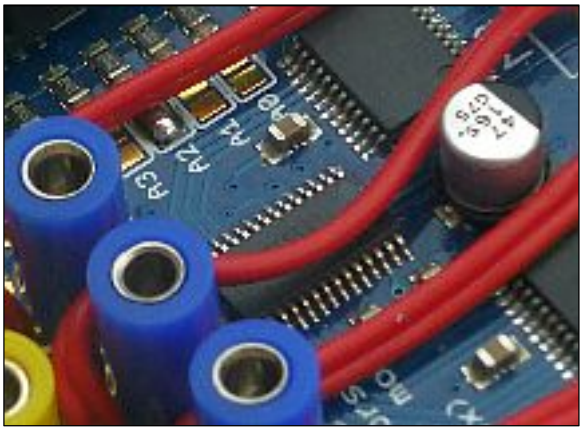
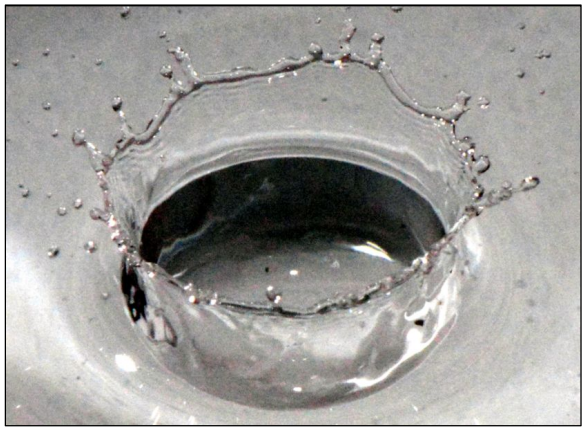
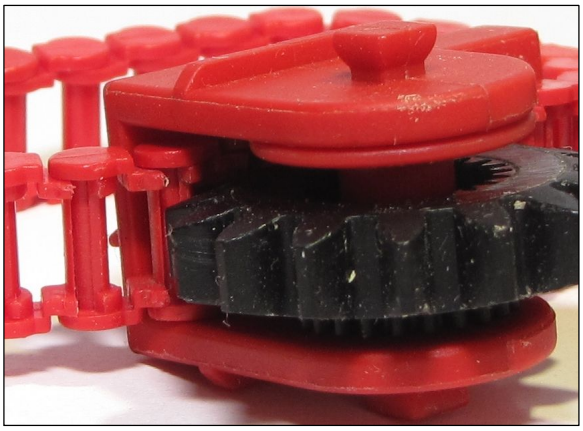
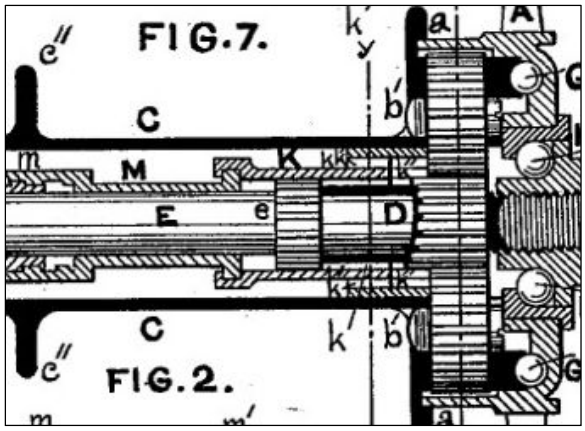
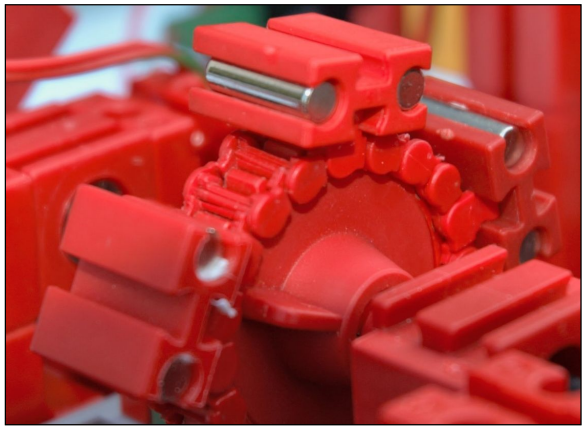
ft:pedia

Heft 2/2016

Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

Die Macher kommen

Dirk Fox, Stefan Falk

„DIY“ (Do-it-Yourself) ist ein Megatrend. In den USA zeichnete sich das bereits um das Jahr 2005 ab, als die „Maker-Communities“ sich zu organisieren begannen und die ersten Maker Faires ihre Tore öffneten. Rund vier Jahre später entstand das erste deutsche FabLab an der RWTH Aachen; heute gibt es bereits mehr als 40 solcher Mini-Fabriken und Maker-Treffpunkte in Deutschland. Zugleich verbreiteten sich Reparatur-Cafés; über 500 wurden Anfang 2016 in Deutschland gezählt. Und 2012 startete der Heise-Verlag die Zeitschrift „c’t Hacks“ (inzwischen: „Make:“) mit zunächst vier und heute sechs Ausgaben pro Jahr. Ein Jahr darauf initiierte Heise die Maker Faire in Hannover. Gut 4.000 Besucher zog die Macher-Messe im ersten Jahr an, 2014 waren es bereits 9.000, 2015 über 10.000 und im Mai 2016 zählte man schon mehr als 16.000 Besucher – eine Vervierfachung in nur drei Jahren, nicht gerechnet die Besucher der „Schwestermessen“ in Berlin, Köln und am Bodensee.

Über die Ursache der DIY-Bewegung lässt sich trefflich spekulieren. Hat die „Generation Kopf unten“ versehentlich das Haupt gehoben und entdeckt, dass das Leben mehr zu bieten hat als virtualisierte Welten? Oder ist sie die monotone Wischerei leid?

Vielleicht ist es aber auch eine Antwort auf die zunehmende Passivität, die uns eine immer künstlichere und intransparentere Welt um uns herum aufzwingt. Die Milch kommt schon lange aus der Tüte und die Pizza aus der Tiefkühltruhe; längst kann man den Fön nicht mehr reparieren und neuerdings lässt sich nicht einmal mehr der Akku des Smartphones tauschen – vom Schrauben am Auto oder dem Frisieren des Mofas (für die Digital Natives: nein, dafür braucht man keinen Friseur) ganz zu schweigen.

Für ein Wesen, das sich als „Krone der Schöpfung“ versteht, ein gewöhnungsbedürftiger Zustand: Die schöne neue Produktwelt beherrscht uns und zwingt uns ihren Willen auf. Der Maker-Trend ist ein Aufstand der Entmündigten: Kollektiv können wir noch mithalten und die Welt mit eigenen Kreationen überraschen. Befeuert wird dieser Trend von technischen Innovationen wie bezahlbaren 3D-Druckern, leistungsfähigen Mini-Computern (wie Arduino oder Raspberry Pi) und einem Riesenangebot an billigen Sensoren.

Jetzt hat die Welle auch die fischertechniker erreicht. Zwar erobern wir uns schon seit vielen Jahren mit Funktionsmodellen das Weltwissen zurück – aber dass die Community innerhalb von nur drei Monaten eine neue TXT-Firmware bereitstellt, der Arduino sich zum „Einsteiger-Controller“ mausert und nun sogar fischertechnik mit einem Selbstbau-3D-Drucker den Maker-Markt ins Visier nimmt, ist eine neue Entwicklung. Freuen wir uns also auf viele interessante Überraschungen, die die Symbiose aus fischertechnikern und Makern in Zukunft noch hervorbringen wird.

Beste Grüße,
Euer ft:pedia-Team

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

Inhalt

Termine

Was?	Wann?	Wo?
fischertechnik FanClub-Tag	24.07.2016	Waldachtal (Tumlingen)
fischertechnik-Convention 2016	24.09.2016	Dreieich (bei Frankfurt)
Clubdag Schoonhoven	29.10.2016	Schoonhoven (NL)
Modellschau in Münster	20.11.2016	Kardinal von Galen Gymnasium (Münster)

Impressum

http://www.ftcommunity.de/ftpedia

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

Autoren: Christian Bergschneider, Daniel Canonica, Matthias Dettmer, Roland Enzenhofer, Stefan Falk, Dirk Fox, Stefan Fuss, Andreas Gail, David Holtz, Raphael Jacob, Jens Lemkamp, Thomas Püttmann, Harald Steinhaus, René Trapp, Dirk Uffmann, Martin Westphal, Dirk Wölffel.

Modell

Mini-Modelle (Teil 12): Mondrakete

Stefan Falk

Nachdem es schon einen über 20 m hohen Turm auf der Convention gab [1] und für den nächsten Fan-Club-Tag eine gigantisch große Brücke angekündigt wurde [2], muss also ein richtiges Monstermodell her: Der fischertechnik-Nachbau einer 110 m hohen Mondrakete [3]!

Für's fantasievolle Spiel im Kinderzimmer wurde die große Apollo-Rakete auf einen etwas praktischeren Maßstab gebracht:

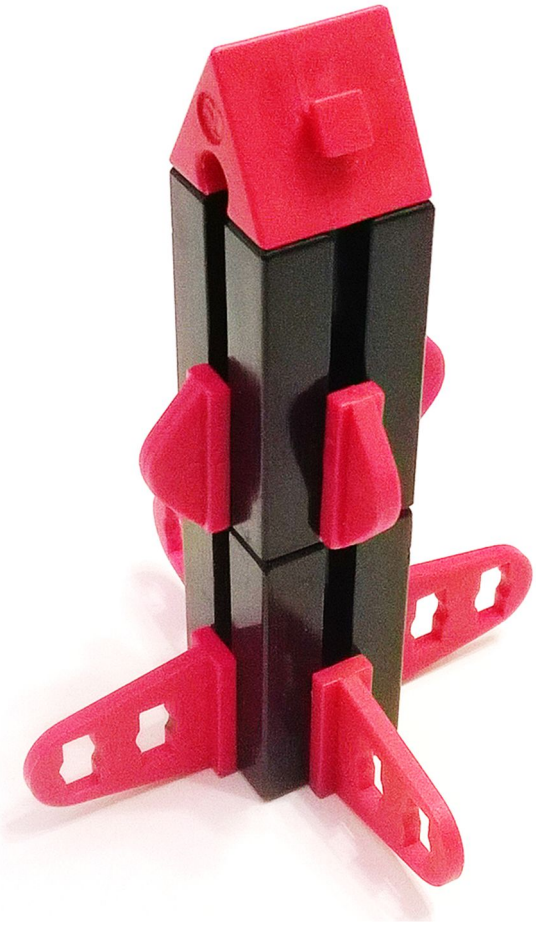


Abb. 1: Für winzige Menschen oder Autos zum Größenvergleich war auf diesem Bild einfach kein Platz mehr

Natürlich hat die Rakete *alles,* um bis mindestens zum Mond und zurück zu gelangen. Ihr mehrstufiges Konzept ist schamlos von der Saturn-Rakete [3] geklaut:

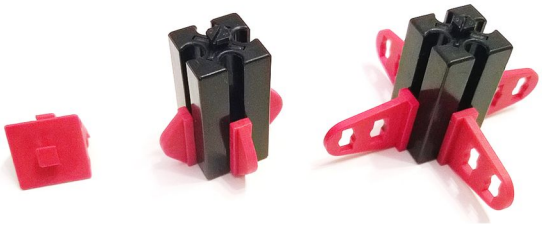


Abb. 2: Raumkapsel, zweite und erste Stufe der Rakete

Hier die Stückliste für all die Astronauten da draußen:

St.	ft-Nr.	Bezeichnung
2	32879	Baustein 30
1	31010	Winkelstein 60°
4	38253	S-Kupplung 15 2
4	31602	Kufe

Referenzen

[1] Stratmann, Michael: *20-m-Turm.* ft-Community-Website, 2014.

[2] Haizmann, Dirk: *Wir brauchen Hilfe.* Aufruf zur Brückaufbau-Mitarbeit im ftc-Forum, 2016.

[3] Wikipedia: *Saturn V*.

Tipps & Tricks

Kaulquappen (Teil 8)

Harald Steinhaus

Frühjahrszeit ist auch Laichzeit für die Frösche, und Wasser gibt es ja diesmal mehr als reichlich. Also ist es kein Wunder, wenn die Kaulquappen schlüpfen. Das Thema „Flügeltüren" und Möglichkeiten um die schwarze Kette herum stehen diesmal im Mittelpunkt.

Durch die Öffnung im großen Seiltrommelhalter passt der Schaft des Z10 nur in der Lage „quer". „Längs" gestellt wie hier bleibt diese Klappe entweder offen oder zu:

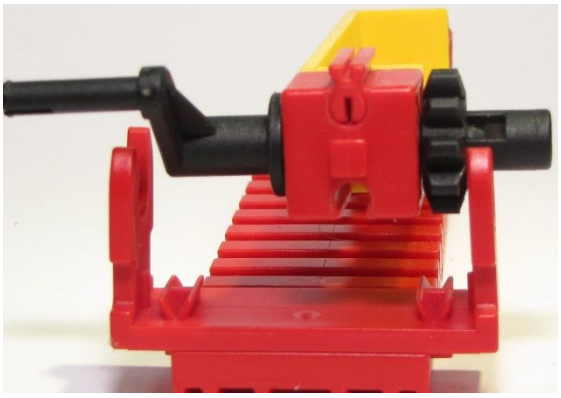


Abb. 1: Verriegelung

Mit der Führungsplatte 32455 kann man Flexschienen auch mittendrin stützen und in Form zwingen:

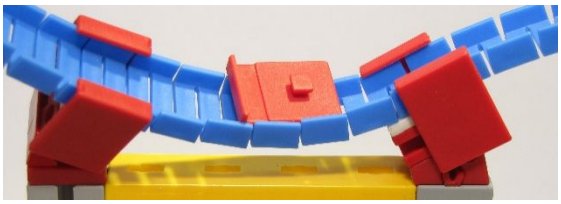


Abb. 2: Flexschiene und 32455

Die Kette (sogar die neue rote mit den Knubbeln außen dran) passt durch den V-Stein 15 hindurch. In der alten Lenkwürfel-Klaue (mit Statikloch) ist bis zum Boden hin genug Platz für die schmale Kette. Ein Z15 mit einer „Hülse mit Scheibe" 35981 hält sie dort fest und kann sie antreiben. Die neue Lenkwürfelklaue hat innen schräge Versteifungen, an denen die Kette nicht vorbei käme.

Abb. 3: Kettenführung (1)

Die großen Knubbel an den schwarzen Kettengliedern kann man einklemmen und damit Hebel betätigen. Die Führungsplatte 32455 eignet sich zum Umlenken und Führen, so wie auch die Lasche 23,5 mm, die dann aber um 45° geschwenkt sein muss.

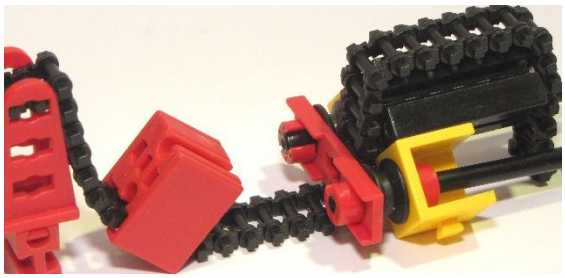


Abb. 4: Kettenführung (2)

Wenn die dicken Knubbel an den schwarzen Kettengliedern schon im richtigen Abstand sitzen, kann man sie natürlich auch mit fischertechnik-Schnecken antreiben. Der ganze Aufwand mit den zwei Lenkhebeln in Abb. 5 (der zweite drückt von unten) dient nur dazu, die Kette an der richtigen Position zwischen den Schnecken zu halten.

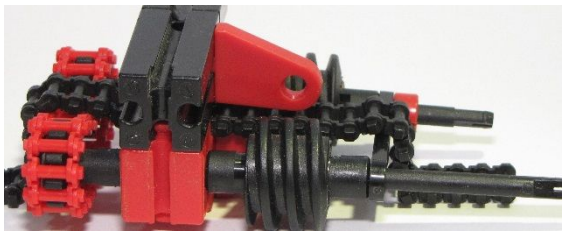


Abb. 5: Kettenzug

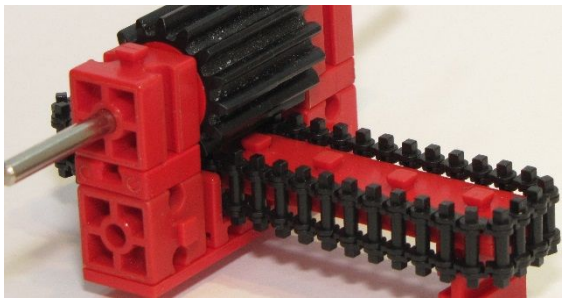


Abb. 6: Hubvorrichtung

Man lege zwei V-Platten 15 · 90 Rücken an Rücken, drapiere die schwarze Kette drum herum, führe das Ganze mittels U-Träger-Adapter, und erhält eine Zahnstange, mit der man ziehen und drücken kann. Das breite Differenzial schiebt sie hin und her. Das Ganze kann in den Löchern der unteren BS15 mit Loch gelagert werden (Abb. 6).

Schließlich zum großen Bild 7: Bei Fahrwerken und Flügeltüren braucht man Hebelpaare, die symmetrisch betätigt werden. Gäbe es fischertechnik-Schnecken mit Linksgewinde, wäre der Fall ziemlich schnell erledigt. Die Hebel hier sind allesamt nicht für große Kräfte geeignet. Für leichte Fälle könnte da schon mal ein Prinz drunter sein.

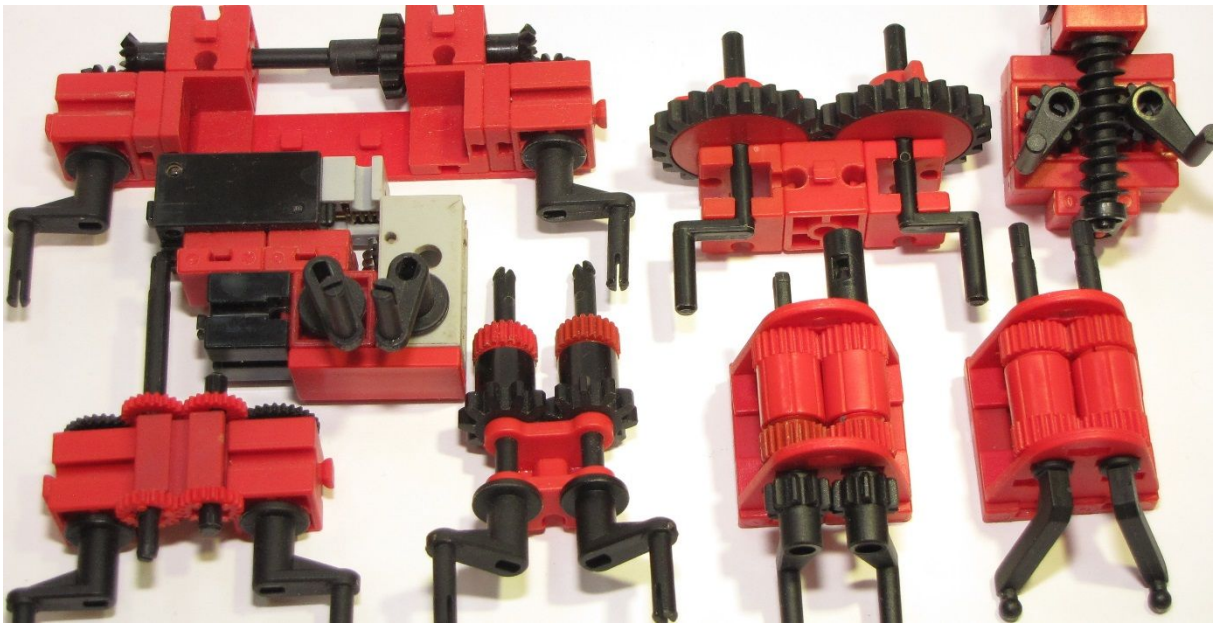


Abb. 7: Symmetrische Hebel

Modell

Gummiring-Pistole

Jens Lemkamp

Plötzlich Regen am Sonntag – und schon muss schnell etwas Spielspaß für die Jungs gebaut werden. Hier der ausführliche Bauvorschlag für eine schnell zu bauende Gummiring-Pistole. Aber Vorsicht! Nicht auf Menschen und Tiere anwenden – Verletzungsgefahr!

In Abb. 1 sieht man die „Pistole“ in ungeladenem Zustand:

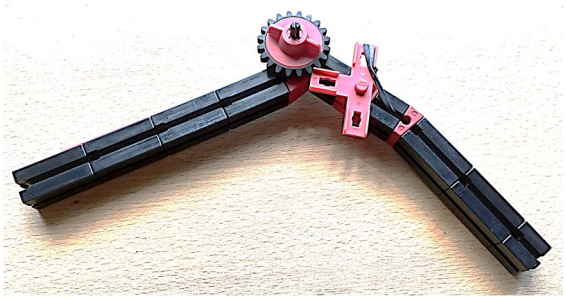


Abb.1: Die ungeladene Pistole

Ein herrlicher Spaß für die Jungs und Mädels – hier nun die Bauanleitung:

Die Munition besteht aus herkömmlichen Haushaltsgummiringen:

Abb. 2: Munition

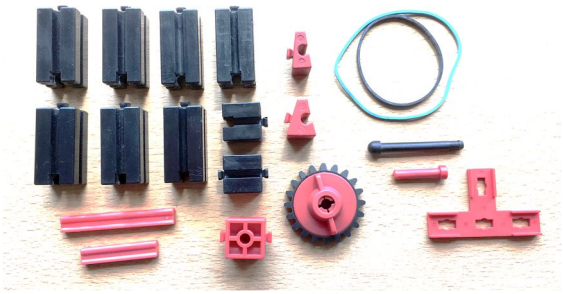


Abb. 3: Baustufe 1 – Einzelteilübersicht

Stückliste:

St.	Bezeichnung
7	Baustein 30 (schwarz/grau)
1	Baustein 15 mit einem Zapfen
1	Baustein 15 mit zwei Zapfen
1	Baustein 15 rot mit Bohrung
1	Winkelstein 15°
1	Winkelstein 30°
1	Zahnrad Z20 mit Flachnabe 25
1	V-Achse 20 Rastachse (31690)
1	Kunststoff-Steckachse 40
1	Statik T-Lasche (31672) oder Statik L-Lasche (31671) mit S-Riegel und Statik-Strebe 30 L
1	Verbinder 30
1	Verbinder 45
2	Gummiring (wie Abbildung)

In Baustufe 2 erkennt man den „Lauf“, die Auslöser-Halterung und den Handgriff:

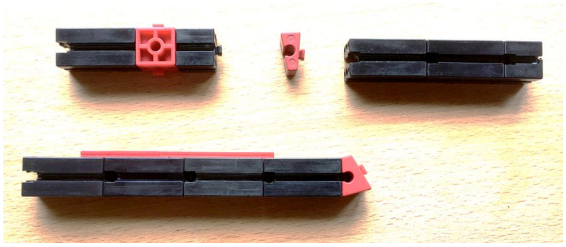


Abb. 4: Baustufe 2

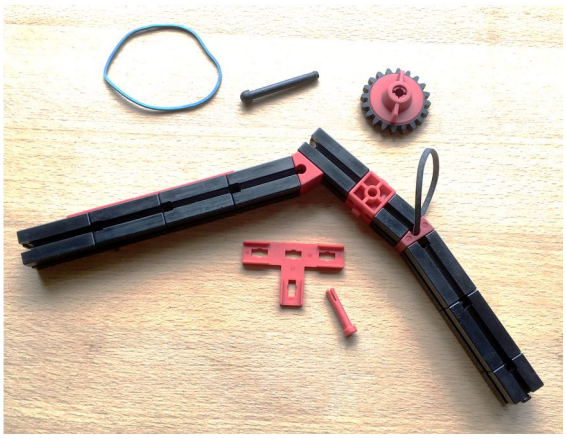
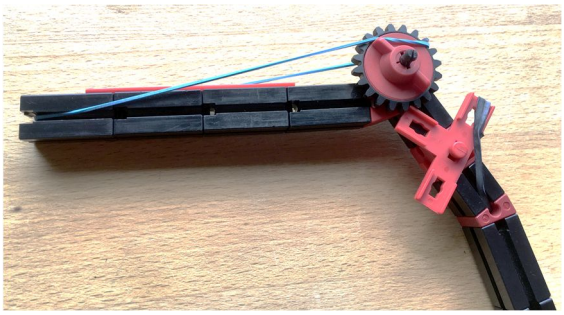


Abb. 5: Baustufe 3

Abb. 6: Das eingehängte Spanngummi

Die weiteren Abbildungen zeigen den Zusammenbau, insbesondere das Einhängen der Gummis:



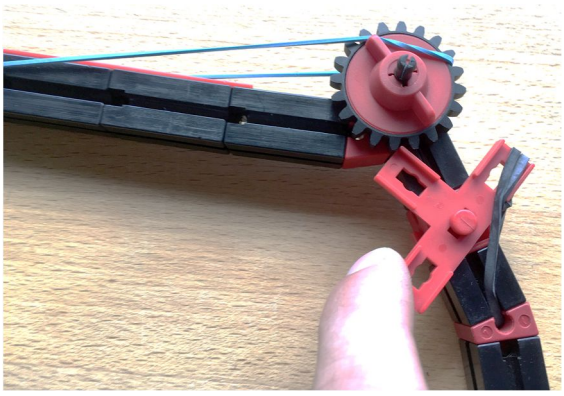


Abb. 7: Auslösen der Pistole

Bei Bedarf können einige Teile auch durch andere ersetzt werden (das erfordert dann ggf. andere Achsen):

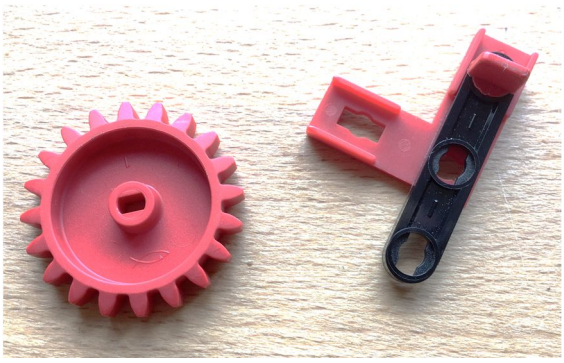


Abb. 8: Alternative Teile für den Abzug

Vielleicht konstruiert ja noch jemand eine Zielscheibe?

Quellen

[1] fischertechnik-Datenbank: http://www.ft-datenbank.de

[2] Austin, John: *Schreibtisch-Artillerie.* Heel Verlag, ISBN 978-3-86852-409-3

Modell
Seilbahn

Daniel Canonica

Bei uns in der Schweiz fährt an jedem besseren Hügel eine Seilbahn hoch. Manche wurden vor mehr als hundert Jahren erbaut. Interessant sind die verschiedenen Antriebstechniken und die Möglichkeiten, ohne Gefährdung von Personen besonders steile Abschnitte zu überwinden.

Gewöhnliche Schienenfahrzeuge können schon bei mehr als zwei bis drei Prozent Steigung die Antriebskraft nicht mehr voll nutzen. Besonders Laub im Herbst oder vereiste Schienen im Winter können auf steilen Streckenabschnitten zu einem Problem werden. Mit Zahnradbahnen oder Seilbahnen lassen sich erheblich größere Steigungen überwinden.

Typen von Seilbahnen

Abgesehen von Luftseilbahnen sind Schienen gebundene Seilbahnen die häufigsten. Auch bei diesen gibt es diverse Arten: bei Standseilbahnen ist die Kabine fest mit dem Zugseil verbunden [1]. Bei den berühmten Cable Cars von San Francisco verbindet sich die Kabine mit einer Art überdimensionierter Zange, dem „Grip“, mit dem ständig laufenden Seil.

Im Folgenden konzentrieren wir uns auf die Standseilbahn. An einem Drahtseil, welches bei der Bergstation (und manchmal auch bei der Talstation) über eine Umlenkrolle läuft, sind zwei Kabinen fixiert, welche ungefähr gleich schwer sind.

Bevor man starke Elektromotoren nutzen konnte, bediente man sich eines einfachen Prinzips, um die talwärts fahrende Kabine schwerer zu machen: Wasserballast. Der Ballasttank war an der Bergstation in wenigen Minuten gefüllt. Mit dem Gewicht des Wassers zog die Kabine die andere hoch, und in der Talstation war der Wassertank noch schneller wieder entleert.

Mithilfe der Abt’schen Weiche können auf eingleisig betriebenen Strecken die Kabinen in der Mitte einander ausweichen [3].

Standseilbahnen können sehr steile Hänge befahren:

Abb. 1: Die Gelmerbahn (Kanton Bern)

Die Gelmerbahn im Haslital im Kanton Bern überwindet eine maximale Steigung von 106% (bei einer mittleren Steigung von knapp 50%). Genau genommen handelt es sich dabei um einen Schrägaufzug, weil nur eine Kabine bewegt wird.

Das Modell

Nach anfänglichen Versuchen mit der Abt'schen Weiche bin ich zu einer einfachen und geraden Streckenführung mit einem Wagen übergegangen.

Der Neigungswinkel beträgt zurzeit etwa 37°, das entspricht etwa 75% Steigung (Steigung = Höhe / Grundlänge in %). Da ich die Schienen nicht beliebig verlängern wollte (und konnte), habe ich die Steigung mithilfe von Gelenksteinen an die Tischhöhe angepasst und könnte diese mit wenigen Handgriffen ändern. Ich habe es nicht getestet, denke aber, bei 120% käme auch das Modell an seine Grenzen …

Zum Glück gibt es noch die alten 30 mm-Flachsteine, um eine schöne Treppe zu bauen:

Abb. 3: Die Talstation

Abb. 2: Das Seilbahn-Modell

Im Gegensatz zur Talstation, welche auf einem festen Fundament ruht, hängt die Treppe der Bergstation sozusagen in der Luft. Mit einigen Kniffen habe ich sie an der Grundplatte der Bergstation fixiert.

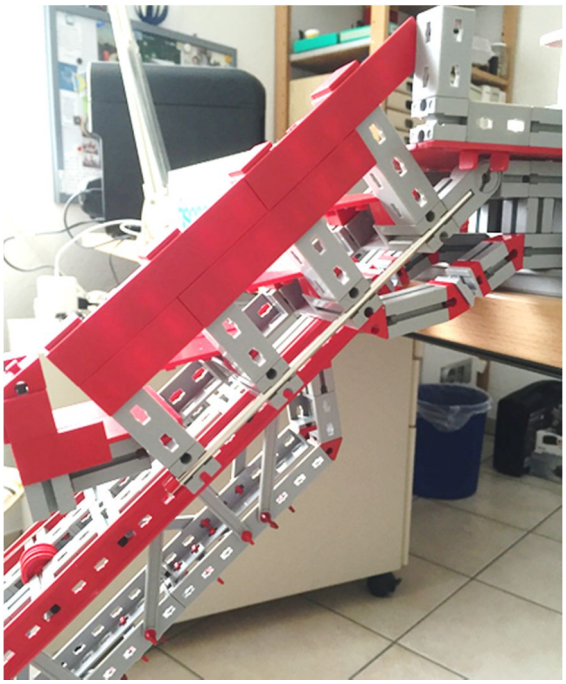


Abb. 4 Die Bergstation

Die Kabine selbst ist mit den Abständen zwischen den einzelnen Plattformen auf eine Neigung von ca. 30° eingestellt. Insbesondere bei den Stationen sollte diese der tatsächlichen Schienenneigung entsprechen; während der Fahrt kann die Neigung der Schienen natürlich durchaus variieren. Es gibt sogar Bergbahnen, bei denen sich der Kabinenboden automatisch an die Steilheit der Strecke anpasst.

Die Kabine läuft auf vier Achsen mit je zwei Spurkranzrädern in den Schienen:

Abb. 5 Die Kabine

Den Motor habe ich anfänglich mit dem alten Polwendeschalter und einem Taster betrieben. Nun steuere ich ihn mit dem gerade neu gekauften Control Set an. Das erlaubt eine stufenlose Anpassung der Geschwindigkeit, was beim Einfahren in die Berg- und Talstation durchaus nützlich ist.

Quellen

[1] Wikipedia: *Seilbahn*.

[2] Wikipedia: *Standseilbahn*.

[3] Geerken, Ralf: *Die Geheimnisse der Turmbergbahn.* ft:pedia 4/2015, S. 5-10.

Modell

Mini-Modelle (Teil 13): Visitenkartenhalter

Martin Westphal, René Trapp

Ein Visitenkartenhalter für fischertechniker.

Mittlerweile hat es sich sicher herumgesprochen, dass ein Stand unserer ft:c die Maker Faire in Hannover bereicherte. Natürlich sollten kleine Visitenkärtchen mit unserer Werbung unters Volk gebracht werden. Einfach einen Stapel davon auf den Tisch zu legen geht bei solch einer großen Veranstaltung allerdings nicht, der Stapel wäre in Windeseile zerfleddert worden. Also haben die Autoren am Vorabend noch schnell den hier vorgestellten Visitenkartenhalter entworfen (Abb. 1).

Abb. 1: Der bestückte Visitenkartenhalter

Und damit er ergonomisch schräg steht, gibt es noch die raffiniert einfache Konstruktion auf der Rückseite (Abb. 2).

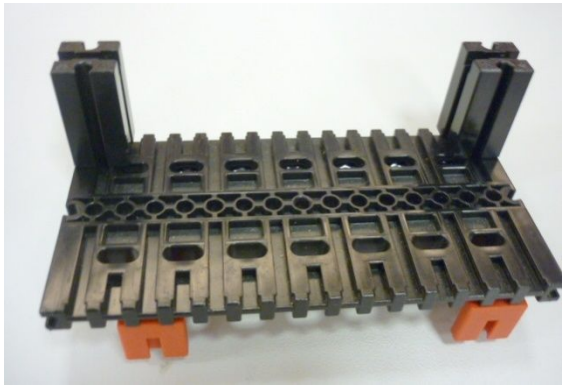


Abb. 2: Die Kartenhalter Rückseite

Diese Teile werden für den vorgestellten Visitenkartenhalter insgesamt benötigt:

St.	ft-Nr.	Bezeichnung
1	35129	Grundplatte 120 x 60 x 7,5
2	38240	V-Baustein 15 Eck
2	32879	Grundbaustein 30

Modell

Urlaubskasten-Modell 2: Schrittförderer

Stefan Falk

Ausschließlich aus Bauteilen des in der ft:pedia-Ausgabe 1/2016 [1] zusammen gestellten Urlaubs-Baukastens besteht das hier vorgestellte einfache Schrittförderwerk.

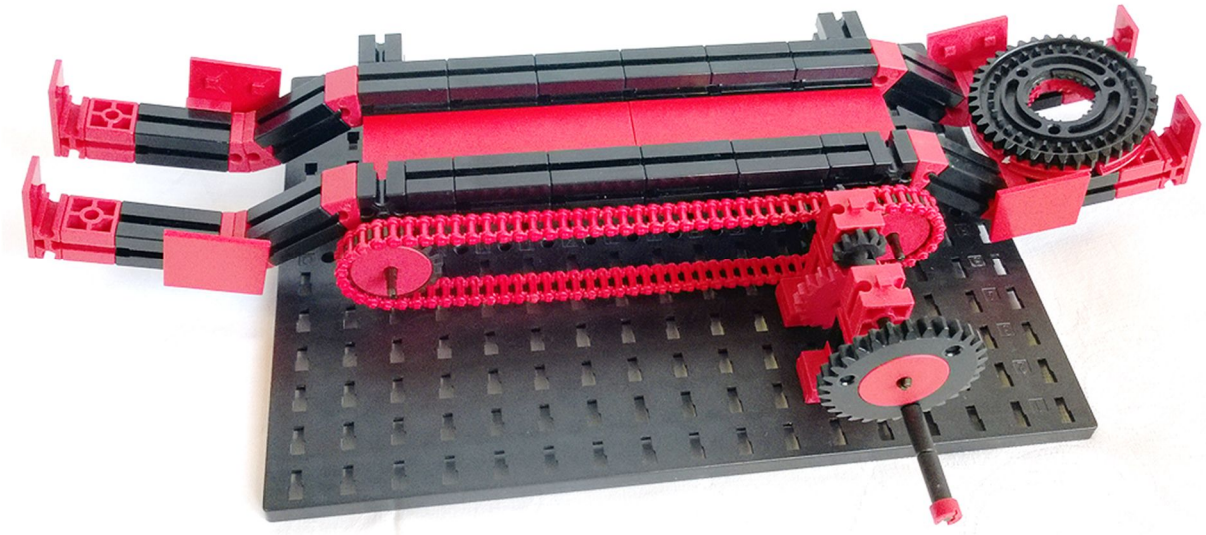


Abb. 1: Schrittförderanlage

Die Maschine besteht aus zwei Rahmenteilen, auf denen das zu befördernde Stückgut aufliegt. Von unten wird es über eine in der Mitte verlaufende lange Platte immer wieder angehoben, ein Stück weiter versetzt und wieder abgelegt. Diesen Ablauf zeigt die Bilderserie in Abb. 3. Tests mit Besuchern ergaben, dass die Leute nur schwer aufhören können, das Stückgut durch Kurbeln hin und her zu befördern.

Zum Bau des Modells

Die beiden Rahmenstränge sind gleich aufgebaut; Abb. 2 zeigt die Details. In einem sind lediglich zwei Baustein 15 mit zwei Zapfen durch solche mit Bohrung ersetzt (siehe Abb. 5). Die quadratischen Bauplatten 30 · 30 sind durch Winkelsteine 15° am Rahmen angebracht. Gebaut wie gezeigt können die Rahmen die zu

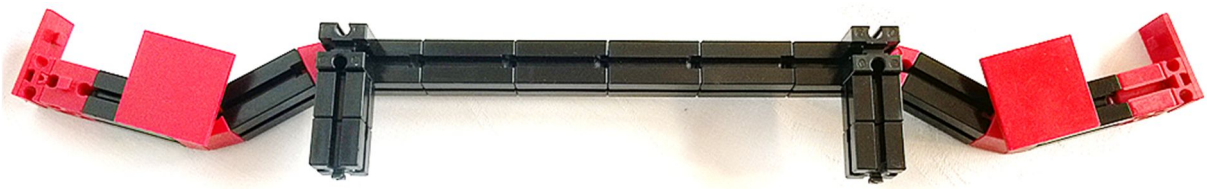


Abb. 2: Eines der beiden baugleichen Rahmenteile

befördernde Kombination aus Drehscheibe und Zahnrad Z40 zuverlässig auffangen.

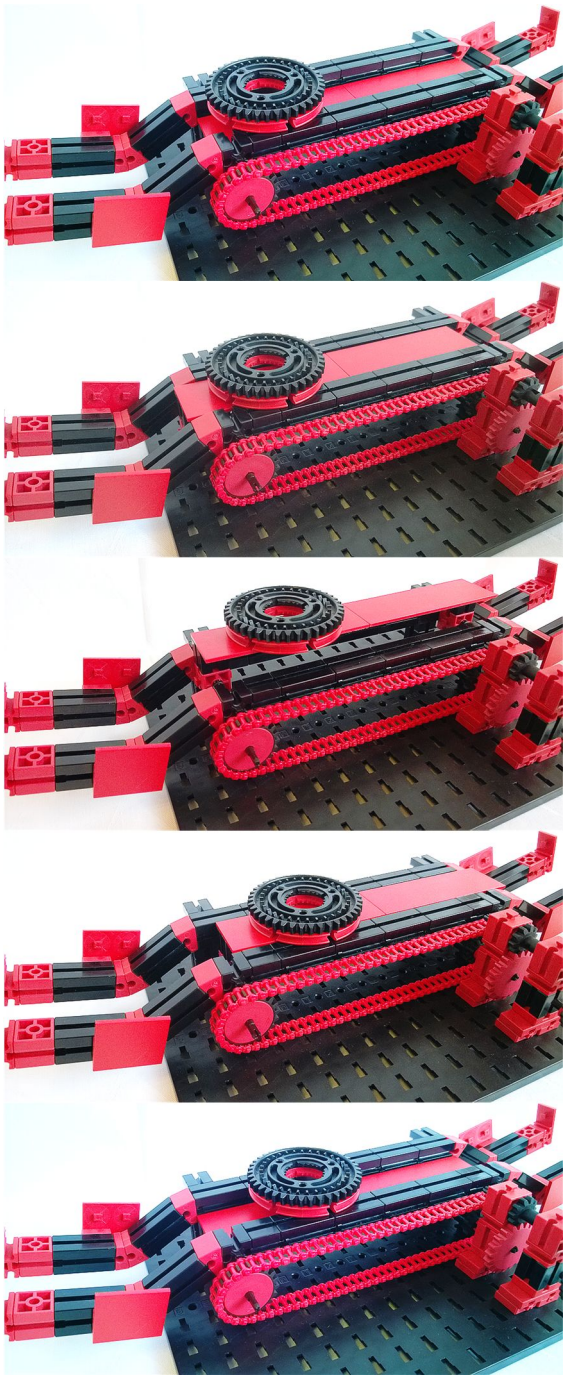


Abb. 3: Wirkungsweise

Parallelkurbel-Getriebe

Der bewegliche Mittelteil besteht einfach aus zwei Platten 30 · 90, die mit einem Baustein 5 15 · 30 3N (38428) verbunden sind. An den Enden stecken je ein Baustein 15 und ein Baustein 15 mit Bohrung.

