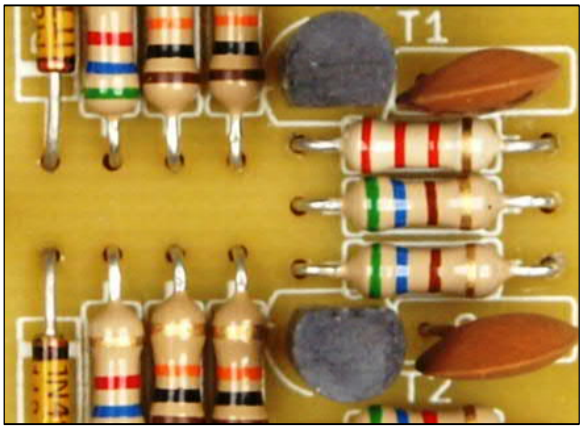
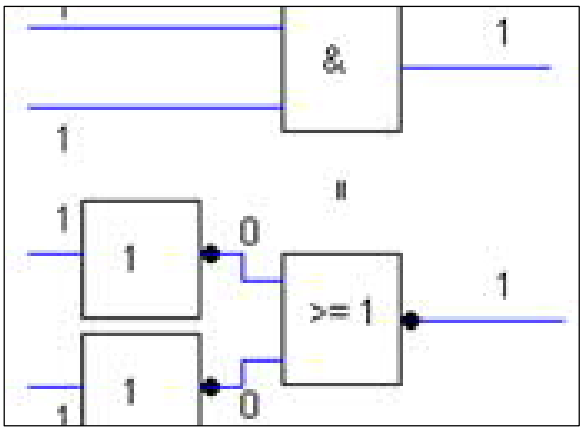
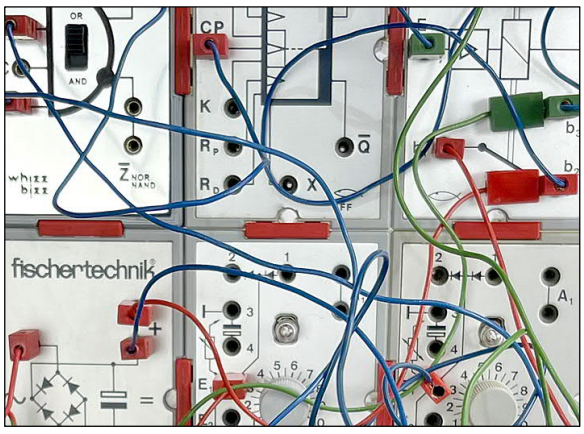
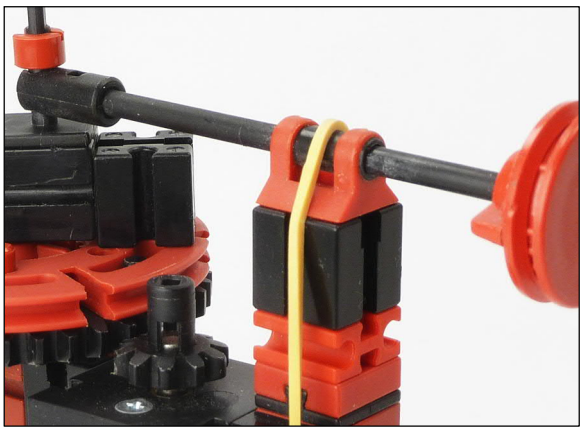
ft:pedia

Heft 4/2021









Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

Bildung bildet

Dirk Fox, Stefan Falk

Es gab eine Zeit in unserer Jugend (also im vergangenen Jahrtausend), in der Bildung als hohes Gut galt – das Versprechen eines erfüllten und besseren Lebens. Unsere Kinder werden jetzt sagen: „Jaja, damals, als ihr noch in Höhlen gelebt habt…“.

Also gut, lassen wir uns einmal auf diese Sicht ein. Wissen wird heute im Internet „zusammengeklickt“, Youtube-Videos zeigen uns, worauf es im Leben ankommt und Lesen wird überbewertet – Lernen muss Spaß machen, sonst hat die Pädagogik versagt. Aber – funktioniert das so?

Erwächst aus zusammengeklickten Informationen tatsächlich Wissen? Ein gut gemachtes Video kann helfen, einen Zusammenhang zu vermitteln, aber hilft es auch, wenn es gilt, gänzlich neue Herausforderungen zu bewältigen? Und „Spaß“ – nun ja, man kann das 1x1 sicherlich mit einem Spiel aufwerten oder Sprache durch eine spannende Geschichte vermitteln, aber wer etwas wirklich beherrschen will, muss üben, üben, üben. Wer sich das nicht zumutet, lernt nichts.

Tun wir unseren Kindern einen Gefallen, wenn wir ihnen vorgaukeln, dass es einen anstrengungsbefreiten Weg zu Wissen gibt? Ich habe da meine Zweifel. Lernen ist harte Arbeit, und darauf lässt man sich nur ein, wenn man einen Sinn dahinter sieht. Und was könnte der anderes sein als – Bildung?

