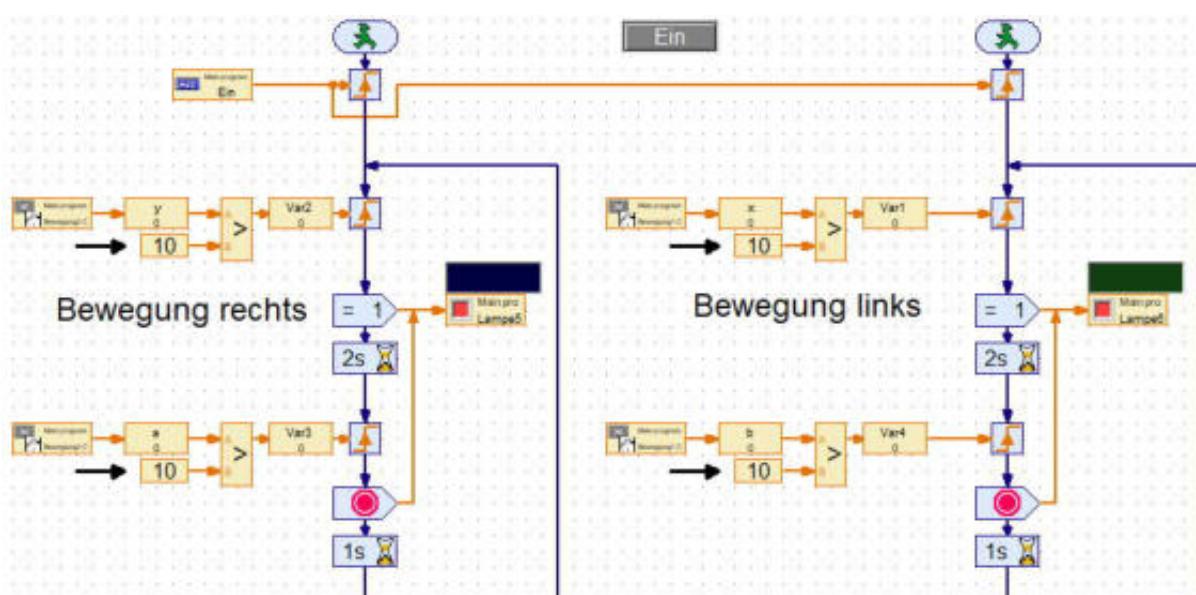


Abb. 7: ROBO Pro-Programm „Gestenerkennung“

Ich habe noch ein weiteres Hobby, die Zauberei. Ich arbeite daran, fischertechnik dort ebenfalls zu nutzen. Denkbar wäre eine Kiste, deren Deckel sich hebt und dann ein Pumuckl herauskommt. Auch Donner oder Blitz könnten von einer Bewegung ausgelöst werden.

## Referenzen

Bisher sind in der Reihe „Sensoren am TXT“ die folgenden Beiträge erschienen:

- [1] Kurt Mexner: *ROBO Pro wird sensibel - Lichtempfindliche Widerstände*. [ft:pedia 3/2021](#), S. 67–69.
- [2] Kurt Mexner: *Sensoren am TXT: IR-Dioden und -Transistoren*. [ft:pedia 4/2021](#), S. 31–34.

Computing

## Tuning der Mecanum-Roboter

Dirk Fox

Der neue fischertechnik-Baukasten „Robotics Hightech“ enthält mehrere Modelle mit den Mecanum-Wheels, angetrieben von je einem Encoder-Motor. Eine vielversprechende Innovation – allerdings lässt die Konstruktion der Roboter „Luft nach oben“, insbesondere bei der Geschwindigkeit.

### Hintergrund

Mit den Mecanum-Rädern hat fischertechnik 2021 eine spannende Neuigkeit auf den Markt gebracht [1].