Die Bausteine 15 mit Bohrung stecken in je einer Rastkurbel. Die Kurbeln sind, wie die Abb. 5 zeigt, einfach in einem der Rahmenteile gelagert – beide also im selben Rahmenteil; das andere hat für den beweglichen Mittelteil keine Bedeutung.

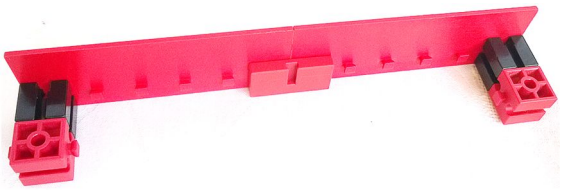


Abb. 4: Förderplatte

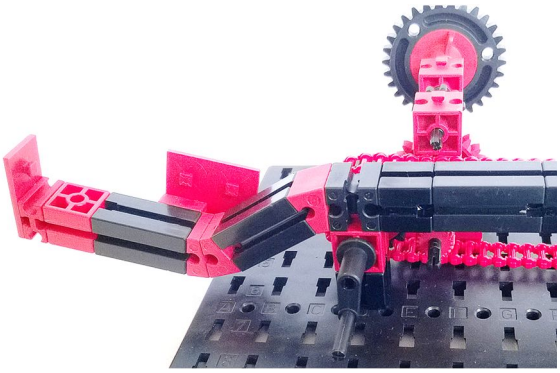


Abb. 5: Lagerung der Kurbeln

Beide Kurbeln sind durch Zahnräder Z20 und Kette so miteinander verbunden, dass sie sich immer gleich bewegen. Da die Förderplatte an beiden Kurbeln eingehängt ist, bewegt sie sich also letztlich kreisförmig. In der oberen Hälfte ihrer Bewegung nimmt sie das Stückgut mit, in der unteren Hälfte der Bewegung liegt das Stückgut auf dem Rahmen auf, während die Förderplatte an die Ausgangsposition zurückkehrt.

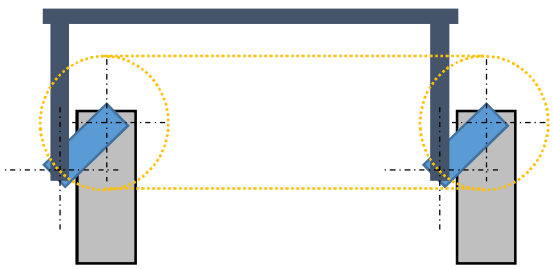


Abb. 6: Prinzip des Parallelkurbelgetriebes

Diese Art von Mechanik, bei der zwei (oder mehr) Kurbeln synchron laufen, nennt man *Parallelkurbelgetriebe* (vgl. auch [2], und

wer's ganz genau wissen will, findet in [3] viele Informationen und Modelle dazu).

Antrieb

Eine der Kurbeln wird von Hand angetrieben – wer einen Motor besitzt, wird ihn nach dem Ausprobieren von Hand sicherlich anbringen und die Förderanlage auch elektrisch betreiben. Abb. 7 und 8 zeigen eine Konstruktionsmöglichkeit:

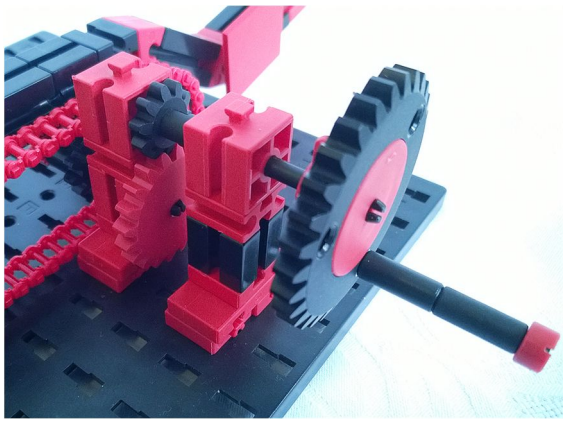


Abb. 7: Antriebskurbel mit Untersetzungsgetriebe

Die Achse im Z30 (zum leichteren Kurbeln mit zwei Hülsen versehen) wird zweimal per Z10 auf Z20 untersetzt. Das letzte Z20 ist das des Parallelkurbeltriebs. Von unten sieht das dann so aus:

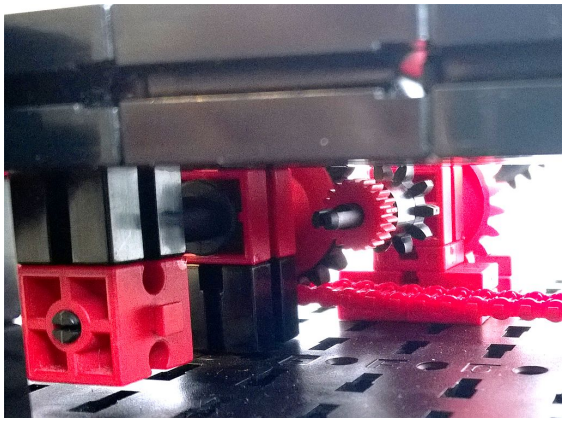


Abb. 9: Blick von unten auf den Antrieb

Quellen

[1] Falk, Stefan: *Der Wohnzimmer-Dienstreisen-Urlaubs-Notfallkasten*. ft:pedia 1/2016, S. 31-36.

[2] Geerken, Ralf: *Die Geradführung einer Viergelenkkette im Einsatz bei einer kleinen Laufmaschine*. ft:pedia 4/2012, S. 4-10.

[3] Fischer-Werke: *Parallelkurbelgetriebe,* in: hobby 2 Begleitbuch Band 5, 1973, S. 18-29.

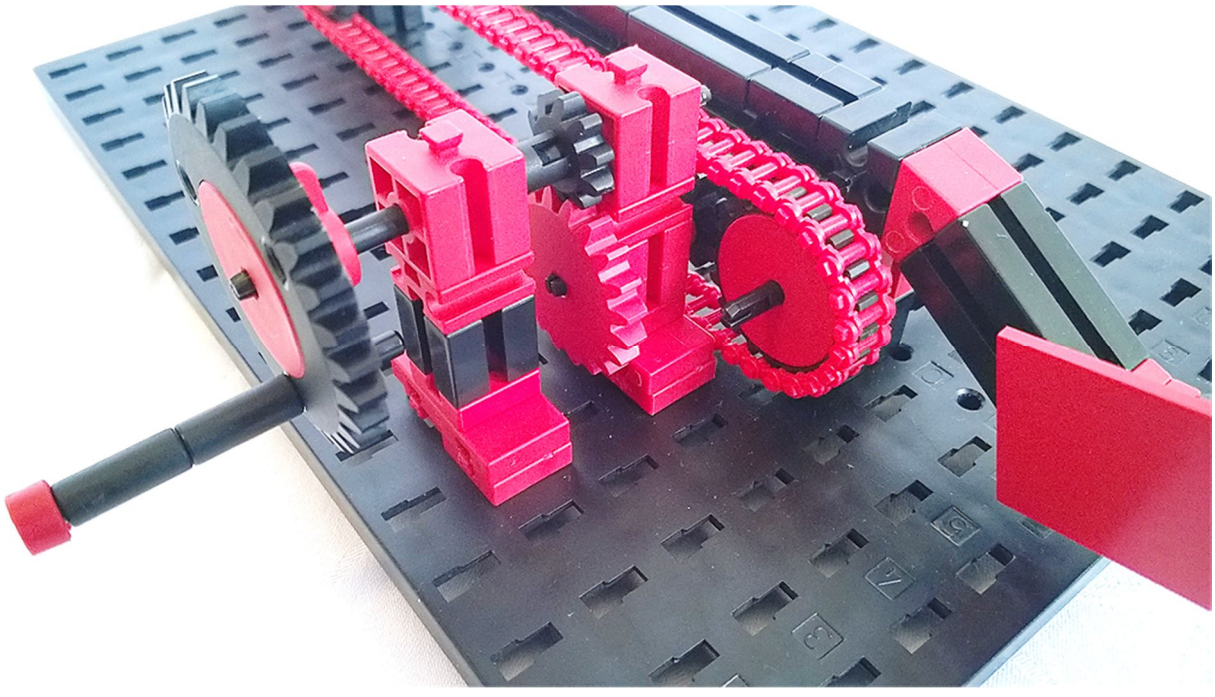


Abb. 8: Kurbelantrieb des Modells

Modell

Mini-Modelle (Teil 14): Brieföffner

René Trapp

Wie ein fischertechniker stilecht seine Post öffnet – mit nur vier Bauteilen.

Ein Brieföffner? Im 21. Jahrhundert? („Papi, was genau ist eigentlich ein Brief?“) Nun ja, mit einem Brieföffner kann man sich auch am Rücken kratzen oder Gegenstände aus schwer zugänglichen Schlitzen retten.

Für fischertechniker eröffnen sich weitere Möglichkeiten: Der vorgestellte Brieföffner eignet sich auch als Einschubwerkzeug für Bausteine in Aluminiumprofile.

Abb. 1: Der Brieföffner

Wer mag (und bereit ist, ein fünftes Bauteil zu investieren) kann noch einen Klemmring an das offene Ende der Seiltrommel setzen.

Und hier die Teileliste:

St.	ft-Nr.	Bezeichnung
1	38241	Bauplatte 15 x 30
1	31004	Baustein 30 mit Bohrungen
1	31016	Seiltrommel
1	31030	Metallachse 150

Bisher erschienen:

[1] René Trapp: *Mini-Modelle (Teil 1): Gabelstapler*. ft:pedia 4/2013, S. 4-5.

[2] Johann Fox: *Mini-Modelle (Teil 2): Panzer*. ft:pedia 2/2014, S. 18-19.

[3] René Trapp: *Mini-Modelle (Teil 3): Scheinwerfer*. ft:pedia 3/2014, S. 11.

[4] Johann Fox: *Mini-Modelle (Teil 4): Hubschrauber*. ft:pedia 3/2014, S. 12-13.

[5] René Trapp: *Mini-Modelle (Teil 5): Traktor*. ft:pedia 4/2014, S. 7.

[6] Johann Fox: *Mini-Modelle (Teil 6): Bagger*. ft:pedia 4/2014, S. 8-9.

[7] Johann Fox: *Mini-Modelle (Teil 7): Hovercraft*. ft:pedia 1/2015, S. 4-5.

[8] René Trapp: *Mini-Modelle (Teil 8): Flugsaurier*. ft:pedia 4/2015, S. 4.

[9] Norbert Doetsch: *Mini-Modelle (Teil 9): Motorrad*. ft:pedia 1/2016, S. 6.

[10] René Trapp: *Mini-Modelle (Teil 10): Jojo*. ft:pedia 1/2016, S. 4.

[11] René Trapp: *Mini-Modelle (Teil 11): Flugzeug*. ft:pedia 1/2016, S. 14.

[12] Stefan Falk: *Mini-Modelle (Teil 12): Mondrakete*. ft:pedia 2/2016, S. 5.

[13] Martin Westphal, René Trapp: *Mini-Modelle (Teil 13): Visitenkartenhalter*. ft:pedia 2/2016, S. 13.

Modell

Urlaubskasten-Modell 3: Gabelstapler

Stefan Falk

Die Bauteile des Urlaubs-Baukastens aus ft:pedia 1/2016 [1] wurden ja so ausgewählt, dass man damit schon nicht-triviale mechanische Konstruktionen herstellen kann. Hier folgt ein Modellvorschlag für einen Gabelstapler, in dem ich mal keinen Kettenzug, sondern eine Hebemechanik mit Zahnstangen einsetzen wollte.

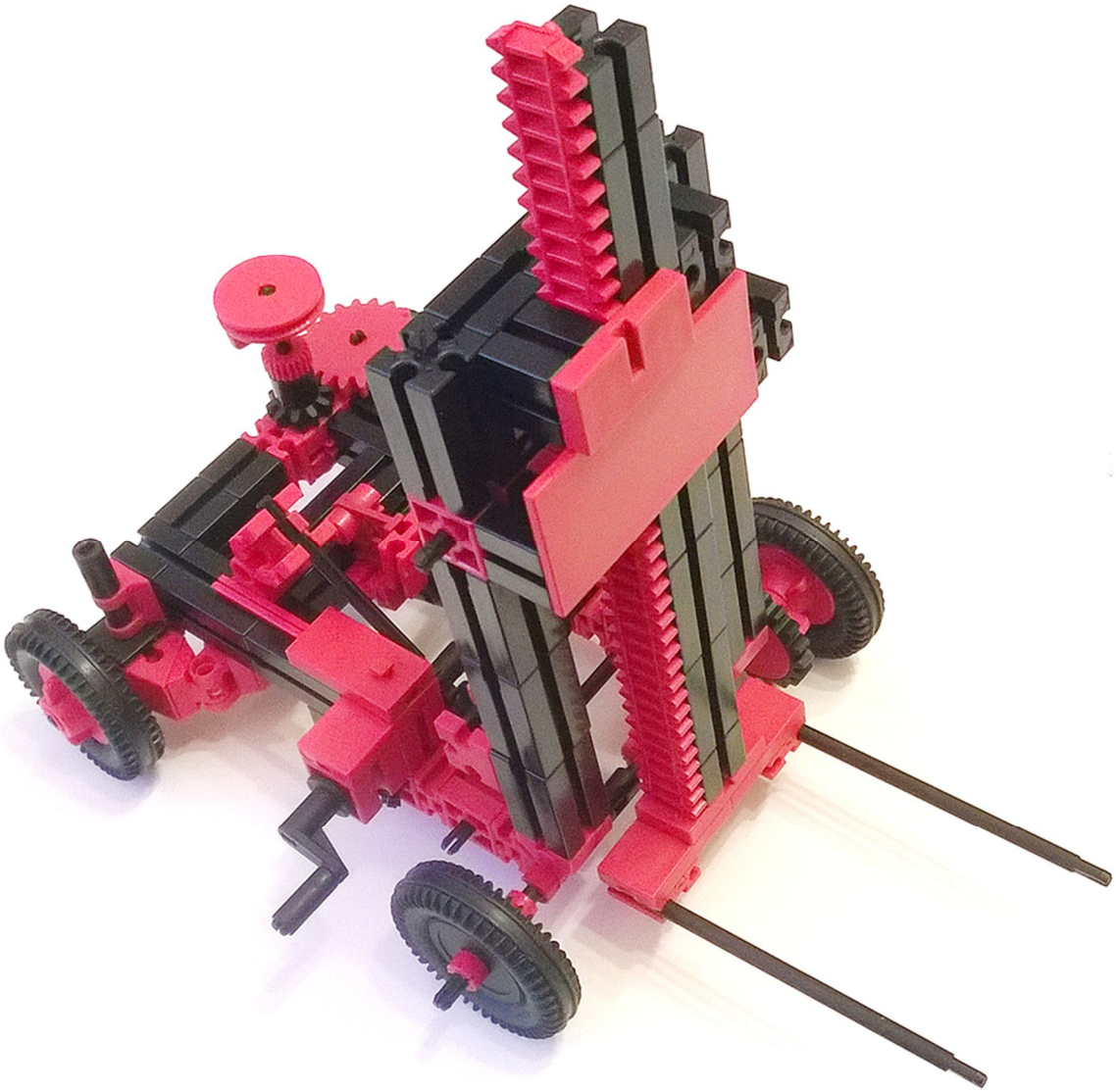


Abb. 1: Der Gabelstapler

Der Gabelstapler ist auf einem von den lenkbaren Hinterrädern bis zum obenliegenden Zahnstangenantrieb durchgehenden Rahmen aufgebaut. Insgesamt bietet das Modell:

- Gelenkte Hinterräder,
- einen per Hebeleinstellung etwas nach hinten kippbaren Aufbau (damit die Ladung nicht herunterfällt),
- Gabelstäbe, die bis ganz nah an den Boden heruntergefahren werden können sowie
- einen kurbelbetriebenen Zahnstangenantrieb zum Anheben der Gabel.

Das Modell ist durchaus spielstabil und sollte also für's Kinderzimmer gut geeignet sein. Beim Aufbau muss aber vermutlich jemand mit etwas fischertechnik-Baupraxis dabei sein – der Teilebestand des „Wohnzimmer-, Dienstreisen-, Urlaubs- und Notfall-Kastens" wird schon ganz schön ausgenutzt.

Aber der Reihe nach.

Der Rahmen

Abb. 3 zeigt einen Großteil des Modells. Geht beim Nachbauen anhand dieses Bildes vor; Details könnt ihr in den weiteren Abbildungen „nachschlagen". Das Bild enthält den vollständigen, normalerweise senkrecht stehenden Teil (oben) sowie den wesentlichen Teil des Fahrwerks-Rahmens. Die Hinterräder und ihre Lenkung sowie die Gabel werden später einfach angesteckt.

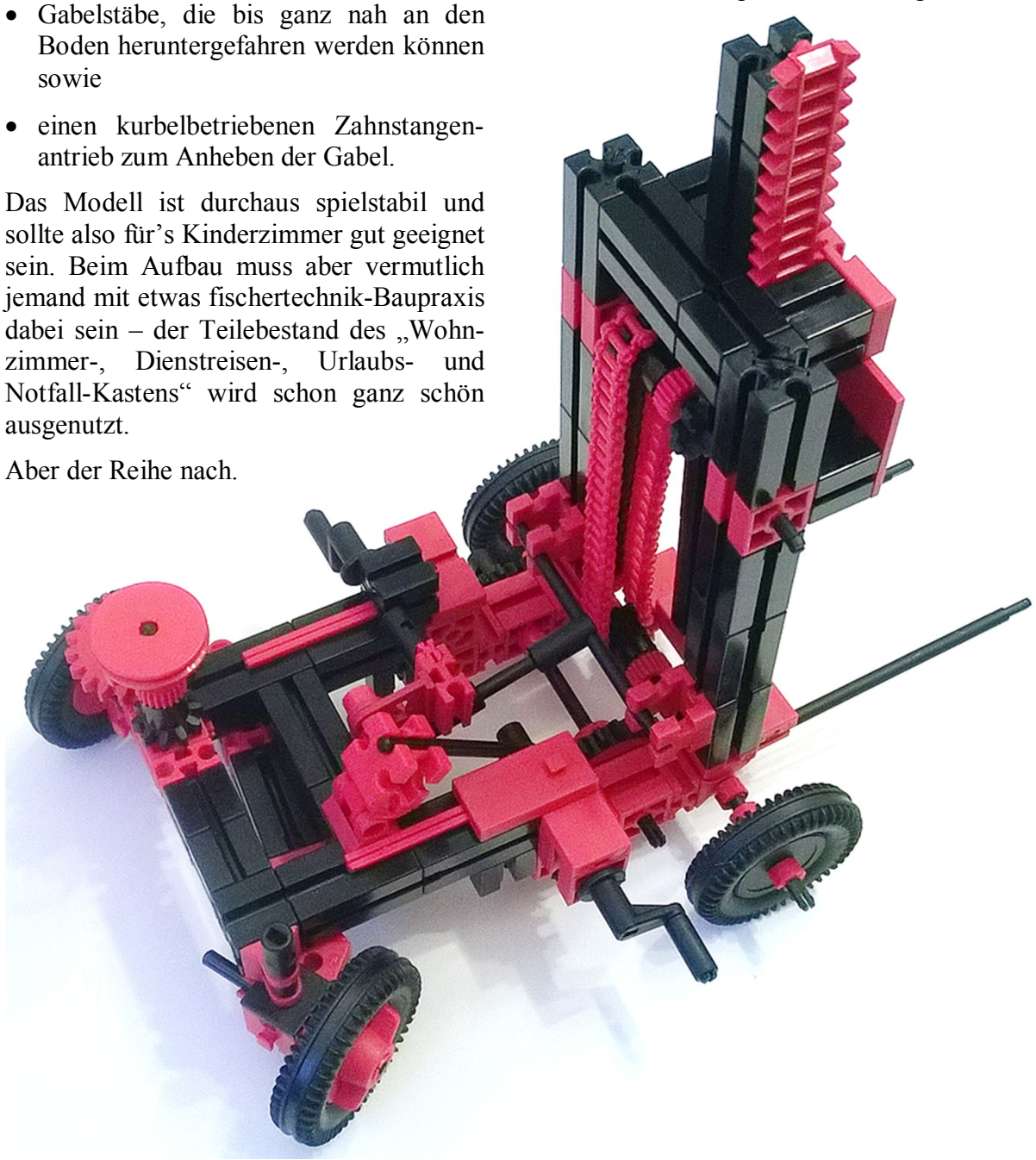


Abb. 2: Jede Menge Mechanik

Die Kurbel rechts in Abb. 3 geht über zwei Z10 auf das Z20 auf einer langen Achse, die links über ein zweites Z10 auf ein weiteres Z20 führt. Dieses Z20 dreht die Ache mit dem Kettentrieb für's Anheben der Gabel.

Damit die lang genug wird, besteht sie aus zwei Rastachsen, die in der Mitte mit einer Rastkupplung verbunden sind. Ebenfalls in der Mitte sitzt das normale Z10 für die Kette. Zwischen Z10 und Gelenkstein sorgt ein Klemmring für den nötigen Abstand, damit die Kette nicht am Rahmen schleift.

Auf derselben Achse sitzen auch die Vorderräder. Wichtig: Damit sich die Räder unabhängig vom Antreiben der Kette beim Fahren drehen können, werden für die Vorderräder die beiden Freilauf-Flachnaben aus dem Urlaubskasten verwendet. Die Achsen und Freilaufräder werden mit einigen Klemmringen gesichert. Wer keine Freilaufnaben besitzt, verwendet normale Flachnaben und zieht diese nur ganz sachte an – schließlich dürfen sich die beim Fahren drehenden Vorderräder und der auf derselben Achse sitzende Kettentrieb nicht stören.

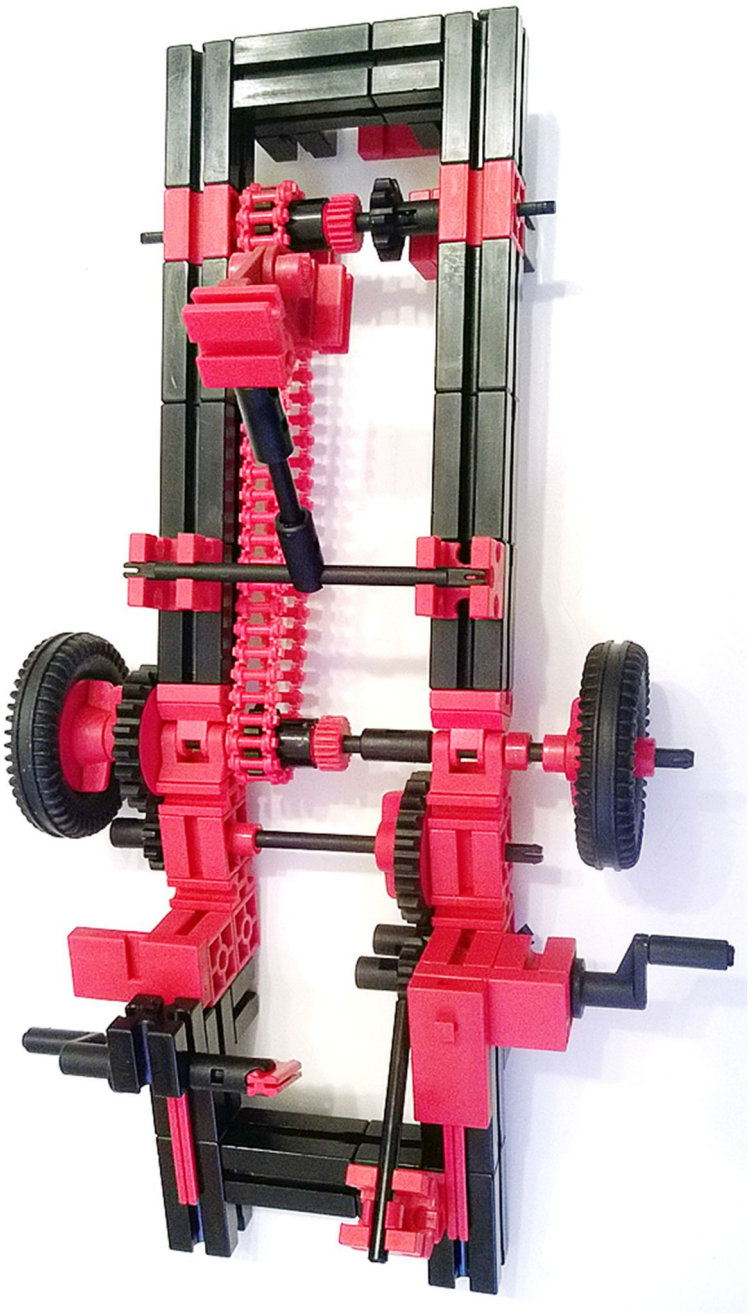


Abb. 3: Der Rahmen – unten fehlen noch die Hinterräder, am oberen Bildrand wird später die Zahnstange eingehängt

Ebenfalls um dieselbe Achse kann der Gabelträger über die beiden Gelenksteine gekippt werden. Für eine solche Mehrfachausnutzung derselben Achse finden sich in [2] noch ein paar Anregungen.

Die Kippmechanik besteht aus zwei bis hierhin noch nicht miteinander verbundenen Teilen: Da wäre einmal die Kurbel links unten in Abb. 3 und als Gegenstück dazu die (später noch daran zu befestigende) Hebel-Mimik, die im Bild gerade hochgeklappt ist. Sie drückt oder zieht später an der quer in die beiden roten Lagerböcke eingespannten Rastachse 75. Die Lagerböcke sind einfach an den unteren BS30 des Gabelträgers befestigt. Die Wirkungsweise der Kippmechanik werden wir später noch genauer betrachten.

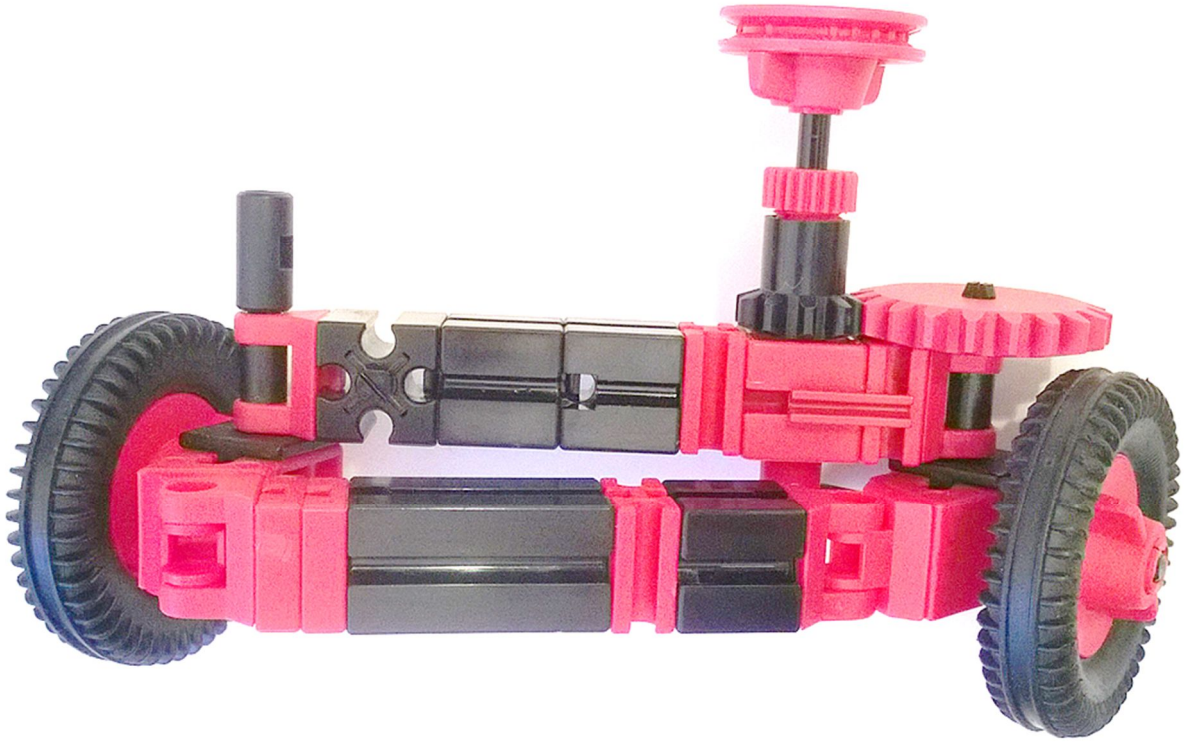


Abb. 4: Die Hinterräder mitsamt Lenkung

Die Hinterradlenkung

Als nächstes wird das Fahrwerk mit den lenkbaren Hinterrädern ergänzt. Dazu wird die Baugruppe in Abb. 4 hergestellt.

Der obere Querträger muss genau nach dem Bild aufgebaut werden. Für den Abstand des Z10 zum Rast-Z20 ist der BS7,5 zwischen der im Bild rechten Gelenkklaue und dem BS15 mit Bohrung wichtig. Für die korrekte Breite des Querträgers ist der zweite BS7,5 wichtig.

Und schließlich werden sowohl der in den BS15 mit Bohrung nach ganz außen eingeschobene Federnocken als auch der nach vorne zeigende Zapfen des BS15 auf der im Bild linken Seite benötigt, um die Hinterrad-Baugruppe am Rahmen zu befestigen. Wir schauen in Abb. 4 also sozusagen vom Fahrer aus nach hinten auf die Hinterräder.

Die Räder selbst sitzen mit ihren durch Klemmringe gesicherten Achsen auf BS7,5 mit Bohrung, die an Rastaufnahmeachsen 22,5 (130593) „hängen“. Die stecken in einer Lagerhülse 15 (36819) in „halben Gelenkbausteinen“. So können die Räder leicht gelenkt werden.

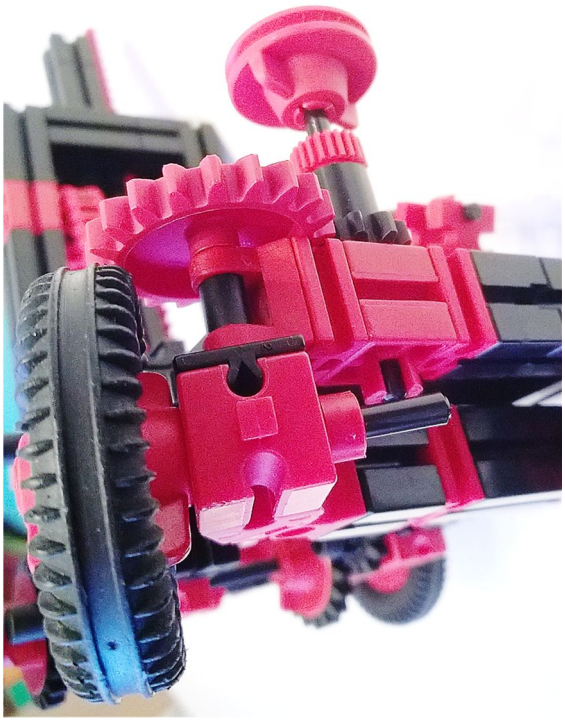


Abb. 5: Aufhängung der Hinterräder

Damit bei der Hecklenkung ein ordentliches Lenktrapez [3] herauskommt (das kurveninnere Rad sollte stärker einschlagen als das äußere), muss die Spurstange kürzer ausfallen als der Abstand zwischen den Gelenkmittelpunkten. Also sitzen auf den BS7,5 mit Bohrung, in denen die Radachsen drehen, erstmal je ein Winkelstein 30°, dann wieder je ein Gelenkstein, und schließlich die Kombination aus BS30, BS15, einem BS7,5 und zwei BS5. Das ergibt eine recht saubere Lenkgeometrie.

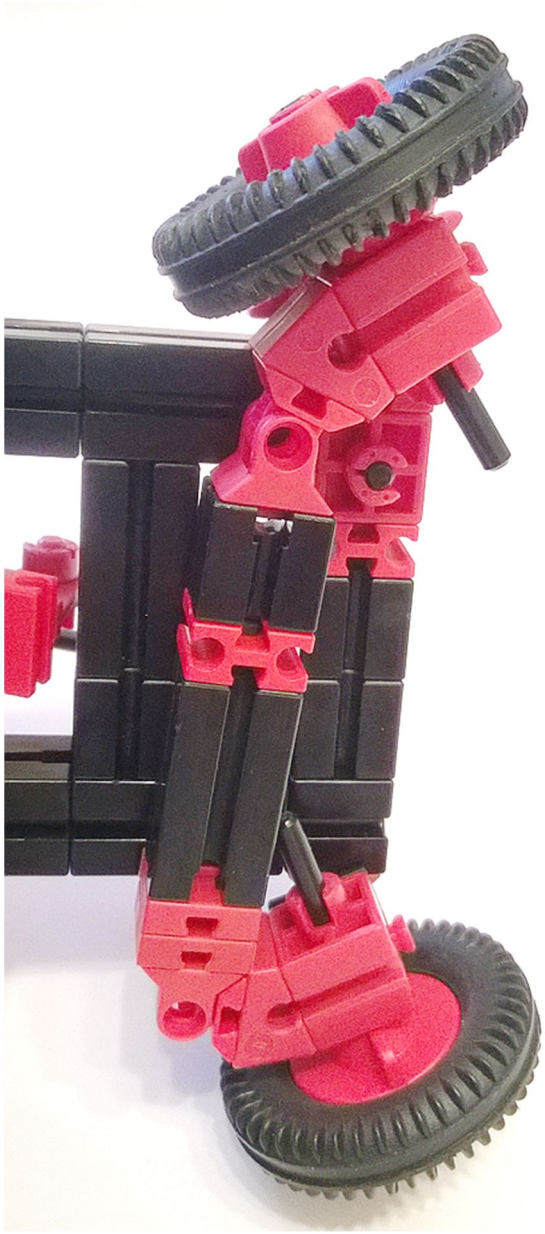


Abb. 6: Die am Rahmen angebaute Lenkung von unten gesehen

Wie Abb. 2 zeigt, wird diese Baugruppe einfach an den Rahmen gesteckt. Dafür ist es wichtig, dass die Nut des linken BS30 am Ende des Rahmens waagerecht ausgerichtet ist – da muss ja der breite Teil des Federnockens aus Abb. 4 rein. Abb. 6 zeigt die eingebaute Lenkung von unten.

Ihr habt alles richtig gebaut, wenn die Lenkung sich am „Lenkrad“ – der nach oben stehenden Flachnabe aus Abb. 4 – ganz leichtgängig bis in die jeweiligen Endlagen betätigen lässt.

Die Kippmechanik

Wenn ein Gabelstapler seine Last aufgenommen und hochgehoben hat, soll sie nicht z. B. beim Bremsen nach vorne von der Gabel rutschen. Deshalb wird der ganze senkrechte Aufbau etwas nach hinten gekippt. Die Gabeln zeigen dann leicht nach oben und verhindern das Abrutschen der darauf liegenden Last.

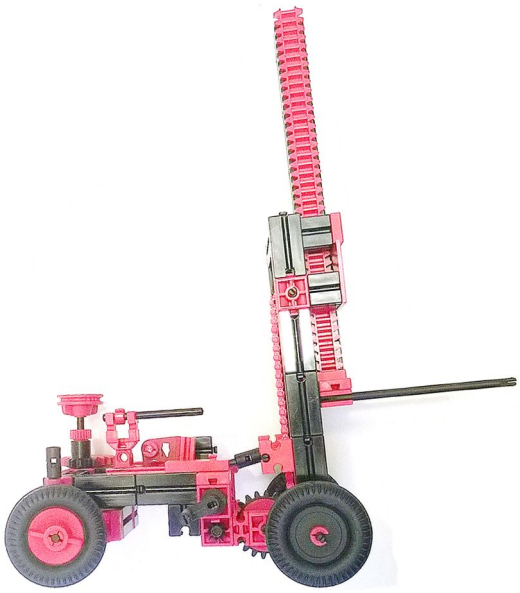


Abb. 7: Der Gabelträger ist leicht nach hinten gekippt

Zu diesem Zweck ist der senkrechte Aufbau an den zwei Gelenksteinen im Rahmen kippbar aufgehängt, denn zum Aufnehmen und Absetzen der Last müssen die Gabeln natürlich möglichst tief unten aufliegen.

Der linke Teil des Rahmens, an dem sich die Kippmechanik befindet, ist so aufgebaut:

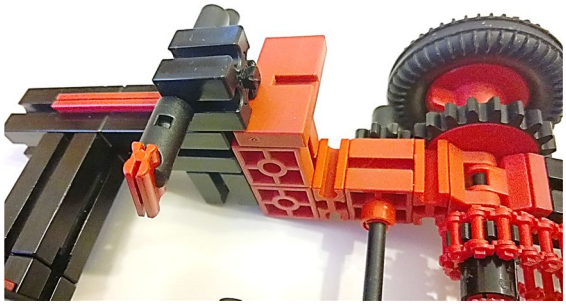


Abb. 8: Lagerung der Kurbel für die Kippmechanik

Der BS15, in dem die Kurbelachse steckt, sitzt also über einen Verbinder 45 auf dem Rahmen und kann so fein nach vorne oder hinten justiert werden. Die Rastachse 30 trägt außer der Kurbel einen Rastadapter mit einem Verbinder 15 vornedran.

Mit dem Gabelträger wird das wie folgt verbunden:

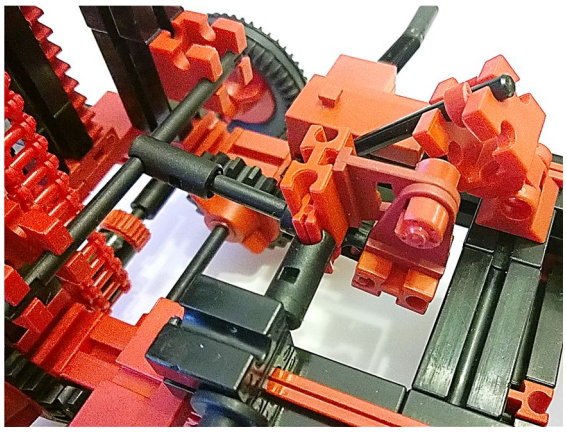


Abb. 9: Ankoppeln des Gabelträgers an die Kipp-Kurbel

Vom eben schon erwähnten Verbinder 15 aus geht die Mechanik also so weiter:

1. Am Verbinder sitzt eine gelenkige Kombination von je einem BS7,5 und einer S-Kupplung 15 2 (38253). Das Gelenk ist mit einem Seilklemmstift (eine andere kurze Achse tut's auch) und zwei Klemmringen ausgeführt.
2. Der letzte BS7,5 ist per Federnocken mit Rastadapter, Rastachse 30 und noch einem Rastadapter verbunden.
3. Letzterer hängt an der Rastachse 75, die über die beiden Lagerböcke mit dem Gabelträger verbunden ist.

Justiert werden müssen nun die Kurbel (nach vorne und hinten) und der BS7,5 am Verbinder 15 (nach oben und unten), sodass es einen Schnapp-Effekt gibt. Wenn die Gabel (beim Fahren des Staplers) nach hinten gekippt ist, muss die Mechanik so stehen:

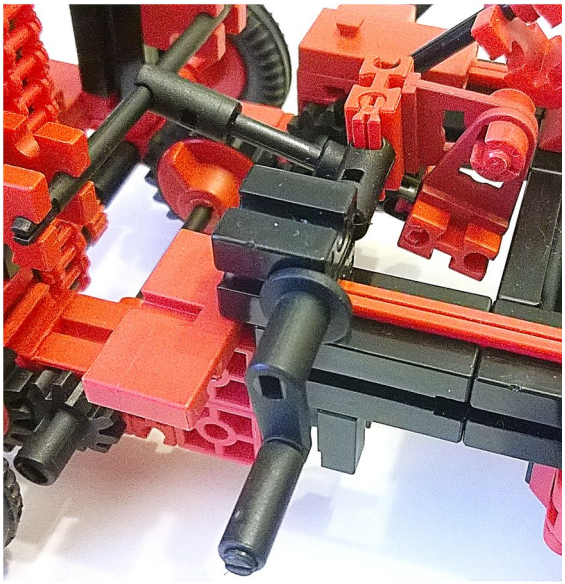


Abb. 10: Gabel nach hinten gekippt

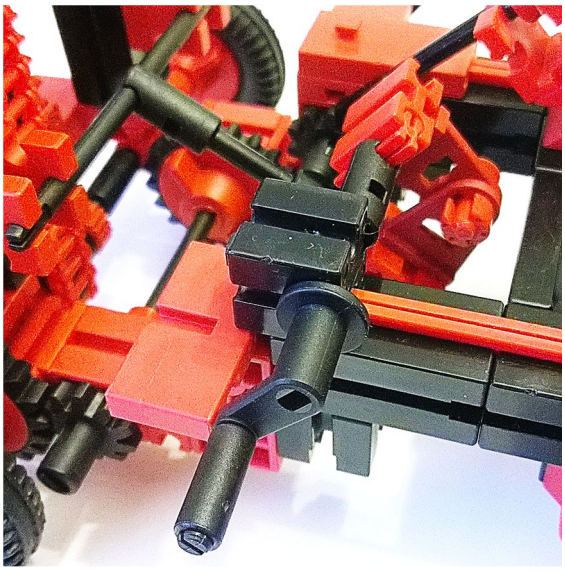


Abb. 11: Gabel nach unten gekippt

Die Kurbeln lassen sich angenehmer drehen, wenn immer eine Lagerhülse 15 (36819) aufgesteckt wird.