Ein engagierter Verfechter dieser Überzeugung ist der Neurologe Manfred Spitzer. In seinem lesenswerten Buch „Die Smartphone-Epidemie“ stellt er zusammen, was wir heute über Lernen und lern- und bildungsfeindliche Einflüsse wissen. Wer die Kurzfassung möchte, dem sei sein Vortrag auf dem Wissensdurst-Festival am 07.10.2019 empfohlen. Darin weist er nach, dass Social Networks, Youtube und andere Mediennutzungen bei Kindern und Jugendlichen zu Aufmerksamkeitsstörungen sowie Kurzsichtigkeit führen und Lernprozesse behindern. Von Google und Co. profitieren nur Kinder (und Erwachsene), die bereits eine hohe Bildung haben – sie suchen gezielter und können gefundene Informationen einordnen und bewerten. Zugleich führt die immer umfangreichere Mediennutzung zu einem Rückgang des Bildungsniveaus und seit Ende der 90er Jahre zu einem Absinken des Intelligenzquotienten. Dabei war der IQ seit 1923 kontinuierlich gestiegen: in Deutschland bspw. um durchschnittlich drei Punkte je Jahrzehnt (Flynn-Effekt).

Wie in der Zeit, in der wir noch in Höhlen lebten, gilt auch heute noch: Bildung ist Voraussetzung für Weltverständnis. Und wie kann man das besser erfahren, als mit einem fischertechnik-Modell, das erst funktioniert, wenn wir es verstanden haben?

Mit der ft:pedia möchten wir einen kleinen Bildungsbeitrag leisten. Wir hoffen, dass ihr nach der Lektüre jeder Ausgabe ein wenig klüger seid als davor. Danke an die über 80 Autoren, die viel Zeit investieren, um ihr Wissen für euch aufzubereiten.

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?
Bescherung	24.12.2021	(fast) überall

Impressum

http://www.ftpedia.de

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

Autoren: Florian Bauer, Michael Biehl, Arnoud van Delden, Stefan Falk, Dirk Fox, Helmut Jawtusch, Peter Krijnen, Kurt Mexner, Thomas Püttmann, Harald Steinhaus.

Modell

fischertechnik-Band

Michael Biehl

Eigentlich hatte ich tolle Bandfiguren aus alten Lampen und allen möglichen Teilen entdeckt, die mir sehr gefielen. Upcycling nennt man das, glaube ich, jetzt auf neudeutsch. Allerdings auch zu enormen Preisen.

Angeregt von den Insekten und den Blumen von Rüdiger Riedel [1, 2, 3] kam ich auf die Idee, dass man da aus fischertechnik etwas zaubern könnte. Also sichtete ich eine Sortierkiste nach der anderen und stellte grob etwas zusammen. Erst war ich bei einer Kopfform angekommen, dann bei zwei Gitarrenspielern und dem Schlagzeuger. Aber je mehr ich wühlte, kam ich auf immer neue Charaktere mit anderen