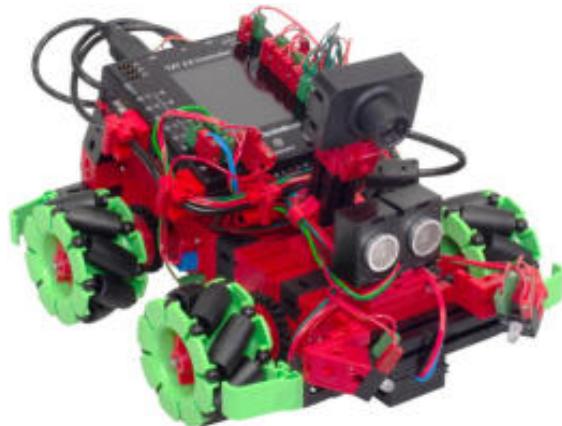


Abb. 1: Mecanum-Roboter aus dem Baukasten „Robotics Hightech“ (Bild: fischertechnik)

Die Mecanum-Roboter (Abb. 1) treten in direkten Wettbewerb mit dem chinesischen Drohnenhersteller und Weltmarktführer [DJI](#) (Da-Jiang Innovations Science and Technology Co., Ltd), der seit 2019 mit seinem „Robomaster“ (Abb. 2) auf den Maker- und Bildungsmarkt zielt. Am gleichnamigen [Robotik-Wettbewerb](#), den DJI 2013 initiierte, nahmen 2020 weltweit über 500 Teams teil. Für diesen Wettbewerb entwickelte DJI den Roboter-Bausatz (Abb. 2). Die beiden Robomaster-Modelle S1 und EP verfügen über eine Kamera, vier Abstandssensoren und können mit handelsüblichen (I<sup>2</sup>C-) Sensoren erweitert werden.

Die Ansteuerung und Programmierung erfolgt wie bei fischertechnik über eine WLAN-Verbindung. Es gibt eine passende [Erweiterung für Scratch](#) und ein [Software Development Kit \(SDK\) für Python](#).



Abb. 2: Robomaster EP (Bild: DJI)

Die [Videos](#), die DJI zum Robomaster ins Netz gestellt hat, sind beeindruckend – und ansprechend. Auch die technischen Daten des Roboters lassen staunen – vor allem seine Geschwindigkeit von bis zu 3,5 m/s bei Vorwärtsfahrt (etwa 12,5 km/h) und 600°/s beim Drehen um die eigene Achse.

Der Controller steuert neben den vier 12V-Antriebsmotoren auch mehrere Servos – für die Bewegung eines Roboterarms mit Greifer (Modell EP) oder einer Abschuss-einrichtung für Gelkugelchen (Modell S1).

Allerdings kann DJI seine Herkunft als Produkthersteller nicht verleugnen: Der Roboter ist mit 3,3 kg sehr schwer, und für den Zusammenbau benötigt man mehrere Stunden – Erfahrung im Modellbau vorausgesetzt. Auch ist der Preis mit rund 550 € (S1) bzw. 950 € (EP) durchaus üppig. So ist der Roboter denn auch mehr Spiel- als Lernobjekt: Konstruktionsvarianten sind nur als Anbau vorgesehen, und der Zusammenbau des Chassis selbst ist nicht Teil des Lernprozesses.

Hier kann fischertechnik punkten: Für etwa die Hälfte des Preises (Smarttech ca. 190 €, Hightech ca. 475 €) erhält man einen Baukasten, der die Konstruktion zahlreicher Modelle ermöglicht – ebenfalls erweiterbar um bis zu drei Servos und zahlreiche I<sup>2</sup>C- und fischertechnik-Sensoren. Der Aufbau der Modelle geht (Erfahrung mit fischertechnik vorausgesetzt) sehr fix von der Hand; Um- und Anbauten sind möglich und erwünscht. Auch das Gewicht ist deutlich niedriger: Der fischertechnik-Roboter bringt inklusive Akku lediglich 1,25 kg auf die Waage.

Die fischertechnik-Mecanum-Räder haben allerdings nur acht Rollen (DJI: 12), daher „gleitet“ der Roboter nicht ganz so elegant wie der Robomaster über den Untergrund. Doch vor allem bei der Geschwindigkeit bleibt das fischertechnik-Modell weit hinter den Robomaster-Modellen zurück: Gefühlt „schleicht“ der Roboter im Vergleich mit seinen fixen Konkurrenten. Das verringert den Spielspaß deutlich und begrenzt auch die Möglichkeiten für den Schul- oder Wettbewerbseinsatz.