Der Schnapp-Effekt ergibt sich, wenn der Seilklemmstift im Gelenk oberhalb der Kurbelachse zu liegen kommt. Dann rastet die Mechanik ein. Der Baustein 5 15 · 30 3N direkt vor der Kurbel dient als Anschlag. Wie man sieht, kann auch er nach Bedarf nach vorne oder hinten justiert werden.

Dreht man die Kurbel nach vorne bis zu diesem Anschlag, wird die Gabel ganz abgekippt (Abb. 11).

Der Kettenzug

Auf der rechten Seite des Staplers befindet sich die Kurbel zum Hochheben der Gabel:

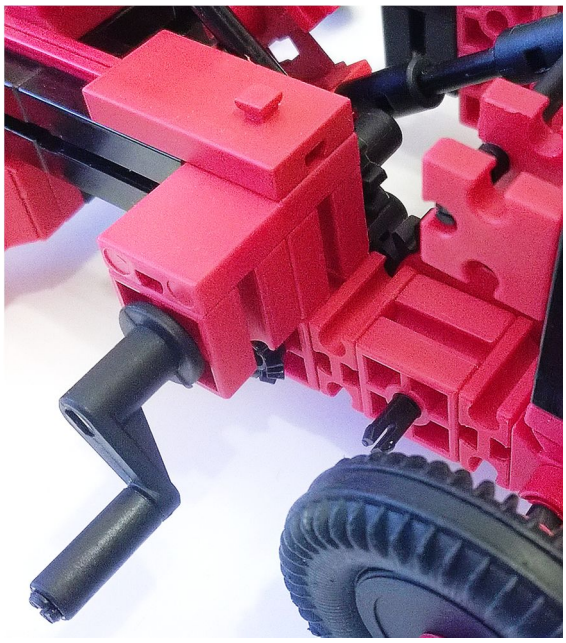


Abb. 12: Die Kurbel für den Kettentrieb

Die Kurbel geht also über eine Rastachse 45 auf ein Rast-Z10. Eine Etage tiefer sitzt wieder ein Rast-Z10, hier auf einer Rastachse 30 (35061) + Kegelzahnrad Z10 m1 45°. Das dient nur als Zwischenzahnrad zum nächsten Z20 (siehe Abb. 13).

Auf der linken Fahrzeugseite geht es (Abb. 14) weiter mit noch einer Z10-auf-Z20-Untersetzung. Abb. 15 zeigt das ganze Modell von unten.

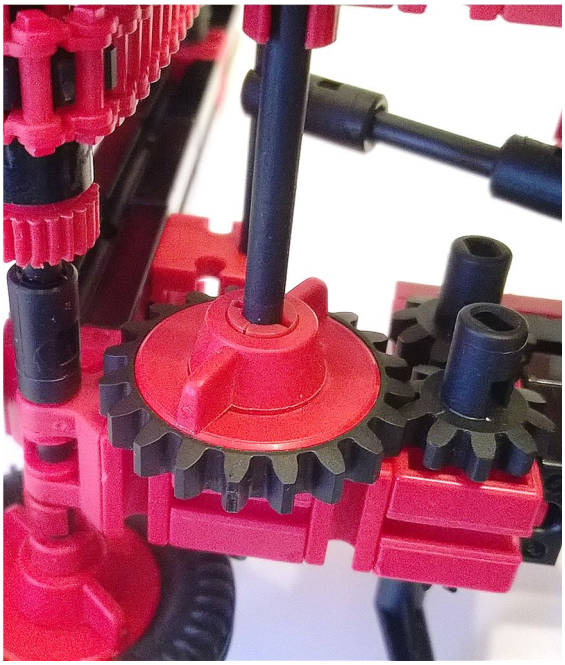


Abb. 13: Zwischen-Z10 und Z20 auf der rechten Fahrzeugseite von unten gesehen

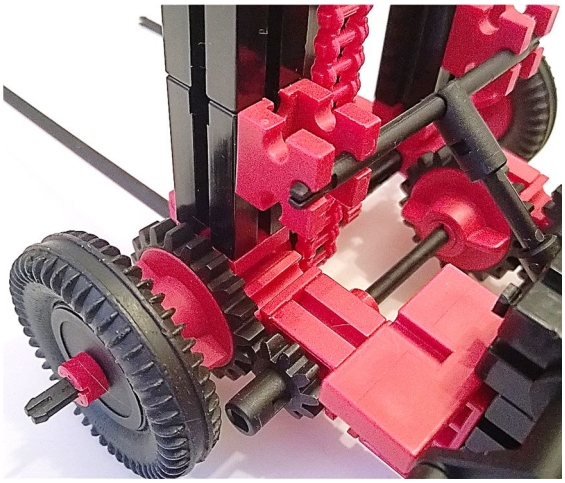


Abb. 14: Untersetzung zur kombinierten Vorderachse/Antriebsachse der Kette

Damit die angehobene Gabel nicht durch das Gewicht der Last wieder nach unten rutscht, brauchen wir noch eine *Sperrklinke* [4]. Man kann immer hochkurbeln, aber zum Herablassen der Last muss man erst die Sperre lösen.

Die Sperrklinke ist schnell aufgebaut: Wie Abb. 16 zeigt, besteht sie lediglich aus Gelenkstein, Lagerbock, Achse und Klemmring. Die Achse greift in das Z10 an der Kurbel ein, das also (im Bild) gegen den Uhrzeigersinn frei gedreht kann, aber nicht

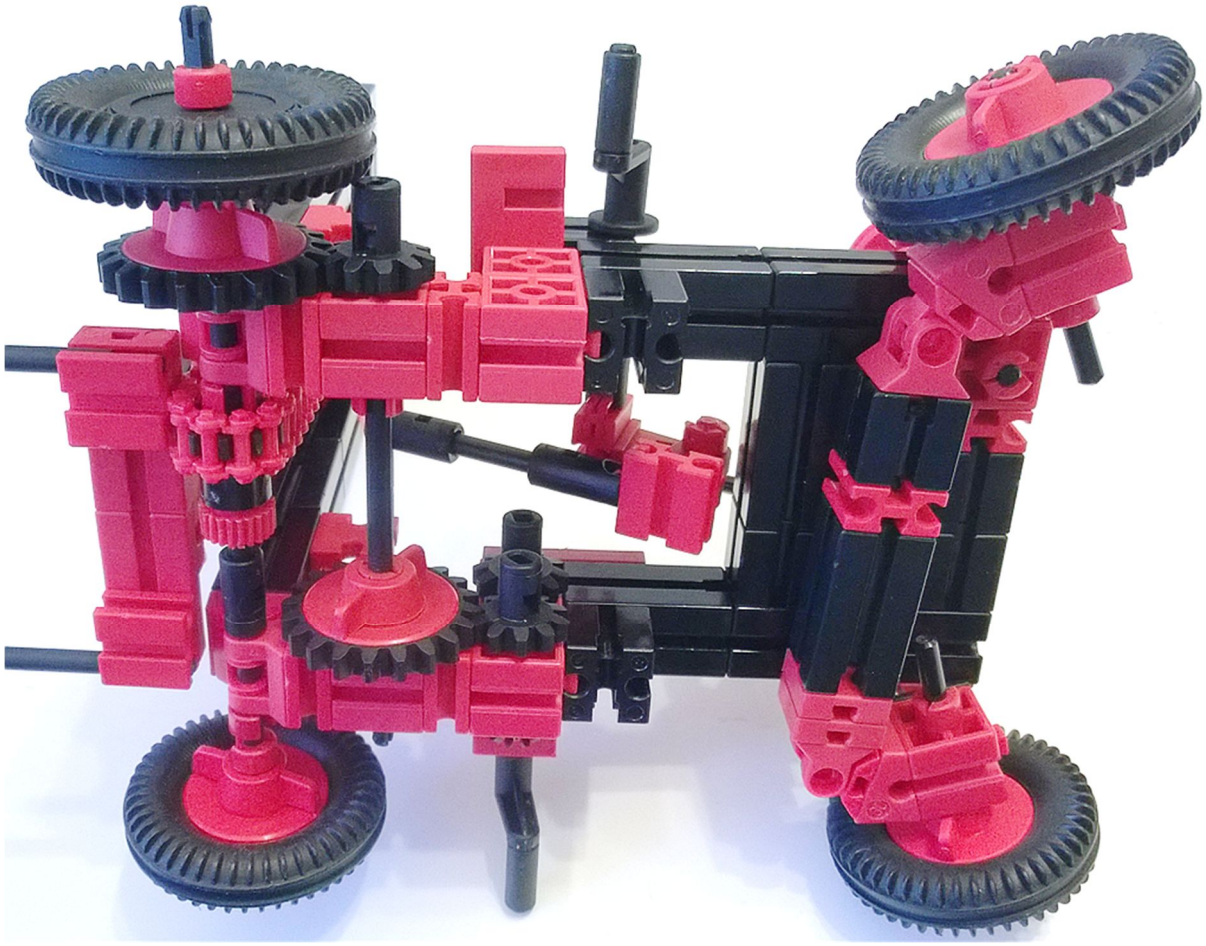


Abb. 15: Gesamtansicht des Fahrzeugbodens

zurück. Will man die Gabel herunterkurbeln, muss man die Sperrklinke erst etwas anheben, um das Z10 freizugeben.

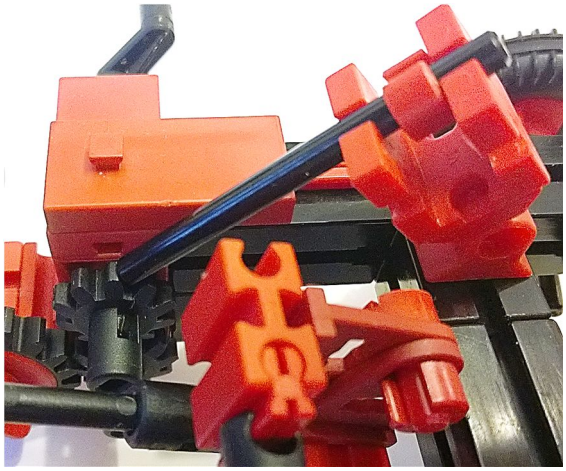


Abb. 16: Sperrklinke

Einhängen der Gabel

Schließlich brauchen wir noch die Gabel selbst. Sie ist anhand Abb. 17 aus wenigen Standardteilen schnell aufgebaut. In die Grundbausteine sind die Zahnstangen eingeschoben. Unten klemmen die Rastachsen 110 (richtig herum gedreht) ohne weitere Befestigung einfach in einer Gruppe aus BS7,5, Federnocken, Baustein 5 15 · 30 (35049) und Baustein 5 15 · 30 3N (38428).

Den oberen Teil des Staplers, in den die Gabel von unten eingeschoben wird, stellt Abb. 18 dar. Zuverlässig drückt er die Gabel gegen das dahinterliegende Z10, und sie hat auch seitlich nicht zu viel Spiel. Wie Abb. 1 schon zeigte, wird danach noch eine Bauplatte 30 · 60 2Z (38249) auf den (in Abb. 18) linken unteren BS15 und den

rechten BS7,5 aufgeschoben. Auf der Rückseite (Abb. 19) sieht man, wie das obere Rast-Z10 gerade passend in die Zahnstange eingreift.

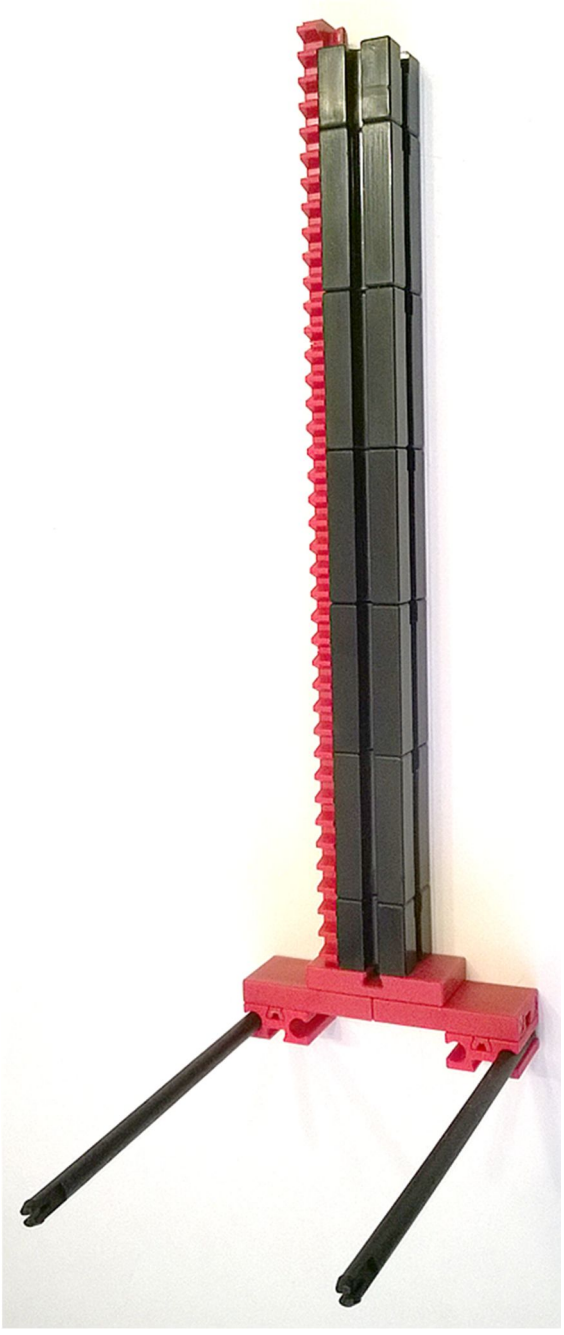


Abb. 17: Die Gabel

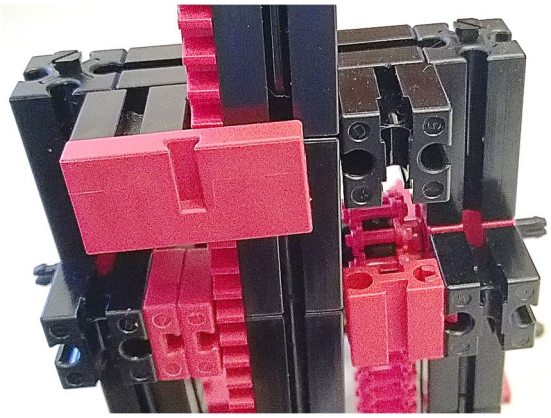


Abb. 18: Einhängen der Gabel

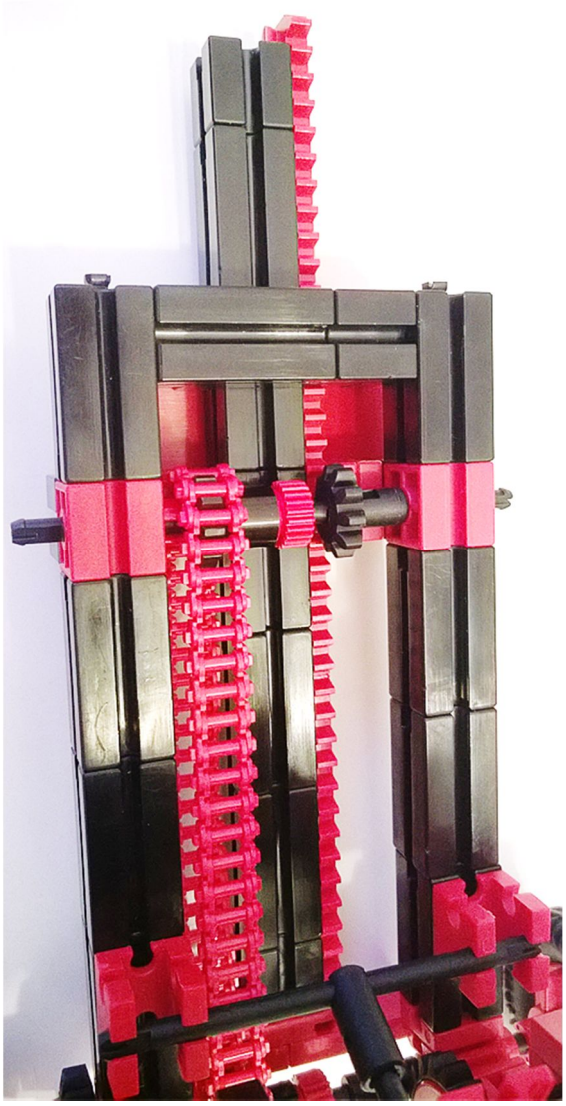


Abb. 19: Eingriff des Rast-Z10 in die Zahnstange

Ist die Gabel korrekt eingehängt, kommt sie unten direkt vor der Vorderachse an:

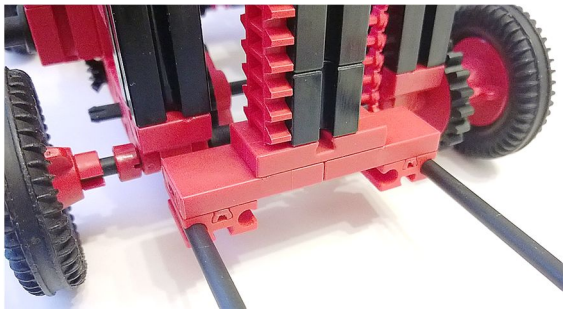


Abb. 20: Die abgesenkte Gabel vor der Vorderachse

Damit ist der Aufbau fertig und es kann losgespielt werden. Die Lagerarbeiter rufen schon.

Quellen

[1] Falk, Stefan: *Der Wohnzimmer-Dienstreisen-Urlaubs-Notfallkasten*. ft:pedia 1/2016, S. 31-36.

[2] Falk, Stefan: *Raffiniertes mit Achsen*. ft:pedia 3/2013, S. 38-39.

[3] Fox, Dirk: *Lenkungen (Teil 1)*. ft:pedia 1/2011, S. 16-21.

[4] Fischer-Werke: *Gesperre,* in: hobby 1 Begleitbuch Band 1. 1972, S. 47-49.

Modell

fischertechnik-Kegelbahn

Dirk Wölffel

Seit einiger Zeit feiern Spielgeräte wie der fischertechnik-Flipper ihr Comeback. Auf Modellbauausstellungen sind diese Modelle immer ein Publikumsmagnet. Vater gegen Sohn, wer hat die meisten Punkte. Daher lag der Gedanke nahe, etwas zu konstruieren, womit man die Besucher fesseln kann, um sie für fischertechnik zu begeistern. Auf der Suche nach einem geeigneten Modell kam mir die Idee, eine Kegelbahn zu bauen. Die Herausforderungen lagen darin, dem Original möglichst nahe zu kommen und Mechanik und Elektronik zu verbinden.

Die Kegelbahn

Beim Durchsuchen der fischertechnik-Datenbank nach einer Bauvorlage bin ich fündig geworden: auf Seite 5 der 1979-3 – Club-Nachrichten (Abb. 1). Auf dem Foto war leider nicht viel zu erkennen, daher war Kreativität gefragt. So galt es zunächst einige Probleme zu lösen.

Abb. 1: Kegelbahn (Club-Nachrichten 1979-3)

Der Aufbau

Das Modell aus den Club-Nachrichten war die Grundlage für den Bau der Kegelbahn. Als Kugelrinnen kamen die Flexschienen wie gerufen, sie passen sehr gut dazu.

Abb. 2: Der Aufbau

Ich habe die Kegelbahn erhöht gebaut, um die Kegelerkennung von unten zu lösen. Die Maße betragen 78 cm (Länge) x 27 cm (Breite) x 33 cm (Höhe), siehe Abb. 3.

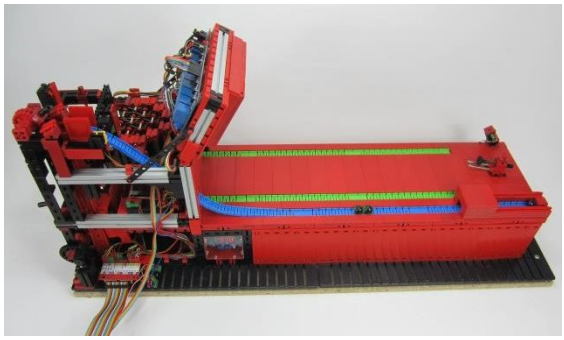


Abb. 3: Kegelbahn seitlich

In Abb. 4 sieht man rechts die *klappbare Kegelanzeige* mit der Elektronik. Diese wurde nachträglich eingebaut, um sie beim Transport nicht zu beschädigen. Darunter liegt der Spannungswandler für die 5 V-Elektronik und links erkennt man die Förderkette für die Murmeln.

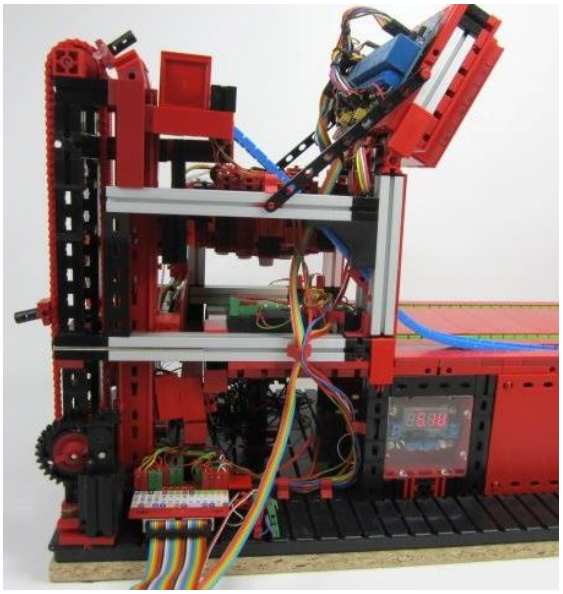


Abb. 4: Kegelbahn hinten

Die Mechanik

Die Murmeln (14 mm Durchmesser) laufen über den Kugelrücklauf in den *Kugelsammelkasten*. Mittig ist eine *bewegliche Abschussrampe* befestigt. Damit lassen sich die Murmeln zielgenauer steuern und verlassen nicht so schnell die Kegelbahn.

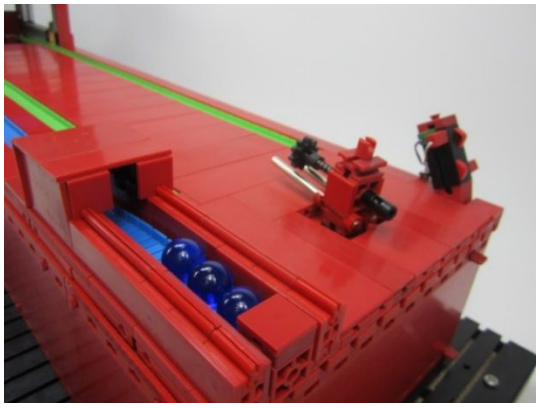


Abb. 5: Kegelbahn vorne

Rechts sind zwei Mini-Taster angeordnet zum Starten und zur Eingabe über das Display (Abb. 5).

Die Murmeln werden über den Mitnehmer der Kette hochgeführt. Anschließend wird über den gelben Fototransistor gezählt (Abb. 6). Pro Wurf bekommt ein Spieler drei Murmeln.

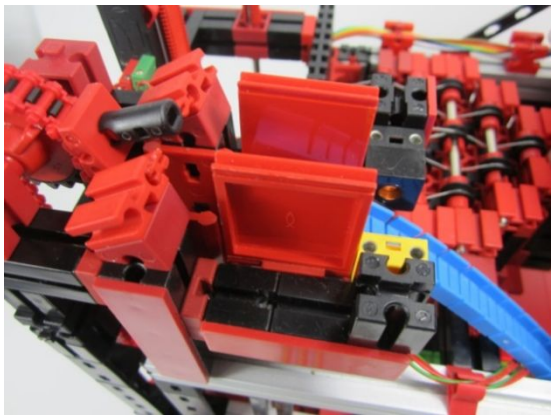


Abb. 6: Lichtschranke zum Zählen der Murmeln

Die Kegel werden mit einem Motor XS mit Hubgetriebe hochgezogen. Aufgestellt werden die Kegel über Seilrollen mit Hilfe eines Seils (Abb. 7).

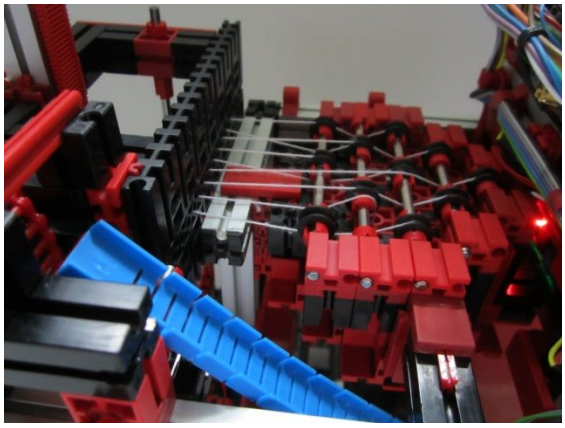


Abb. 7: Seilführung

Die Kegel

Nach langer Suche hatte ich endlich die passenden Kegel gefunden. Die Kegel stammen aus einem Miniatur-Bowling-Spiel für Kinder (Abb. 8). Die Variante mit „Mensch-ärger-Dich-nicht“-Spielfiguren aus den Club-Nachrichten hatte nicht

funktioniert, da die Auflagefläche der Spielfiguren zu breit ist und diese daher nicht umfallen.

Abb. 8: Miniatur-Bowling

Die Kegelerkennung

In einer ersten Variante verwendete ich eine Pixy-Camera (rechts im Bild) zur Erkennung der Kegel [1]. Die Kegel standen dazu auf einer Acrylglasscheibe und hatten auf der Unterseite einen farbigen Punkt. Leider musste ich diese Version verwerfen, da immer Störlicht von oben einfiel und dadurch die Bilderkennung fehlerhaft war (Abb. 9).

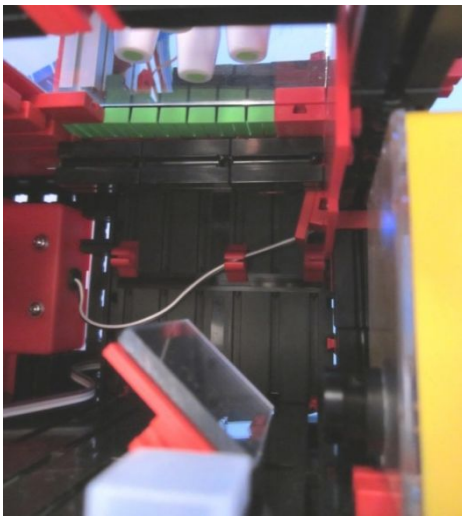


Abb. 9: Versuch 1: optische Bilderkennung

In der zweiten Variante habe ich dann neun Reedkontakte verwendet (Abb. 10). In die Kegel wurde ein 1 mm-Loch für das Seil gebohrt und von unten ein runder Ferritmagnet eingesetzt (Abb. 11). Diese Lösung erwies sich als sehr zuverlässig bei der Kegelerkennung.

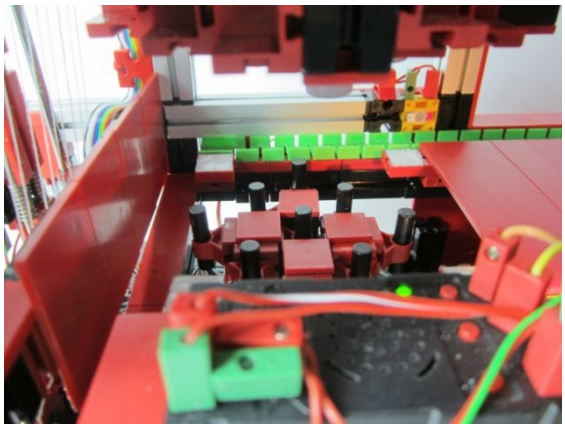


Abb. 10: Neun Reedkontakte

Abb. 11: Kegel mit Ferritmagnet

Die Anzeige

Als Anzeige dienen neun Leuchtsteine mit alten weißen Leuchtkappen und ein LCD1602 16x2 Display, das über das I²C-Protokoll angesteuert wird (Abb. 12).

Abb. 12: Leuchtsteine mit I²C-Display

Es kann mit einem oder zwei Spielern gekegelt werden. Jeder hat drei Würfe. Der Name wird über die beiden Taster vorne

eingegeben. Die Anzeige gibt die Anzahl der Würfe, die Punktzahl des Wurfs und die Summe der Würfe aus (Abb. 13).

Abb. 13: I²C-Display-Anzeige

Die Elektronik

Über den gelben Fototransistor (Abb. 14) wird erkannt, wann ein Wurf durchgeführt worden ist.

Abb. 14: Lichtschranke vorn

Die Kegelerkennung funktioniert über ein 8-Kanal-Relais und zwei I/O-Port-PCF8574 (Abb.15). Nach jedem Wurf werden die Reed-Kontakte ausgelesen und aufsummiert. Angesteuert wird dies über den I²C-Anschluss des Robo TX Controllers.

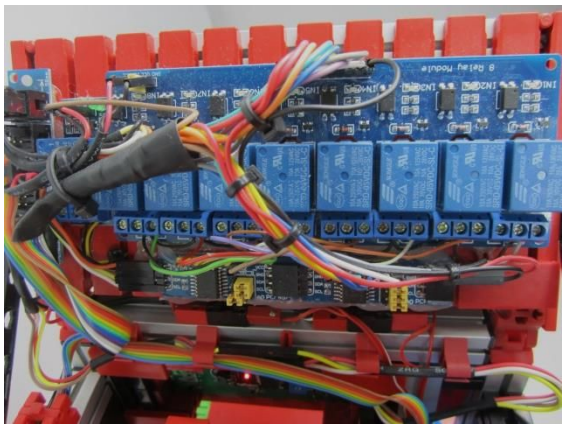


Abb. 15: 8-Kanal-Relais mit zwei PCF8574

Der Einsatz

Die fischertechnik-Kegelbahn kam zum ersten Mal auf der Modellbaumesse Neumünster vom 05.-06.03.2016 zum Einsatz.

Im Dauereinsatz zeigte sich die eine oder andere Schwäche der ursprünglichen Konstruktion, die in der hier vorgestellten Variante verbessert wurden – z. B. sind Glasmurmeln geeigneter als die fischertechnik-Metallkugeln, da die Ferrit-Magnete der Kegel nicht an ihnen hängenbleiben. Auch die Anzeige der Ergebnisse zweier Spieler erhöht den Spielspaß.

Ein Video der Kegelbahn findet ihr auf YouTube.

Referenzen

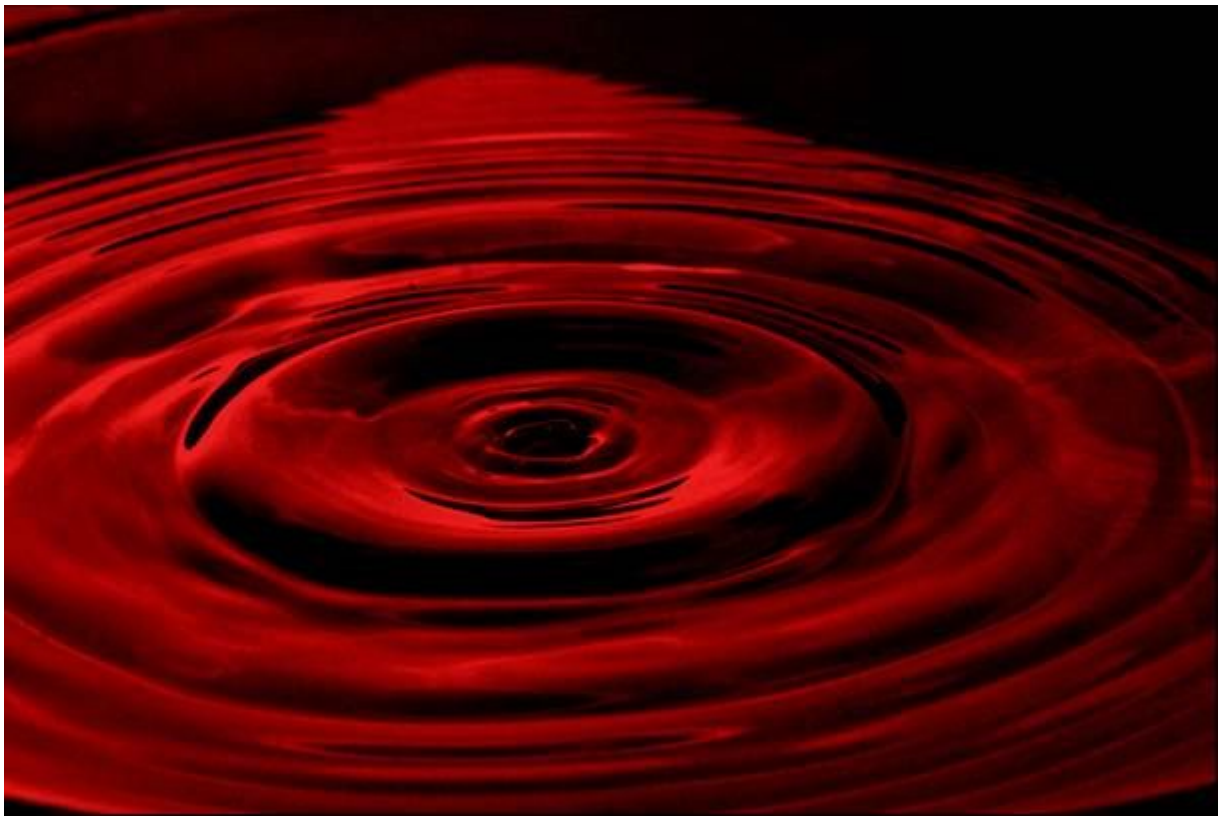
[1] Dirk Wölffel, Dirk Fox: *I²C mit dem TX – Teil 11: Die Pixy-Kamera*. ft:pedia 4/2014, S. 43-51.

Modell

Tropfen-Fotografie

Andreas Gail

Man glaubt es kaum, aber auch beim Fotografieren von Wassertropfen kann fischertechnik eine wichtige Rolle spielen. Im vorliegenden Beitrag werden einzelne Baugruppen aus vorangegangenen ft:pedia Beiträgen kombiniert.



Was wird benötigt?

Eine Reihe von Baugruppen aus vorangegangenen ft:pedia-Beiträgen werden in diesem Anwendungsfall miteinander kombiniert. Neben diversen fischertechnik-Teilen werden folgende Hauptkomponenten benötigt:

- Eine Schlauchquetschpumpe [1] zur Erzeugung von aufeinander folgenden Wassertropfen,
- ein Punktlaser [2] für eine Lichtschranke zur Wassertropfenerkennung,
- ein Modul [3] zur Kamera-Ansteuerung über IR,
- eine Kamera mit einem Objektiv zur Aufnahme kleiner Objekte,
- ein Blitzgerät, zur Kamera passend,
- eine Glasschüssel mit Wasser und
- ein Bildbearbeitungsprogramm.

Gesamtaufbau

Abb. 1 zeigt den Gesamtaufbau, der für die Aufnahme von Wassertropfen verwendet wurde. Die Durchführung selbst erfolgte jedoch bei deutlich reduzierter Umgebungshelligkeit. Sowohl die Kamera als auch der Blitz können bei der späteren Aufnahme notfalls auch in der Hand gehalten werden. Mindestens ein Blitz wird benötigt, wobei ein Kamera integrierter Blitz auch ausreichend ist.

Die Wasserschalte befindet sich unten im Turm. Auf dem etwa 70 cm hohen oberen Aufbau befinden sich gemäß Abb. 2 die Schlauchquetschpumpe und die Laser-Lichtschranke.

Wasser (ohne Zusätze) wird dabei durch den Wasserschlauch (Silikon, Außendurchmesser 4 mm) gepumpt. Etwa alle acht Sekunden bildet sich ein Tropfen, der sich ablöst und herunterfällt. Dieser Wassertropfen wird von der Laserlichtschranke detektiert (siehe Abb. 3).

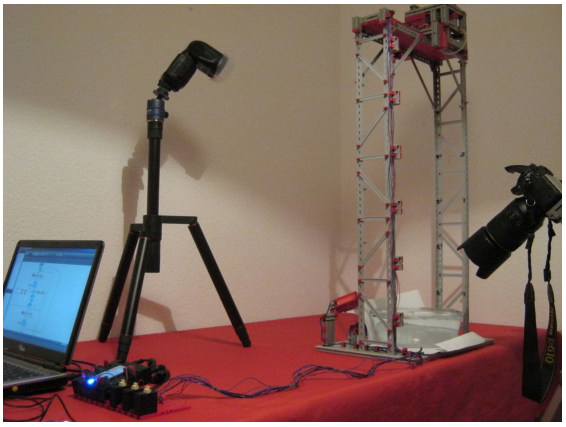


Abb. 1: Gesamtaufbau zur Aufnahme von Tropfen, die in eine mit Wasser gefüllte Schale fallen

Das Signal der Fotodiode wird von einem Robo TXT-Controller (TXT) ausgewertet und akustisch signalisiert. Wenn eine Aufnahme erfolgen soll, leitet der TXT einen Impuls an das Modul zur Kamera-Ansteuerung (Abb. 4).