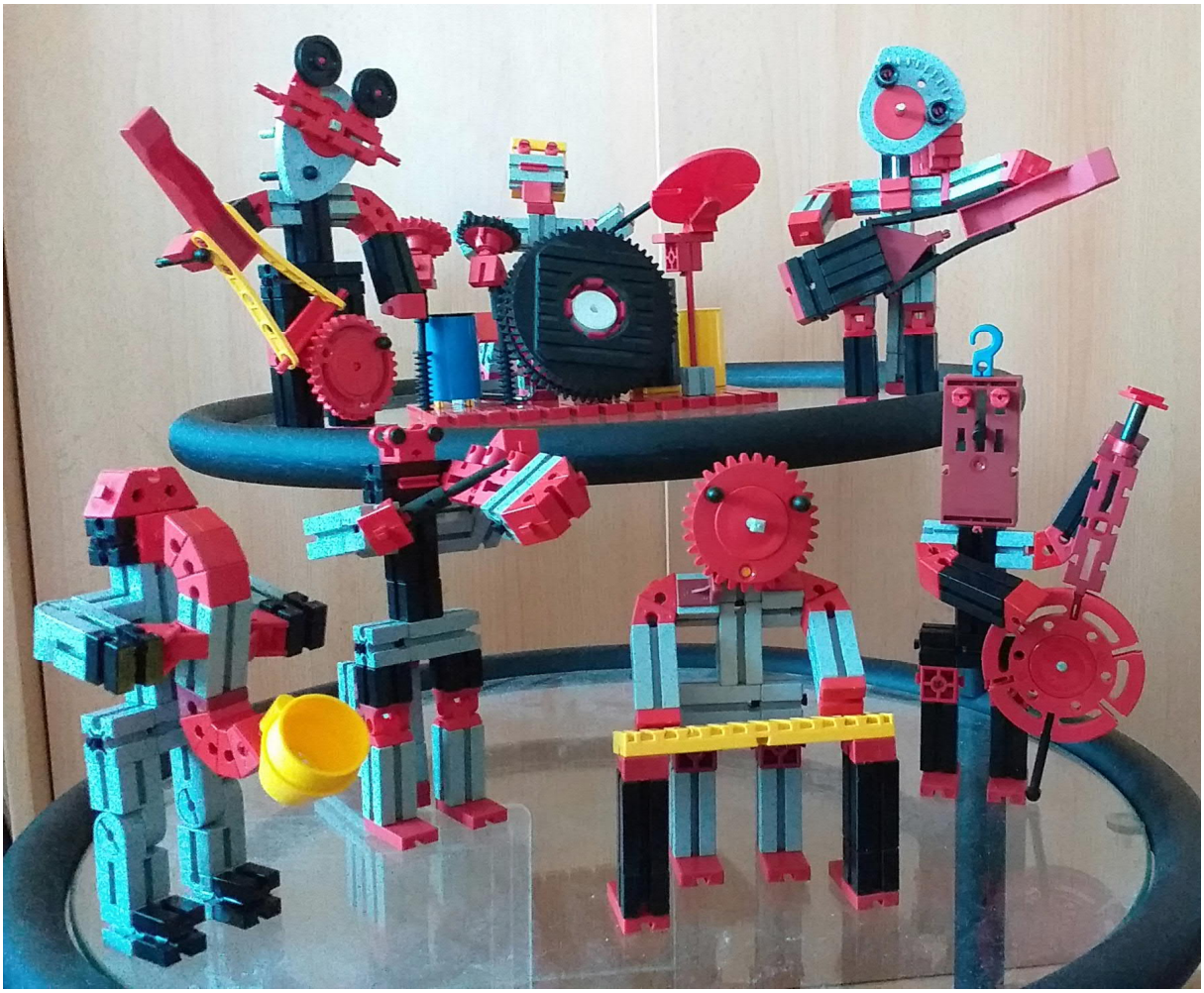


Abb. 1: Die komplette Band „sechs Seiten“

Gesichtern und Instrumenten. Jetzt ist es schon ein kleines Orchester geworden, das Platz in der Vitrine bekommen hat. Und es geht vielleicht noch mehr.

Die bereits gebauten Musiker zeigen die folgenden Abbildungen:

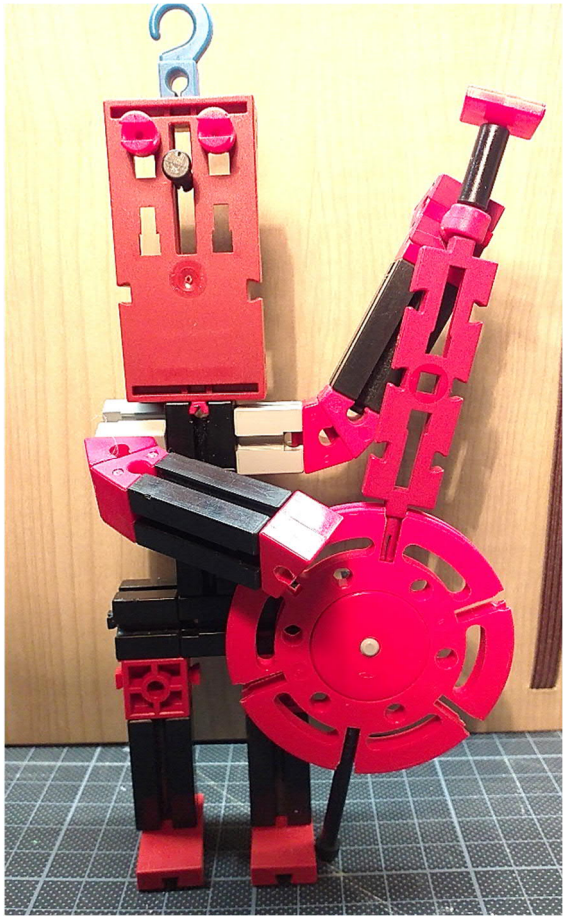


Abb. 2: Kontrabassist Question

Abb. 3: Schlagzeuger Roland

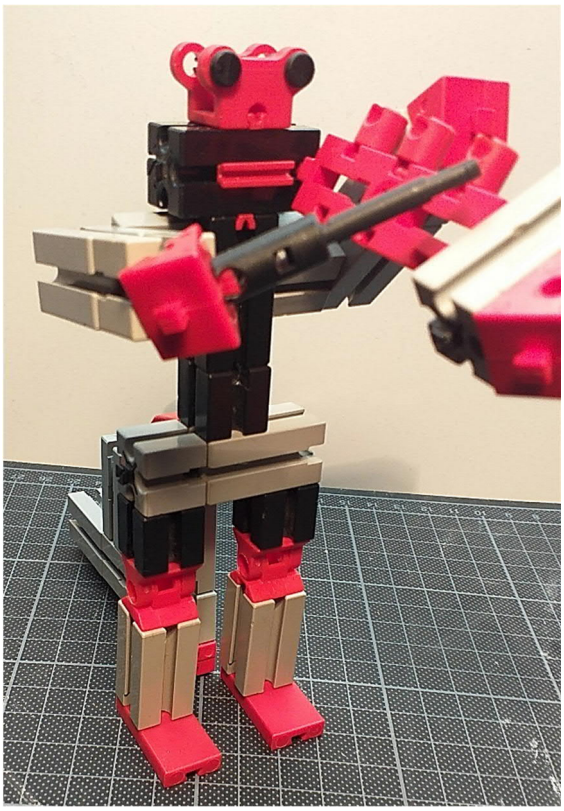


Abb. 4: Violinistin Mantis

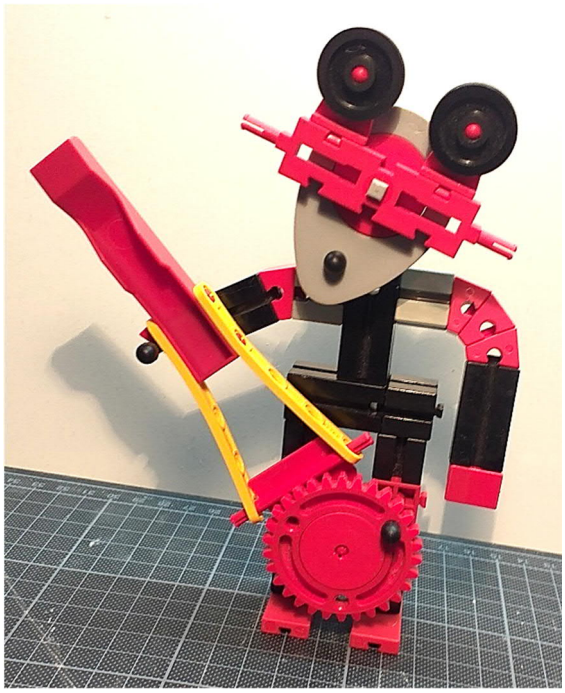


Abb. 5: Gitarrist Mouse

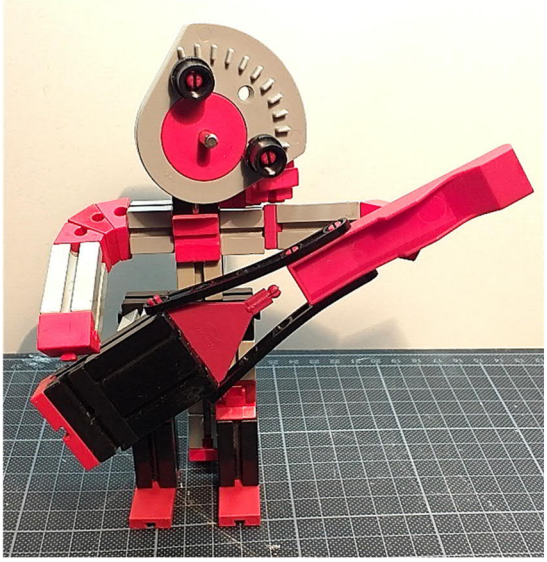


Abb. 6: Gitarrist Groot Marvel

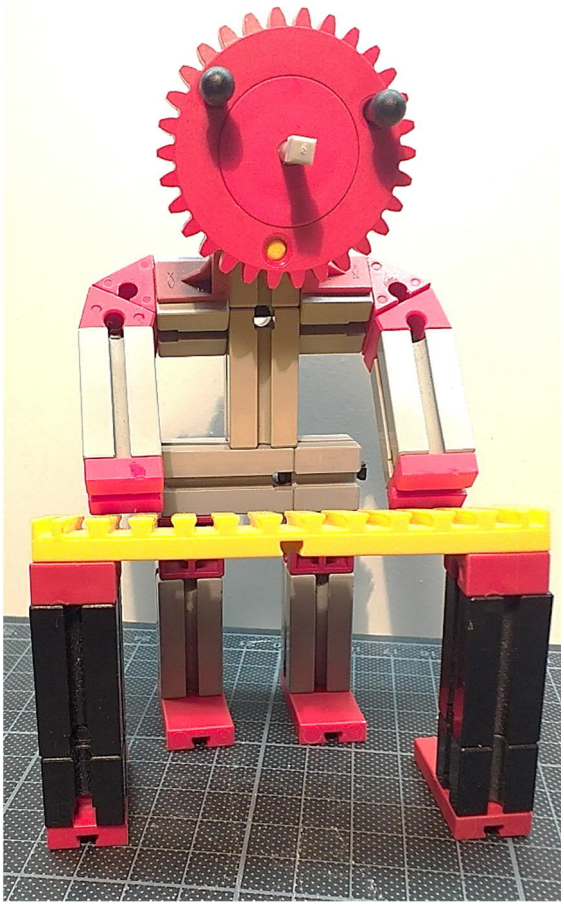


Abb. 7: Keyboarderin Sunny

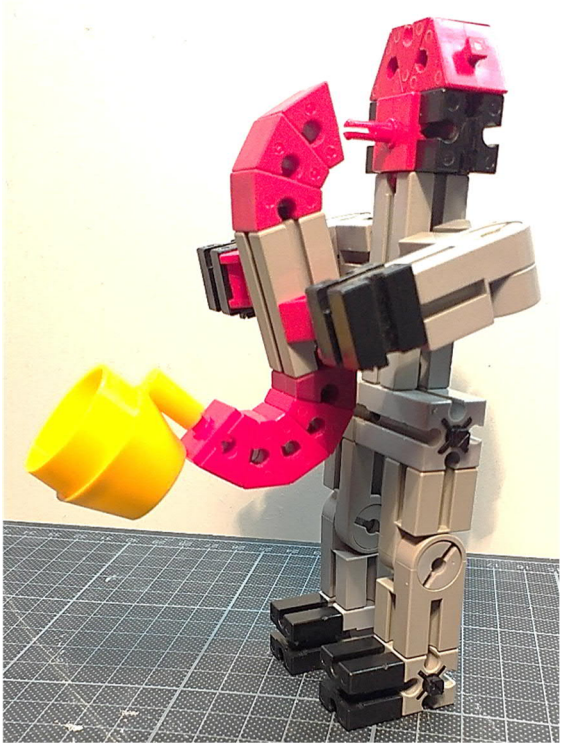


Abb. 8: Saxophonist Antoine

Referenzen

[1] Rüdiger Riedel: *Schmetterling*. ft:pedia 1/2021, S. 5–6.

[2] Rüdiger Riedel: *Getier im Frühling*. ft:pedia 1/2021, S. 13–19.

[3] Rüdiger Riedel: *Blumen*. ft:pedia 3/2021, S. 4–10.

Modell

Massagepistole

Thomas Püttmann

Muskeln können krampfen, sich verhärten, verkürzen oder sogar zusammenwachsen. Eine Massage lockert die Muskulatur und unterstützt den Lymphfluss. Viele Therapeuten, Sportler und Berufsmusiker setzen inzwischen Massagepistolen ein. Unsere fischertechnik-Massagepistole kann sich zwar nicht ganz mit den kommerziellen Geräten vergleichen, bringt aber doch einen spürbaren Lockerungseffekt.

Wer kennt das nicht: Nachdem man viele Stunden konzentriert und ohne größere Bewegung am Schreibtisch oder am Computer gesessen hat, hat sich die Nackenmuskulatur verkrampft. Nichts hilft in diesem Fall besser als gezielte ausgleichende Dehnung und Bewegung. Bisweilen ist das Problem jedoch so weit fortgeschritten, dass zusätzliche Maßnahmen sinnvoll sind.