Aber: Ist die Konstruktion der fischertechnik-Roboter tatsächlich schon das Ende der Fahnenstange – oder kann man da noch etwas herausholen? Denn anders als bei DJI sind die Mecanum-Roboter kein unveränderbares Modell, sondern eine von zahlreichen Konstruktionsvarianten.

## Ansatzpunkte

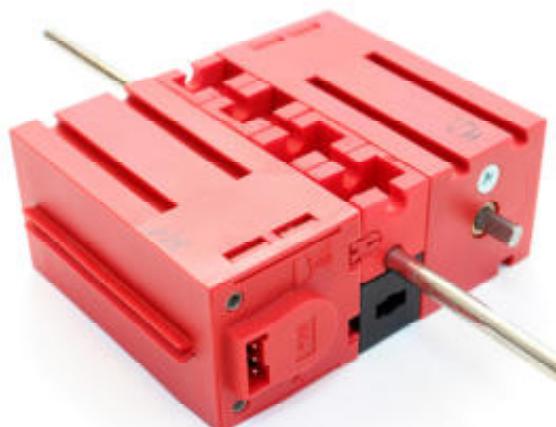
Die Geschwindigkeit des Roboters können wir vor allem durch drei Maßnahmen erhöhen:

- Verringerung der Verluste (Reibung)
- Verringerung des Gewichts
- Erhöhung der Achsdrehzahl

### Verringerung der Verluste

Verluste entstehen vor allem an zwei Stellen: in den Achslagern und im Getriebe. Reibung im Getriebe lässt sich vermeiden, wenn man die Mecanum-Räder (wie bei vielen Mecanum-Modellrobotern) direkt an der Motorwelle befestigt. Da die Welle aber nur mit Rastachsen verlängert werden kann, benötigt man ein Außenlager für jede Achse, damit die sich nicht durchbiegt. Das würde zusätzliche (statt weniger) Reibung verursachen. Ideal wäre daher eine „Rastachsen-Nabe“, die auf die Welle gesteckt werden könnte – vielleicht ist das ja eine Idee für ein neues fischertechnik-Bauteil.

Bleibt die Reibung der Achslager. Im Originalmodell von fischertechnik drehen die Achsen in je zwei Rollenlagern ([37636](#), Abb. 3). Das macht die Konstruktion sehr stabil, wenn einem Ästheten auch aufstoßen dürfte, dass die Achsen etwa 2 mm oberhalb der Motorwelle liegen.



*Abb. 3: Lagerung der Achsen im Originalmodell von fischertechnik*

Je nach Fertigungstoleranzen kann es hier aber zu deutlich messbarer Reibung kommen. Für die Lagerung der Achsen würde jeweils ein Rollenlager ausreichen. Immerhin kann man die Reibung reduzieren, indem man die Achsen und Rollenlager sorgfältig auswählt (die Lager sollten frei auf der Achse drehen) und die Achsen so montiert, dass sie nur in je drei Führungen aufliegen.

### **Verringerung des Gewichts**

Etwa 70% des Gewichts des Mecanum-Roboters (878 g) lassen sich nicht beeinflussen: Wir brauchen die vier Mecanum-Räder (137 g), die Motoren (4 x 106 g), den Akku (209 g) und den Controller (108 g).<sup>3</sup> Man könnte den Controller durch einen leichten Arduino (75 g) oder einen ftDuino (95 g) ersetzen, aber die Einsparung wäre mit 10-30 g eher mickrig.

Immerhin können wir auf einige etwas „gewichtigere“ Bauelemente verzichten oder sie durch leichtere Bausteine ersetzen. Dazu zählen vor allem die massiven Grundbausteine, die man alle weglassen oder durch Bausteine 15 Eck bzw. Winkelträger 15 ersetzen kann. Damit sparen wir etwa 100 g Gewicht ein. Die Metallachsen durch Rastachsen zu ersetzen würde ebenfalls helfen, aber den Antrieb labil machen.

Immerhin kann man mit einer leichten Anpassung der Antriebskonstruktion etwas kürzere Achsen wählen. Schließlich kann man noch auf ein wenig „Schnickschnack“ in den An- und Aufbauten verzichten; das spart weitere 100 g.