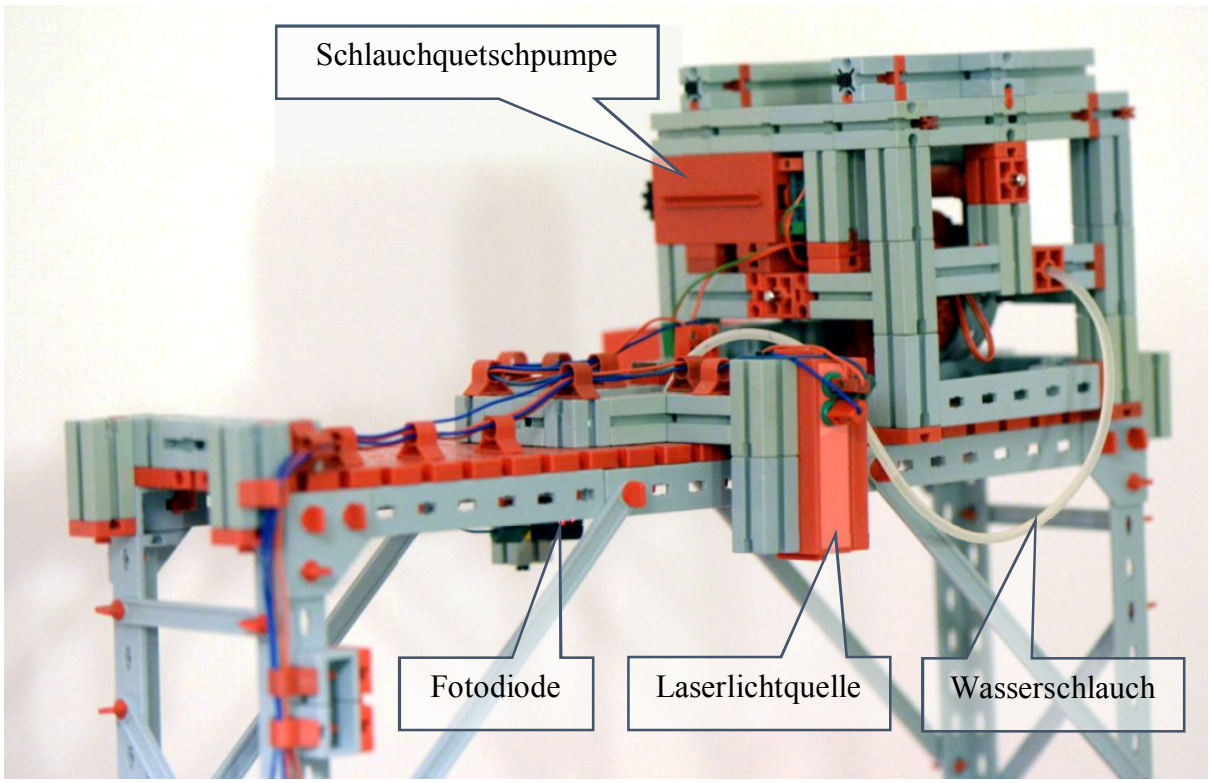


Abb. 2: Oberer Aufbau zur Erzeugung und Erkennung von Wassertropfen

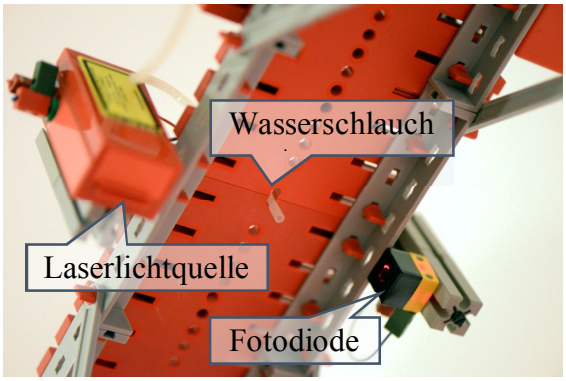


Abb. 3: Laserlichtschranke und Schlauchende zur Erkennung der erzeugten Wassertropfen

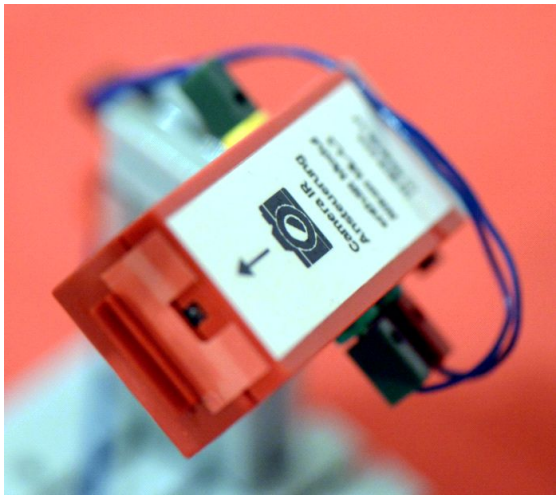


Abb. 4: IR-Modul zur Kamera-Ansteuerung

Geschwindigkeitsproblem

Die Zeitspanne, in der ein Wassertropfen 70 cm tief vom oberen Aufbau in die darunter befindliche wassergefüllte Schale fällt, ist bei der verwendeten Kamera deutlich zu kurz, um eine Aufnahme vorzubereiten und durchzuführen. Deshalb muss zunächst das Zeitintervall zwischen den einzelnen Tropfen gemessen werden. Dann ist nach einem erkannten Tropfen genügend Zeit, die Aufnahme zu starten, weil nun bekannt ist, wann der nachfolgende Tropfen kommen wird. Zur Zeitmessung der Tropfenabstände wurde das folgendes Robo Pro-Programm verwendet:

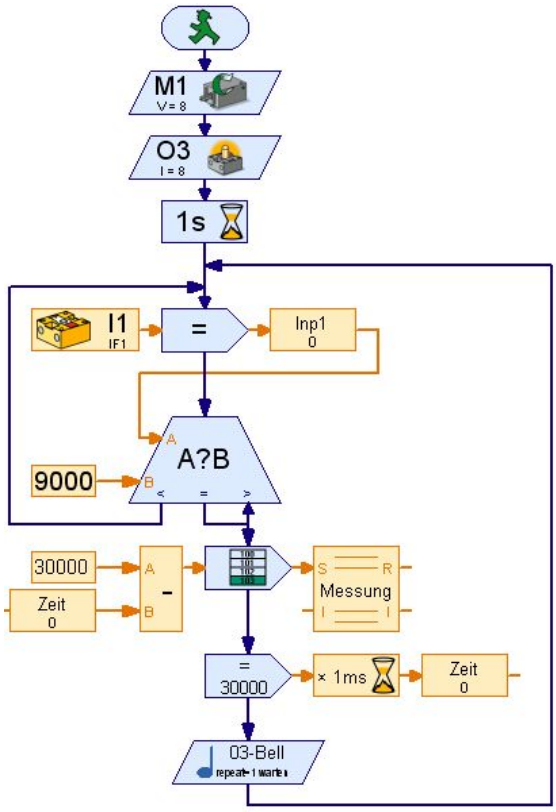


Abb. 5: Robo Pro-Programm zur Messung und Aufzeichnung der Zeitabstände zwischen einzelnen Tropfen

M1 im Programm steht für die Schlauchquetschpumpe und O3 für den Laser. Am Eingang I1 ist die Fotodiode angeschlossen, wobei dieser Eingang als „A 5k“, d. h. analoge Widerstandsmessung eingestellt wurde. Die zeitlichen Messergebnisse werden in die Liste *Messung* eingetragen und nach Programmende in eine externe Datei im CSV-Format exportiert. Aus mindestens 20 Messungen sollte eine mittlere Zeitspanne errechnet werden. Die dabei auftretenden Ungenauigkeiten kommen einerseits durch die Gleichlaufschwankungen der Schlauchquetschpumpe und andererseits durch das Laufzeitverhalten des TXT zustande. Diese Ungenauigkeiten haben jedoch Vorteile bei der späteren Aufnahmeserie, weil so automatisch verschiedene zeitliche Abschnitte beim Eintropfen fotografiert werden.

Einstellung der Kamera

Verwendet wurde eine Vollformat-Kamera (Sensorgröße 24 mm · 36 mm) mit einem 105 mm-Makro-Objektiv (Lichtstärke 2,8). Folgende Einstellungen wurden vorgenommen:

- M (manuelle Belichtungseinstellung, Autofokus aus)
- Verschlusszeit 1/200 s
- Blende 36 (minimale Einstellung)
- Empfindlichkeit ISO 6400
- interner Blitz abgeschaltet, externer Blitz manuell (1/128)

Obwohl wie bereits oben erwähnt ein Stativ für die Kamera nicht unbedingt erforderlich ist, so ist es doch empfehlenswert. Zunächst muss nämlich eine manuelle Scharfeinstellung vorgenommen werden. Das geht einfach, indem ein beschriebenes Papierstück kurzzeitig auf die Wasseroberfläche in der Wasserschale aufgelegt und dann auf seine Beschriftung scharfgestellt wird. Das sollte an der Stelle sein, an der der Tropfen ins Wasser eintaucht. Danach wird das Papierstück wieder entfernt. Bevor mit den Aufnahmen begonnen wird, sollte sichergestellt sein, dass das Wasser in der Schale sauber ist, also keine Trübstoffe oder Fasern enthält. Nun kann mit den ersten Testaufnahmen begonnen werden. Dabei sind die Vorlaufzeit zur Aufnahmeauslösung auszuprobieren und die Belichtung zu überprüfen. Zur Korrektur sind ggf. die ISO-Empfindlichkeit, Blende oder Blitzleistung anzupassen.

Einfluss der ISO-Empfindlichkeit

Grundsätzlich bewirkt die Einstellung eines kleineren ISO-Wertes eine bessere Bildqualität hinsichtlich des Bildrauschens.

Einfluss der Blendenzahl

Die maximale Blendenzahl ist gleichbedeutend mit einer minimalen Blendenöffnung. Das hat den Vorteil, dass das aufgenommene Bild eine große Tiefenschärfe hat. Somit ist der aufgenommene Tropfen nicht nur an der Stelle scharf abgebildet, an dem zuvor manuell die Fokussierung vorgenommen wurde, sondern auch in näher oder weiter entfernt liegenden Bereichen. So können auch kreisförmige Wellenbewegungen um die Eintauchstelle des Tropfens scharf abgebildet werden.

Einfluss der Verschlusszeit

Dieser Einstellwert ist vergleichsweise unbedeutend. Wichtig ist nur, dass die Kamera die Verschlussöffnung und die Blitzauslösung synchronisieren kann.

Einfluss der Blitzleistung

Dieser Einstellung kommt eine besondere Bedeutung zu. Die Bewegung des Wassertropfens verläuft sehr schnell; trotzdem soll möglichst alles scharf abgebildet werden. Deshalb wird die Blitzleistung möglichst klein eingestellt. Eine kleine Blitzleistung ist dabei gleichbedeutend mit einem sehr kurzen Blitz. Die oben genannte Einstellung (1/128 der Leistung) hat somit zur Folge, dass bei dem verwendeten Blitz die Blitzdauer nur 1/38500 s dauert. Länger als 1/20000 s sollte der Blitz für scharfe Abbildungen möglichst nicht dauern.

Aufnahmen

Wenn alles eingestellt ist, kann mit der mehr oder weniger automatischen Aufnahmeserie begonnen werden. Die Position des Blitzgerätes sollte dabei immer wieder mal verändert werden. Einstellungen in Richtung der Kamera, gegen die Kamera, von oben, von unten, seitlich, in die Schale hinein usw. ergeben unterschiedliche Effekte. Dabei kann das Blitzgerät durchaus in der Hand gehalten werden. Auch die Kameraposition sollte immer wieder mal verändert werden. Nach 200 bis 300 Aufnahmen sollte man alles, was es prinzipiell zu sehen gibt, im Kasten haben.

Bildoptimierung

Wenn man sich die aufgenommenen Bilder betrachtet, sehen diese mitunter recht öde aus. Jetzt kommt ein passendes Bildbearbeitungsprogramm zum Zuge. Aus meiner Sicht sehr empfehlenswert ist das frei zugängliche Bildbearbeitungsprogramm GIMP [4]. Die nachfolgenden Beispiele wurden mit der Version 2.8.14 bearbeitet.

Abb. 6: Startfenster des freien Bildbearbeitungsprogramms GIMP 2.8.14

Ein unbearbeitetes Bild zeigt Abb. 7:

Abb. 7: Rohaufnahme

Nachdem die Darstellung so verändert wurde, so dass der volle zur Verfügung stehende Farbraum ausgenutzt wird, hat das Bild bereits deutlich klarere Strukturen, wie in Abb. 8 deutlich wird. Im Bildbearbeitungsprogramm ist dazu die Funktion *Farben* | *Werte...* | *Automatisch* | *OK* verwendet worden. Wenn ein Farbstich entfernt werden soll, so ist dieses über *Bild* | *Modus* | *Graustufen* und anschließend mit *Bild* | *Modus* | *RGB* möglich.

Abb. 8: Aufnahme nach Farbraumanpassung

Durch Zoomen auf einen Bildausschnitt der Gesamtaufnahme kommt man zu Abb. 9. Hierzu verwendet man die Funktion *Bild* | *Leinwandgröße...*

Abb. 9: Aufnahme nach Farbraumanpassung und Ausschnittsvergrößerung

Um die Sache noch interessanter aussehen zu lassen, kann man alles noch etwas einfärben.

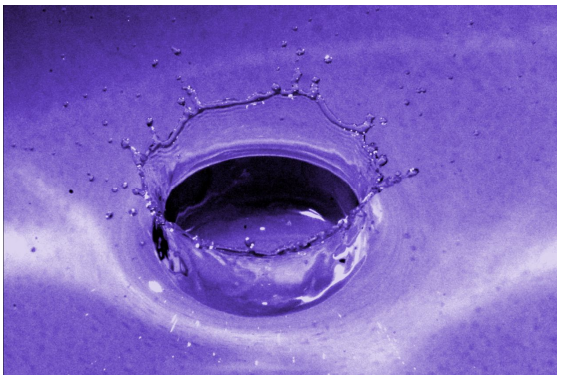


Abb. 10: Aufnahme nach Farbraumanpassung, Ausschnittsvergrößerung und Einfärbung

Hier sind alle erdenklichen Farben, Helligkeiten und Kontraste möglich. Dazu stehen die Funktionen *Farben* | *Einfärben* oder auch *Farben* | *Helligkeit/Kontrast* bereit. Weitere Beispiele zeigen Abb. 11 und 12:

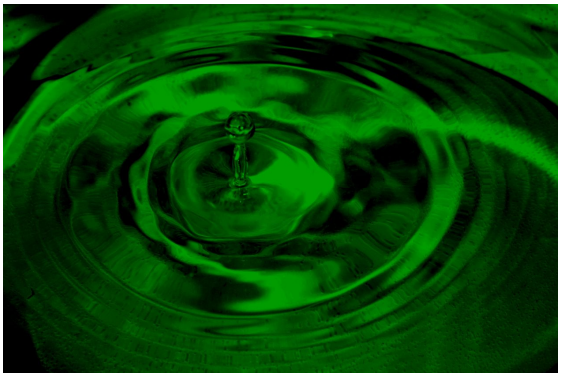


Abb. 11

Abb. 12

Referenzen

[1] Gail, Andreas: *Schlauchquetschpumpe*. ft:pedia 1/2016, S. 40-41.

[2] Gail, Andreas: *Einstieg in Experimente mit Lasern*. ft:pedia 2/2014, S. 14-17.

[3] Gail, Andreas: *Nikon-Kamera-Ansteuerung über IR*. ft:pedia 4/2015, S. 40-43.

[4] Bildbearbeitungsprogramm *GIMP*. https://www.gimp.org

[5] Gail, Andreas: *fischertechnik: Wassertropfen Fotographie mit Nikon*. youtube-Video, 2016.

Grundlagen

Planetengetriebe

Thomas Püttmann

Wenn Getriebe mehr als zwei An-/Abtriebe besitzen, handelt es sich um Planetengetriebe. Wir stellen zwei Beispiele vor und erklären daran, wie solche Getriebe funktionieren. Zum Schluss beschreiben wir ein Funktionsmodell eines 2-Gang-Schaltgetriebes.

Geschichte

Bei einem Planetengetriebe sind mehrere Drehbewegungen um eine zentrale Achse miteinander verkoppelt. Einige Zahnräder (Planeten) drehen sich um andere (Sonnen), die sich selbst in der Regel auch noch drehen. Der Name *Planetengetriebe* deutet nicht nur auf den Aufbau dieser Getriebe hin, sondern auch auf deren erste bekannte Anwendung: astronomische Vorgänge mechanisch nachzubilden. Zu diesem Zweck befanden sich Planetengetriebe im *Mechanismus von Antikythera* (ca. 100 v. Chr.) und in den astronomischen Uhren ab ca. 1300 n. Chr.

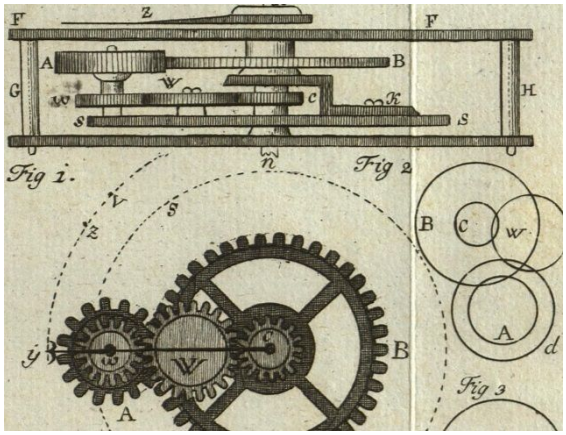


Abb. 1: Auszug aus dem Buch Neues Rädergebäude [1] *von 1791 (Quelle: NTK Prag)*

Die erste systematische Abhandlung über Planetengetriebe hieß *Neues Rädergebäude* [1] und erschien im Jahr 1791. In ihr beschrieb der Barfußmönch *Frater David Sancto Cajetano* (1726-1796), wie man mit möglichst wenigen Zahnrädern Übersetzungen wie 1:1009 erzielt, die zur mechanischen Nachbildung bestimmter astronomischer Vorgänge gebraucht werden. Dieses Wissen hatte er schon ab den 60er Jahren des 18. Jahrhunderts zum Bau herausragender astronomischer Uhren eingesetzt.

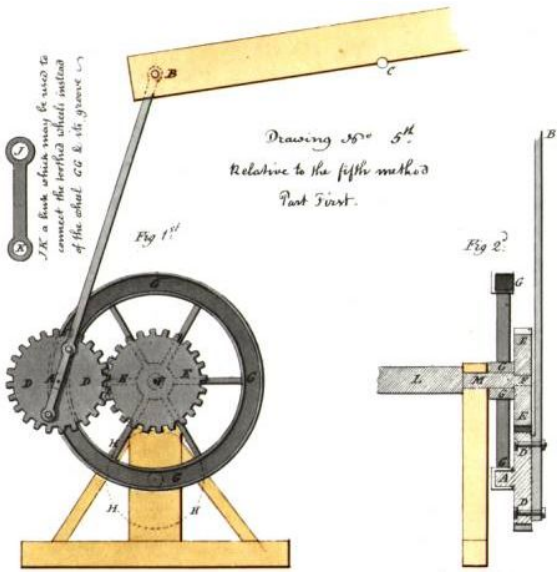


Abb. 2: Auszug aus James Watts Patentschrift aus dem Jahr 1781 [2]

Der Schotte *James Watt* (1736-1819) verwendete ein Planetengetriebe zu einem ganz anderen Zweck: Im Jahr 1781 wandelte er die oszillierenden Auf- und Abbewegung eines Kolbens in seiner Dampfmaschine in eine Drehbewegung um [3].

Das theoretische Verständnis und der gezielte Entwurf von Planetengetrieben verbreitete sich vor allem mit dem Buch *Principles of Mechanisms* [4] von *Robert Willis* (1800-1875) aus dem Jahr 1841.

Die ersten Patente auf Nabenschaltungen in Fahrrädern erhielten laut Wikipedia der Amerikaner *Seard Thomas Johnson* im Jahr 1895 und der Engländer *William Reilly* im Jahr 1896 [5].

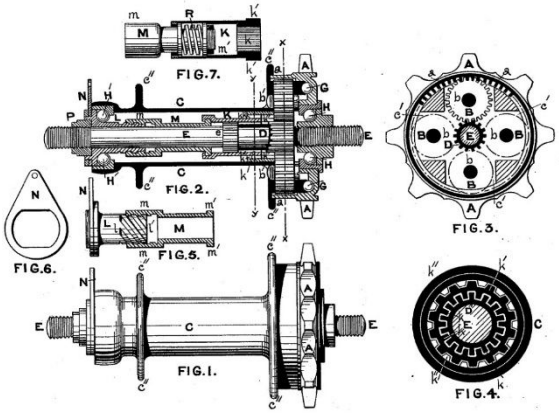


Abb. 3: Auszug aus William Reilly's und Charlton Haigh's Patentschrift aus dem Jahr 1897 (US Patent 588657, Quelle: Google)

Heute finden sich Planetengetriebe an vielen weiteren Stellen im Fahrzeugbau, zum Beispiel in Automatikgetrieben oder bei Hybridantrieben.

Übersetzung 5:2

Eine der ersten Anwendungen von Planetengetrieben war das Erzielen von vorgegebenen Übersetzungen mit wenigen Zahnrädern, die nicht zu viele Zähne haben sollten [1].

Bei Verwendung der fischertechnik-Zahnräder bekommt dieses Problem noch mehr Gewicht, da es nur wenige Zahnradtypen gibt. Somit gibt es nur sehr wenige Übersetzungen, die mit einer einzelnen Stirnradgetriebestufe erzielt werden können. Im Wesentlichen sind dies die Übersetzungen 2:1, 3:1, 4:1, 3:2 und 4:3. Es gibt zum Beispiel keine direkte Möglichkeit, eine Übersetzung von 5:2 oder 15:1 zu realisieren.

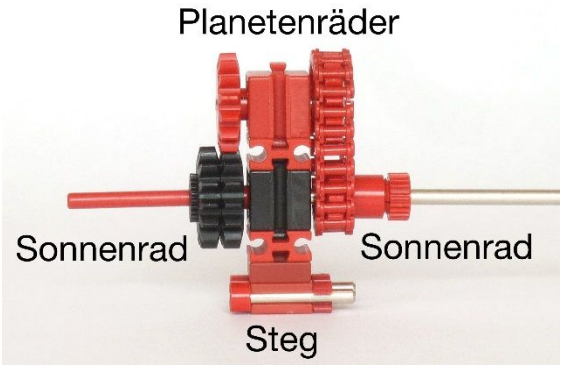


Abb. 4: Planetengetriebe, mit dem eine Übersetzung von 5:2 erzielt wird

Unser erstes Planetengetriebe liefert genau diese Übersetzung von 5:2, siehe Abbildung 4. Die zentrale Komponente ist der *Steg* oder *Planetenträger*. In dessen Mitte befindet sich ein Baustein mit Ansenkung, in den von links eine Kunststoffachse mit Vierkant gesteckt ist. Auf dieser Achse drehen sich zwei Zahnräder Z15 frei. Sie bilden das erste Sonnenrad. Von rechts steckt in dem Baustein mit Ansenkung frei drehbar eine Metallachse, auf der ein Z10 als zweites Sonnenrad fungiert.

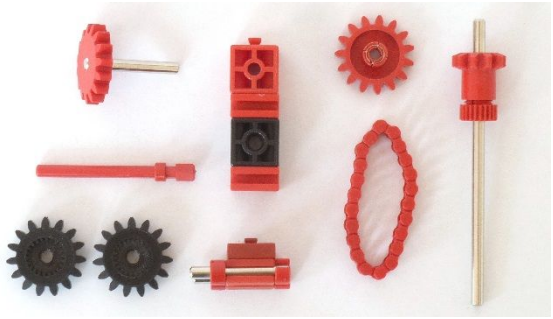


Abb. 5: Komponenten des Getriebes

Die beiden Planeten sind Z15 mit Klemmring. Alternativ kann man freilaufende Z15 mit Papier auf der Metallachse 30 festklemmen. Das Z10-Sonnenrad rechts ist über eine Kette mit einem der beiden Z15-Planeten verbunden. Dadurch ergibt sich bei festgehaltenem Steg eine Übersetzung von 3:2 vom rechten Z10-Sonnenrad auf die Planeten, d. h. wenn das rechte Sonnenrad drei Umdrehungen macht, machen die Planeten zwei Umdrehungen, siehe Abbildung 6. Auf der linken Seite wird nun die Drehbewegung von den

Planeten auf das linke Z15-Sonnenrad übertragen. Dabei findet nur eine Umkehrung des Drehsinns statt. Bei festgehaltenem Steg liegt also eine Übersetzung von 3:-2 zwischen dem rechten und dem linken Sonnenrad vor.

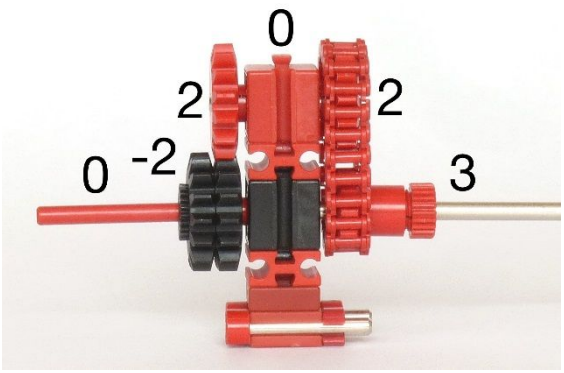


Abb. 6: Festgehaltener Steg

Dreht man nun das gesamte Getriebe starr zwei Umdrehungen weiter, so hat man das schwarze Z15-Sonnenrad so zurückgedreht, als hätte es sich insgesamt gar nicht bewegt. Der Steg hat insgesamt zwei Umdrehungen gemacht und das rechte Sonnenrad fünf. Daran erkennt man, dass die Übersetzung zwischen dem rechten Sonnenrad und der mit dem Steg fest verbundenen roten Kunststoffachse mit Vierkant bei festgehaltenem schwarzen Z15-Sonnenrad 5:2 beträgt. Ob man nämlich die beiden Bewegungen (Drehung bei festem Steg und starre Drehung des gesamten Getriebes) nacheinander ausführt oder gleichzeitig überlagert, liefert das gleiche Ergebnis.

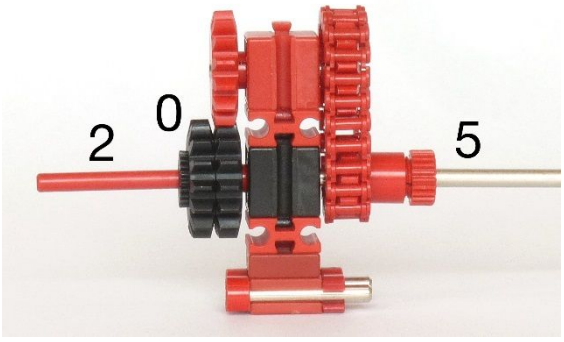


Abb. 7: Festgehaltenes linkes Sonnenrad

In der Praxis wird man somit das linke der beiden schwarzen Z15 an der Lagerung der roten Kunststoffachse befestigen und die Übersetzung von 5:2 zwischen den beiden fluchtenden Achsen nutzen.

Übersetzung 15:-1

Eine der wichtigsten Anwendungen von Planetengetrieben ist das Erzielen hoher Übersetzungsverhältnisse mit wenigen Zahnrädern und auf kleinem Raum. Als Beispiel realisieren wir die Übersetzung von 15:-1, die man sehr gut bei Uhren als eine Stufe der 60:1-Übersetzung zwischen Sekunden- und Minutenzeiger benutzen kann. Das Getriebe in Abbildung 8 ist aber auch rein an sich wegen der stark verschiedenen Drehgeschwindigkeiten der Zahnräder und Wellen ein faszinierendes Anschauungsobjekt, an dem auch kleinere Kinder mit großer Begeisterung kurbeln.

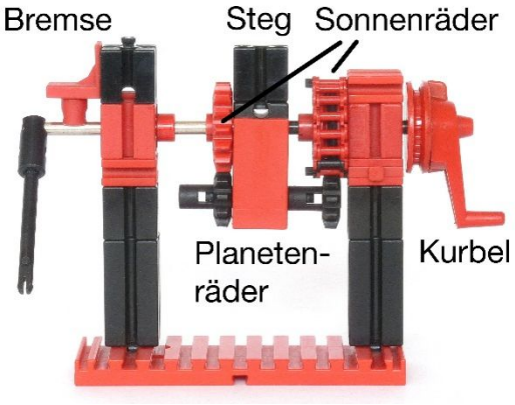


Abb. 8: Zwischen Kurbel und Zeiger liegt eine Übersetzung von 15:-1 vor.

Speziell ist an diesem Modell der Einsatz eines Selbstbau-Z16. Es besteht aus einer geschlossenen Kette aus 14 roten Kettengliedern und zwei schwarzen Kettengliedern. Diese Kette wird auf eine Seilrolle mit O-Ring aufgezogen. Ist kein passender O-Ring vorhanden, so kann man auch einen Pneumatikschlauch auf 64 mm ablängen und zwischen Seilrolle und Kette einpassen. Dabei muss man allerdings aufpassen, dass beide Enden des Schlauchs nicht durch die Löcher der Kette herausstechen. Idealerweise verklebt man die Enden des Pneumatikschlauchs.

Das Z16 kann sich an und für sich frei auf der schwarzen Kunststoffachse mit Vierkant frei drehen. Die seitlichen Enden der beiden schwarzen Rastkettenglieder passen aber genau in die Aussparungen zwischen den Federnocken und den Bausteinen 7,5 an der rechten Lagerung. Das Z16 kann somit regelrecht in den Lagerblock eingesteckt werden, siehe Abbildung 9.

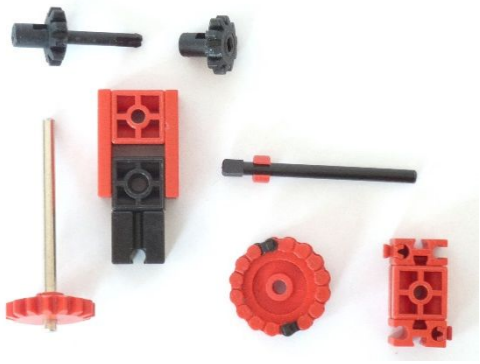


Abb. 9: Komponenten des Getriebes

Der Steg ist bei diesem Getriebe fest mit der Kurbel verbunden. Hält man die Kurbel und damit den Steg fest, so liegt zwischen Z16 und Z15 offensichtlich eine Übersetzung von 15:16 vor, d. h. drehen wir das Z16 15-mal, so dreht sich das Z15 16-mal im gleichen Sinn. Drehen wir nun das gesamte Getriebe starr 15-mal im Gegensinn, so haben wir das Z16 zurückgedreht, das Z15 hat insgesamt eine Umdrehung gemacht und Kurbel und Steg insgesamt 15 im Gegensinn. Damit liegt bei festem Z16 eine Übersetzung von -15:1 oder 15:-1 von Kurbel auf Zeiger vor.

Eine Stärke der Planetengetriebe sind die fluchtenden Wellen, die ein leichtes Hintereinanderschalten von Getrieben ermöglichen. Schaltet man beispielsweise zwei der Getriebe aus Abbildung 8 hintereinander, so hat man mit 225:1 schon eine recht große Übersetzung ins Langsame.

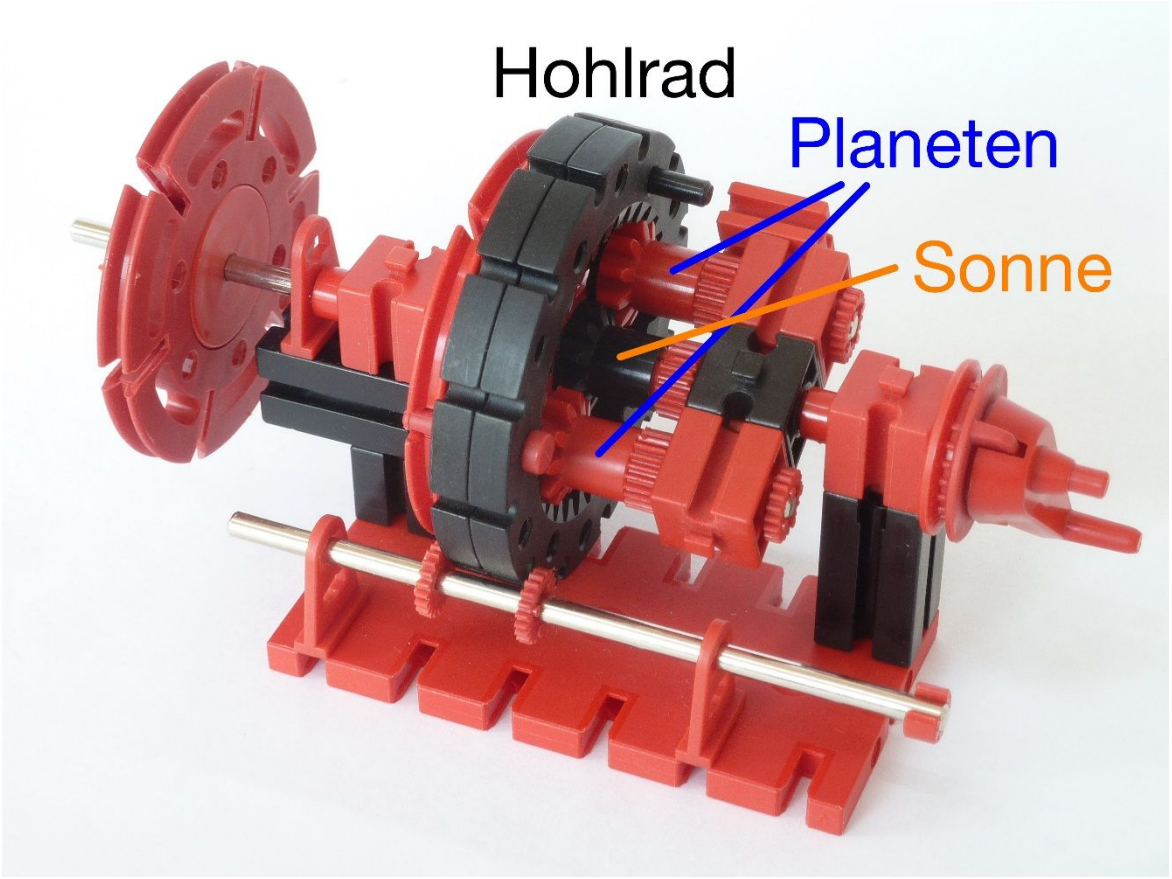


Abb. 10: Zwei-Gang-Schaltgetriebe

Zwei-Gang-Schaltgetriebe

Das letzte Modell dieses Beitrags ist ein Zwei-Gang-Schaltgetriebe mit dem üblichen fischertechnik-Hohlrad-Planetengetriebe, siehe Abbildung 10. Man kann an ihm die Vorgänge in einer Fahrradnabenschaltung grundsätzlich verstehen. Für die praktische Verwendung in einer Nabe eignet sich unser Getriebe allerdings nicht. Dort ist das Schalten durch das Ausziehen des zentralen Nabenelements günstig.

In unserem Modell werden die Innenzahnräder mit der exzentrischen Schaltstange vor dem Getriebe nach rechts oder links bewegt. Im normalen Gang (Innenzahnräder rechts) liegt eine Übersetzung von 1:1 vor, im schnellen Gang (Innenzahnräder links) eine Übersetzung von 1:4.

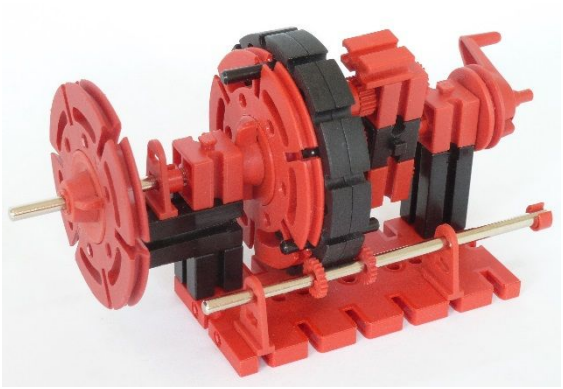


Abb. 11: Zweite Ansicht des Schaltgetriebes

Die Innenzahnräder sind durch eine Rastachse 20 und zwei Kunststoffachsen 30 miteinander und mit der Drehscheibe 60 verbunden. Die Kunststoffachsen 30 fungieren als Mitnehmer. Wie weit sie in welcher Richtung aus dem Verbund aus Drehscheibe und Innenzahnrädern herausragen, ist wesentlich und muss genau eingestellt werden.

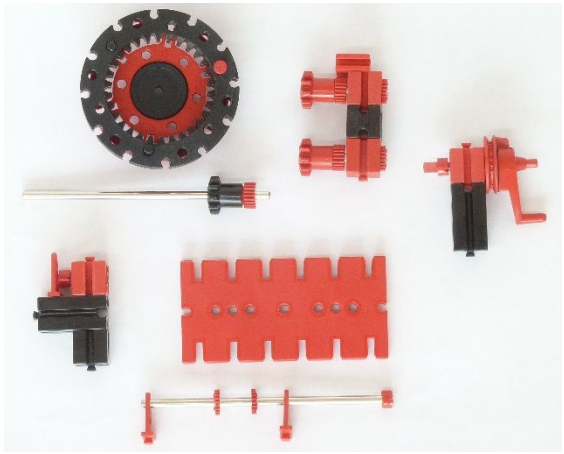


Abb. 12: Komponenten des Schaltgetriebes

Im normalen Gang sind die Innenzahnräder rechts. Sonnenrad und Planetenräder liegen an der Drehscheibe an. Einer der Mitnehmer legt sich unter Last an den Steg an, siehe Abbildung 13. Das ganze Getriebe bewegt sich dadurch starr, und somit liegt eine Übersetzung von 1:1 von der Kurbel auf die Abgangswelle vor.

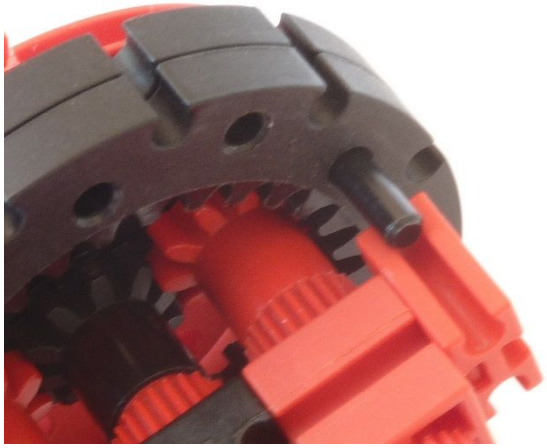


Abb. 13: Mitnahme zwischen Steg und Innenzahnrädern im normalen Gang

Im schnellen Gang sind die Innenzahnräder links. Der andere Mitnehmer schlägt an den linken unteren schwarzen Baustein 15 an und hält dadurch die Innenzahnräder fest, siehe Abbildung 14. Jetzt ergibt sich eine Übersetzung von 1:4 von der Kurbel auf die Abgangswelle.

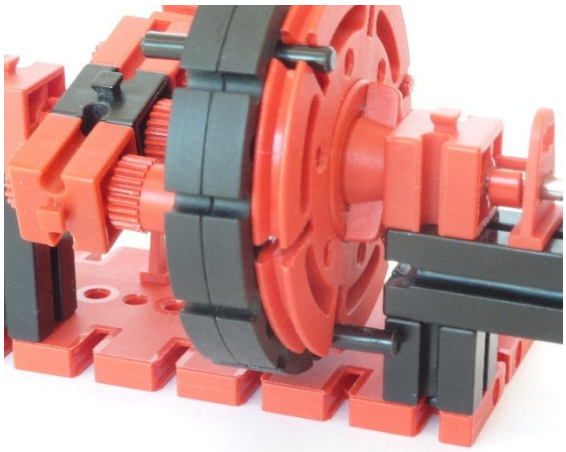


Abb. 14: Im schnellen Gang hält der Mitnehmer die Bewegung der Innenzahnräder auf

Wie die Übersetzung von 1:4 zustande kommt, kann man Abbilddung 15 entnehmen. Hält man Kurbel und Steg fest, so ergibt sich vom Sonnenrad zu den Innenzahnrädern eine Übersetzung von 3:-1. Dreht man nun das ganze Getriebe starr eine Umdrehung im gleichen Sinn weiter, so hat die Kurbel insgesamt eine Umdrehung gemacht, die Innenzahnräder keine und die Abgangswelle vier. Somit liegt bei festen Innenzahnrädern eine Übersetzung von 1:4 zwischen Kurbel und Abgangswelle vor.

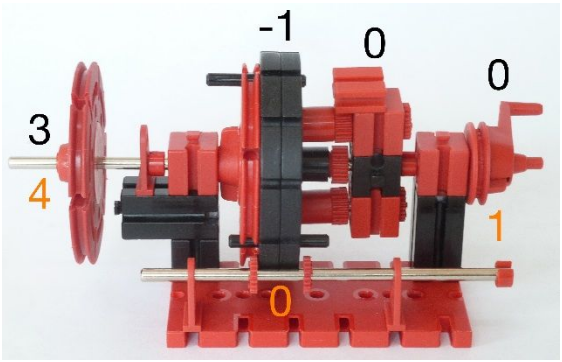


Abb. 15: Erklärung der 1:4-Übersetzung im schnellen Gang

Literatur und Links

[1] David a Sancto Cajetano: *Neues Rädergebäude*, Joseph Edlen von Kurzbeck, Wien, 1791, abrufbar in der NTK Digital Library Prag.

[2] Eric Robinson, A. E. Musson: *James Watt and the Steam Revolution*. London, 1969.

[3] Dirk Fox und Thomas Püttmann: *Technikgeschichte mit fischertechnik*, dpunkt.verlag, Heidelberg, 2015.

[4] Robert Willis: *Principles of Mechanisms*, John W. Parker, London, 1841, abrufbar unter http://archive.org.

[5] William Reilly und Charlton Haigh: *Two-Speed-Driving-Gear for Bicyles*, US- Patent 588657.

Elektromechanik

Anwendungen für Magneten (2): Rotationstransformator

Andreas Gail

Zu einer Zeit, zu der man bewegte Videobilder noch auf Tonbandkassetten aufzeichnete, waren Rotationstransformatoren in privaten Haushalten sehr gebräuchlich: Sie waren in den Kopftrommeln von Videorecordern verbaut. Auch wenn genau diese Anwendung heute keine Bedeutung mehr hat, haben Rotationstransformatoren durchaus das Potential, für spätere Anwendungen genau die passende Problemlösung zu werden.

Die Kopftrommel eines Videorecorders

Bei den damaligen Videorecordern wurde das Magnetband an einer schnell rotierenden Trommel, der Kopftrommel, vorbeigeführt. Damit erfolgte die Aufzeichnung des bewegten Videobildes auf das Magnetband. Genauso wurde auch das Auslesen des bereits beschriebenen Bandes durchgeführt. Die Übertragung der Videosignale von der Elektronik des Videorecorders zur schnell rotierenden Kopftrommel bzw. umgekehrt wurde dabei von Rotationstransformatoren übernommen [1].

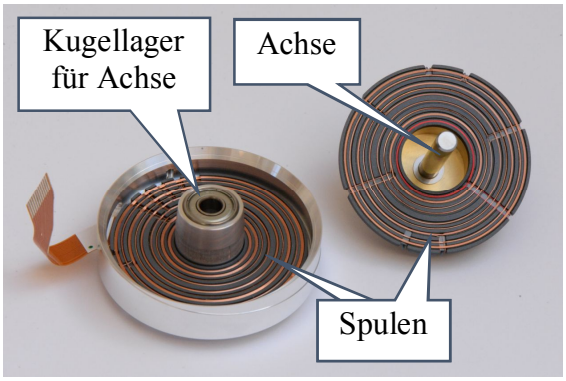


Abb. 1: Rotationstransformator der Kopftrommel eines Videorecorders: Links die fest stehende Spule (Stator), rechts die rotierende (Rotor)

Abb. 1 zeigt einen Rotationstransformator in auseinandergenommener Form. Wird die Achse in die Wellendurchführung des Kugellagers eingesteckt, befinden sich Rotor und Stator genau übereinander, d. h. die Spulen des Rotors liegen genau über den Spulen des Stators. In Abb. 2 sind die einzelnen Spulen gut erkennbar. Insgesamt verfügt der abgebildete Rotationstransformator über neun Spulenpaare, die übereinander rotieren können. Für eine gute Signalübertragung ist es dabei wichtig, dass die Spulen beider Seiten einen minimalen Abstand zueinander haben.