Eine professionelle Massage kann eine deutliche Linderung bewirken. Allerdings steht sie nicht immer dann zur Verfügung, wenn man sie braucht.

Seit ein paar Jahren gibt es im Handel Massagepistolen zu kaufen, mit denen man schmerzende oder verhärtete Stellen mit relativ hohen Frequenzen bearbeiten kann.

Unsere fischertechnik Massagepistole erreicht eine solche hohe Schlagzahl nicht, kann aber trotzdem die Muskulatur spürbar lockern.

Zu beachten gibt es relativ wenig. Man sollte keinen großen Druck ausüben und sie möglichst nicht am Kopf, am Hals oder direkt auf Knochen anwenden.

Abb. 1: Die Massagepistole aus fischertechnik

Bauanleitung

Beginne mit dem Schlagrad. Seine Achse ist in einem Baustein mit Bohrung gelagert. Ein Abstandsring sorgt für die richtige Höhe des Zahnrads Z40 (Abb. 2).

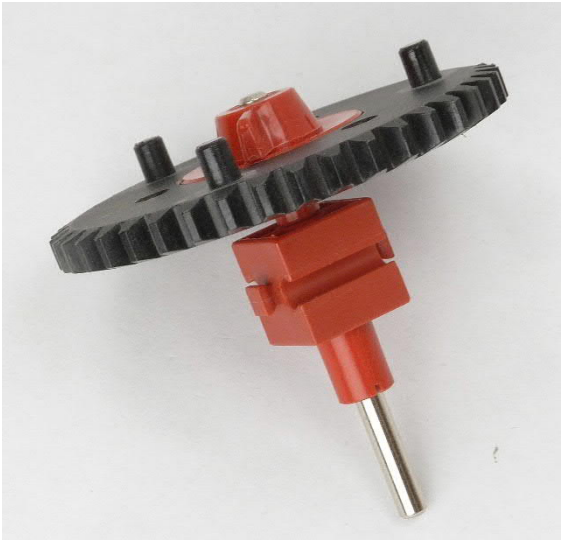


Abb. 2: Die Achse des Schlagrads mit Z40

Stecke auf das Z40 eine Drehscheibe 60 und sichere sie mit einem Klemmstift oder einem Verbindungsstopfen (Abb. 3).

Abb. 3: Die aufgesteckte Drehscheibe

Baue als nächstes die Exzentereinheit auf die Drehscheibe (Abb. 4).