Die insgesamt erreichbare Gewichtsreduktion liegt damit bei etwa 200 g. Das ist ein nennenswerter Teil (rund 16 %) des Gesamt- und über die Hälfte des beeinflussbaren Gewichts. Die Auswirkung auf die Geschwindigkeit ist jedoch eher marginal,

solange das Fahrzeug keine Steigung bewältigen muss (und das ist mit Mecanum-Wheels ohnehin schwierig). Die Gewichtsreduktion hilft ein wenig; eine deutliche Auswirkung auf die Geschwindigkeit sollte man sich von ihr aber nicht versprechen.

### **Erhöhung der Achsdrehzahl**

Den entscheidenden Effekt können wir uns von einer Erhöhung der maximalen Achsdrehzahl erhoffen. Das können wir auf zwei Wegen bewirken:

- Verbesserung der Effizienz der Motoren
- Verwendung stärkerer Motoren

Um festzustellen, ob wir bei den Encoder-Motoren Spielraum für Effizienzsteigerung haben, müssen wir prüfen, ob die Motoren bei der aktuellen Konstruktion bereits im optimalen Leistungsbereich arbeiten.

Wie das geht hat René Trapp 2015 in einem sehr schönen Beitrag in der ft:pedia erklärt [2]. Mit Motorkennlinien und einer Messung des Stroms unter Last können wir den Antrieb so optimieren, dass die Motoren möglichst wenig Verlustleistung (sprich: Wärme) erzeugen, ohne dabei „unter ihren Möglichkeiten“ zu bleiben – der Nennbetrieb. Nun ist die Messung des Stroms bei einem fahrenden Roboter nicht so einfach. Daher greifen wir auf die von René vorgestellte Faustregel zurück: Der Motor läuft im Nennbetrieb, wenn sich die Motorwelle mit etwa 75% der Leerlaufdrehzahl dreht.

Im Leerlauf konnte ich bei den roten Encodern Drehzahlen von 155-180 RPM messen. Damit liegt der optimale Wirkungsgrad des Motors bei etwa 116-135 U/min. Wenn die Drehzahl der Motoren höher liegt, ist bei der Last noch „Luft nach oben“.

Alternativ könnten wir stärkere Motoren verwenden, beispielsweise die Power-Mo-

<sup>3</sup> Tatsächlich könnten wir die Motoren reduzieren – wie das geht, wäre ein Thema für einen eigenen

Beitrag –, aber damit würden wir auch die Kraft des Antriebs verringern.

toren. Die lassen sich allerdings in Erman-gelung eines Gehäuses schlecht verbauen. Für das „Competition Set“ hat fischertechnik gerade einen [schwarzen Encoder-Motor](#) (186175) mit 0,61 Nm und 230 RPM im Nennbetrieb herausgebracht; bei Santjo-hanser ist er bereits für 24 € [erhältlich](#).

## Vorarbeiten

Um verschiedene Modellvarianten testen zu können habe ich ein kleines [Testprogramm in ROBO Pro](#) geschrieben, das die verschiedenen Bewegungsarten des Mecanum-Roboter über eine Fernsteuerung (ROBO Pro-Bedienfeld) im Online-Mode via WLAN zugänglich macht (Abb. 4):