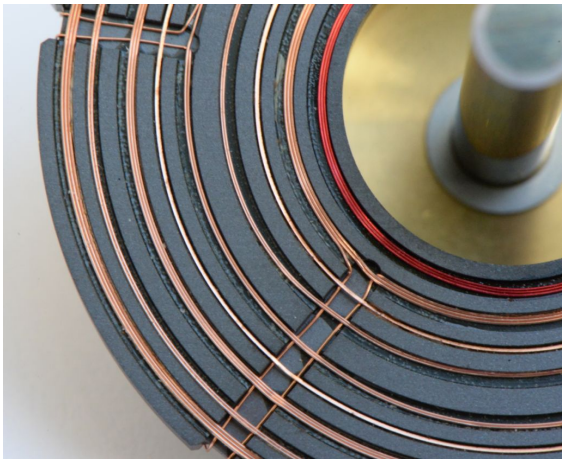


Abb. 2: Einzelne Spulen eines Rotationstransformators auf der Seite des Rotors

Ein erster Prinziptest

Zu den klassischen fischertechnik Bauteilen gehörten Elektromagnete:

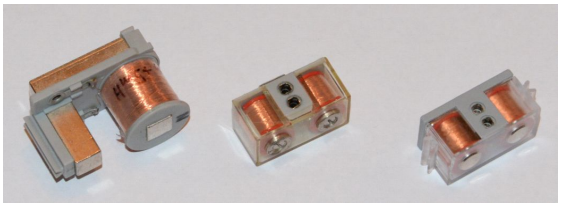


Abb. 3: Elektromagnete von fischertechnik

Mit diesen Elektromagneten können wir den ersten Prinziptest machen. Der Aufbau ist ähnlich der bereits früher gezeigten Induktionssensor-Messstrecke [2]. Der Unterschied ist, dass diesmal der induzierte Strom nicht gemessen, sondern mithilfe einer LED direkt gezeigt wird. Mit diesem Aufbau kann man selbst die obige Behauptung überprüfen, dass der Abstand der beiden Magneten möglichst gering sein sollte, wenn man maximalen Energieübergang erreichen möchte.

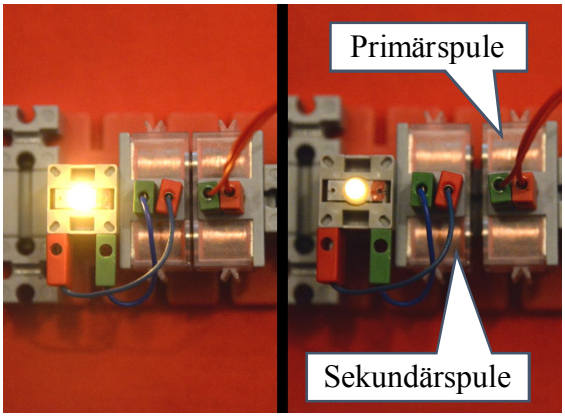


Abb. 4: Die LED-Helligkeit ist umso größer, je kleiner der Abstand der Elektromagnete zueinander ist. Links: Große Helligkeit bei geringem Abstand, rechts geringere Helligkeit bei größerem Abstand

Als Wechselstromquelle wird ein alter Transformator oder der Robo(tics) TX(T) Controller verwendet [2]. Dieser Versuch funktioniert nur mit Wechselspannung. Bei den beiden baugleichen Magneten wird eine gleiche Anzahl an Spulenwindungen unterstellt. Deshalb entspricht die angelegte Wechselspannung auf der Primärspule auch der Spannung, die auf der Sekundärspule induziert wird. Dieser Aufbau entspricht einem Trenntransformator [3].

Herstellung eines passenden Magnetpaares

Die Elektromagneten von Abb. 3 sind zum Aufbau eines Rotationstransformators ungeeignet. Doch auch hier lässt sich fischertechnik einfach erweitern, wie die folgende Abbildung zeigt:

Abb. 5: Elektromagnete zur Nutzung mit fischertechnik

Verbaut wurden Magneten mit 20 N Haltekraft, 12 V DC, 2,5 W (Conrad Art.-Nr. 502290). Bei der Verklebung ist die exakte Positionierung wichtig, damit später alles rund läuft. Das erreicht dieser Hilfsaufbau:

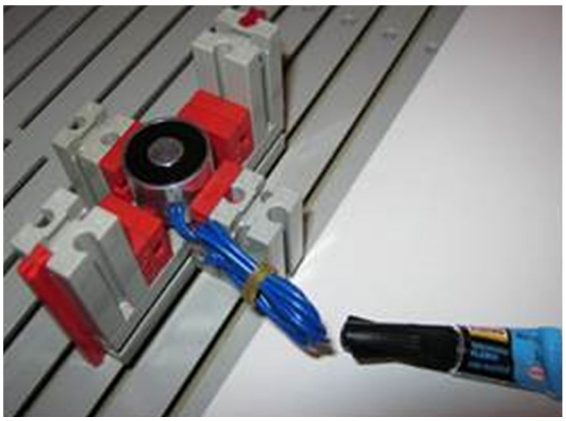


Abb. 6: Positionierung bei der Verklebung

Aufbau des Rotations-transformators

Der Aufbau geschieht ähnlich Abb. 4, allerdings mit den Magneten der Abb. 5. Wenn die beiden Magneten zueinander drehbar gelagert sind, ist der entscheidende Teil fertig:

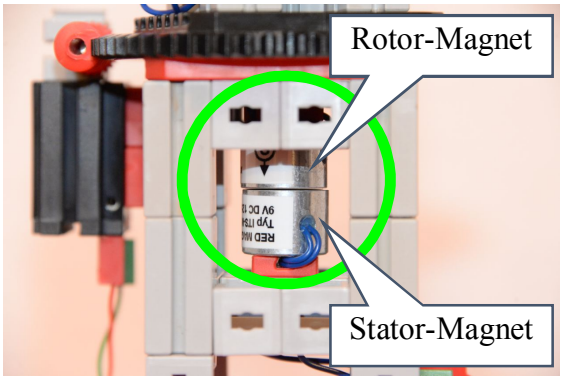


Abb. 7: Der eigentliche Rotationstransformator ist grün markiert

Die demontierte Detailansicht des Rotationstransformators ist in Abb. 8 und 9 zu sehen. Hier sind die Magneten konzentrisch aufgebaut. Entsprechend des Zustands der Wechselspannung wird mittig der Nordpol des Magneten entstehen und am Rand der Südpol oder eben umgekehrt (im steten Wechsel). In Abb. 8 sind die Pole blau und gelb dargestellt.

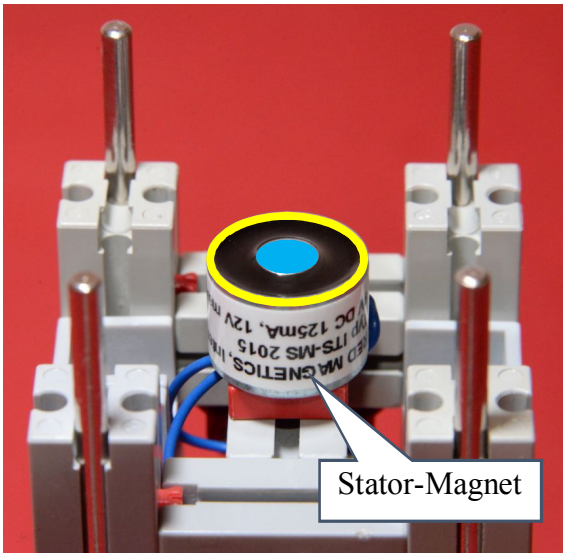


Abb. 8: Stator des Rotationstransformators

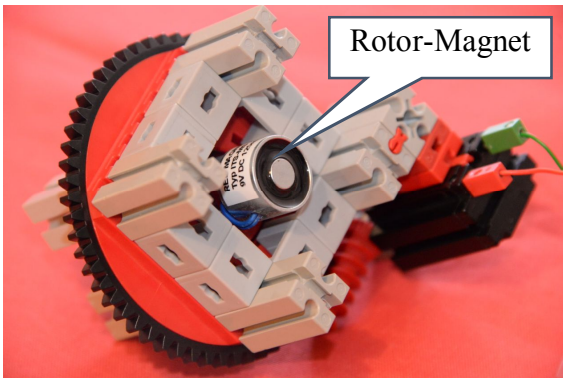


Abb. 9: Rotor des Rotationstransformators

Der Motor dreht die obere Plattform mit dem Rotor-Magneten und der LED. Nochmal erwähnt sei, dass der untere Magnet (der Stator) mit Wechselspannung versorgt wird – sonst kann der Transformator nicht wirken. Übrigens ist die Energieübertragung umso besser, je höher die Frequenz der Wechselspannung ist.

Quellen

[1] Wikipedia: *Rotationstransformator*.

[2] Gail, Andreas: *Anwendungen für Magneten (1): Induktionssensor*. ft:pedia 2/2015, S. 40-43.

[3] Wikipedia: *Trenntransformator*.

[4] Gail, Andreas: *fischertechnik: Rotationstransformator*. YouTube-Video, 2016.

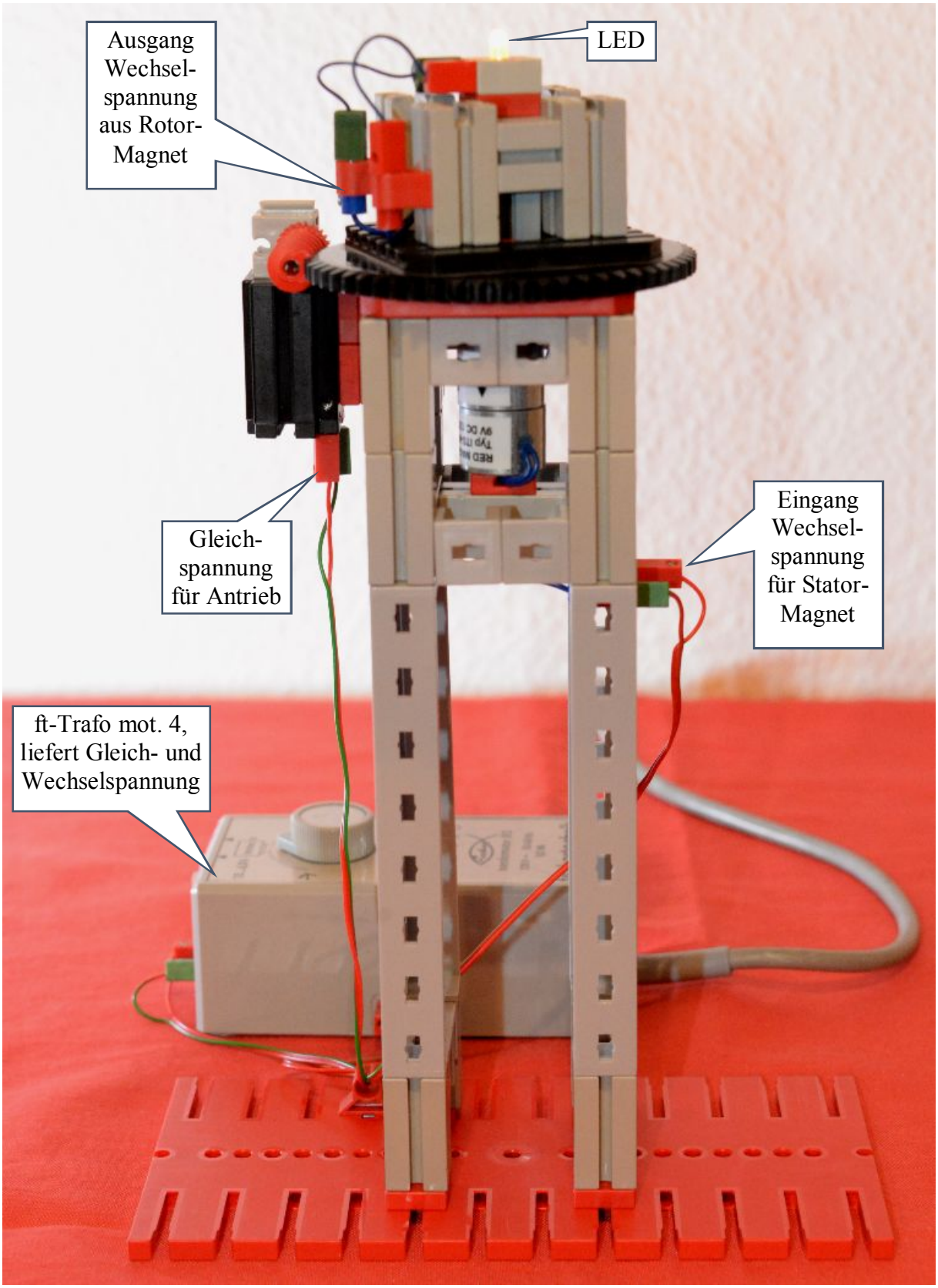


Abb. 10: Gesamtaufbau mit Nutzung eines Rotationstransformators

Elektromechanik

Synchronmotoren

Matthias Dettmer

Habt ihr schon mal eine Zeitreise gemacht? Also ich schon, mindestens zwei Mal in diesem Jahr. Nachdem ich mir Ende letzten Jahres das Buch zur „Technikgeschichte mit fischertechnik“ [1] gekauft hatte, war zuerst die Turmuhr aus Kapitel 4 (ab Seite 51) dran. Einen ersten Versuch eine Pendeluhr zu bauen, hatte ich vor etwa 43 Jahren unternommen, mit damals eher bescheidenem Erfolg. Und dann ist da das Kapitel 10 zum Elektromotor. Die beiden dort beschriebenen Synchronmotoren, also Motoren die „nur“ mit der Netzfrequenz von 50 Hertz betrieben werden, haben mich dann um 26 Jahre zurückgeschickt.

Anfang der 90er Jahre hatte ich beruflich mit dem Aufbau und der Programmierung von Prüfständen für elektrischen Maschinen zu tun, und ich dachte, dass man einen Synchronmotor eigentlich auch mit fischertechnik nachbauen kann. Heraus kam, hauptsächlich inspiriert von der Drehscheibe 60 (31019) mit ihrer 6-er Teilung, ein Motor mit drei Polpaaren. Der war von zwei Elektromagneten „umschlossen“ und von meiner Lötstation (Achtung! Empfindliche Geister bitte jetzt nicht weiterlesen, sondern erst weiter unten) mit 12 V Wechselstrom bestromt. Aber das Ding lief nach dem Anwerfen satt mit den erwarteten 1000 Umdrehungen pro Minute, nachgewiesen über eine Lichtschranke und ein Oszilloskop. Die Spulen haben die 12 V übrigens überlebt, wenn sich auch ein wenig Kondenswasser an den Innenseiten gebildet hatte. Leider habe ich damals keine Fotos gemacht, das war mangels Digitalkameras auch nicht so einfach wie heute.

Gleich nach dieser Zeitreise ging ich daran, auch noch andere Drehzahlen mit Synchronläufern umzusetzen. Bahnbrechend dazu war die Verwendung von Neodym-Stabmagneten mit 10 mm Länge und 4 mm Durchmesser wie im Kapitel 10 (Seite 218, Abb. 10-15, und Fußnote) gezeigt.

Neben dem leicht veränderten Modell aus dem Buch (Polpaarzahl sechs, resultierend in 500 U/min, ganz rechts) zeigt mein Modell jetzt auch die Polpaarzahlen drei (wieder 1000 U/min, ganz links) und fünf mit 600 U/min (in der Mitte des Bildes).

Inzwischen etwas „gesitteter“ was die Belastung der Bauteile anbelangt: Hier kommt für die Stromversorgung aller Synchronläufer wieder ein altes Netzgerät mit seinen 6,8 V zum Einsatz.

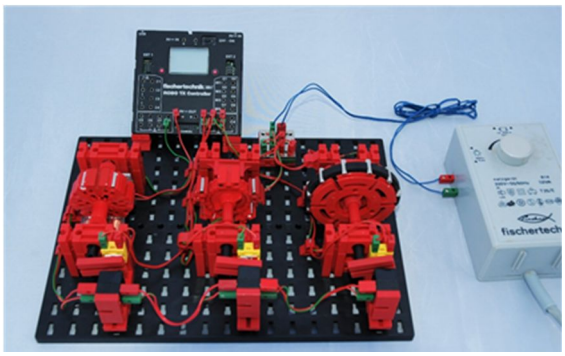


Abb. 1: Synchronläufer mit Polpaarzahlen von 3, 5 und 6

Beim Aufbau der drei Motoren wollte ich den Grundrahmen möglichst immer gleich, aber etwas flexibler als beim „Urvater“ aus dem Artikel gestalten. Dort sind nämlich die Spulen in einem konstruktionsbedingt festen Abstand angebracht, und ich wollte mehr Einstellmöglichkeiten haben.

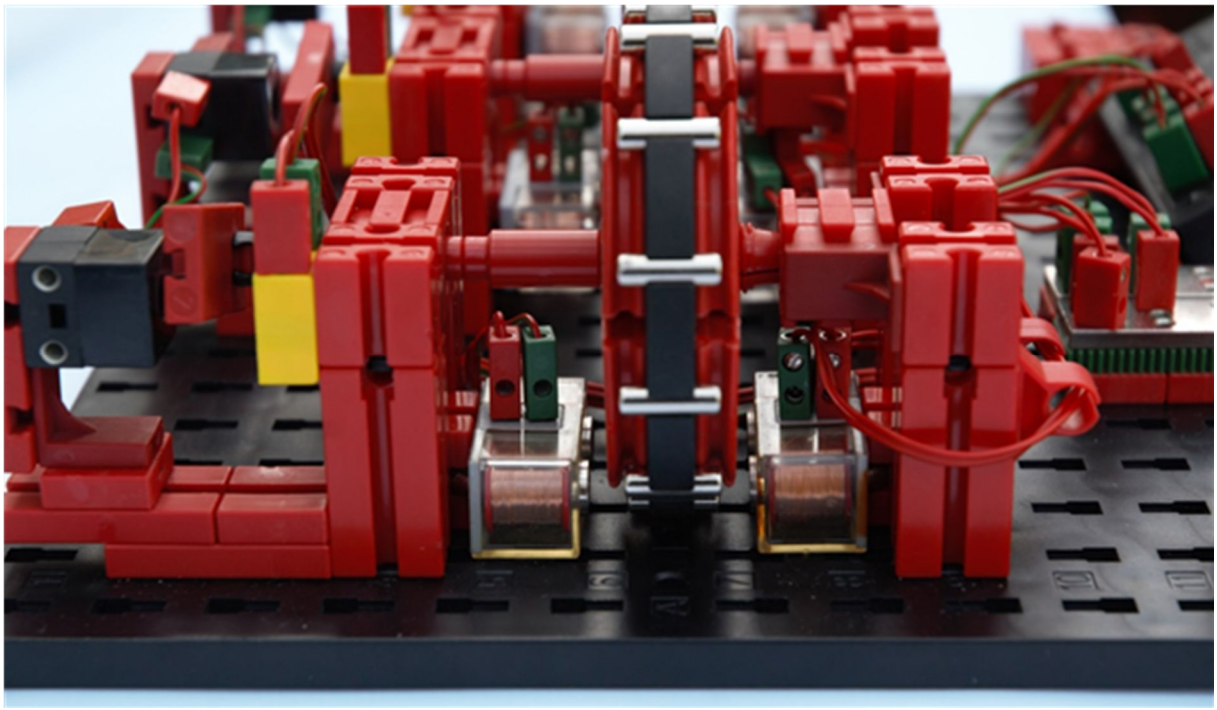


Abb. 2: Die Spulen sind innerhalb einiger Millimeter gut positionierbar, und auch die Lagerung der Hauptwelle erlaubt verschiedene Ansätze zum Befestigen des Läufers auf der Achse.

Die Spulen sollten in der Entfernung zum Läufer einstellbar sein, und die Lagerung des Läufers sollte auch andere „Kraftkoppelung“ an der Achse erlauben als mit den Naben sonst üblich.

Erste Hürde bei anderen Polpaarzahlen: Wie stelle ich sicher, dass die Pole in etwa gleichen Abständen auf dem Umfang liegen? Und zweitens: wenn ich mit Winkelsteinen gleiche Abstände hinbekomme: Wie befestige ich das dann auf der Achse?

Hürde eins war mittels Winkelsteinen schnell übersprungen. Es ist wirklich faszinierend, wie viele Möglichkeiten es hier gibt. Nahezu alles „durch drei teilbare“ lässt sich so umsetzen. Die einfachste Methode bieten die Winkelsteine 60°, wahrscheinlich ginge Ähnliches auch wieder mit der Drehscheibe. Hürde zwei war schon schwieriger, konnte aber mit zwei gegeneinander auf der Achse verspannten Kegelzahnrädern umgesetzt werden.

Das führte dann aber zu Hürde Nummer drei: Wie hält man die Neodym-Magneten sinnvoll in den Nuten der Bausteine fest?

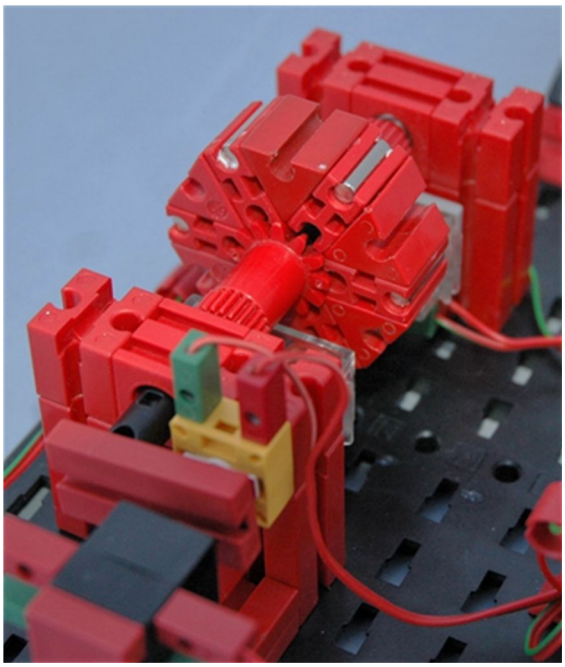


Abb. 3: Die Befestigung des Läufers über zwei Kegelzahnräder sieht wackelig aus, ist aber absolut symmetrisch.

Folgt mein erstes Geständnis: Ich habe geschummelt. Ein Fetzchen Papier liefert die nötige Spannkraft, und wenn man ein wenig fummelt, dann kommt das Fetzchen so unter dem Magnetstab zu liegen, dass man es nicht sieht. Das gelingt aber nicht immer, siehe Abb. 3, ist aber immer noch besser – weil lösbar – als mit Cyanacrylat.

Der Begriff „Polpaarzahl“ hilft bei elektrischen Maschinen, eine Idee von der zu erwartenden Drehzahl zu bekommen. Synchronmotoren funktionieren nur dann, wenn die Anzahl der Pole auf dem Durchmesser auch gerade ist, es also „ganzzahlige“ Polpaarzahlen gibt.

Der verwendete Wechselstrom hat 50 Hertz (Dimension: 1/s). Will man die Umdrehungen statt „pro Sekunde“ in „pro Minute“ haben, dann muss man die 50 1/s mit 60 multiplizieren, das ergibt dann 3000 1/min. Dividiert man diese durch die Polpaarzahl, dann erhält man die zu erwartende Drehzahl des Motors:

$$Drehzahl\ \left[\frac{1}{min}\right] = \frac{Netzfrequenz\ \left[\frac{1}{s}\right] \cdot 60\ [\frac{s}{min}]}{Polpaarzahl}$$

Damit ergeben sich die in Tab. 1 zusammengestellten möglichen Drehzahlen.

Als meine „anderen“ Synchronmotoren prinzipiell liefen, habe ich eine weitere Idee aus dem Buch umgesetzt und alle Achsen in Kugellagern (anders als im Buch mit 9 mm Durchmesser, die passen in die Schneckenlager für die Rastachsen-Schnecken) gelagert.

Pole	Polpaare	Drehzahl [U/min]	Besonderheit
2	1	3000	Fast nicht umzusetzen
4	2	1500	Ist mir einmal gelungen, lief aber nicht dauerhaft
6	3	1000	Befestigung auf der Achse mit Kegelzahnrädern
8	4	750	Im Modell nicht umgesetzt
10	5	600	Das sind 10 Umdrehungen pro Sekunde → Uhr!
12	6	500	… wie im Buch
38	19	157,89	Siehe Text
40	20	150	... auch wie im Buch

Tab. 1: Drehzahlen der Synchronmotoren

Dann ist mir ein kleiner technischer Kniff aufgefallen. Im Uhrenmodell in Kapitel 10 (ab S. 219) sind nur 38 Magneten im Einsatz, zwei Magneten fallen wegen der beiden Mittelspeichen (S. 221 Abb. 10-18) weg. Mit einer Polpaarzahl von 19 würde die Frequenz um einige Hertz „nicht stimmen“ – 150 U/min (entsprechend 2,5 Hertz) gibt es nur bei einer Polpaarzahl von genau 20. Da mag jemand fragen: Und warum läuft es dann trotzdem? Ganz einfach: Weil insgesamt zwei Elektromagneten im Einsatz sind, die um 60° = ⅓ · 180° (zwei fehlende Magneten) versetzt sind. Wenn also eine der beiden „Fehlstellen“ an einer der Spulen vorbeikommt,

dann übernimmt die jeweils andere das „Weiterschubsen“. Es gibt also 38 Magneten plus zwei „Fehlstellen“, und das ergibt eine Polpaarzahl von 20. Na bitte.

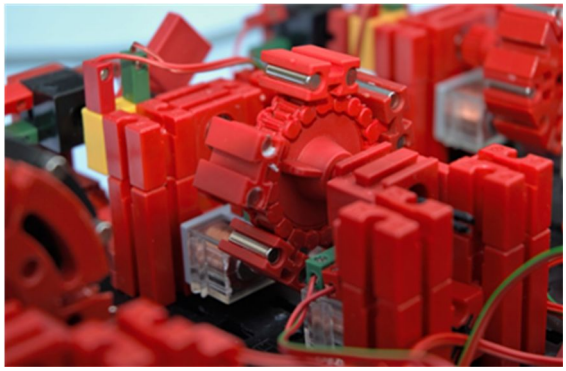


Abb. 4: Jeweils drei „normale“ und ein „Förderglied“ 37192 multipliziert mit fünf passen genau auf ein Zahnrad mit 20 Zähnen.

Star meiner Sammlung ist für mich der Motor mit der Polpaarzahl fünf. Mit seinen 600 Umdrehungen pro Minute dreht er sich also zehn Mal pro Sekunde. Auch hier sind wir wieder beim Thema Uhr, wobei ich noch nicht ausprobiert habe, ob sich mit fischertechnik-Zahnrädern vernünftige Teilungen herstellen lassen und man das verwerten kann.

Es ist schon ein enorm zufriedenstellendes Geräusch, wenn alle drei Motoren vor sich hin schnurren. Ein letzter Akt bleibt noch: Stimmt denn das mit den Drehzahlen? Der Nachweis sollte über Lichtschranken und den TX-Baustein erbracht werden.

Sicher hätte man die Drehzahlmessung auch „diskret“ über kleine Zählprogramme umsetzen können. Da ich nicht weiß, wie oft pro Sekunde ich den TX „messen lassen“ kann, mussten die für die Encoder-Motoren gedachten Zählereingänge herhalten.

Also: Motor und damit Zähler zurücksetzen (drei Mal für die drei Drehzahlen), dann Motor mit mittlerer Geschwindigkeit rechts starten, sechs Sekunden warten, Zähler auslesen und anzeigen, drei Sekunden warten, Anzahl Messungen Zähler hochzählen und das Ganze von vorne. Auf die Multiplikation mit zehn (wegen der Minute) habe ich verzichtet.

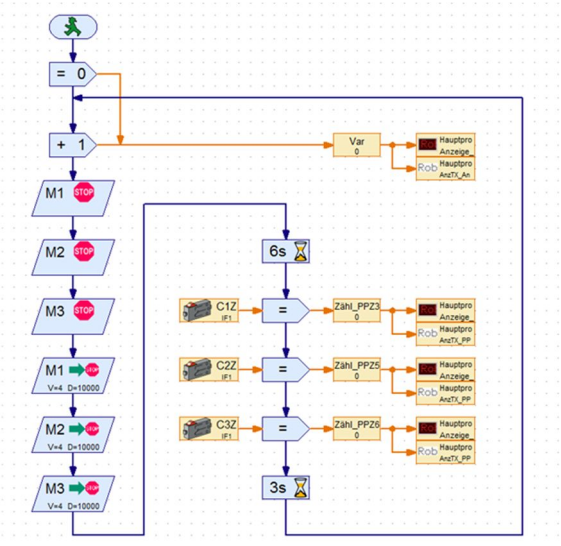


Abb. 5: Schade eigentlich, dass die Zählereingänge sich nur über die Encoder-Motoren ansprechen lassen. M1 bis M3 sind hier nämlich gar nicht wirklich angeschlossen – einen Zählereingang ohne Encoder Motor zu benutzen heißt, auf einen Motorausgang zu verzichten.

Das funktionierte zuerst allerdings gar nicht. Irgendwie wurde für die schnelleren Motoren immer eine höhere – genau doppelt so hohe – Drehzahl angezeigt.

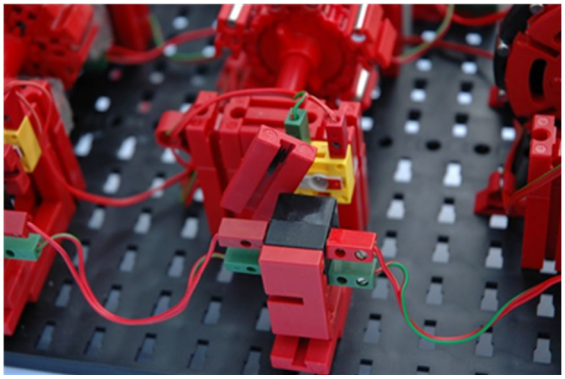


Abb. 6: Schnell mal Nachmessen? Lichtschranke und Unterbrecher sind so einfach wie möglich gehalten.

Und schon wieder muss ich eine kleine Schummelei beichten: Ursache für die „doppelte“ Anzeige war die Nut des „Unterbrechers“ der Lichtschranke. Mit einem

schwarzen Colli-Marker-Stift wurde das behelfsmäßig geändert (siehe Abb. 6).

Folgt noch der Ausblick auf weitere Modellvariationen und eine kleine Manöverkritik. Im Moment sind jeweils nur zwei Elektromagneten „im Einsatz“. Würde man weitere Spulen in der den Polpaarzahlen entsprechenden Teilungen anbringen, dann würde das jeweilige erzielbare Drehmoment deutlich ansteigen.

Als Erweiterungen sind etwa eine „Anlaufhilfe“ oder eine berührungslose, elektronische Steuerung mit einer Art Frequenzumrichter denkbar – sicher werde ich den TX damit mal an seine Grenzen treiben. Die oben beschriebene Drehzahlformel würde dann durchaus auch höhere Drehzahlen und eine Anlaufhilfe (mit niedriger Frequenz beginnen und dann steigern) ermöglichen.

Die Polpaarzahlen zwei (1500 U/min) und eins (3000 U/min) sind da schon kniffliger. Ohne Kugellager und ein Konzept zum Anwerfen, etwa durch Seilzug, wird das vermutlich nicht gehen.

Der Aufbau der drei Varianten hat eine Menge Spaß gemacht, jede Variante (auch die aus dem Buch!) hat mindestens drei Konstruktionsstufen durchlaufen bis es wirklich „rund“ lief. Manchmal hilft es auch, nochmal völlig neu mit einer leeren Platte zu beginnen.

Alle Modelle sind am Ende direkt vom Beispiel aus dem Buch abgeleitet. Mit einer anderen als dieser Positionierung (seitlich vom Rotor und unten) kommt man auch zum Ziel, das ist aber umständlicher und braucht mehr Platz.

Das Zählprogramm zählt manchmal „einen Puls zu wenig“. Insbesondere, wenn die „Beruhigungszeit“ von drei Sekunden weggelassen wird. Würde man dabei noch mit zehn multiplizieren, dann würde man den Fehler ebenfalls verzehnfachen.

Referenzen

[1] Dirk Fox, Thomas Püttmann: *Technikgeschichte mit fischertechnik*. dpunkt Verlag, 2015.

Computing

TXT Controller – Tipps & Tricks (2): Screenshots

Raphael Jacob

So genannte „Screenshots“ (Bilder vom Bildschirminhalt) helfen, Fehler zu dokumentieren oder Vorgehensweisen zu veranschaulichen – z. B. im fischertechnik community forum. In diesem Beitrag erkläre ich, wie ihr Screenshots vom TXT erstellt und anschließend in ein ‚handelsübliches‘ Format konvertiert.

Screenshot erstellen

Vorbedingung

Um Screenshots zu erstellen braucht ihr eine SSH- und SFTP-Verbindung zum TXT. Wie man diese einrichtet, kann man in [1] nachlesen.

Die Arbeit am TXT

Am TXT geht es recht schnell. Man sollte sich bereits als „ROBOPro“ mit dem Passwort „ROBOPro“ eingeloggt haben („root“ ist nicht erforderlich, funktioniert aber ebenfalls). Jetzt muss nur ein Kommando ausgeführt werden:

```
cp /dev/fb0 /
opt/knobloch/screenshot.raw
```

Damit ist die Arbeit am TXT bereits abgeschlossen und man kann zum nächsten Schritt übergehen.

Dateiübertragung

Jetzt muss die Datei `screenshot.raw` zum Computer übertragen werden. Dazu verwende ich die Software Filezilla [1]. Auch hier ist es wiederum egal, mit welchem der beiden Accounts man sich einloggt. Links navigiert man zu dem Ordner, in dem man später die raw-Datei abspeichern möchte. Dann verbindet man sich mit dem TXT, indem man auf den kleinen Pfeil am Icon des Servermanagers drückt und dann den TXT auswählt. Jetzt navigiert man rechts zum Ordner `/opt/knobloch` und lädt die Datei mit einem Doppelkick herunter.

Bildkonvertierung

Die Bildkonvertierung kann sowohl unter Linux als auch unter Windows erfolgen.

Konvertierung unter Linux

Um diese raw-Datei in ein nutzbares Format zu bringen, benötigt man die Software `avconv` (enthalten in `libav-tools`). Zum Installieren gebe ich im Ubuntu-Terminal folgendes Kommando ein:

```
sudo apt-get install libav-tools
```

Jetzt kopiere ich die raw-Datei mittels USB-Stick auf den Ubuntu-Computer und verschiebe sie in einen Ordner nach Wahl. Um mit dem Terminal dorthin zu navigieren, gibt man folgendes Kommando ein (wobei „…“ durch Ordnernamen zu ersetzen ist):

```
cd /…/…/…/
```

Um die Datei nun zu konvertieren, gibt man folgendes Kommando ein:

```
avconv -vcodec rawvideo -f rawvideo
-pix_fmt rgb565 -s 240x320 -i
[Dateiname der Ursprungsdatei] -f
image2 -vcodec png [Dateiname der
Ausgabedatei(PNG)]
```

In unserem Fall:

```
avconv -vcodec rawvideo -f rawvideo
-pix_fmt rgb565 -s 240x320 -i
screenshot.raw -f image2 -vcodec
png screenshot.png
```

Jetzt hat man eine png-Datei, die man nun beispielsweise mit einem USB-Stick auf den Windows-PC kopieren kann.

Konvertierung unter Windows

Unter Windows kann man sich des bekannten Programms IrfanView (Download: http://www.irfanview.com/; nicht die 64-Bit-Version) bedienen. Die Konvertierung ist denkbar einfach. In IrfanView muss man oben links auf

File > Open As > Raw File

drücken und dann die raw-Datei öffnen. Im nächsten Dialog müssen einige Einstellungen getroffen werden (siehe Abb. 3). Nach dem Festlegen der Eigenschaften und einem Klick auf „OK“ kann man den Screenshot bereits sehen. Um das Bild jetzt auch als PNG nutzen zu können muss man unter

File > Save As

einen Speicherort angeben und folgende Einstellungen rechts wählen (Abb. 1):

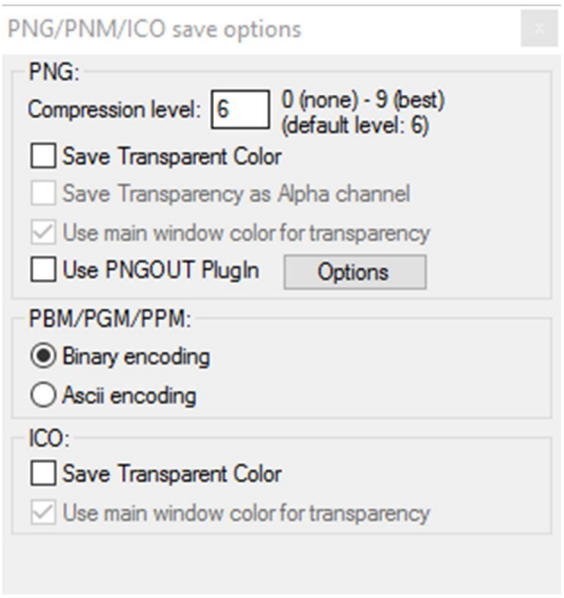


Abb. 1: Die Einstellungen für den Export

Quellen

[1] Raphael Jacob: *TXT Controller – Tipps und Tricks (1): Das Root-Passwort*. ft:pedia 1/2016, S. 65-68.

[2] MasterOfGizmo: *TXT-Tricks #2: Screenshots*. ft:c fischertechnik community forum, 31.05.2015.

Abb. 2: Das Ubuntu Terminal:
1.Ordnerinhalt vor dem Konvertieren,
2. Kommando zum Konvertieren (Ausgabe weggeschnitten),
3. Ordnerinhalt nach dem Konvertieren

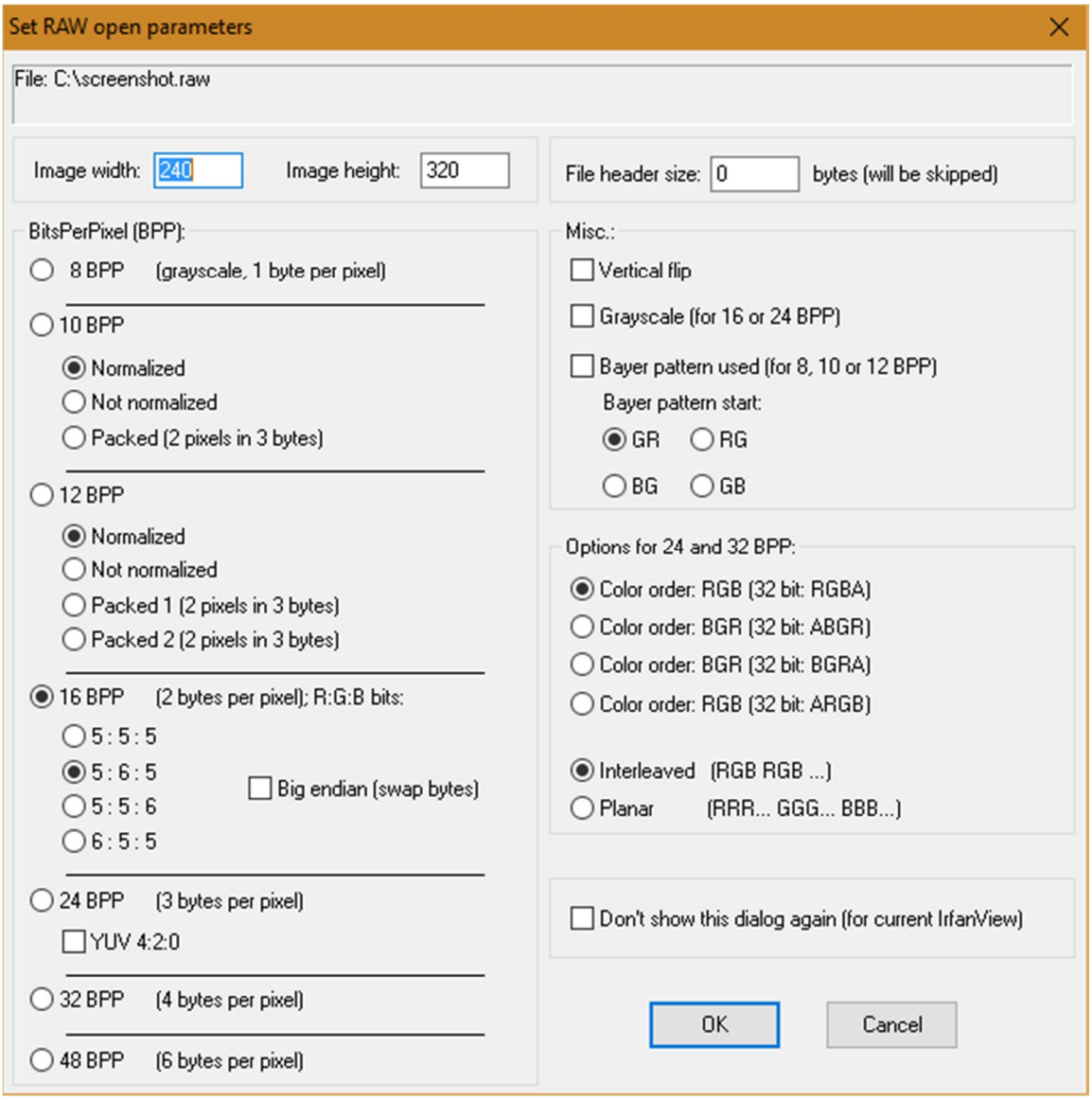


Abb. 3: Der Dialog zum Festlegen der Eigenschaften der raw-Datei

Computing

Alternative Controller (1): Der Arduino

David Holtz

In der Reihe „Alternative Controller“ werden wir einige Projekte vorstellen, die zeigen, dass und wie man fischertechnik-Modelle mit unterschiedlichen Microcontrollern (an)steuern kann. In diesem ersten Beitrag wird die Arduino-Plattform eingeführt, die ja schon Gegenstand früherer Beiträge war, und ein Vergleich mit dem TXT-Controller unternommen.