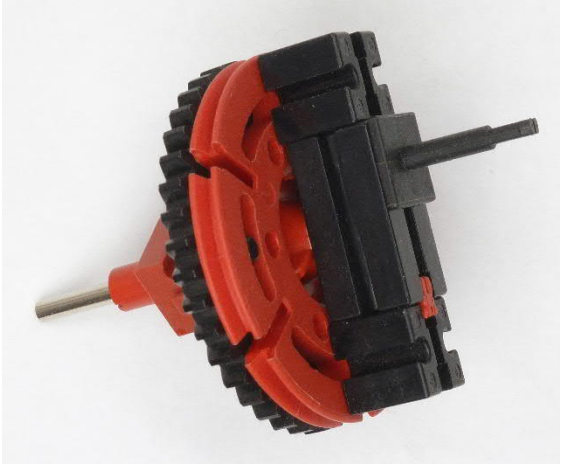


Abb. 4: Die Exzentereinheit

Zum Anbau an den XM-Motor benötigst du noch ein Zwischenstück (Abb. 5). Zusammen mit einer Bauplatte 5 mit drei Teilnuten sorgt es für ausreichend Stabilität im Dauerbetrieb.

Abb. 5: Das Zwischenstück

Auf der gegenüberliegenden Seite des XM-Motors bringst du die drehbare Führung des Schlägers an (Abb. 6).

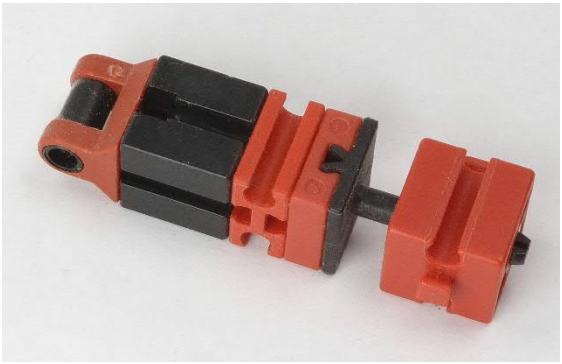


Abb. 6: Die drehbare Führung des Schlägers

Der Schläger selbst besteht aus einer Rastkupplung, einer Rastachse 90 und einer kompletten Nabe.

Sichere die drehbare Führung mit einem Gummiring. Ein Rastadapter oder eine Rastkupplung lösen sich im harten Einsatz zu oft von der Rastachse mit Platte.

Setze neben den Baustein mit Bohrung, der die drehbare Führung lagert, einen Baustein 15 und verbinde beide mit einem Verbindungsstück 15. So bekommt die drehbare Führung ausreichend Stabilität.