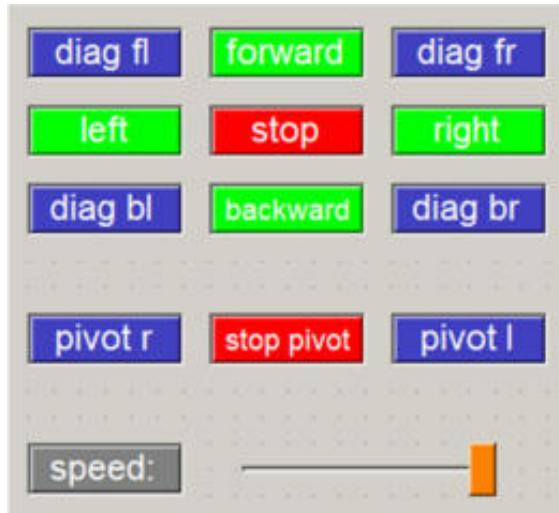
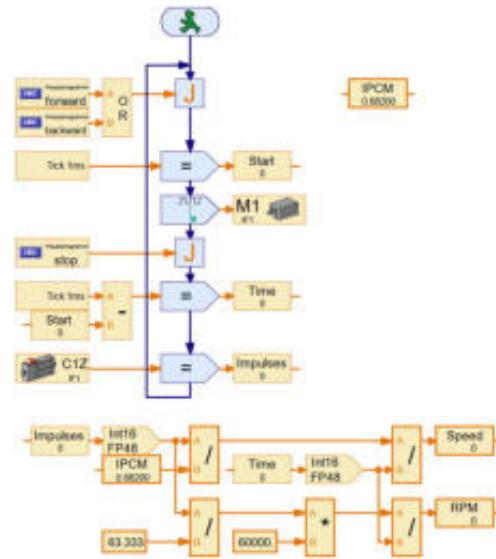


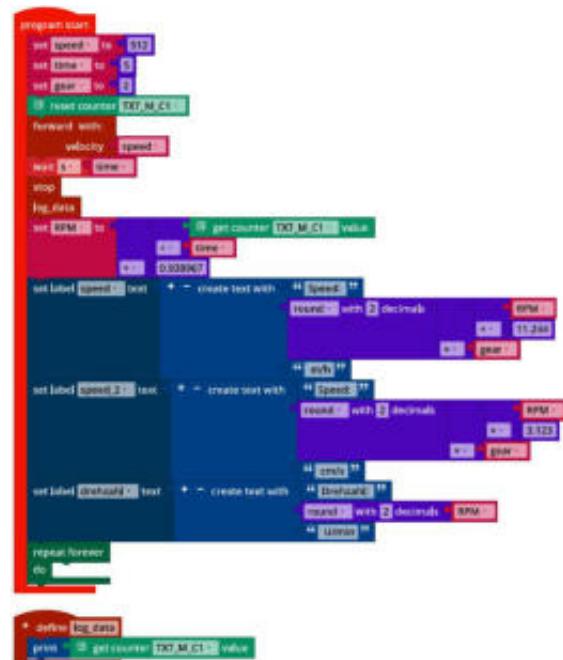
Abb. 4: ROBO Pro-Fernsteuerung für Mecanum-Roboter

Ergänzt habe ich das Programm um eine Stoppuhr, die bei Geradeausfahrt des Robo-ters die verstrichene Zeit misst. Die dabei gezählten Impulse des Encoder-Motors M1 teile ich durch die Anzahl Impulse, die beim Mecanum-Roboter bei einer Geradeaus-fahrt über eine Strecke von 1 cm Länge anfällt, und das Ergebnis teile ich wiederum durch die Messung der Stoppuhr (in ms). Das liefert mir die Durchschnittsgeschwin-digkeit der Fahrt. Aus Impulszahl und Stoppuhr bestimme ich zugleich die Drehzahl der Motorwelle (RPM) (Listing 1).

Dasselbe funktioniert natürlich auch in ROBO Pro Coding für den TXT 4.0 ([Listing 2](#)). Die Zahl der Impulse (IPCM), die der Encoder für 1 cm Fahrstrecke benötigt (resp. liefert), habe ich experimentell über eine Teststrecke von zwei Metern Länge bestimmt. Bei der originalen fischertechnik-Konstruktion sind das 0,682 Impulse.



Listing 1: Stoppuhr, Drehzahl- und Geschwindigkeitsmessung in ROBO Pro



Listing 2: Drehzahl- und Geschwindigkeitsmessung in ROBO Pro Coding

## Modellvarianten

Alle im Folgenden vorgestellten Modellvarianten habe ich sowohl mit dem Robotics TXT Controller als auch mit dem TXT 4.0 gesteuert und jeweils Geschwindigkeit und Umdrehungszahl gemessen. Mit dem TXT sind die Modelle rund 50 g schwerer als die Varianten mit dem TXT 4.0; die Messergebnisse unterscheiden sich auch deshalb ein wenig.