Hintergrund

Im Jahr 2005 entwickelten Studenten aus Italien mit ihrem Dozenten Massimo Banzi das erste Arduino-Board. David Mellis schrieb die passende Software dazu, mit der man das Board in C/C++ programmieren kann und die unter die Creative-Commons-Lizenz gestellt wurde [1, 2].

Von der ersten kleinen Auflage des Boards waren 50 Stück für eine Schule bestimmt. In den Folgejahren wurden mehr und mehr Boards entwickelt und verkauft [1]. So eroberte der Arduino als einfaches und günstiges Mikrocontroller-Board, das zusammen mit einer IDE (engl. *integrateted development environment*) ausgeliefert wurde, Klassenzimmer und die Herzen von Bastlern und Quereinsteigern weltweit.

Seit dem Rechtsstreit um die Marke Arduino im Jahr 2015 werden die Boards von zwei offiziellen Stellen entwickelt und vertrieben [3]: Die Arduino LLC [4] vertreibt die Produkte unter dem Namen Arduino in den USA und unter dem Namen Genuino im restlichen Teil der Welt. Smart Projects S.R.L [5] besitzt die Namensrechte an der Marke Arduino in einigen europäischen Ländern, darunter Italien und Deutschland.

Mittlerweile gibt es sehr viele unterschiedliche Arduino-Boards. Einige sind mit Funktionen wie Ethernet, WLAN, einem SD-Kartenslot oder Ähnlichem erweitert worden.

Hardware

Prozessor und Speicher

Der Arduino UNO R3 ist der am weitesten verbreitete Arduino. Er basiert wie die meisten Arduino-Boards auf einem Atmel AVR Mikrocontroller. Dieser wird über einen 16-MHz-Schwingungsquarz getaktet. Für Programme stehen 32 kB Speicher zur Verfügung; außerdem verfügt er über 2 kB RAM [6].

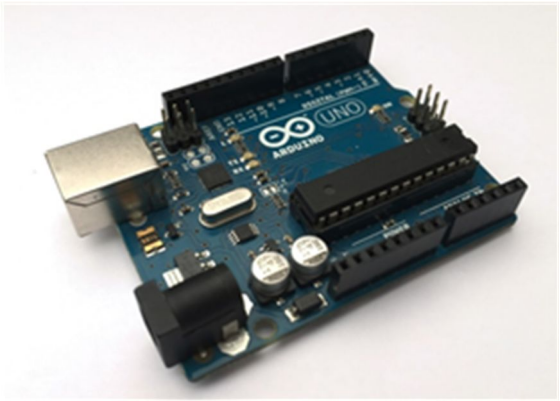


Abb. 1: Arduino Uno R3

Im Vergleich zum TXT-Controller erscheinen die Taktrate und der Speicher winzig: Der TXT-Controller verfügt über den ARM Cortex A8 Prozessor mit 600 MHz Taktrate, er hat 64 MB Flashspeicher und 128 MB RAM.

In der Praxis reichen die Ressourcen des Arduino jedoch vollkommen aus. Für anspruchsvollere Aufgabe gibt es Boards mit mehr Speicher (z. B. den Arduino Mega) oder mit mehr Rechenleistung (z. B. den Arduino Due). Dennoch kommen alle diese Alternativen nicht an die Rechenleistung des TXT-Controllers heran, weshalb rechenlastige Aufgaben ausgelagert werden sollten.

Während der TXT-Controller zur Objekterkennung mittels Kamera in der Lage ist, sollte am Arduino eine Kamera mit eigenem Bildverarbeitungsprozessor verwendet werden, wie z. B. die Pixy Cam [7].

Verbindung zum PC

Des Weiteren zeichnen sich Arduino-Boards durch ihre USB-Schnittstelle aus, über die sie komfortabel programmiert werden können. Aufgrund des vorprogrammierten Boot-Loaders ist kein externes Programmiergerät zum Beschreiben des Chips notwendig. Serielle Kommunikation zwischen dem PC und dem Arduino (beispielsweise zur Übertragung von Messwerten) ist über ein USB-Kabel möglich.

Verschaltung von Aktoren und Sensoren

Kennzeichnend für alle Arduino-Boards ist die mehr oder weniger große Zahl an Pins. Der zentrale Unterschied zum TXT besteht darin, dass alle Pins sowohl als Eingang als auch als Ausgang genutzt werden können. Ein als INPUT konfigurierter Pin kann Sensorwerte durch Bestimmen des anliegenden Potentials einlesen; ein Pin, der als OUTPUT definiert ist, kann Aktoren steuern.

Einige der Pins verfügen über spezielle Eigenschaften:

- Sechs Pins können das anliegende elektrische Potential mit 10 Bit-Auflösung (analog) messen.
- Bei allen Pins, die als INPUT definiert werden, kann ein interner 10-kΩ-Pullup-Widerstand aktiviert werden.
- Sechs der Pins unterstützen, als OUTPUT initialisiert, die Pulsweitenmodulation (PWM), mit der eine analoge Spannung zwischen 0 und 5 Volt durch schnelles An- und Ausschalten moduliert wird (markiert durch eine Tilde: ~).
- Einige der Pins lassen sich als externer Interrupt nutzen.
- Einige der Pins ermöglichen die Kommunikation über I²C, SPI und UART.

Die obige Aufzählung trifft auf Arduino-Boards mit einem Atmel AVR Mikrocontroller zu. Bei anderen Boards (Arduino Zero/Due) können die Funktionen abweichen.

Bei der Verschaltung sollte unbedingt darauf geachtet werden, dass die Spannung an den Pins je nach Board 3,3 oder 5 V beträgt. Da die Stromstärke je Pin beim Arduino Uno 20 mA (in der Spitze 40 mA) nicht überschreiten darf, ist die Ansteuerung von fischertechnik-Hardware wie Motoren nicht ohne Weiteres möglich.[1] Zum Betreiben von fischertechnik-Motoren benötigt man daher eine Half-Bridge, entweder in Form einer separaten Schaltung oder eines Motor Shields.

Für Aktoren, die nur die Zustände ‚an' und ‚aus' einnehmen können (Magnetventile, Glühlämpchen), sollte eine Verstärkerschaltung verwendet werden, um die einzelnen Pins nicht zu überlasten und den Mikrocontroller dadurch zu zerstören.

[1] Zur Veranschaulichung: Durch ein fischertechnik-Glühlämpchen fließen etwa 100 mA Strom, bei Powermotoren bis zu 1 A. LEDs mit einer Stromaufnahme von rund 20 mA können hingegen direkt angeschlossen werden.

Zusätzliche Schaltungen bedeuten bei der Umsetzung eines Projekts mehr Aufwand. Dies ist beim TXT-Controller nicht notwendig, da er bereits Single- und Half-Bridges zur Aktoransteuerung besitzt. Wie dies beim TX-Controller funktioniert, hat Stefan Brunner in der ft:pedia beschrieben [8].

Der Anschluss von fischertechnik-Sensoren ist vergleichsweise simpel. Es muss lediglich darauf geachtet werden, dass der Sensor mit der Board-Spannung betrieben wird (in der Regel sind das 5 V). Man kann den Arduino als vielseitige Schnittstelle zu verschiedenen Sensoren oder Aktoren nutzen: Da die Protokolle SPI (*Serial Peripheral Interface*) und TWI (*Two Wire Interface* bzw. I²C) unterstützt werden, steht der Ansteuerung von Kameras wie z. B. der Pixy Cam [7] oder ähnlicher Hardware nichts im Wege. Die in der Reihe „I²C mit dem TX“ in der ft:pedia vorgestellten Sensoren und Aktoren können auch mit den Arduino-Boards gesteuert werden. In dieser Hinsicht ist der Arduino dem TXT-Controller überlegen, da er flexibler genutzt werden kann.[2]

Welcher Arduino passt zu mir?

Die Wahl des Arduino-Boards richtet sich nach dem Einsatzgebiet. Der Arduino Uno ist das klassische Board zum Steuern eines Roboters mit zwei Motoren und ein paar Sensoren. Kauft man eines der Originale und unterstützt die Organisation (*Made in Italy*), liegt man preislich bei maximal 30 €. Ein Upgrade auf den Arduino Mega mit deutlich mehr Pins und Speicher (ca. 40 €) lohnt sich, wenn man den Arduino als wirkliche Alternative zum TXT-Controller verwenden möchte. Für Basteleien gibt es kleinere Boards, die preislich etwas unter dem Uno liegen. Über Shields kann man Ethernet, WLAN, einen SD Kartenslot und vieles mehr nachrüsten.

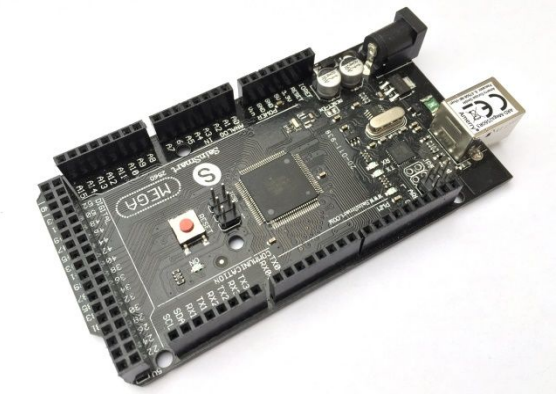


Abb. 2: Arduino Nachbau SainSmart Mega 2560

Da alle Boards *Open Source* sind, gibt es zahlreiche günstigere Nachbauten (vgl. Abb. 2), deren Kosten sich im einstelligen bis niedrigen zweistelligen Euro-Bereich bewegen.

Software

Integrierte Entwicklungsumgebung

Die Arduino-Plattform bietet eine quelloffene C/C++-Entwicklungsumgebung, mit der man ein Programm (im Arduino Jargon „Sketch“ genannt) schreiben, kompilieren und auf das Board laden kann. Sie steht für Linux, Windows und MacOS zur Verfügung. Die Robo Pro-Software für den TXT-Controller, eine graphische Entwicklungsumgebung, ergänzt um den Interfacetest und die Möglichkeit des Online-Betriebs, gibt es nur für Windows-Systeme.

Compiler

Als Compiler wird der avr-gcc Compiler aus der *GNU Compiler Collection* verwendet. Dieser bindet vor dem Übersetzen des Programmcodes in eine Hex-Datei automatisch einige Arduino-Standard-Bibliotheken ein, die dem Programmierer beispielsweise das Abfragen oder Schalten

[2] Allerdings bleibt abzuwarten, wie sich der TXT-Controller mit der Community-Firmware weiterentwickelt.

einzelner Pins sowie die Bildschirmausgabe am Computer erleichtern [9].

Programmstruktur

Jeder Sketch besteht aus dem *void setup*-Teil und der *void loop*. Der Programmabschnitt *setup* wird unmittelbar nach dem Starten des Boards einmal ausgeführt, der Abschnitt *loop* im Anschluss daran als Endlosschleife.

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Abb. 3: Einfacher Arduino-Sketch

Der oben abgebildete Sketch lässt eine LED an Pin 13 blinken. Zunächst wird Pin 13 mit dem Befehl *pinMode* als OUTPUT definiert. In *void loop* wird Pin 13 über den Befehl *digitalWrite* auf 5 V (HIGH) und auf 0 V (LOW) gesetzt, wodurch die LED an- und ausgeschaltet wird.

Besonderheiten

Im Gegensatz zum TXT-Controller wird bei den Arduino-Boards mit Atmel AVR Mikrocontroller kein Multithreading unterstützt. Das gleichzeitige Ausführen zweier Prozesse wird dadurch erschwert; trotzdem können beispielsweise mehrere Bewegungen Interrupt gesteuert parallel ausgeführt werden.

Arduino-Boards haben einen sehr präzisen Timer, der im einstelligen µs-Bereich auflöst. Die Auflösung des Timers des deutlich leistungsfähigeren TXT-Controllers wird hingegen durch Robo Pro auf den ms-Bereich begrenzt.

Ausblick

Der TXT-Controller ist dank seiner vollen Kompatibilität zu fischertechnik die Plug-and-Play-Lösung, mit der man, egal ob Einsteiger oder Fortgeschrittener, komfortabel programmieren kann.

Arduino-Boards können eine kostengünstige und flexible Alternative sein, wenn man einige Hürden beim Verbinden mit der fischertechnik-Hardware überwunden hat. Dieser Aufwand ist je nach Projekt und Vorkenntnissen überschaubar bis groß.

In einigen Beiträgen in der ft:pedia wurden bereits verschiedene Möglichkeiten vorgestellt, wie man mit Arduino-Boards fischertechnik-Modelle steuern kann. Das werden wir in weiteren Beiträgen fortsetzen.

Referenzen und Links

[1] Wikipedia: *Arduino*.

[2] Alexander Weber: *Shake, rattle 'n' roll*. c't 16/2009, S. 164 ff.

[3] Peter König: *Arduino vs. Arduino – der Graben wird tiefer*. heise.de Make:, 30.07.2015.

[4] arduino.cc: *Arduino.cc*.

[5] arduino.org: *Arduino.org*.

[6] arduino.cc: *Arduino Uno*.

[7] Dirk Wölffel, Dirk Fox: *I²C mit dem TX – Teil 11: Pixy-Kamera*. ft:pedia 4/2014, S. 43-51.

[8] Stefan Brunner: *Die Ein- und Ausgänge des TX Controllers*. ft:pedia 4/2012, S. 24-31.

[9] arduino.cc: *BuildProcess*.

Computing

Alternative Controller (2): Infrarot-Empfänger

David Holtz

Dieser Beitrag der Reihe „Alternative Controller“ stellt eine Selbstbaulösung für einen alternativen Infrarot-Empfänger für fischertechnik vor und erklärt, wie die Kommunikation zwischen dem Handsender und dem Empfänger zustande kommt.

Zielsetzung

Mein jüngstes Projekt, ein dreiachsiger Mobilkran, dessen Achsen sich unabhängig voneinander lenken lassen sollen, hat eine Steuerung erfordert, die man mit dem Empfänger aus dem Control Set nicht umsetzen konnte. Eine Selbstbaulösung für einen alternativen Infrarotempfänger sollte Abhilfe schaffen. Folgende Anforderungen wurden daran gestellt:

- Komfortable Möglichkeit zur Fernsteuerung des Modells
- Mehrere unabhängige Servoausgänge
- Einfacher Anschluss von Aktoren (z. B. für Motoren oder Beleuchtung)
- Flexible Nutzung des Empfängers auch für andere Modelle

Ich habe mich entschlossen, die oben formulierten Anforderungen mit einem Arduino UNO als Mikrocontroller umzusetzen. Die in [12] geschilderten Hürden hinsichtlich der Kompatibilität von fischertechnik und der Arduino-Plattform sollten durch ein eigens dafür entwickeltes Shield überwunden werden.

Funktionsprinzip einer Infrarotfernsteuerung

Das Signal einer Infrarotfernbedienung besteht aus Lichtimpulsen aus dem für das menschliche Auge nicht sichtbaren, infraroten Teil des Spektrums, typischerweise mit einer Wellenlänge von 870 bis 950 nm. Einige Digitalkameras ohne Infrarotlichtfilter stellen dieses Licht in Falschfarben dar, wie die folgende Abbildung zeigt:

Abb. 1: Infrarot-Fernsteuerung

Durch zeitlich versetztes Senden der Impulse können Daten übertragen werden. Im Handsender wird dazu, meistens über einen integrierten Schaltkreis (IC), die Position der Joysticks bzw. ein Tastendruck in einen Binärcode umgewandelt. Ähnlich wie beim Morsen wird dieser Binärcode durch ein- und ausschalten einer Infrarot-LED übertragen (vgl. den Abschnitt *Dekodierung* weiter unten in diesem Beitrag).

Modulation

Zusätzlich zum zeitlich versetzten Senden werden die Impulse einer Signalfolge, die eine logische 1 darstellen, moduliert. Modulation bedeutet, dass die Infrarot-LED mit einer Frequenz von 30 bis 56 kHz an- und ausgeschaltet wird, um ein Signal mit

der logischen 1 zu erzeugen. Die Frequenz wird über einen passenden Oszillator generiert [1].

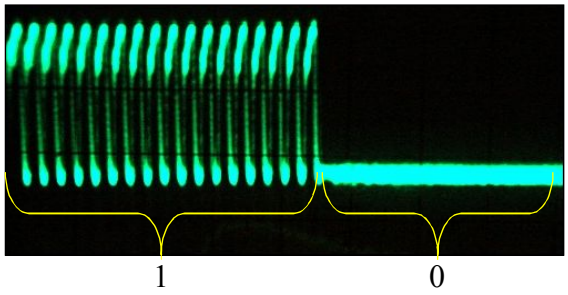


Abb. 2: Oszilloskop-Bild einer modulierten Signalfolge 1-0 (Quelle: Wikipedia [1])

Die Modulation bewirkt, dass das Infrarotsignal weniger anfällig auf Störungen durch Umweltfaktoren wie Sonnenlicht oder anderer Wärmestrahlung reagiert, da sich die kurzen Impulse stärker von der Umgebungsstrahlung abheben. Eine häufige Störquelle sind 50 Hertz Netzschwingungen, die beispielsweise von Energiesparlampen erzeugt werden. Ein Hochpassfilter im Empfänger verringert die Störwirkung niedriger Frequenzen und lässt nur die hohen Frequenzen des modulierten Signals passieren.

Die Länge eines Signals variiert je nach verwendetem Sender und Protokoll, bewegt sich aber in einer Größenordnung von maximal 2500 µs. Die Länge der Signalfolge hängt von der übertragenen Datenmenge und der Länge eines Signals ab. Nach jeder Signalfolge folgt eine einige Millisekunden lange Pause, sodass auch andere Fernbedienungen senden können. Typischerweise können komplette Signalfolgen etwa zehnmal pro Sekunde übertragen werden.

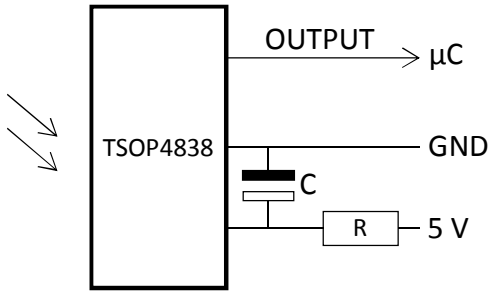
IR-Signale empfangen

Damit ein Infrarotsignal empfangen werden kann, muss Sichtkontakt zwischen Sender und Empfänger bestehen. Es muss aber nicht unbedingt direkter Sichtkontakt sein, da die emittierte Infrarotstrahlung von unterschiedlichen Oberflächen reflektiert wird. Besonders gut funktioniert das an metallischen Oberflächen.

Hardware

Das Empfangen eines Signals gelingt am einfachsten, wenn man ein spezielles Empfängerbauteil verwendet. Dessen Fotodiode reagiert weniger empfindlich auf Störungen durch Umwelteinflüsse, da sichtbares Licht durch einen Sperrfilter herausgefiltert wird. Zusätzlich übernimmt eine integrierte Schaltung die Demodulation der Trägerfrequenz und liefert ein digitales Ausgangssignal, das direkt von einem Mikrocontroller verarbeitet werden kann.

In diesem Beitrag wird das „IR Receiver Module TSOP4838“ [7] für 38 kHz-Signale verwendet. Grundsätzlich lässt sich der Empfänger direkt an einen 5 V-Mikrocontroller anschließen, allerdings empfiehlt das Datenblatt [8] einen optionalen 100 Ω-Widerstand und einen 0,1 µF-Kondensator zur besseren Abschirmung gegen Überlastungen.



R = 100 Ω, C = 0,1 µF

Abb. 3: Schaltplan für TSOP4838

Um das Infrarotsignal auswerten zu können, sollte der Mikrocontroller einen präzisen Timer mit einer Auflösung im µs-Bereich besitzen. Der Arduino UNO erfüllt diese Voraussetzung; grundsätzlich eignen sich alle Arduino Boards und vergleichbare Mikrocontroller.

Ein erstes Programm

Mittels eines einfachen Arduino-Programms (Sketch) soll gezeigt werden, wie

man eine Infrarotsignalfolge sichtbar machen kann.

Dazu programmiert man eine Unterbrechungsroutine (engl. *interrupt service routine*, kurz ISR). Diese wird immer dann aufgerufen, wenn sich der Zustand an dem Pin ändert, an dem der TSOP4838 angeschlossen ist (Modus „change“, d. h. bei fallender oder steigender Flanke).

Die Arduino-Entwicklungsumgebung bietet darüber hinaus eine benutzerfreundliche Funktion, um die Zeit zu messen:

```
uint_32t zeit = micros();
```

Die Funktion *micros()* weist der Variable `zeit` diejenige Zeit in Mikrosekunden zu, die seit dem Starten des Arduinos vergangen ist.

Die ISR könnte so programmiert werden, dass bei jedem Aufruf die Differenz aus der neuen und der alten Zeit berechnet und gespeichert wird, also die Zeit, die zwischen einer fallenden und einer steigenden Flanke (bzw. umgekehrt) vergeht. Die gespeicherten Zeiten kann man sich am seriellen Monitor ausgeben lassen und zur Veranschaulichung in ein x/y-Diagramm übertragen (vgl. Abb. 5).

```
12 // ISR wird bei jedem Flankenwechsel aufgerufen
13 ISR (PCINT0_vect)
14 {
15   // Es werden 100 Flankenwechsel in einem Array
16   // gespeichert, danach erfolgt die Ausgabe auf
17   // dem seriellen Monitor.
18
19   if (currentPulse < 100)
20   {
21     newTime = micros();
22     pulses[currentPulse] = newTime - oldTime;
23     oldTime = newTime;
24     currentPulse ++;
25   }
26   else
27   {
28     // Freigabe für die Ausgabe am Bildschirm
29     printPulses = 1;
30   }
31 }
```

Abb. 4: Interrupt Service Routine

Dekodierung

Zur Dekodierung des empfangenen Signals ist das dazugehörige Protokoll erforderlich. Bis heute sind die für Fernbedienungen verwendeten Protokolle offiziell nicht standardisiert worden, allerdings hat das NEC-Protokoll bei den heutigen TV-Geräten den größten Marktanteil. Im Folgenden werden drei gängige bzw. für fischertechnik relevante Protokolle vorgestellt.

RC-5-Protokoll

Das RC-5-Protokoll wurde Ende der 1980er Jahre von der Firma Philips zur Steuerung von Alltagselektronik wie TV- oder Audiogeräten entwickelt und war lange Zeit in Europa sowie in den USA das am häufigsten zum Einsatz kommende Protokoll. Anfangs bot es 64 Befehle, durch ein Field-Bit wurde der Umfang auf

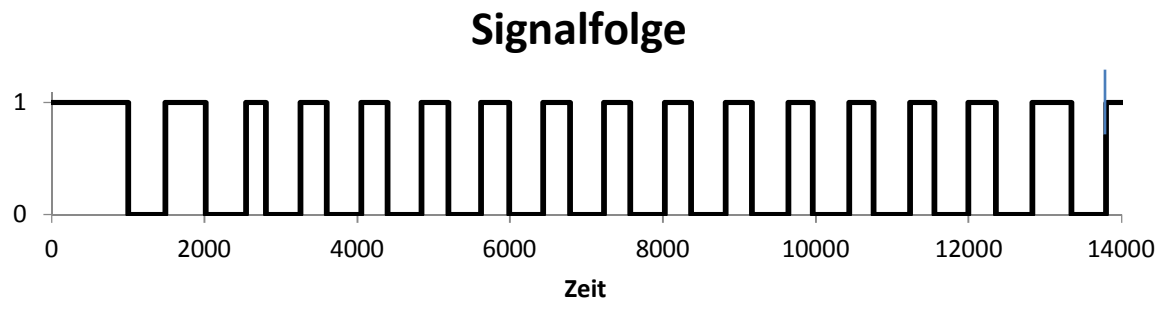


Abb. 5: Ausschnitt aus einer Signalfolge

zweimal 64 Befehle erweitert. Mit dem Nachfolger RC-6 umfasst das Protokoll 256 Befehle. Beim Senden werden keine Informationen zum angesprochenen Gerät übertragen (nur RC-5), sodass eine beliebige RC-5-Fernbedienung ein beliebiges RC-5-Gerät steuern kann [2].

Manchester-Code

Das RC-5-Protokoll nutzt den Manchester-Code zur Datenübertragung, der an der University of Manchester entwickelt wurde. Die Besonderheit des Manchester-Codes liegt darin, dass er selbst getaktet ist, was es wiederum ermöglicht, eine Nachricht drahtlos oder über eine galvanische Trennung (induktiv) zu übertragen. Die eigentliche Bitfolge, die die Nachricht darstellt (Data), wird mit der getakteten Verbindung (Clock) über eine „exklusive Oder“ Verknüpfung (XOR) zum Manchester-Code kodiert (vgl. Abb. 6).

Um den Manchester-Code zu dekodieren, analysiert man die Flankenwechsel zur Mitte der Periode (in der Abbildung zwischen den grauen vertikalen Markierungen). Eine steigende Flanke bedeutet eine logische 1, eine fallende Flanke stellt die logische 0 dar. Da der Manchester-Code die Taktrate enthält, ist er selbstsynchronisierend. Das Taktsignal kann umgekehrt wieder aus dem Code abgeleitet werden [3].

Das RC-5-Protokoll ist aufgrund des Manchester-Codes daran erkennbar, dass eine Pulslänge und eine genauso lange Pausenlänge sowie jeweils die doppelten Puls- und Pausenlängen auftreten können. Möchte man einen Empfänger für verschiedene Protokolle programmieren, so kann über diese Eigenschaft das RC-5-Protokoll erkannt werden.

NEC-Protokoll

Heutzutage ist das NEC-Protokoll das am häufigsten eingesetzte Infrarotprotokoll. Im Gegensatz zum RC-5-Protokoll übermittelt der Handsender einen 8-Bit-Adresscode, damit der Empfänger prüfen kann, ob das Signal für ihn bestimmt war. Dies sorgt dafür, dass nicht versehentlich ein anderes Gerät gesteuert wird (und verhindert, dass man in Elektronikmärkten alle Geräte über eine einzige Fernbedienung ausschalten kann).

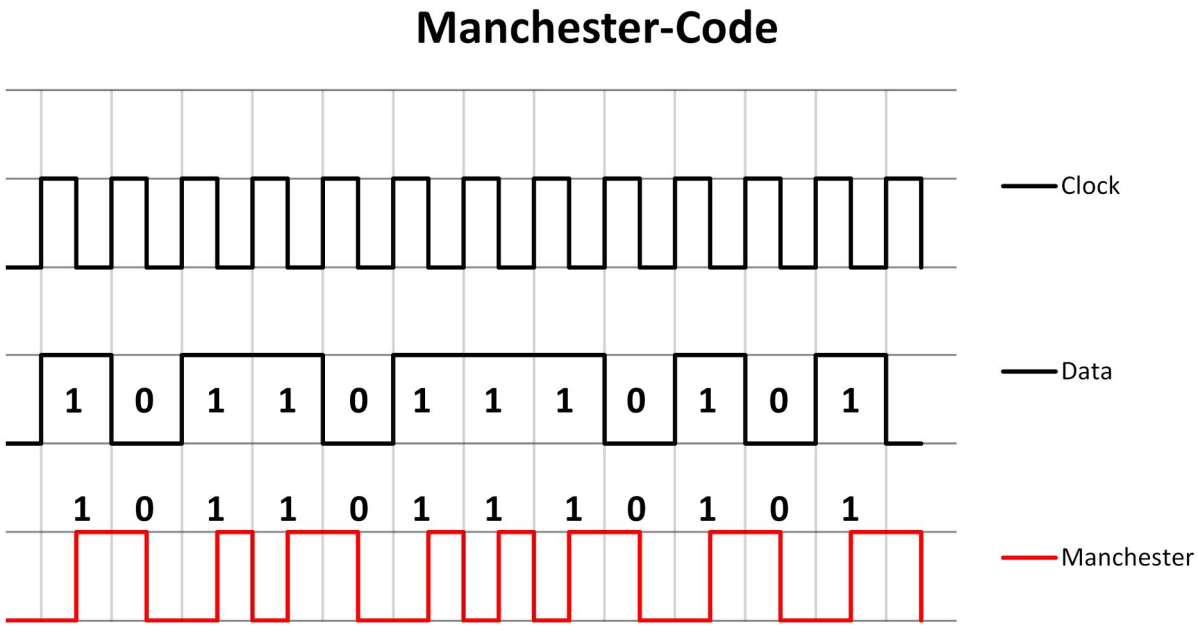


Abb. 6: Veranschaulichung: Manchester-Code

Die acht Bit lange Nachricht kann 256 verschiedene Befehle übertragen, zusätzlich existiert ein „extended mode“, der die Länge der Nachricht verdoppelt und dadurch rund 65000 verschiedene Befehle bietet.

Ein weiterer Unterschied zum RC-5-Protokoll besteht in der Kodierung: Das NEC-Protokoll nutzt Pulsdistanzkodierung, um eine logische 0 beziehungsweise eine logische 1 zu übertragen. Eine „Distanz“ von 1,125 ms zwischen zwei aufsteigenden Flanken stellt die logische 0 dar, liegen zwischen zwei aufsteigenden Flanken 2.25 ms auseinander, handelt es sich um eine logische 1. Da ein Puls mit 560 µs stets dieselbe Länge hat, bestimmt die Pausenlänge zwischen den Pulsen die Pulsdistanz [4].

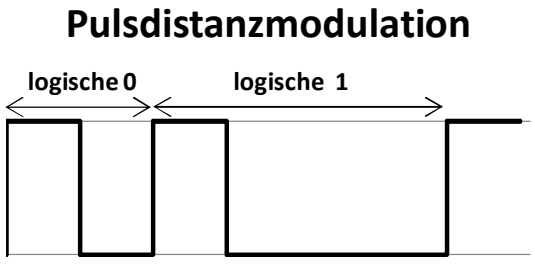


Abb. 7 Pulsdistanzmodulation beim NEC Protokoll

In [9] findet man eine Arduino-Library zur Dekodierung des NEC-Protokolls.

RC-MM Protokoll

Das RC-MM-Protokoll der Firma Philips ist im Gegensatz zu den beiden erstgenannten Protokollen eher unbekannt. Es ist an dieser Stelle aber aus dem Grund erwähnenswert, da der Handsender aus dem fischertechnik Control Set ein vom Prinzip her ähnliches Protokoll nutzt.

Wie beim NEC-Protokoll wird die Nachricht über Pulsdistanzkodierung übertragen. Dieses Verfahren wird jedoch so erweitert, dass jede Distanz zwischen zwei aufsteigenden Flanken eine zweistellige Bitfolge (00, 01, 10 oder 10) darstellt. Hierfür gibt es eine jeweils konstante Pulslänge und vier verschiedene Pausenlängen, sodass insgesamt vier unterscheidbare Abstände zwischen zwei aufsteigenden Flanken auftreten können [5].

Praktischerweise wurde das im fischertechnik Control Set eingesetzte Protokoll bereits über „reverse engineering“ analysiert. Der genaue Aufbau der Nachricht kann unter [10] nachgelesen werden.

IR Protokolle: Fazit

Je nachdem, welches Protokoll man implementiert hat, kann man verschiedene Fernbedienungen zum Steuern der Modelle verwenden. Man sollte sich bei der Wahl der Fernbedienung nach den Anforderungen an die Steuerung richten: TV-Fernbedienungen bieten mit ihrem Nummernfeld und den zahlreichen Funktionstasten die Möglichkeit, viele unterschiedliche Befehle zu übermitteln. Der Handsender aus dem Control Set eignet sich dank seiner Joysticks besonders für die Steuerung von Fahrzeugen. Dank der drei Infrarot-LEDs ist er außerdem deutlich lichtstärker und besser für den Außeneinsatz geeignet, als es TV-Fernbedienungen sind. Zudem sollte man beachten, dass die RC-5- und NEC-Protokolle jeder Taste einen Binärcode zuordnen, anstatt ein Bit für jede Taste zu reservieren. Dies bedeutet, dass es nicht möglich ist, mehrere gleichzeitig gedrückte Tasten zu erfassen.

Shield-Bau

Wie in [12] beschrieben, lassen sich fischertechnik-Aktoren häufig nicht direkt mit dem Arduino verbinden. Die dafür benötigte Schaltung kann man auf einer Experimentierplatine aufbauen. Das ist bei komplexeren Schaltung jedoch schwierig und unübersichtlich.

Für die Umsetzung des IR-Empfängers wurden alle notwendigen elektrischen Bauteile auf einer für den Arduino UNO abgestimmten Platine [11] untergebracht,

die sich einfach auf das Board stecken lässt. In der Fachsprache bezeichnet man diese Platinen als *Shields*.

Verschaltung

Um die Kompatibilität zu fischertechnik herzustellen, muss an den Ausgängen des Shields eine Spannung von 9 V anliegen. An den Pins des Arduino UNO liegen aber nur 5 V an.

Da der Arduino einen integrierten Festspannungsregler besitzt, kann man ihn direkt mit der fischertechnik-Spannung von 9 V betreiben. Dazu verbindet man die Plus-Leitung mit dem VIN-Pin und die Minus-Leitung mit einem der Masse-Pins.

Für die Motoren benötigt man eine *half bridge*; im Selbstbauempfänger ist hierfür der L293B verbaut. Es werden drei Pins je Motor benötigt, um die Richtung des Motors und die Geschwindigkeit (über Pulsweitenmodulation, PWM) zu steuern.

Grundsätzlich ist für die Ansteuerung von fischertechnik-Servomotoren am Arduino keine zusätzliche Hardware erforderlich. Man verbindet die Masseleitung mit der Masse des Arduinos und die Signalleitung mit einem I/O-Pin des Boards. Das Empfänger-Shield nutzt sicherheitshalber statt der 5 V Board-Spannung des Arduinos die über einen externen Festspannungsregler auf 5 V herabregulierte Versorgungsspannung (9 V von ft). Dadurch soll vermieden werden, dass der Festspannungsregler auf dem Arduino-Board durch eine hohe Stromaufnahme beim Betrieb mehrerer Servos überlastet und möglicherweise beschädigt wird.

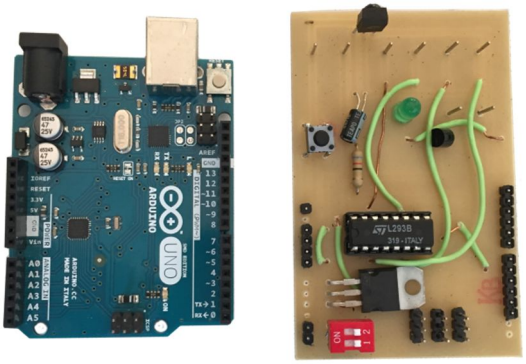


Abb. 8: Arduino Uno und Empfänger-Shield

Herstellung

Über Platinen-Design-Software kann man die Platine am Computer entwerfen.

Für die Herstellung des Shields habe ich eine CNC-Maschine verwendet, die in eine Hartpapierplatte mit 0,35 µm Kupferbeschichtung Isolierbahnen in die Kupferbeschichtung um die eigentlichen Leiterbahnen herum fräst und im Anschluss die Löcher für die Bauteile bohrt.

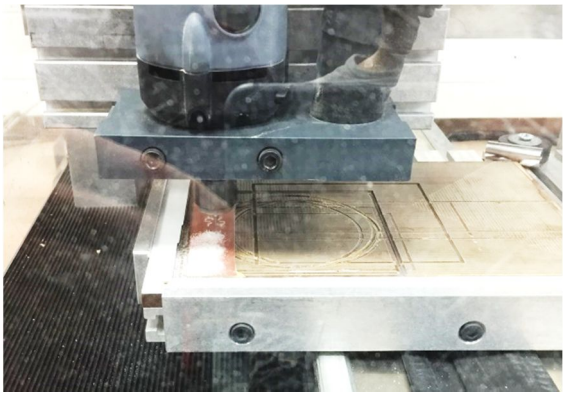


Abb. 9: Platinenherstellung an der CNC Fräse (ältere Aufnahme)

Alternativ bietet es sich an, Platinen zu ätzen. Wer das nicht selbst bewerkstelligen kann, kann einen Dienstleister im Internet mit der Herstellung beauftragen und erhält für rund 12 € eine fertige Platine inklusive Bohrungen.

Eine Lochrasterplatine wäre die schnellste und günstigste Lösung. In diesem Fall müssen jedoch alle Verbindungen einzeln hergestellt werden [6].

Abschließend wurde für das fertig bestückte Shield und den Arduino ein Kunststoffgehäuse in fischertechnik-Abmessungen (90 · 60 mm) angefertigt [11].

Empfänger im Einsatz

Die Anschlüsse für Stromversorgung, Motoren und Servos sind auf die jeweiligen fischertechnik-Stecker abgestimmt. Die Abbildungen 10 und 11 zeigen den Infrarotempfänger, an den die Servomotoren für die einzelnen Achsen und der Antriebsmotor des Krans angeschlossen sind.

Abb. 10: Der fertige IR-Empfänger

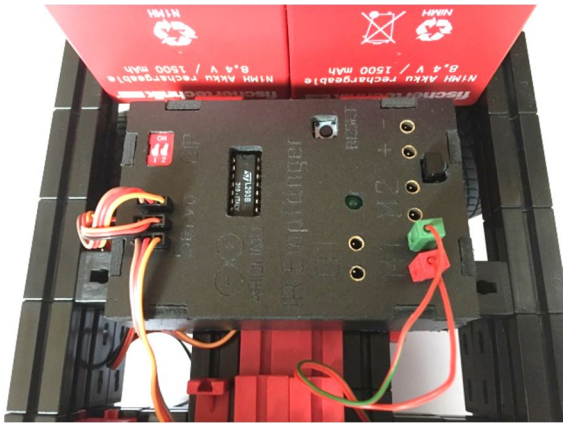


Abb. 11: Der IR-Empfänger im Einsatz

Für den Einsatz im Mobilkran-Modell läuft auf dem Arduino ein passendes Programm, das die Infrarotsignale des Handsenders aus dem fischertechnik Control Set empfängt und die Aktoren entsprechend steuert.

Jede der drei Achsen ist über ein Servo lenkbar. Über den im Empfänger integrierten DIP-Schalter kann man einen der vier verfügbaren Lenkmodi (*Straßenlenkung, Gleichlauflenkung, Hundegang-lenkung* und *verzögernde Hundegang-lenkung*) wählen. Das Arduino-Programm regelt jedes Servo entsprechend des vorgegebenen Lenkwinkels und des gewählten Modus, sodass ein proportionales Lenkverhalten entsteht.

Dazu wird die X-Position des rechten Joysticks in den Lenkwinkel umgerechnet. Im Anschluss wird jedes Servo einzeln mit einem Wert, der sich aus dem Lenkwinkel und dem Lenkindex zusammensetzt, gesteuert. Die Variable *Lenkindex* gibt für jede Achse (je nach gewähltem Modus) die Richtung (± Vorzeichen) und die Intensität (in %) des Lenkeinschlags an.

```
94    // Berechne Lenkwinkel
95    lenkwinkel = map(RightXPosition, -15, 15, -5, 5);
96
97    // Berechne Lenkeinschlag je Achse
98    // Steuere Servos
99    achse1.write(92+lenkwinkel*lenkindex1);
100   achse2.write(87+lenkwinkel*lenkindex2);
101   achse3.write(91+lenkwinkel*lenkindex3);
```

Abb. 12: Programm für Lenkung

Abb. 13 zeigt den Kran im Lenkmodus *Hundegang*. In diesem Fall ist der Lenkindex für alle Achsen gleich und beträgt +100 %, da jede Achse proportional zum vorgegebenen Winkel in dieselbe Richtung einschlagen soll.