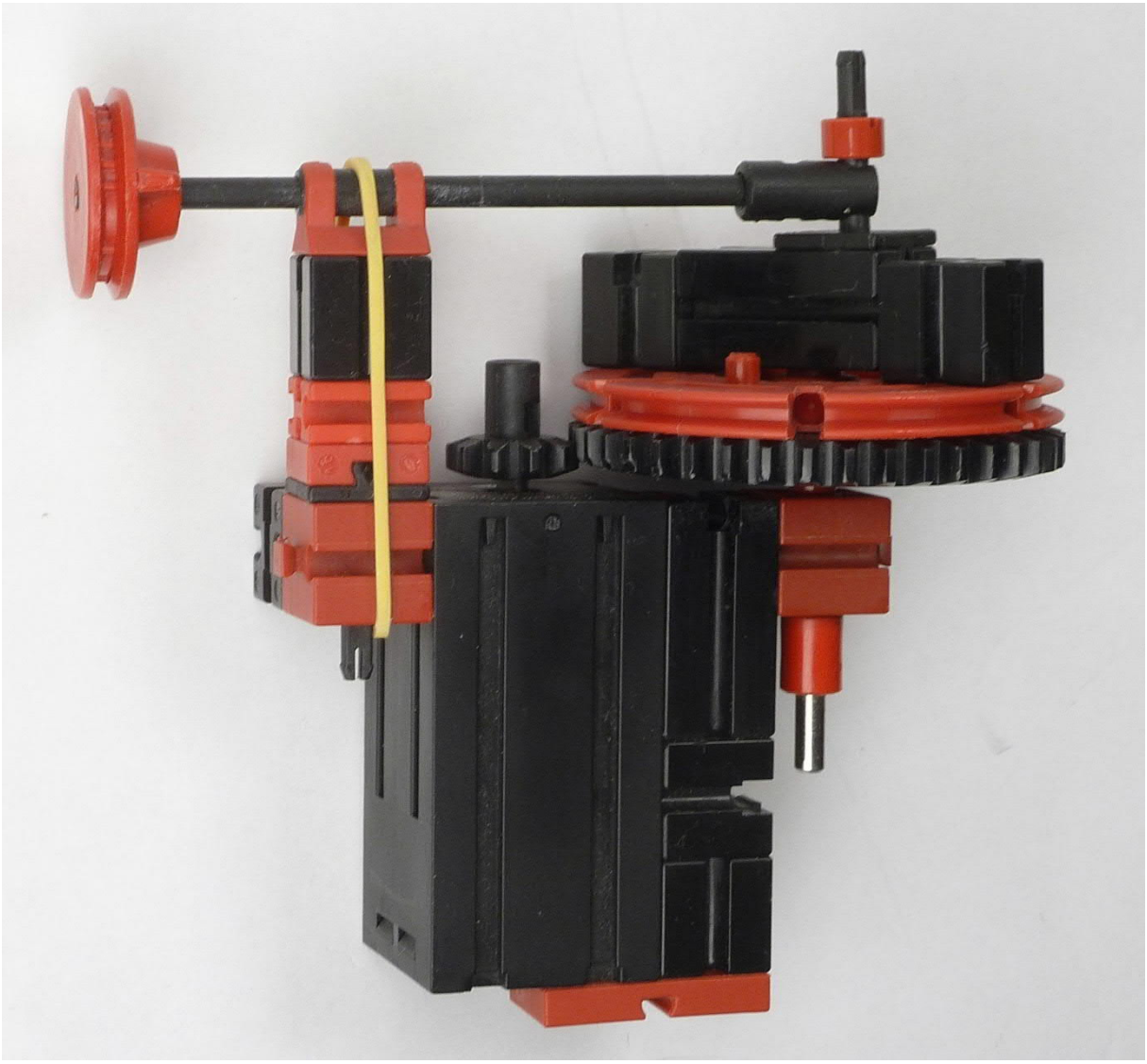


Abb. 7: Die fertige Massagepistole

Modell

Citroën DS régénérée

Harald Steinhaus

Die Citroën Déesse hatte ich vor fast 10 Jahren schon mal gebaut, siehe im Bilderpool unter [1]. Das Kürzel „DS“ wird im Französischen wie „Déesse“ ausgesprochen und bedeutet dann „die Göttin“. Götter werden nicht „reloaded“, und mangels Sterblichkeit auch nicht wieder geboren. Dann eben „régénérée“. Und jetzt liegt der Ball bei Citroën, PSA oder Stellantis: es wird Zeit für eine neue DS, mit Elektroantrieb!

In ihren Grundzügen geht die DS auf die Citroën Traction Avant („Frontantrieb“, vom Schnitt her ein „Gangsterauto“ im Stil der 1930er Jahre, [2]) zurück, so unterschiedlich das Äußere erscheinen mag: Die Traction Avant hat eine angetriebene Vorderachse mit einzeln aufgehängten Radschwingen, die von vorstehenden „Hörnern“ der Karosserie getragen werden. Der Motor liegt hinter der Vorderachse, das Getriebe davor.

Die Déesse war bahnbrechend beim hydropneumatischen Fahrwerk: Jedes Rad wird mit einem Zylinder abgestützt, der teilweise mit Öl gefüllt ist und darüber eine mit Stickstoff gefüllte Kammer aufweist. Je nach Ölmenge liegt die Zelle höher oder tiefer, was zur Niveauregulierung bei variabler Beladung ausgenutzt wird. Bei starker Beladung wird der Stickstoff stärker komprimiert und die Federung wird härter. Weitere Erläuterungen findet man unter [3]. Außerdem kommt die Déesse beim Radwechsel ohne Wagenheber aus: man hebt die Déesse komplett an, stellt auf einer Seite einen Bock unter und senkt sie wieder ab: nun schweben zwei Räder in der Luft und können getauscht werden.

Dieser Artikel zeigt Details und Werdegang der „neuen“ Déesse als eine Reihe von Bildern mit zugehörigen Beschreibungen. Wie das Vorbild hat das Modell Frontantrieb, Servolenkung, mitlenkende Scheinwerfer, Blinker vorn und hinten (unter der Dachkante), rote Rückleuchten in der Stoßstange, Türen rund herum, Ledersitze und

Abb. 1: Die Déesse 2

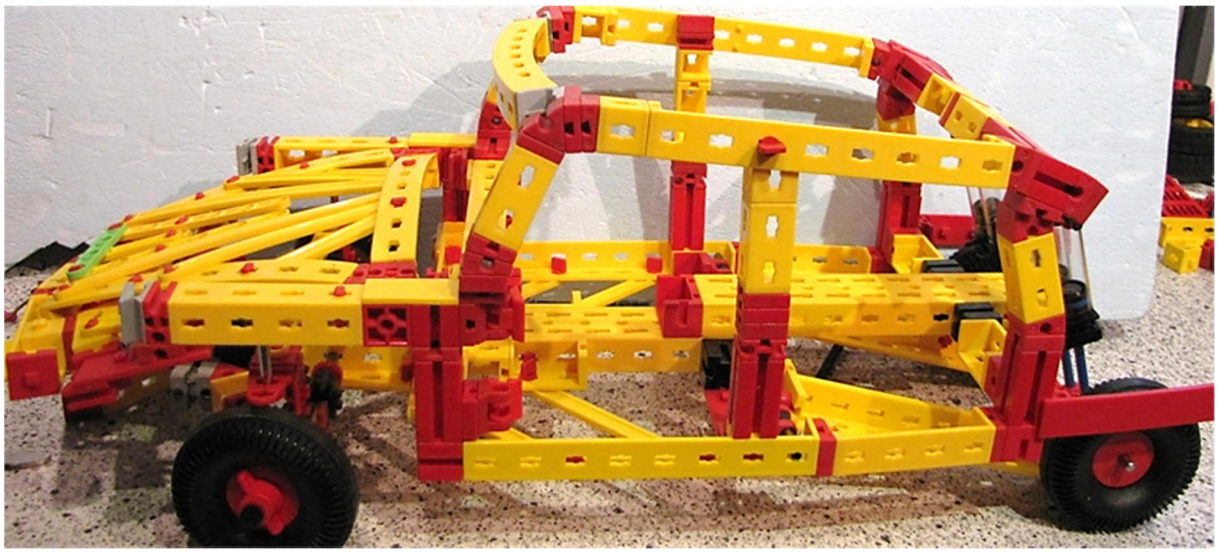


Abb. 2: Erste Studie

pneumatisch abgestütztes Fahrwerk vorn und hinten. Auch die Spaltmaße sind sehr originalgetreu ausgefallen.