Um Störeinflüsse durch unterschiedliche Ladestände der Akkus auf die Messergebnisse auszuschließen, habe ich die Roboter mit einer 9V-Stromversorgung über ein langes fischertechnik-Kabel versorgt.

### Originalversion

Als Referenz habe ich zunächst die Geschwindigkeit und die Drehzahl (RPM) des originalen Baukasten-Antriebs gemessen. Zuvor habe ich das Gewicht des Roboters bestmöglich reduziert und versucht, beim Zusammenbau der Achslagerung die Reibung zu minimieren. Die Z20 habe ich mit einem Abstanderring montiert und das Rast-Z10 nicht bis zum Anschlag aufgesteckt.

Das Ergebnis ist ernüchternd: Mit 0,18 m/s ( $\approx 0,64 \text{ km/h}$ ), einem 20stel der Höchstgeschwindigkeit eines vorbeifahrenden Robomaster, sieht das fischertechnik-Modell mit TXT bestenfalls dessen Kondensstreifen. Die TXT-4.0-Variante ist ein klein wenig schneller; sie erreicht 0,22 m/s ( $\approx 0,8 \text{ km/h}$ ) – immer noch Schnekkentempo.



Abb. 5: Leicht modifizierte Originalversion des Mecanum-Antriebs

Aber die Motorwellen drehen mit ca. 144 RPM oberhalb des optimalen Bereichs – und vertragen damit also noch etwas zusätzliche Last.

### Variante 1

In einer ersten Konstruktionsvariante habe ich die 2:1-Untersetzung ins Langsame durch einen 1:1-Antrieb (Z20 auf Z20) ersetzt. Dazu musste ich den Abstand zwischen den Motorblöcken auf zwei Baulängen (= 30 mm) vergrößern. Die Achsen habe ich dafür in Rollenböcken ([32085](#)) gelagert. Um die Achsen zusätzlich zu stabilisieren habe ich statt zwei separaten Achsen ein altes Differential ([31043](#)) eingebaut (Abb. 6). Natürlich funktioniert die Konstruktion auch mit getrennten Achsen.

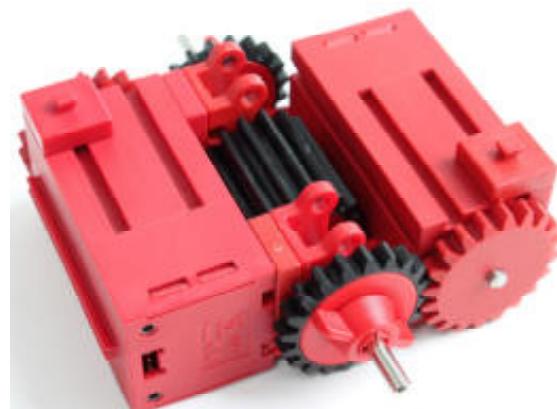
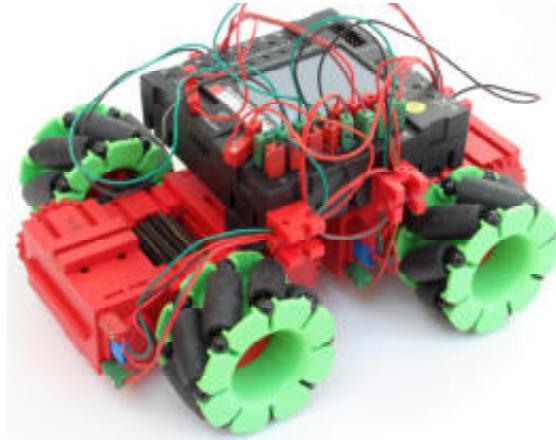


Abb. 6: [Antriebsvariante 1 mit Differential](#)

Auf die Motorachse habe ich jeweils ein Rast-Z20 aufgesteckt und auf den Achsen je ein Z20 mit Nabe montiert. Mit dieser Modifikation ist also eine Verdoppelung der Geschwindigkeit des Roboters zu erwarten. Abb. 7 zeigt die Gesamtansicht des Roboters mit allen vier Motoren.