Ausblick

Ob spezielle Lenkprogramme für ein mehrachsiges Fahrzeug, Tempomat, oder integrierte Blinksteuerung: Ein Selbstbau-Empfänger für fischertechnik-Modelle lässt sich flexibel einsetzen. Praktisch ist, dass der Arduino über die USB-Schnittstelle einfach umprogrammiert werden kann, um das Programm an das aktuelle Modell anzupassen.

Nutzt man nicht alle Servoausgänge des Selbstbauempfängers, kann man die unbelegten Pins auch für Sensoren nutzen.

Damit könnte man Modelle beispielsweise um Fahrerassistenzsysteme wie eine Einparkhilfe mittels Ultraschallabstandssensor erweitern.

Quellen

[1] Wikipedia: *Fernbedienung*, Abschnitt *IR-Modulation*.

[2] Wikipedia: *RC-5* (Netzwerkprotokoll).

[3] Wikipedia: *Manchester-Code*.

[4] ELV Elektronik AG: *Die wichtigsten Infrarot-Codeverfahren*.

[5] SB-Projects: *Philips RC-MM Protocol*.

[6] Bartmann, Erik: *Arduino Projekt. Kapitel 44 – Der Shieldbau*.

[7] reichelt elektronik GmbH & Co. KG: *IR Receiver Module TSOP4838*.

[8] Vishay Intertechnology, Inc.: *IR Receiver Modules for Remote Control Systems*. Datenblatt, 2012.

[9] Targett, Chris: *Arduino-IR-remote*. Library auf GitHub, 2011.

[10] Van der Weiden, Ad: *IR on RI (or IR Control Set and the Robo Interface)*. fischertechnik Community-Wiki, 2009.

[11] Holtz, David: *Alternativer IR-Empfänger*. ft-Community-Website.

[12] Holtz, David: *Alternative Controller (1): Der Arduino*. ft:pedia 2/2016, in dieser Ausgabe.

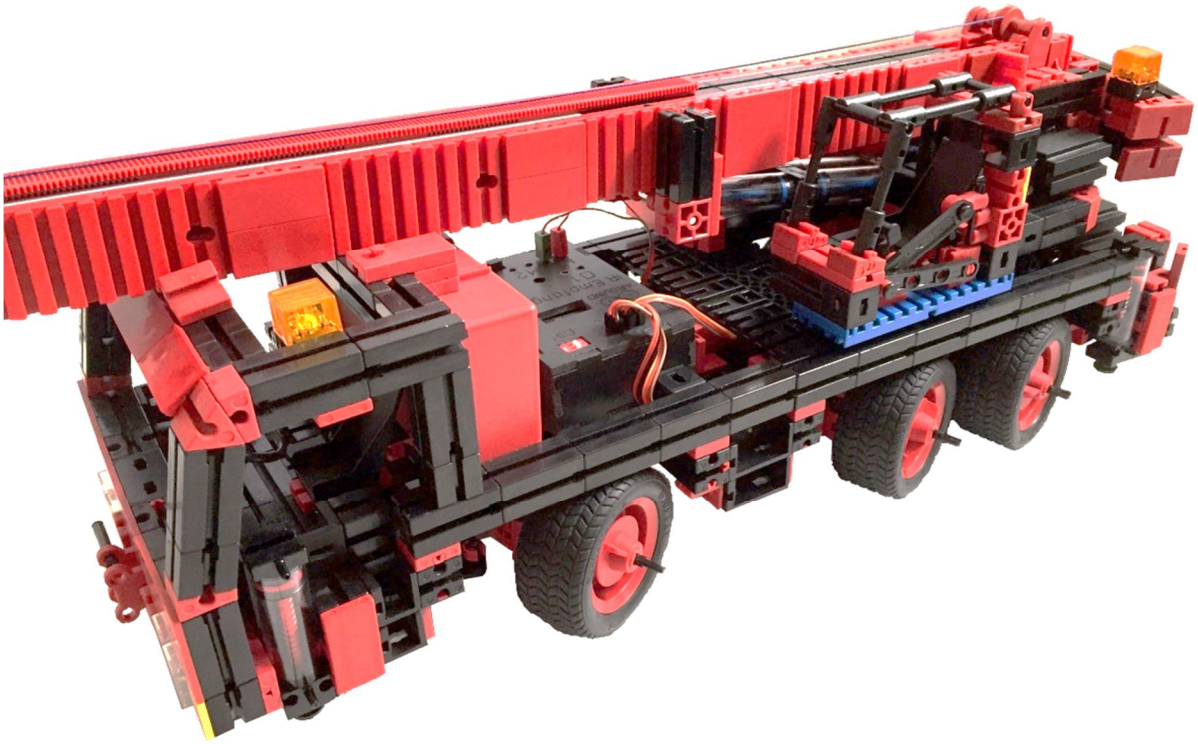


Abb. 13: Mobilkran im Lenkmodus „Hundegang“

Computing

Alternative Controller (3): Der ftPi – ein Motor Shield für den TX(T)

Christian Bergschneider, Stefan Fuss

Am TX(T)-Controller sind die vier Motorausgänge schnell belegt. Aber was, wenn das Modell etwas größer werden soll? Servos sind für Roboter klasse, lassen sich am TX(T) aber nicht direkt anschließen. Da liegt die Idee nahe, einen Motorsteuerungs-Bausatz mit PWM-Baustein an den I2C-Bus des TX(T)-Controllers anzuschließen. Vier zusätzliche Motor- und vier Servo-Ausgänge liefert uns unser ftPi. Aus einer Runde „An-Den-Lötkolben-Fertig-Los“ wurde schnell ein kleines Elektronikprojekt: zwar ist die Schaltung nicht kompliziert, aber der Platz für die zusätzlichen Bausteine beschränkt.

Hintergrund

DC-Motoren

Die Geschwindigkeit eines DC-Motors lässt sich einfach über die Spannung regeln. Je höher die Spannung, desto schneller dreht sich der Motor. Die Spannung eines Motors über einen digitalen Ausgang zu steuern, ist hingegen relativ komplex. Als Abkürzung nutzt man deshalb gerne die Trägheit des Motors aus. Schaltet man einen digitalen Ausgang in sehr kurzen Abständen an und aus, so stellt sich dies für den Motor auch als Spannungsreglung dar.

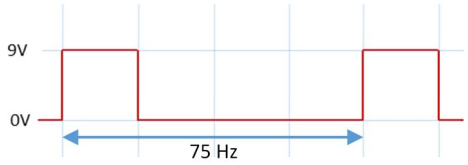


Abb. 1: Beispiel für ein PWM-Signal

Ist z. B. das Steuersignal ¼ der Zeit ein- und ¾ der Zeit ausgeschaltet, so wird der Motor auf ¼ der Maximaldrehzahl laufen. Der Motor verhält sich so, als ob er mit 2,25 V statt 9 V Gleichspannung betrieben würde.

Die Frequenz eines solchen Pulsweitenmodulation- (PWM-) Signals wirkt sich nicht auf die Drehzahl aus. Ist sie allerdings zu gering, so läuft der Motor nicht rund. Je höher die Frequenz des PWM-Signals ist, desto besser läuft der Motor. Bei hohen Frequenzen treten jedoch Wirbelstromeffekte auf; der Motor kann dann die aufgenommene elektrische Leistung nicht mehr in Drehmoment umsetzen.

Im Internet findet man ganze Diskussionsforen darüber, welches die richtige PWM-Frequenz ist. Da diese stark vom Motor und der Anwendung abhängt, ist Ausprobieren die einfachste Lösung.

fischertechnik-Motoren

Alle Motoren von fischertechnik sind DC-Motoren. Sie werden mit 9 V betrieben und lassen sich mit PWM-Technik sehr gut regeln. Für den Artikel haben wir S-, XS-, Encoder- und den klassischen grauen Motor getestet: Mit einer PWM-Frequenz zwischen 70 und 1.500 Hz laufen die Motoren rund.

Servo-Motoren

Servomotoren sind normale Gleichstrommotoren mit einer integrierten Steuerung. Sie haben drei Anschlüsse: Zwei für die Spannungsversorgung, der dritte Eingang ist das Steuersignal.

Als Steuersignal wird ein PWM-Signal eingesetzt. Im Unterschied zum DC-Motor wird das Signal jedoch von der internen Steuerung ausgewertet und in eine Winkelposition des Servos umgesetzt. Im Standard wird das Signal alle 20 ms (entsprechend 50 Hz) wiederholt.

Die Servo-Steuerung interpretiert die Impulsbreite: Ist das Signal 1 ms breit, so stellt sich der Servo in die linke, bei 1,5 ms in die mittlere und bei 2 ms in die rechte Position.

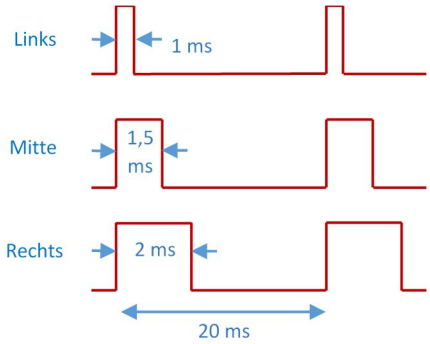


Abb. 2: Die Impulsbreite steuert den Servo

Die Frequenz muss nicht genau 50 Hz betragen. Einige Servo-Typen können mit bis zu 400 Hz angesteuert werden.

fischertechnik-Servos

Die Spannungsversorgung bei den fischertechnik-Servos muss zwischen 4,8 und 6 V liegen. Die maximale Speisespannung von 6 V darf nicht überschritten werden, da sonst die interne Steuerung Schaden nehmen kann. Das ist ungewohnt, da fischertechnik normalerweise mit 9 V arbeitet.

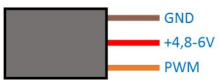


Abb. 3: Steckerbelegung der ft-Servomotoren

Die Steuerelektronik ist auf 50 Hz ausgelegt. Sie arbeitet nur in einem kleinen Frequenzbereich bis ca. 85 Hz stabil; danach steigt die Steuerung aus und der Servo brummt. Das Steuersignal muss nicht genau der Speisespannung entsprechen. Eine Logikspannung von 3,3 V ist ausreichend – und die liefert unser „fischertechnik Pi“ (ftPi).

Der fischertechnik Pi (ftPi)

Nur vier Motorausgänge bieten TX und TXT – und keine Möglichkeit, einen fischertechnik-Servo anzuschließen. Dafür beherrschen beide Controller das I²C-Protokoll. Warum also nicht ein geeignetes Motor Shield aus der Arduino- oder Raspberry Pi-Welt fischertechnik-tauglich machen?

Tatsächlich erhalten wir mit wenigen Modifikationen aus einer marktgängigen Motorsteuerung ein leistungsfähiges fischertechnik Motor Shield (kurz: ftPi), mit dem wir die fischertechnik-Controller um vier Gleichstrom- und vier Servo-Motorausgänge erweitern können – dank I²C sogar bis zu 32 mal.

Die Adafruit DC Motor KITs

Von Adafruit gibt es zwei Motorsteuerungen, die sich für die Anbindung an den TX(T) eignen, da sich beide über die I²C-Schnittstelle ansteuern lassen: das *Arduino Motor Shield* und das *Motor HAT Mini Kit* für den Raspberry Pi [1, 2]. Kern der Schaltung ist der programmierbare PWM-Modulator PCA9685, ein 16-Port-LED-Treiber mit I²C-Bus-Interface.

Alle 16 Ausgänge des Treiberbausteins können mit unterschiedlichen PWM-Signalen beschaltet werden. Die Taktfrequenz ist programmierbar, allerdings ist diese für alle PWM-Ausgänge gleich. Als Leistungsendstufe für DC-Motoren kommen auf der Platine zwei TB6612 zum Einsatz. Die Ansteuerung pro Motor erfolgt über drei LED-Ausgänge.

IN1	IN2	PWM	Motor
L	H	H/L	⇦
H	L	H/L	⇨
L	L	H	STOP

Tab. 1: Die Ansteuerung des TB6612

Die Leistung des PWM-Modulators ist für LEDs ausgelegt. Somit können die Steuersignale für die Servomotoren direkt angeschlossen werden.

Stromversorgung

Beide Motor Kits können über das Netzgerät des Power Sets oder einen fischertechnik-Akku mit 9 V betrieben werden. Auf der Platine wird die Stromversorgung für die Leistungsendstufen bereits geglättet.

Der LED-Treiber erhält in der Originalbeschaltung seine Betriebsspannung vom Raspberry Pi bzw. dem Arduino. Ohne ihn muss diese über einen 3,3 V Spannungsregler (LP2950CZ3) erzeugt werden (Abb. 4).

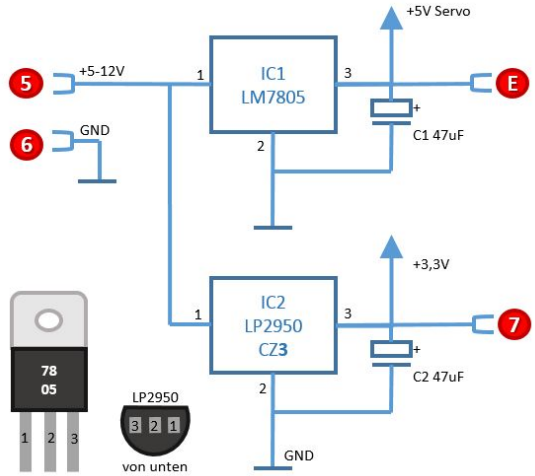


Abb. 4: Spannungsversorgung

Die Servo-Motoren müssen ebenfalls versorgt werden. Dies kann über den 5 V-Spannungsregler (LM7805) erfolgen.

TX mit und ohne T

Das I²C-Interface am TX arbeitet mit 5 V, der TXT benötigt 3,3 V. Der Umbau zwischen TX und TXT ist einfach, da der PCA9685 mit beiden Spannungen betrieben werden kann.

Für den TXT wird für die Logikspannung ein 3,3 V-Regler (LP2950CZ3) und für den TX die 5 V-Variante (LP2950CZ5) des gleichen Reglers eingesetzt.

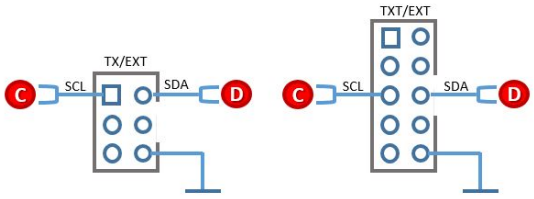


Abb. 5: Wannenstecker für TX und TXT

Jetzt muss man nur noch SDA, SCL und GND über Flachbandkabel mit dem TX(T) verbinden.

Anpassung der Motor Shields

Die Bausätze werden von Adafruit vorbestückt geliefert, d. h. die SMD-Bausteine sind bereits auf der Platine aufgelötet. Die beiliegenden Steckverbinder werden nicht benötigt.

In unserem Projekt kam [1] zum Einsatz. Um Gewicht und Platz zu sparen wurden alle zusätzlichen Bauteile auf der Platine untergebracht. Der Prototypenbereich bietet nicht ausreichend Platz, daher wurden auch Stromversorgungs- und Raspberry-Pi-Anschlüsse umgebaut. Dazu mussten u. a. Leiterbahnen auf der Platine durchtrennt werden. Hierfür ist etwas Erfahrung erforderlich. Für den Nachbau empfiehlt es sich daher, die zusätzlichen Bauteile auf einer getrennten Prototypenkarte aufzubauen und beide Platinen mit Kabeln zu verbinden.

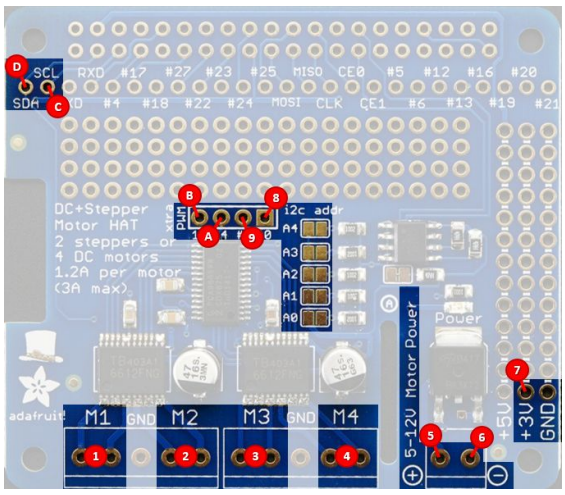


Abb. 6: Die relevanten Anschlüsse des Adafruit Motor HAT [1]

Die Motorausgänge sind auf der Platine mit M1 bis M4 markiert. Die GND-Anschlüsse werden beim DC-Motor nicht benötigt. Für die fischertechnik-Stecker setzt man auf der Platine einen Lötnagel ein und schraubt darauf eine normale Zwergbananenbuchse fest. Die Abstände der Originalanschlüsse reichen allerdings nicht aus, so dass die Motoranschlüsse mit einem kurzen Kabel in den Prototypenbereich verlegt werden müssen.

Der Prototypenbereich bei [2] ist etwas größer; es befinden sich auch schon zwei Steckverbinder für Servo-Motoren auf der Platine. Montiert man hier den 5 V-Spannungsregler stehend, so reicht der Platz auf der Platine aus (Abb. 7).

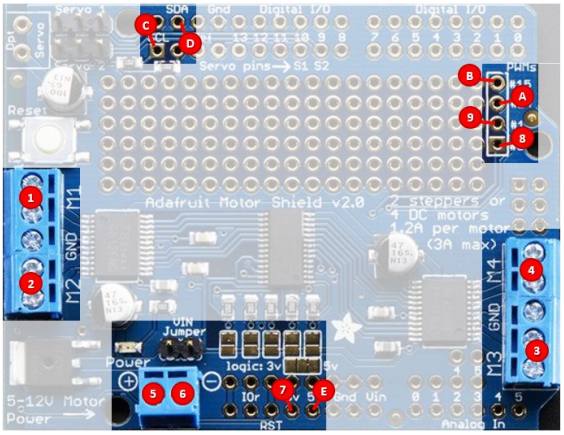


Abb. 7: Die relevanten Anschlüsse des Adafruit Arduino Motor Shield [2]

Die Stromversorgung des Power Sets wird bei (5) und (6) angeschlossen. Dabei bitte unbedingt auf die Polung achten!

Die beiden Spannungsregler werden an die 9 V-Versorgung angeschlossen. Auf der Ausgangsseite werden die Signale mit je einem Elko mit 47 µF geglättet (siehe Abb. 4). Für den TX wird bei I²C die 5 V-Variante eingesetzt und dennoch mit dem 3 V-Anschlüssen der Platine verbunden.

Die PWM-Signale für die Servos werden an (8), (9), (A) und (B) angeschlossen. Bei [2] sind die Anschlüsse für S1 und S2 bereits auf der Platine vorhanden und können entfallen (Abb. 7).

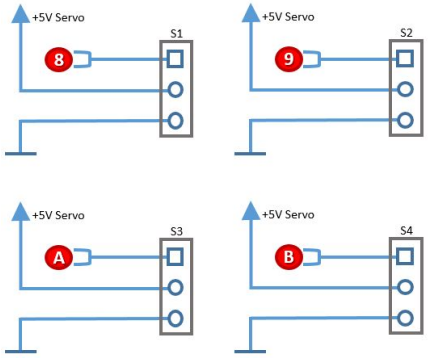


Abb. 8: Beschaltung der Servo-Ausgänge

Der TX(T) wird über eine Wannenleiste angeschlossen. SDA und SCL stehen bei (C) und (D) zur Verfügung.

Programmierung des ftPi

Für die Programmierung steht die RoboPro-Bibliothek ftPi.rpp zur Verfügung. Mit dem Bespielprogramm ftPi-Example.rpp können erste Tests der Schaltung erfolgen. Die Programme finden sich im Download-Bereich der ft:Community.

In jedem Programm muss der PWM-Baustein zunächst initialisiert werden. Dazu muss die Funktion INIT aufgerufen werden. Die Funktion bestimmt die Beschaltung der LED-Ausgänge und aktiviert den internen Oszillator. Dieser gibt den Takt für das PWM-Signal vor. Leider kann für alle Ausgänge nur eine Frequenz eingestellt werden.

Die Bibliothek stellt den Frequenzteiler im Register 0xFE auf 0x50; dies entspricht 76 Hz. Werden keine Servos im Modell eingesetzt, so kann über eine höhere Frequenz der Betrieb der DC-Motoren optimiert werden. Der Frequenzteiler kann mit der folgenden Formel berechnet werden:

$$prescale = \frac{25\,000\,000}{4096 \cdot Sollfrequenz} - 1$$

Zum Abschluss der Funktion werden alle Motoren ausgeschaltet und die Servos auf die mittlere Position gestellt.

Die Ansteuerung der Motoren übernehmen die Unterprogramme M5 bis M8. Die Drehzahl kann in 256 Stufen eingestellt werden

(0 = aus, 255 = maximale Drehzahl). Mit dem zweiten Eingang wird die Drehrichtung (> 0 rechts, ≤ 0 links) gewählt.

Die Servo-Motoren werden über die Unterprogramme S1 bis S4 gesteuert. Sie können mit Werten zwischen -150 (linke Position) und 150 (rechte Position) angesteuert werden.

Mehrere ftPis an einem Controller

Die I²C-Geräteadresse des PCA9685 kann über fünf Jumper frei eingestellt werden. Theoretisch können so bis zu 32 ftPis an einen TX(T)-Controller angeschlossen werden. Die Basisadresse des PWM-Bausteins ist 0x60.

RoboPro kann leider nicht mit flexiblen I²C-Adressen arbeiten. Beim Einsatz von mehreren fischertechnik Pis an einem TX(T)-Controller muss die Bibliothek mehrfach kopiert und die jeweils eingestellte I²C-Geräteadresse ausgetauscht werden.

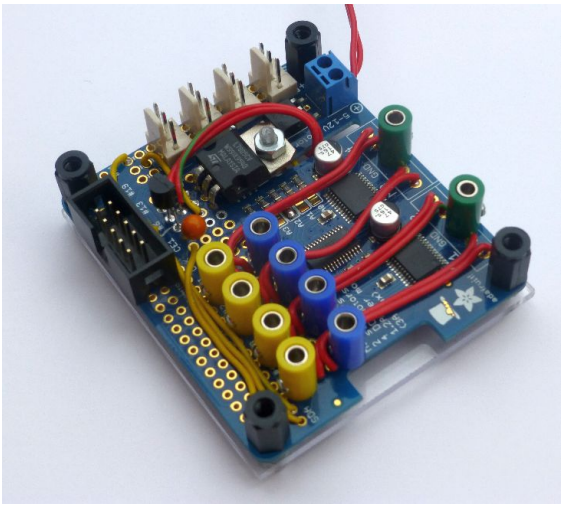


Abb. 9: Der fertige ftPi

Inbetriebnahme des ftPi

Bei der Inbetriebnahme empfiehlt es sich, zunächst die Platine ohne TX(T) zu testen und die Anschlüsse der beiden Spannungsregler kontrollieren, denn sie sind unterschiedlich belegt.

Dazu schließt man die Stromversorgung an und prüft mit einem Multimeter die Ausgangsspannung an den Spannungsreglern und den Servo-Anschlüssen. Dann erst sollte man den TX(T) anschließen.

Risiken und Nebenwirkungen

Obwohl die Schaltung recht einfach ist, müssen ein paar Dinge berücksichtigt werden:

- Ist der ftPi für den TX-Betrieb mit einem 5 V-Spannungsregler ausgestattet, so darf er nicht mit einem TXT-Controller verbunden werden.
- Die Buchsen der Servos haben zwar einen Verpolungsschutz, allerdings ist es schwierig, den passenden Stecker zu bekommen. Beim Anschließen der Servos daher immer auf die Polung achten, um Schäden an der Platine und den Servos zu vermeiden.
- Der PWM-Baustein arbeitet weitgehend autark. Er benötigt den TX(T)-Controller nur zur Parametrierung. Wird das Robo Pro-Programm abgebrochen oder der TX(T)-Controller abgeschaltet, so laufen die Motoren und Servos am ftPi weiter. Zum Not-Aus die Spannungsversorgung des ftPi unterbrechen. Typ [2] hat hierfür einen Reset-Taster auf der Platine.

Der Nachbau erfolgt auf eigenes Risiko – für Schäden übernehmen die Autoren keine Haftung. Bei Fragen und Anregungen könnt ihr uns gerne eine E-Mail an ftpi@gmx.de schicken.

Quellen

[1] Adafruit: *DC & Stepper Motor HAT for Raspberry Pi – Mini Kit*

[2] Adafruit: *Adafruit Motor/Stepper/Servo Shield for Arduino v2 Kit – v2.3*

Computing

Wiederbelebung eines fischertechnik-Buggy-Modells von 2002

Dirk Uffmann, Roland Enzenhofer

Ein fischertechnik-Buggy von Economatics aus dem Jahr 2002, dessen Ursprünge bis in das Jahr 1983 zurückreichen, erreichte nie seinen ursprünglichen Einsatzweck in einer englischen Schule. Wir erzählen, wo er von Roland entdeckt und erworben wurde – und was wir getan haben, um ihn zum „Leben" zu erwecken.

Herkunft

Als begeisterter Sammler von fischertechnik ist Roland immer wieder auf der „Jagd" nach Raritäten. Der Zufall wollte es, dass ein verirrtes Inserat samt Foto auf dem Bildschirm aufblitzte, welches zwar fischertechnik zum Inhalt hatte, aber nicht fischertechnik hieß. So stöberte Roland, bis er zu den Wurzeln von „Economatics" fand, und da war sein Interesse geweckt. Ein Problem beim Erwerb der Sachen war, dass die Sammlung in einer aufgelassenen Schule in England war und insgesamt über 200 kg Gewicht hatte (siehe die Bilder auf den ft:c-Seiten). Als Logistiker hat Roland auch dieses Problem gelöst und die Sammlung fand unversehrt den Weg von England nach Österreich in sein Raritätenkabinett.

Neben den vielen anderen Kuriositäten und Seltenheiten rund um fischertechnik sollte der fischertechnik PIC Buggy von Economatics (Abb. 1) aber in der Sammlung kein funktionsloses Dasein ohne Steuerprogramm fristen müssen. Dadurch und Dank der ft:c-Gemeinde kam der Kontakt zu Dirk zustande, der sich begeistert an die Arbeit machte, den Buggy in Gang zu bringen und insbesondere dem PIC Mikrocontroller ein paar Funktionen „einzuhauchen".

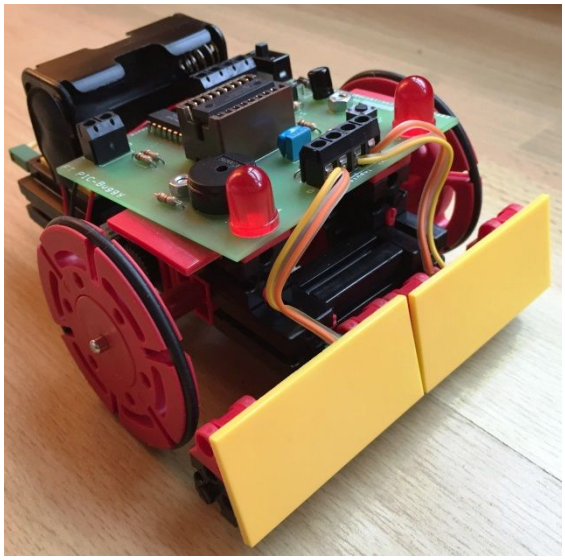


Abb. 1: Der Fischertechnik PIC Buggy von Economatics aus dem Jahr 2002

Historie

Die Firma Economatics aus Sheffield in England hat von ca. 1981 bis 2008 Elektronik- und Robotik-Produkte für den Schulbereich angeboten. Bekannt geworden ist der sogenannte BBC-Buggy [1] aus dem Jahr 1983, der per Flachbandkabel vom BBC Micro, einem Heimcomputer von der britischen Firma Acorn, per Basic-Programm gesteuert wurde. Ein modifizierter Nachbau des BBC Buggy von xbach ist auf den Seiten der ft:c zu bewundern. Dieser wurde bereits mit einem PIC-Mikrocontroller gesteuert. Aus dem Jahr 2002 stammt

der hier vorgestellten ft-Buggy. Economatics hat dazu auch die Software-Entwicklungsumgebung angeboten mit einer Flowchart-basierten Programmierumgebung namens „PIC-Logicator Software“ und einem dedizierten Hardware-Programmer für den PIC, der dessen Speicher beschreiben kann. Economatics ging leider 2008 in Konkurs. Dabei hat der industrielle Bereich den Ausbildungsbereich mit in die Pleite gezogen. Der Ausbildungsbereich wurde von der Firma Revolution Education Ltd. übernommen. Bis heute wird über PICAXE ein kleiner Buggy namens PICAXE-20X2 Microbot als Produkt angeboten – leider nicht mehr aus fischertechnik. Die Logicator Software wird von PICAXE weiterentwickelt.

Aufbau des Buggy

Die Konstruktion des Economatics-Buggy ist minimalistisch gehalten (siehe Abb. 2 bis 4). Das Fahrgestell beinhaltet zwei S-Motoren für den Antrieb, die zugleich als tragende Elemente dienen. Zum Anbringen der Platine mit zwei Schrauben wurde eine rote Bauplatte mit zwei Löchern versehen. Der Batteriehalter wurde auf eine gelbe Bauplatte geklebt. Vorne dienen zwei gelbe Bauplatten mit zwei Tastern zur Detektion von Kollisionen.

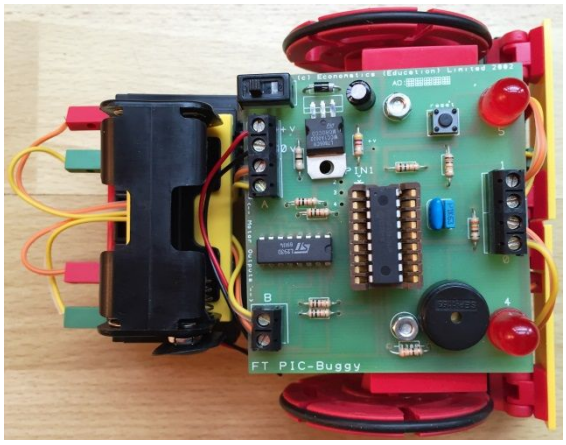


Abb. 2: ft-Buggy von oben

Die Drehscheiben 60 werden mit je einem O-Ring zum Antriebsrad. Passende O-Ringe in hervorragender Qualität (NBR70) und unterschiedlichen Dicken gibt es beispielsweise bei HUG Industrietechnik (Art. Nr. 4306000500, 60 x 5 mm).

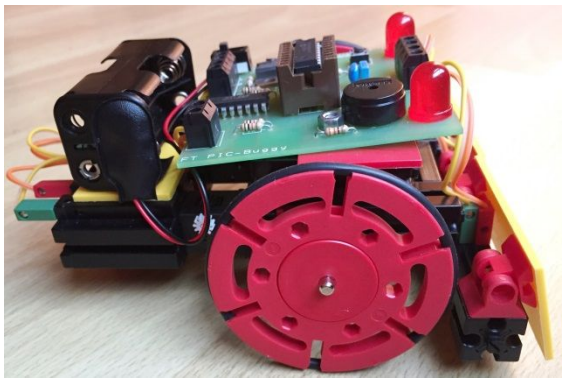


Abb. 3: ft-Buggy von der Seite

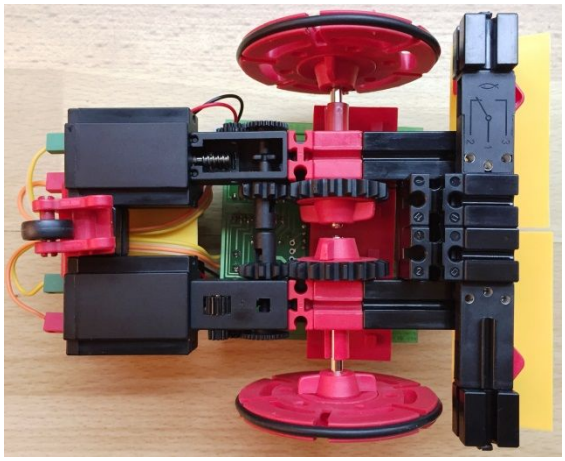


Abb. 4: ft-Buggy von unten

Steuerungsplatine

Auf dem Buggy ist zur Steuerung eine Platine mit einem PIC Flash-Mikrocontroller 16F628 und L293D Motortreiber angebracht (Abb. 5 und 6). Im Hobby-Bereich waren im Jahre 2002 Embedded Flash Mikrocontroller noch bei weitem nicht so etabliert wie heute. Bei dem 16F628 handelt es sich um einen sogenannten Mid-Range Typen, der über 3,5 kByte Flash-Speicher (2048 Programmworte à 14 Bit), 228 Byte RAM und 128 Byte EEPROM verfügt, sowie einen PWM-Ausgang und mehrere Eingänge mit analoger Komparator-Funktion. Dieser Typ ist sogar heute noch bei Microchip in Produktion, wird aber nicht mehr für neue Designs empfohlen. Tab. 1 zeigt die Pinbelegung am

PIC µC. Vier Anschlüsse am PIC µC sind noch frei und könnten für Erweiterungen genutzt werden.

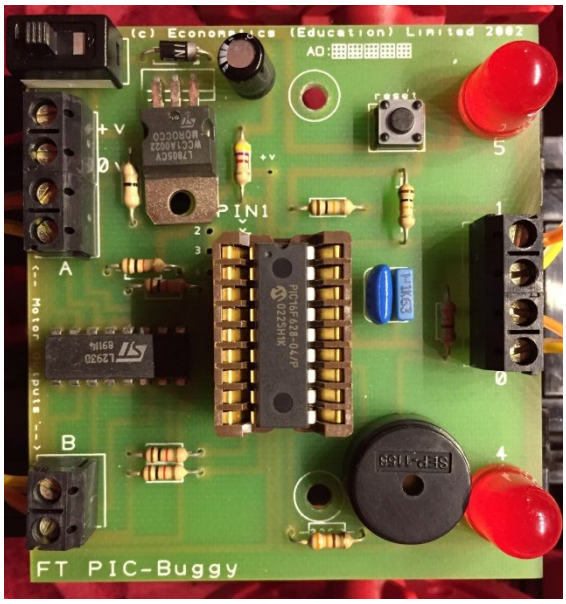


Abb. 5: Oberseite der Platine des ft-PIC-Buggy mit PIC 16F628 und L293D Motortreiber-IC, Ausgangsklemmen für zwei Motoren, zwei LEDs, ein Piezo-Lautsprecher und Eingangsklemmen für die Taster zur Hinderniserkennung

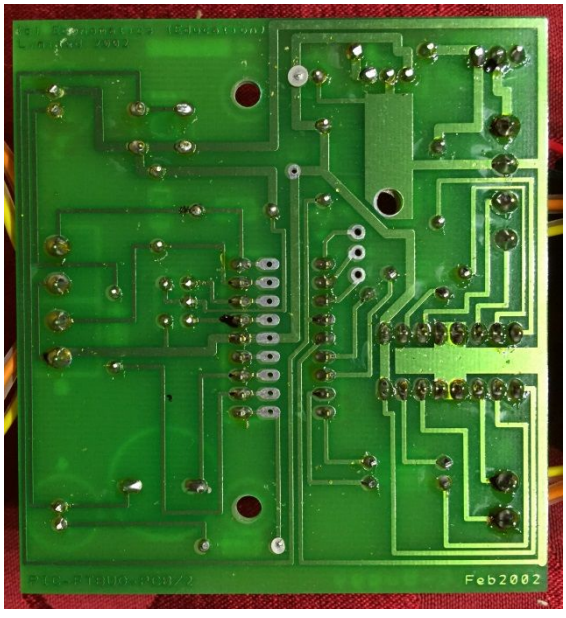


Abb. 6: Unterseite der Platine des ft-Buggy

Die Platine verfügt über zwei Eingänge für die Taster zur Detektion von Kollisionen sowie vier Ausgänge für den Motortreiber L293D zur Steuerung der beiden Antriebsmotoren. Über weitere drei Ausgänge können zwei LEDs und ein Piezo-Lautsprecher angesteuert werden.

Pin	Port / Funktion	Anschluss / Komponente
1	RA2 / AN2 / Vref	frei
2	RA3 / AN3 / CMP1	frei
3	RA4 / TOCKI / CMP2	frei
4	RA5 / MCLR /Vpp	Reset-Taster
5	Vss	Masse
6	RB0 / INT	L293D In1 / Motor A (links)
7	RB1 / RX /DT	L293D In2 / Motor A (links)
8	RB2 / TX / CK	L293D In3 / Motor B (rechts)
9	RB3 / CCP1	L293D In4 / Motor B (rechts)
10	RB4 / PGM	LED rechts
11	RB5	LED links
12	RB6 / T1OSO / T1CKI / PGC	Piezo-Summer
13	RB7 / T1OSO / PGD	frei
14	Vdd	+5V
15	RA6 / OSC2 / CLKOUT	4 MHz Keramik-Oszillator
16	RA7 / OSC1 / CLKIN	4 MHz Keramik-Oszillator
17	RA0 / AN0	Taster rechts
18	RA1 / AN1	Taster links

Tab. 1: Pin-Zuordnung am PIC 16F628 zu den Komponenten

Programmierung

Leider stand weder die originale Version der Software PIC-Logicator von 2002 noch das Programmiergerät von Economatics zur Verfügung; es war nicht Bestandteil der Lieferung aus England. Von der ältesten verfügbaren Version der Software wird der 16F628 nicht mehr direkt unterstützt; es ginge nur über einen Umweg: Man müsste zunächst den erstellten Flow-Chart als Basic-Code exportieren und diesen dann mit einem lizenzpflichtigen Basic-Compiler übersetzen. Auf diesen Umweg haben wir verzichtet und einen alternativen Weg beschritten: Wir erstellten mit der kostenlosen Entwicklungsumgebung MPLAB X IDE und dem ebenfalls kostenlosen C-Compiler XC8 ein Programm und brannten es mit einem PICKit2-Clone (veraltetes Programmier-Gerät für PICs), inklusive Adapterplatine mit Nullkraftsockel für 10 € aus China beschafft, auf den Baustein. Auf der Adapterplatine musste zunächst ein Verdrahtungsfehler beseitigt werden, bis

die PIC16F628 erkannt wurde und sich in den Programmiermodus versetzen ließ.

Steuerprogramm

Ziel des Steuerprogramms ist es, die wesentliche Funktion des Buggy zu implementieren, nämlich das Erkennen von Kollisionen mit Hindernissen und deren Umfahrung (Ausweichmanöver). Abb. 7 zeigt das Ausweichmanöver schematisch.

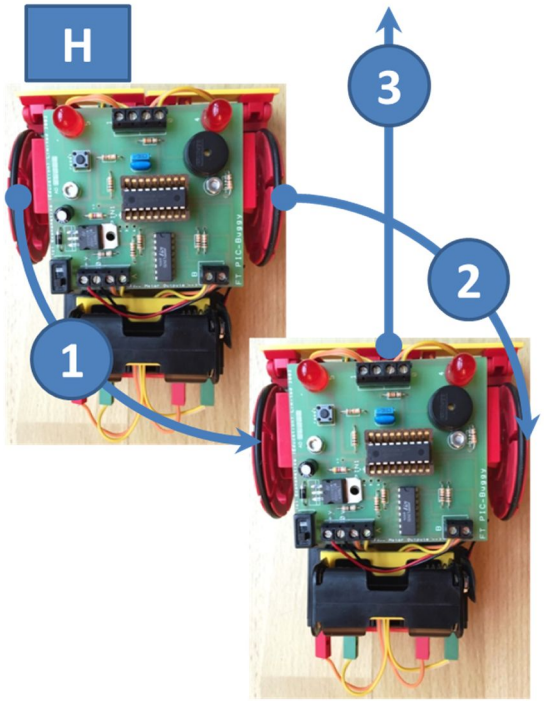


Abb. 7: Schema des einfachen Ausweichmanövers in drei Schritten bei Kollision mit einem Hindernis (H) auf der linken Seite

Bei einer Kollision mit einem Hindernis *H* auf der linken Seite fährt zunächst nur das linke Rad rückwärts, bis der Buggy sich um 90° gedreht hat (Schritt 1). Die Drehachse ist dabei der Aufpunkt des rechten Rades am Boden. Dann fährt nur das rechte Rad rückwärts, bis der Buggy sich wieder um 90° zurück gedreht hat (Schritt 2). Die Drehachse ist dabei nun der Aufpunkt des linken Rades am Boden. Nun wird die Vorwärtsfahrt mit beiden Rädern wieder aufgenommen (Schritt 3). Für die rechte Seite ist es genau spiegelverkehrt implementiert. Wenn beide Taster gleichzeitig eine Kollision anzeigen, hat das Ausweichmanöver für Hindernis links Vorrang.