Und tatsächlich: Der von einem TXT gesteuerte Roboter (Abb. 7) erreicht bei meinen Messungen eine Geschwindigkeit von 35 cm/s ( $\approx 1,26 \text{ km/h}$ ) mit 120 RPM. Mit dem TXT 4.0 konnte ich sogar 52 cm/s ( $\approx 1,88 \text{ km/h}$ ) messen; die Umdrehungszahl der Motorachse stieg dabei auf 167 RPM.

Das ist immer noch nur ein Siebtel der Höchstgeschwindigkeit des Robomaster S1, aber eine erkennbare Verbesserung.



*Abb. 7: Konstruktionsvariante 1  
(Gesamtansicht)*

## Variante 2

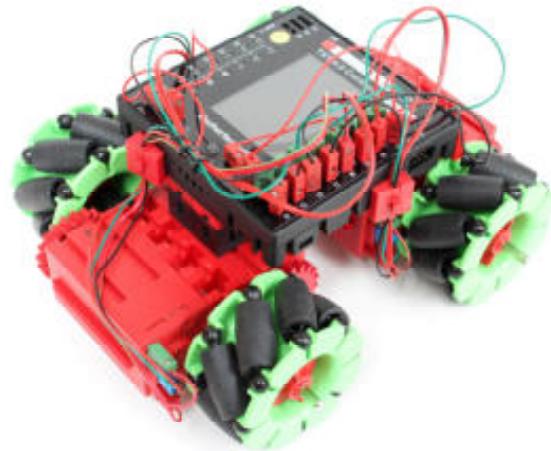
Die Messergebnisse der ersten Konstruktionsvariante motivierten mich zur Ersetzung der 1:1- durch eine 1:2-Übersetzung ins Schnelle. An dem originalen Antriebsmodul von fischertechnik musste ich dafür gar nicht viel ändern: Wenn man das Rast-Z10 auf der Motorwelle durch ein Rast-Z20 ersetzt und das Z20 mit Nabe durch ein Z10 (Abb. 8), sollte sich die Geschwindigkeit des Fahrzeugs gegenüber dem Original etwa vervierfachen.



*Abb. 8: Antriebsvariante 2 mit einer  
Übersetzung 1:2 ins Schnelle*

Das erreicht die Modellvariante auch fast: Die Geschwindigkeit des vom TXT gesteuerten Roboters stieg auf 0,65 m/s ( $\approx 2,34$  km/h) bei 106 RPM – schon eine spürbare zusätzliche Last. Mit einem TXT 4.0

erreichte das Modell 76 cm/s ( $\approx 2,75$  km/h) bei 122 RPM; der Anstieg der Motorlast ist erkennbar. Damit arbeitet der Motor aber immer noch im Nennbereich. Die Maximalgeschwindigkeit der Modellvariante 2 (Abb. 9) erreicht also etwa ein Fünftel der Höchstgeschwindigkeit des Robomaster.



*Abb. 9: Konstruktionsvariante 2  
(Gesamtansicht)*

Beim Zusammenbau des Antriebsmoduls muss man darauf achten, dass zwischen dem Rast-Z20 und dem Motor etwas Platz bleibt, damit es nicht am Gehäuse reibt.

## Weitere Optionen

Motiviert von den Resultaten habe ich auch mit einer Übersetzung 1:3 ins Schnelle experimentiert. Der Geschwindigkeitsgewinn ist allerdings gering, da die Umdrehungszahl spürbar sinkt und der Motor stark in die Überlast geht. Keine gute Idee also. Weitere Geschwindigkeitssteigerungen sind daher nur mit stärkeren Motoren zu erreichen.

## Referenzen

- [1] Dirk Fox: *Mecanum-Räder und Omniwheels*. [ft:pedia 1/2021](#), S. 66–74.
- [2] René Trapp: *fischertechnik-Motoren richtig betreiben*. [ft:pedia 3/2015](#), S. 34–38.
- [3] fischertechnik: [\*Datenblatt Encodermotor 9V, Art.-Nr. 53422\*](#). FT-T-KN, 15.08.2017.



*Sechsbeiniger Skorpion aus dem mit dem Toy Award ausgezeichneten Baukasten „Animal Friends“*



*Skorpion von Rüdiger Riedel ([ft:pedia 1/2021](#))*