Um die Fahrbewegungen des Buggy etwas interessanter zu gestalten, habe ich eine Software-PWM-Steuerung über den Interrupt des Timer2-8-bit-Überlauf implementiert. Beide Motoren können unabhängig voneinander in zehn Leistungsstufen angesteuert werden. Über die Variable `Beschleunigung` kann wahlweise die Leistungsstufe alle 260 ms (viermal Timer1-16-bit-Überlauf) um eine Stufe erhöht oder erniedrigt werden (innerhalb definierter Unter- und Obergrenzen). Bei der Vorwärtsfahrt beschleunigt der Buggy von Leistungsstufe 3 auf 9 innerhalb von 1,56 Sekunden. Beim Ausweichmanöver wird im ersten Schritt das Rad von Leistungsstufe 3 auf 9 beschleunigt und dann im zweiten Schritt das andere Rad von Leistungsstufe 9 auf 3 verlangsamt, jeweils innerhalb von 1,56 Sekunden. Damit sollen die Steuermöglichkeiten für die Geschwindigkeit demonstriert werden.

Daneben soll von den beiden LEDs und vom Piezo-Lautsprecher Gebrauch gemacht werden, und zwar soll ein Warnsignal erzeugt werden, während ein Ausweichmanöver gefahren wird. Das Warnsignal ist ein wechselndes Blinken der LEDs und eine in der Tonhöhe zwischen 400 und 600 Hz auf- und ablaufende Sirene.

Tab. 2 bis 5 zeigen den erstellten C-Code.

Video

Das Verhalten des wiederbelebten ft-PIC-Buggy und die Reaktion auf Hindernisse kann in diesem Video beobachtet werden.

Referenzen

[1] fischertechniker: *Wie weit fischertechnik vor 32 Jahren seiner Zeit voraus war*. fischertechnik-blog.de

```
#include <xc.h>
// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.
// CONFIG
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator: Crystal/resonator on
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = ON       // Power-up Timer Enable bit (PWRT enabled)
#pragma config MCLRE = ON       // RA5/MCLR pin function select (RA5/MCLR pin function is MCLR)
#pragma config BOREN = ON       // Brown-out Reset Enable bit (BOD Reset enabled)
#pragma config LVP = OFF        // Low-Voltage Programming Enable bit (RB4/PGM pin has digital I/O function)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection off)
#pragma config CP = OFF         // Code Protection bits (Program memory code protection off)

#define _XTAL_FREQ 4000000
#define pwm_max 9
#define pwm_min 3
#define Ausweichzeit 25         // in Mehrfachen von 65,5 ms
#define Ton_tief 156
#define Ton_hoch 100
#define links 0
#define rechts 1
#define vorwaerts 2
#define rueckwaerts 1
#define stop 3

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

volatile uint_fast8_t Motor_Bits_PWM=0;
volatile uint_fast8_t Motor_Bits_Merker=0;
volatile uint_fast8_t PWM_Zaehler = 0;
volatile uint_fast8_t PWM_Tabelle[2];
volatile uint_fast8_t Zeitschritte_65ms = 0;
volatile uint_fast8_t Zeitschritte_Warnsignal = 0;
volatile uint_fast8_t Zeitschritte_Beschleunigung = 0;
volatile uint_fast8_t Ton_Richtung = 0;
volatile uint_fast8_t Hindernis = 0;
volatile uint_fast8_t Hindernis_links = 0;
volatile uint_fast8_t Hindernis_rechts = 0;
volatile uint_fast8_t Ton = Ton_tief;
volatile int_fast8_t Ton_Schritt = -4;
volatile uint_fast8_t Warnsignal = 0;
volatile int_fast8_t Beschleunigung = 0;

void init (void) {
        __delay_ms(1000);   // 1 Sekunde warten
        CMCON = 0x07;       //PORTA als Eingänge (Komparatoren abschalten)
        TRISB = 0;          //PortB als Ausgänge
        OPTION_REG = 0xC2;
       // T0CS = 0; PSA = 0; PS2=0; PS1=1; PS0=0
       // i.e. clock/8 = 125 kHz bzw. 8 µs Periode
        INTCONbits.T0IE = 1; // Interrupt einschalten für den TIMER0 Überlauf
       // TIMER1 (16 bit) läuft ohne Prescaler
       // i.e. 65,5 ms Periode (16 Wiederholungen ergeben 1 Sekunde)
        T1CONbits.TMR1ON = 1; // TIMER1 (16 bit) aktivieren
        PIE1bits.TMR1IE = 1; // Interrupt einschalten für den TIMER1 Überlauf
        PR2 = 125;          // Überlauf für TIMER2, i.e. 500 Hz (1 MHz / (16*125)
        T2CON = 0x06;       // TIMER2 einschalten, Prescaler 1:16
        PIE1bits.TMR2IE = 1; // Interrupt einschalten für den TIMER2 Überlauf
        INTCONbits.GIE = 1;  // Sämtliche Interrupts global einschalten
}
```

Tab. 2: C-Code erster Teil

```
/*****************************************************************************
        Motor () stellt die Funktion zur Manipulation von Motor_Bits_Merker zur
        Steuerung einzelner Motoren bereit. Der pwm-Wert gilt auch bei
        Einzelbeschaltung der Ausgänge immer für beide Ausgänge.
        Motor_Nummer = 0; //linker Motor
        Motor_Nummer = 1; //rechter Motor
        pwm von 0 bis 10
        Zustand=0 => Motor aus / Minus an beiden Motorausgängen
        Zustand=1 => Motor an / rückwärts
        Zustand=2 => Motor an / vorwärts
        Zustand=3 => Motor aus / Plus an beiden Motor-Kabeln
*****************************************************************************/
void Motor (uint_fast8_t Motor_Nummer, uint_fast8_t Zustand, uint_fast8_t pwm) {
//      Abfangen unsinniger Werte
        if (Motor_Nummer > 1) return;
        if (Zustand > 3) return;
        PWM_Tabelle[Motor_Nummer] = pwm;
        Motor_Nummer = Motor_Nummer << 1;
//      Umrechnen von Motor_Nummer auf die Bitposition
//      Zunächst wird "Zustand" in "Motor_Bits_Merker" an der gewünschten Position
//      hinzugefügt, dazu werden zunächst die jeweiligen 2 Bits an den Bit-
//      Positionen für Motor_Nummer auf 0 gesetzt
        Motor_Bits_Merker &= ~(3 << Motor_Nummer);
//      und dann die 2 neuen Bits dem Motor_Bits_Merker hinzugefügt
        Motor_Bits_Merker |= Zustand << Motor_Nummer;
}

void Ausweichen (void) {
        uint_fast8_t x=links, y=links;
        if (Hindernis_links) x=rechts;
        else y=rechts;
        Motor(x, stop, 0);      // Motor x abschalten
        Motor(y, rueckwaerts, pwm_min); //Motor y rückwärts
        Warnsignal = 1;
        Zeitschritte_65ms = 0;
        while (Zeitschritte_65ms < Ausweichzeit); // Zeit für die Fahrt
        Motor(y, stop, 0);      // Motor y abschalten
        Beschleunigung = -1;
        Motor(x, rueckwaerts, pwm_max); //Motor x rückwärts
        Zeitschritte_65ms = 0;
        while (Zeitschritte_65ms < Ausweichzeit); // Zeit für die Fahrt
        Warnsignal = 0;
        Beschleunigung = 1;
        Motor(links, vorwaerts, pwm_min);
        Motor(rechts, vorwaerts, pwm_min); // Vorwärtsfahrt
}

int main(void) {
        init();
        Motor(links, vorwaerts, pwm_min);
        Motor(rechts, vorwaerts, pwm_min); // Vorwärtsfahrt
        Beschleunigung = 1;

        while(1) {
                if (Hindernis) {
                        Ausweichen();
                }
        }
}
```

Tab. 3: C-Code zweiter Teil

```
void interrupt Routine (void) { // interrupt function
        if (INTCONbits.T0IF) { // Prüft, ob die TIMER0 Überlauf Flagge gesetzt wurde
        // Dieser Code-Teil wird ausgeführt alle: Ton * 8µs, das entspricht der halben Periode für den Piezo-Summer
        // -> Ton = 156 -> 2.5 ms volle Periode oder 400 Hz Ton-Frequenz
        // -> Ton = 100 -> 1.6 ms volle Periode oder 600 Hz Ton-Frequenz
                INTCONbits.T0IF = 0; // lösche die Interrupt Flagge
                if (Warnsignal)  {
                        TMR0 -= Ton; //TIMER0 vorladen, um eine bestimmten Tonhöhe zu erzeugen
                        if (RB6) RB6=0; //Ausgang für den Piezo-Summer umschalten
                        else RB6=1;
                }
        }

        if (PIR1bits.TMR1IF) {  // Prüft, ob die TIMER1 Überlauf Flagge gesetzt wurde
                PIR1bits.TMR1IF = 0; // lösche die Interrupt Flagge
                Zeitschritte_65ms ++;
                Hindernis = 0x03 & PORTA; //Kollision?
                Hindernis_rechts = RA0; // Kollision rechts?
                Hindernis_links = RA1; // Kollision links?
                Zeitschritte_Beschleunigung ++;

                if (Zeitschritte_Beschleunigung == 4)  {
                        Zeitschritte_Beschleunigung = 0;
                        if (Beschleunigung > 0) {
                                if (PWM_Tabelle[0] < pwm_max) PWM_Tabelle[0] ++;
                                 if (PWM_Tabelle[1] < pwm_max) PWM_Tabelle[1] ++;
                        }
                        else if (Beschleunigung < 0) {
                                 if (PWM_Tabelle[0] > pwm_min) PWM_Tabelle[0] --;
                                 if (PWM_Tabelle[1] > pwm_min) PWM_Tabelle[1] --;
                        }
                }

                if (Warnsignal)  {
                        if (Ton == 156) Ton_Schritt = -4; // Sirene zwischen 400 und 600 Hz
                        if (Ton == 100) Ton_Schritt = 4;
                        Ton += Ton_Schritt;
                        Zeitschritte_Warnsignal ++;
                        if (Zeitschritte_Warnsignal == 4) {  // LED Blinken
                                Zeitschritte_Warnsignal = 0;
                                if (RB4)  {
                                        RB5 = 1;
                                        RB4 = 0;
                                }
                                 else  {
                                        RB5 = 0;
                                        RB4 = 1;
                                }
                        }
                }
                else  {
                        RB4 = 0;
                        RB5 = 0;
                        Ton = 156;
                        Ton_Schritt = -4;
                }
        }

//Fortsetzung der Interrupt-Service-Routine auf der nächsten Seite
```

Tab. 4: C-Code dritter Teil, Beginn der Interrupt-Service-Routine

```
//... Fortsetzung der Interrupt-Service-Routine von der Vorseite

        if (PIR1bits.TMR2IF) {  // Prüft, ob die TIMER2 Überlauf Flagge gesetzt wurde & der Interrupt aktiv ist
                // Dieser Code-Teil wird alle 2 ms ausgeführt (10 Stufen -> PWM Frequenz 50Hz)
                // 1µs (Instruction Clock) * 125 (Überlauf gemäß PR2) * 16 (Prescaler)) = 2 ms

                PIR1bits.TMR2IF = 0; // lösche die Interrupt Flagge
                uint_fast8_t temp = 0;
                PWM_Zaehler++;
                if (PWM_Zaehler == 10) {  //Ausgänge gemäß der Variable Motor_Bits_Merker einschalten
                        PWM_Zaehler = 0;
                        temp = PORTB; //PortB einlesen
                        temp &= 0xF0; //unteres Nibble löschen
                        temp |= Motor_Bits_Merker; //Motor_Bits einfügen
                        PORTB = temp; //rausschreiben auf den Port B
                }

                if (PWM_Tabelle[0] == PWM_Zaehler)  {  // Ausgänge gemäß der PWM Tabelle [] abschalten
                        RB0 = 1;
                        RB1 = 1;
                }

                if (PWM_Tabelle[1] == PWM_Zaehler)  {  // Ausgänge gemäß der PWM Tabelle [] abschalten
                        RB2 = 1;
                        RB3 = 1;
                }
        }
}
```

Tab. 5: C-Code vierter Teil, Ende der Interrupt-Service-Routine

Computing

Economatics BBC-Buggy mit moderner Elektronik im Linien-Labyrinth

Dirk Uffmann, Roland Enzenhofer

Die englische Firma Economatics brachte 1983 den BBC-Buggy heraus, der damals von einem Heimcomputer gesteuert wurde. Hier zeigen wir eine bezüglich der Elektronik modernisierte Variante im Linien-Labyrinth mit einer Aufgabenstellung, die auch damals schon bearbeitet und in einem Fernsehbeitrag von BBC vorgestellt wurde.

Einleitung

Nachdem Dirk Fox unseren vorhergehenden Beitrag in dieser Ausgabe der ft:pedia zum fischertechnik PIC-Buggy von Economatics aus dem Jahr 2002 gelesen hatte, teilte er uns seine Anregung mit, diesen Fahrroboter doch als interessante Aufgabe ein Linien-Labyrinth lösen zu lassen. Wir sinnierten zunächst darüber, den kleinen Buggy um die nötigen Funktionen zu erweitern, also Rad-Encoder und einen Liniensensor hinzuzufügen. Doch dann kam uns die Idee, den BBC-Buggy von 1983 nachzubauen und mit moderner Elektronik und einem Display auszustatten, denn der BBC-Buggy von 1983 hat das von ihm erkundete Linien-Labyrinth auch auf dem Display des Heimcomputers angezeigt. Das kann man in dem BBC-Fernsehbeitrag „Making the Most of the Micro (8): Everything Under Control“ sehen.

Es gibt interessante Seiten zum BBC-Buggy im Netz. Besonders gefallen mir die Seiten von Neil Fazakerley, wo man sogar eine komplette Bauanleitung des BBC-Buggies findet, und der Blog von Dirk Fox zum Thema „Wie weit fischertechnik vor 32 Jahren seiner Zeit voraus war“.

Aufbau des Modells

Roland hat sich sogleich begeistert an den Aufbau des Modells gemacht und es nach München gesendet (Abb. 1). Seine umfangreiche Fischertechnik-Sammlung hatte fast alle benötigten Teile vorrätig, sogar die benötigten Schrittmotoren (Abb. 2). Das einzig fehlende Spezialteil, das Lager für die Rollkugel hinten, hat Roland auf seinem 3D-Drucker angefertigt (Abb. 3 und 4). Später hat er zusätzlich noch einen kompletten BBC-Buggy im Originalzustand erworben.

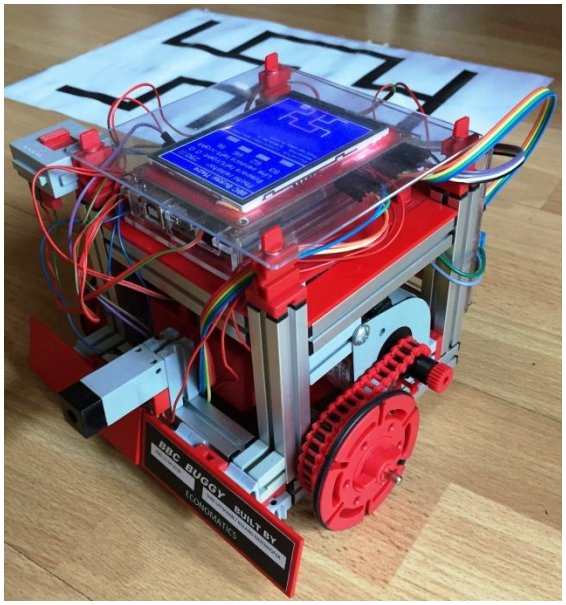


Abb. 1: Nachbau des BBC-Buggies

Abb. 2: Schrittmotoren

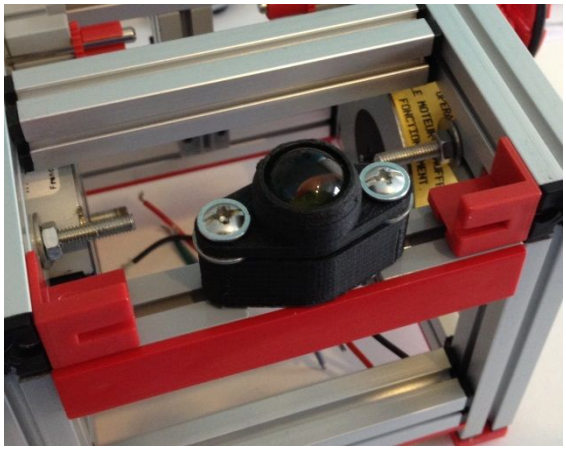


Abb. 3: Kugel-Lager aus Rolands 3D-Drucker mit Glasmurmel (wird noch ersetzt durch eine Stahlkugel mit 16 mm Durchmesser)

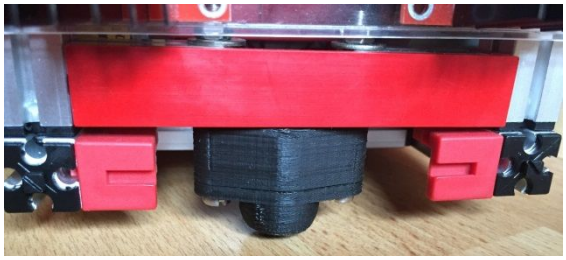


Abb. 4: Kugel-Lager im Fahrbetrieb

Steuerelektronik

Über einen Power-MOSFET (Metall-Oxid-Halbleiter-Feldeffekttransistor) wird der gesamte Versorgungsstrom des Systems ein- und ausgeschaltet. Abb. 5 zeigt die Platine für die Stromverteilung zu den einzelnen Modulen. Der Buggy zieht mit laufenden Schrittmotoren und dem Display fast 2 A Strom. Würde zum Schalten dieses Stromes direkt der Motorschalter eingesetzt, würde dessen Spannungsabfall mit bis zu 0,3 V von der Akkuspannung abgehen. Mit dem Power-MOSFET muss er nur noch den Steuerstrom von < 100 µA schalten.

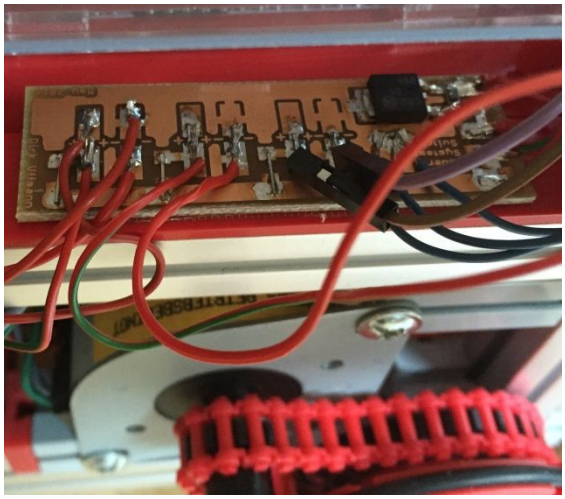


Abb. 5: Stromverteilungsplatine mit Power-MOSFET

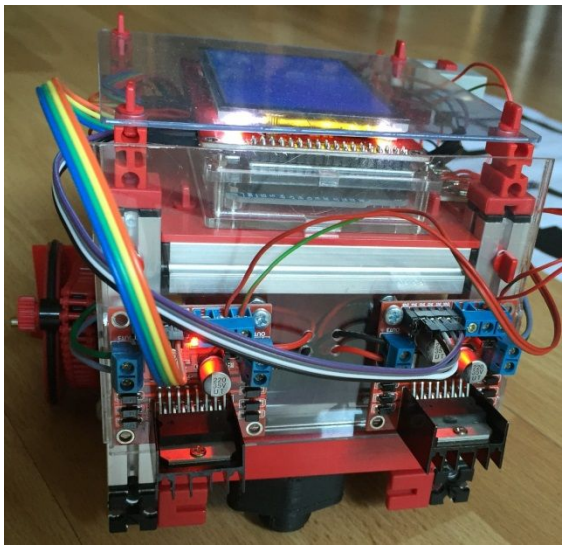


Abb. 6: Ansicht des Fischertechnik BBC-Buggies von hinten mit den beiden Motortreiberplatinen L298, die auf dem Zuschnitt einer CD-Hülle montiert sind.

Für die Steuerung wird ein Arduino Mega-Board mit dem ATMEGA 2560 und einem 240 x 400 Pixel Farb-Display mit einem ILI9327 als Controller eingesetzt. Über den PORT K des ATMEGA 2560 werden mittels zweier Motortreiberplatinen L298

(Abb. 6) die beiden Schrittmotoren des Antriebs gesteuert.

Die Schrittmotoren sind zweiphasige, bipolare Typen mit 7,5° Schrittwinkel von Crouzet (Bestell-Nr. 82920001). Wir steuern diese mit Halbschritten an. Es ergibt sich für jeden Motor an den vier Anschlüssen folgendes periodische Muster (4-Bit Nibble als hexadezimale Zahl) mit acht Halbschritten je Periode:

`0xE, 0xA, 0xB, 0x9, 0xD, 0x5, 0x7, 0x6.`

Tab. 1 zeigt die C-Funktion zur Ansteuerung der Schrittmotoren.

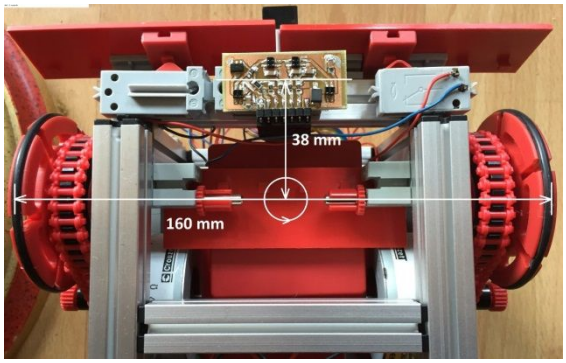


Abb. 7: Geometrie mit Rotationszentrum

Mit der Geometrie des Modells (Abb. 7) ergeben sich dann folgende Daten für die Bewegungssteuerung:

- Raddurchmesser (inkl. Gummiringe, 2 mm): 62 mm
- Radabstand: 160 mm
- Getriebeuntersetzung: 1:3
- Vollständige Radumdrehung: 96 Halbschritte
- Drehung um 90 Grad: 3•96•(160/4)/62 = 186 Halbschritte
- Weg je Halbschritt: π•62 mm/96/3 = 0,676 mm
- Weg 10 mm: 15 Halbschritte (1,5% Fehler)
- Mittlerer Abstand zwischen Linien-Sensor-Elementen und dem Buggy-Rotationspunkt zwischen den Rädern: 38 mm = 56 Halbschritte.

Über PIN F des ATMEGA2560 werden alle Sensoren eingelesen, das sind:

- zwei fischertechnik-Kollisionstaster
- ein fischertechnik-Fotowiderstand
- ein eigens konstruierter Linien-Sensor mit vier Sensor-Elementen ITR8307.

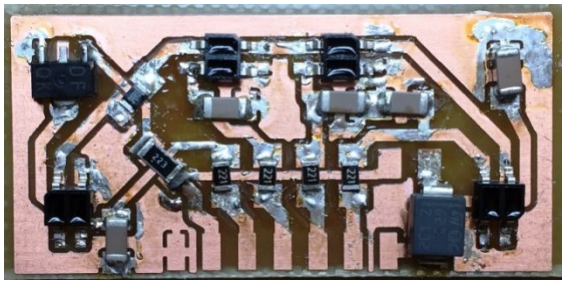


Abb. 8: Linien-Sensor mit vier Sensor-Elementen ITR8307

Abb. 8 zeigt den Linien-Sensor und Abb. 9 die zugehörige Schaltung. Die Sensor-Elemente sind so platziert, dass die beiden mittleren mit einem Abstand von 8 mm vollständig über der schwarzen Linie (Breite 15 mm) zu liegen kommen können.

Die Spurregelung nutzt die Messwerte dieser beiden Sensoren, um auf der Linie zu bleiben. Die äußeren Sensorelemente liegen 10 mm dahinter mit einem Abstand von 30 mm zueinander und sollen die Abzweigungen erkennen. Sie liegen deshalb nach hinten versetzt, weil sich so einfacher ermitteln lässt, ob bei einer Abzweigung auch noch eine Linie geradeaus weiterführt.

Die Schaltung und das Messprinzip sind angelehnt an den Polulu-Sensor QTR-8RC. Die Messung an den Sensorelementen funktioniert so:

- An einem digitalen Ausgangs-Pin am Mikrocontroller wird der 10 nF-Kondensator am Fototransistor zunächst auf ca. 5 V aufgeladen, Ladezeit > 50 µs.
- Dann wird der Pin auf Eingang umgeschaltet (ohne Pull-up Widerstand) und die Zeitmessung für die Entladung gestartet. Dazu wird in einer Schleife für

maximal eine Millisekunde abgefragt, ob der Pin von *high* auf *low* gewechselt hat; wenn ja, wird der zugehörige Zeitpunkt (Timerwert) abgespeichert und als Messergebnis genutzt.

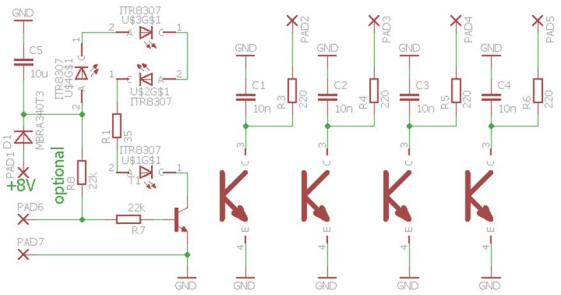


Abb. 9: Schaltung des Linien-Sensors

Der Widerstand R8 ist optional und kann weggelassen werden. Man braucht ihn nur, wenn die Infrarot-LEDs ständig leuchten sollen (z. B. weil man einen digitalen Pin zum Einschalten der LEDs einsparen möchte). Der bipolare Transistor ist vom Typ BCX54-16 (Stromverstärkung 250 typisch). Die LEDs werden mit 50 mA betrieben und alle 5 ms für 1 ms eingeschaltet (Tastverhältnis 20%). In diesem Pulsbetrieb könnte man die LEDs auch mit bis zu 250 mA betreiben; dazu müsste man den Widerstand R1 von 35 Ω auf 7 Ω und den Widerstand R7 von 22 kΩ auf 4,7 kΩ reduzieren (den optionalen Widerstand R8 darf man dann nicht einbauen).

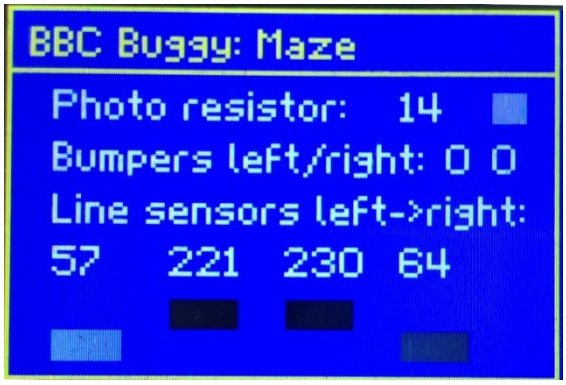


Abb. 10: Ausschnitt des Displays mit den Messwerten. Unten die Messwerte des Linien-Sensors, die auch als Grauwerte dargestellt sind. Hier ist der Sensor mittig über einer schwarzen Linie platziert.

Der Timer läuft in Schritten von 4 µs. Dann erhält man mit einem 8-bit-Timer Werte zwischen ca. 60 auf weißem Papier und 250 (Limit bei 1 ms) auf schwarzem Papier. Tab. 2 zeigt den für die Messroutine verwendeten C-Code. Abb. 10 zeigt den Ausschnitt des Displays mit den Messwerten.

Labyrinth

Wie das Original von 1983 erkundet der Buggy ein Linien-Labyrinth bis zum Zielpunkt und berechnet danach den kürzesten Pfad zwischen Start und Ziel. Diesen Pfad fährt er dann vom Zielpunkt zurück bis zum Startpunkt (d. h. statt links wird rechts abgebogen und umgekehrt). Das Labyrinth wurde mit der Freeware *inkscape* erstellt, in Größe A3 ausgedruckt und mit Tesafilm auf dem Boden festgeklebt (Abb. 11).

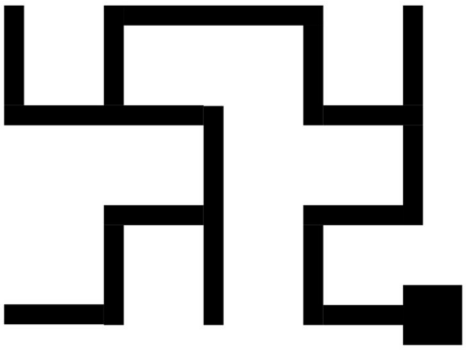


Abb. 11: Verwendetes Labyrinth: Der Zielpunkt ist durch das große schwarze Rechteck markiert (Linienbreite: 15 mm, Raster für die Abzweigungspunkte: 75 mm)

Am Anfang des Labyrinths sollte der Buggy mit dem Liniensensor in etwa mittig auf einer schwarzen Linie stehen und mindestens einer der Sensoren vollständig schwarz und ein anderer vollständig weiß sehen. Zu Beginn wird nämlich eine Kalibrierung durchgeführt. Dazu wird der Schwellenwert, der für die Mustererkennung zwischen schwarz und weiß unterscheidet, genau mittig zwischen den maximalen und den minimalen Messwert gelegt.

Für die Erkundung des Labyrinths auf dem Hinweg vom Startpunkt zum Zielpunkt

gelten an den Abzweigungen folgende Prioritäten:

1. Links abbiegen (L = *Left* = 1)
2. Geradeaus fahren (S = *Straight* = 2)
3. Rechts abbiegen (R = *Right* = 3)
4. Zurück fahren (B = *Back* = 4)

Die an den Abzweigungspunkten gewählten Wege für die Weiterfahrt werden in einem Feld *track [intersection]* gespeichert und stehen später für die Berechnung des kürzesten Pfades *track_reduced [intersection]* zur Verfügung.

Nachdem eine Abzweigung erkannt wurde, fährt der Buggy noch so weit geradeaus vor, bis das Zentrum seiner Drehbewegung über dem Abzweigungspunkt zu liegen kommt. Erst dann führt er die Drehung aus.

Sobald der Buggy den Zielpunkt erreicht hat, berechnet er aus dem gespeicherten Hinweg den kürzesten Pfad vom Startpunkt zum Zielpunkt. Dafür werden schrittweise ein Element des Hinweges *track []* in den reduzierten Pfad *track_reduced []* kopiert und dann die jeweils letzten drei Elemente des reduzierten Pfades *track_reduced []* überprüft und ggf. nach folgendem Schema ersetzt:

LBR → B	143 → 4
LBS → R	142 → 3
RBL → B	341 → 4
SBL → R	241 → 3
SBS → B	242 → 4
LBL → S	141 → 2

Dafür wird der in Tab. 3 dargestellte C-Code genutzt. Abb. 12 zeigt den Ausschnitt des Displays mit der Darstellung des erkundeten Labyrinths in weiß und dem kürzesten Pfad in rot. Oben sind auch die Pfade *track []* und *track_reduced []* als Buchstabenfolgen dargestellt.

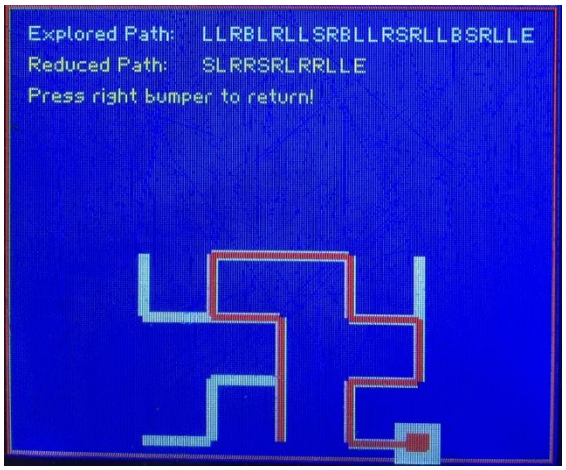


*Abb. 12: Display mit dem erkundeten Labyrinth (**weiß**) und dem kürzesten Pfad (rot)*

Video

In diesem Video kann die Fahrt des BBC-Buggies im Labyrinth bewundert werden.

```
//*****************************************************************************
// Function to move the buggy with halfsteps of the stepper motors
// turns the step motors of the robot by a number of steps in the specified directions
// dir = 1 forward, dir = 0 stop, dir = -1: backward
//*****************************************************************************
void move (uint8_t steps, int8_t dir_left, int8_t dir_right) {
        const uint8_t phase_right[8] = {0xEF, 0xAF, 0xBF, 0x9F, 0xDF, 0x5F, 0x7F, 0x6F};
        const uint8_t phase_left[8] = {0xFE, 0xFA, 0xFB, 0xF9, 0xFD, 0xF5, 0xF7, 0xF6};
        uint8_t phase_index_left = 0;
        uint8_t phase_index_right = 0;
        while (steps) {
                while (collision_sensor_left || collision_sensor_right) sleep_mode(); //stop
                steps_left += dir_left;
                steps_right+= dir_right;
                phase_index_left = (uint8_t) steps_left & 0x0007;
                phase_index_right = (uint8_t) steps_right & 0x0007;
                PORTK = phase_left[phase_index_left] & phase_right[phase_index_right] ;
                wait_ms(10);
                steps--;
        }
}
```

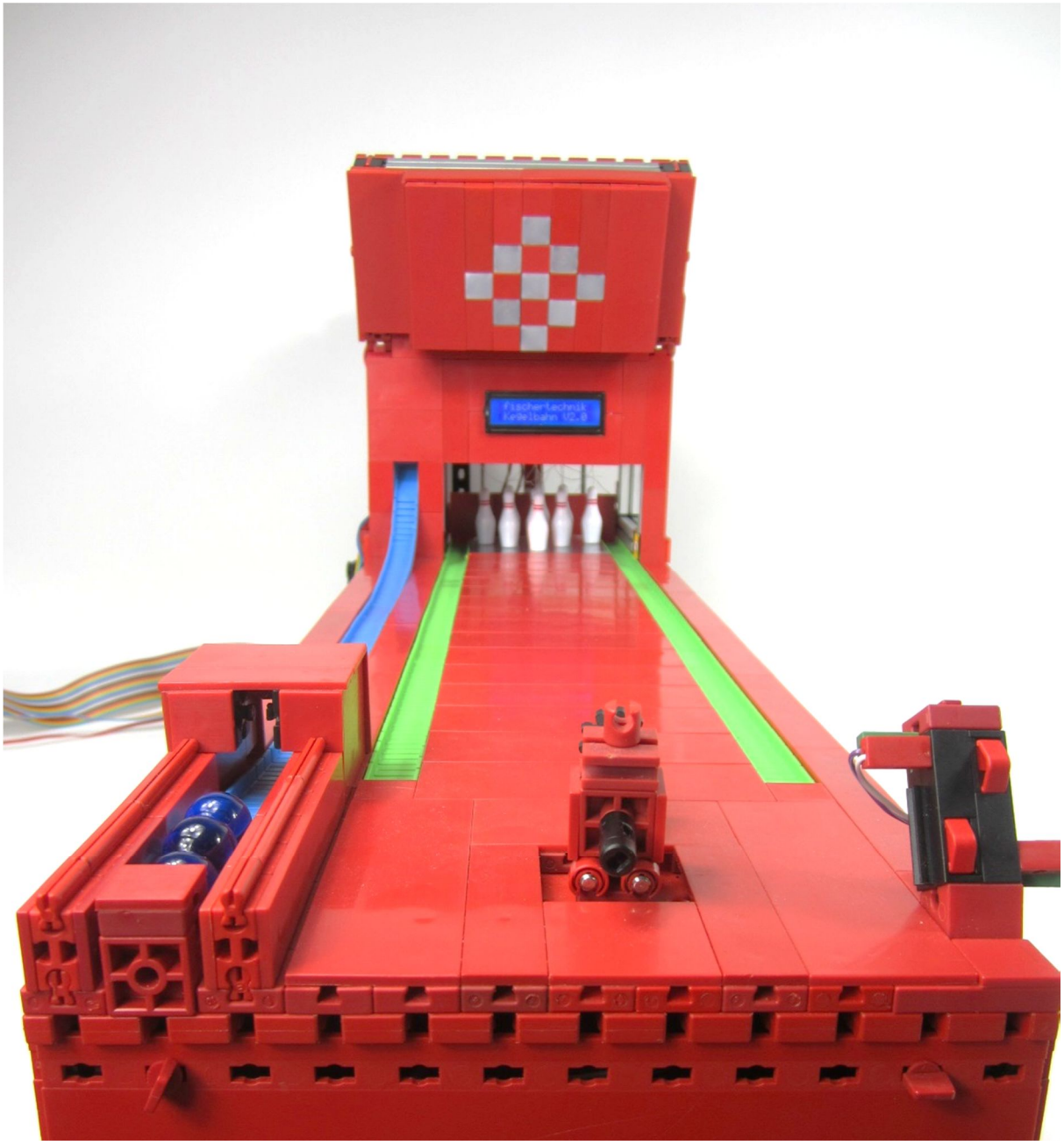
Tab. 1: C-Funktion zur Ansteuerung der Schrittmotoren

```
//*************************************************
// Measures the track sensors
// Function execution within ca. 1 ms
//*************************************************
uint8_t measure_track_sensor (void) {
        uint8_t status_input = 0;
        uint8_t timestamp = 0;
        uint8_t i = 0;
        uint8_t flag_to_measure = 0x2F; //mark inputs for measurement
        uint8_t pattern = 0;
        light_sensor = 250;
        while (i<4) {
                track_sensor [i] = 250; //init with max value for time-out
                i++; }
//switch on pullups on the tracking sensors and the light sensor input lines to charge the sensor lines
        PORTF |= 0x2F;
// switch sensor lines to output to charge sensor lines to high (tau = 220 Ohm * 47 nF = 10 µs)
        DDRF |= 0x2F;
        TCNT0 = 0;
        while (TCNT0 < 40); //wait for 160µs
        TCNT0 = 0; // 4 µs per timer step, i.e. 64 system clocks, max value 250 i.e. 1 ms
        DDRF &= 0xD0;
//switch off pullups on the sensor input lines to discharge the sensor lines via Phototransistor
        PORTF &= 0xD0;
        PORTF |= _BV(PF4); //pulse IR LEDs
        do {
                status_input = PINF; //read sensor lines
                timestamp = TCNT0;
                if (flag_to_measure & 0x01) {
                        if (!(status_input & 0x01)) {
                                track_sensor[3] = timestamp;
                                flag_to_measure &= ~(0x01); } }
                if (flag_to_measure & 0x02) {
                        if (!(status_input & 0x02)) {
                                track_sensor[2] = timestamp;
                                flag_to_measure &= ~(0x02); } }
                if (flag_to_measure & 0x04) {
                        if (!(status_input & 0x04)) {
                                track_sensor[1] = timestamp;
                                flag_to_measure &= ~(0x04); } }
                if (flag_to_measure & 0x08) {
                        if (!(status_input & 0x08)) {
                                track_sensor[0] = timestamp;
                                flag_to_measure &= ~(0x08); } }
                if (flag_to_measure & 0x20) {
                        if (!(status_input & 0x20)) {
                                light_sensor = timestamp;
                                flag_to_measure &= ~(0x20); } }
        } while (timestamp < 250);
        PORTF &= ~(_BV(PF4)); //switch off IR LEDs

//      detect pattern
        pattern = 0;
        i=4;
        while (i)
        {
                i--;
                if (track_sensor [i] > threshold) pattern |= 1 << i; }
        return pattern;
}
```

Tab. 2: C-Code der Messroutine für den Linien-Sensor

```
//      copy first two items from original track
        track_reduced[1] = track[1];
        track_reduced[2] = track[2];
        i = 3;
        j = 3;
        while (track[i]) {
//              copy one further item from original track
                track_reduced[j] = track[i];
//              check the last three items for reduction
                if (track_reduced[j-1] == 4) {
                        if (track_reduced[j-2] == 1) {
                                switch (track_reduced[j]) {
                                        case 1:
                                                track_reduced[j-2] = 2;
                                                j--;
                                                break;
                                        case 2:
                                                track_reduced[j-2] = 3;
                                                j--;
                                                break;
                                        case 3:
                                                track_reduced[j-2] = 4;
                                                j--;
                                                break;
                                        default: //no reduction possible
                                                j++;
                                }
                        }
                        else if (track_reduced[j-2] == 2) {
                                switch (track_reduced[j]) {
                                        case 1:
                                                track_reduced[j-2] = 3;
                                                j--;
                                                break;
                                        case 2:
                                                track_reduced[j-2] = 4;
                                                j--;
                                                break;
                                        default: //no reduction possible
                                                j++;
                                }
                        }
                        else if (track_reduced[j-2] == 3) {
                                switch (track_reduced[j]) {
                                        case 1:
                                                track_reduced[j-2] = 4;
                                                j--;
                                                break;
                                        default: //no reduction possible
                                                j++;
                                }
                        }
                        else j++;
                }
                else j++;
                i++;
        }

//      delete reduced items (max two may occur)
        track_reduced[j] = 0;
        track_reduced[j+1] = 0;
```

Tab. 3: C-Code zur Ermittlung des kürzesten Pfades



fischertechnik-Kegelbahn (Foto: Dirk Wölffel)