Den Anfang der neuen Déesse machte eine Art Drahtgerüst, das die Grundzüge der Geometrie andeutet. Die Motorhaube ist geblieben, nur jetzt mit Streben beplankt. Hier liegt der Motor vor der Vorderachse, es gibt vorne noch keine Hubmechanik, der mittlere Längsträger hat eine unschöne Stufe, die Hinterradschwingen sind ziemlich lang ausgefallen und die P-Zylinder stützen sich steil nach oben an der hinteren Dachkante ab. Da wird sich noch einiges ändern.

Vor der Vorderachse eingebaut sind, in Abb. 4 von links nach rechts: Schalter für den Kompressor, Lenkservo, Antriebsmotor, Kompressor. Die Antriebswellen verlaufen durch Gelenksteine hindurch, die zur Hubmechanik gehören.

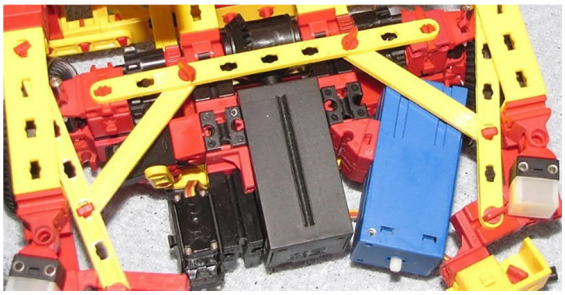


Abb. 4: Motorraum

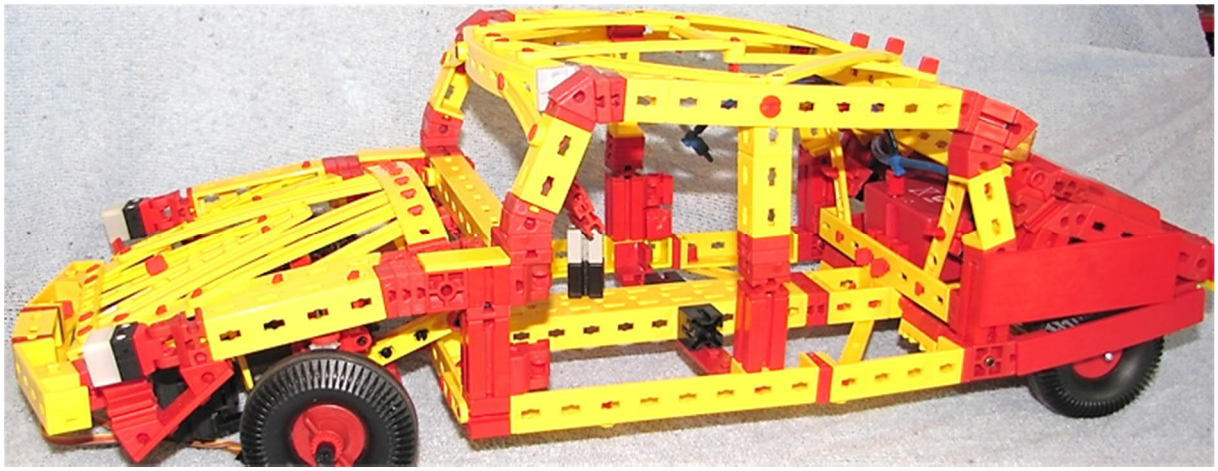


Abb. 3: Etwas weiter fortgeschritten

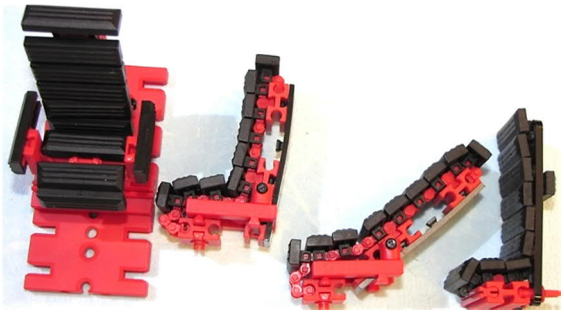


Abb. 5: Vordersitze

Abb. 6: Fast fertig

Abb. 7: Heck

Die Vordersitze (Abb. 5) haben verstellbare Rückenlehnen und sitzen auf einem Drehzapfen, um das Einsteigen zu erleichtern. Die starre Variante mit Armlehnen (links und rechts außen) stammt aus dem Fendt GTA reloaded [4].

Mit nunmehr fast durchgehend roten Winkelträgern und mit allen Türen ist die Déesse fast fertig (Abb. 6). Hinter der Stoßstange hat ein Überdruckschalter für die

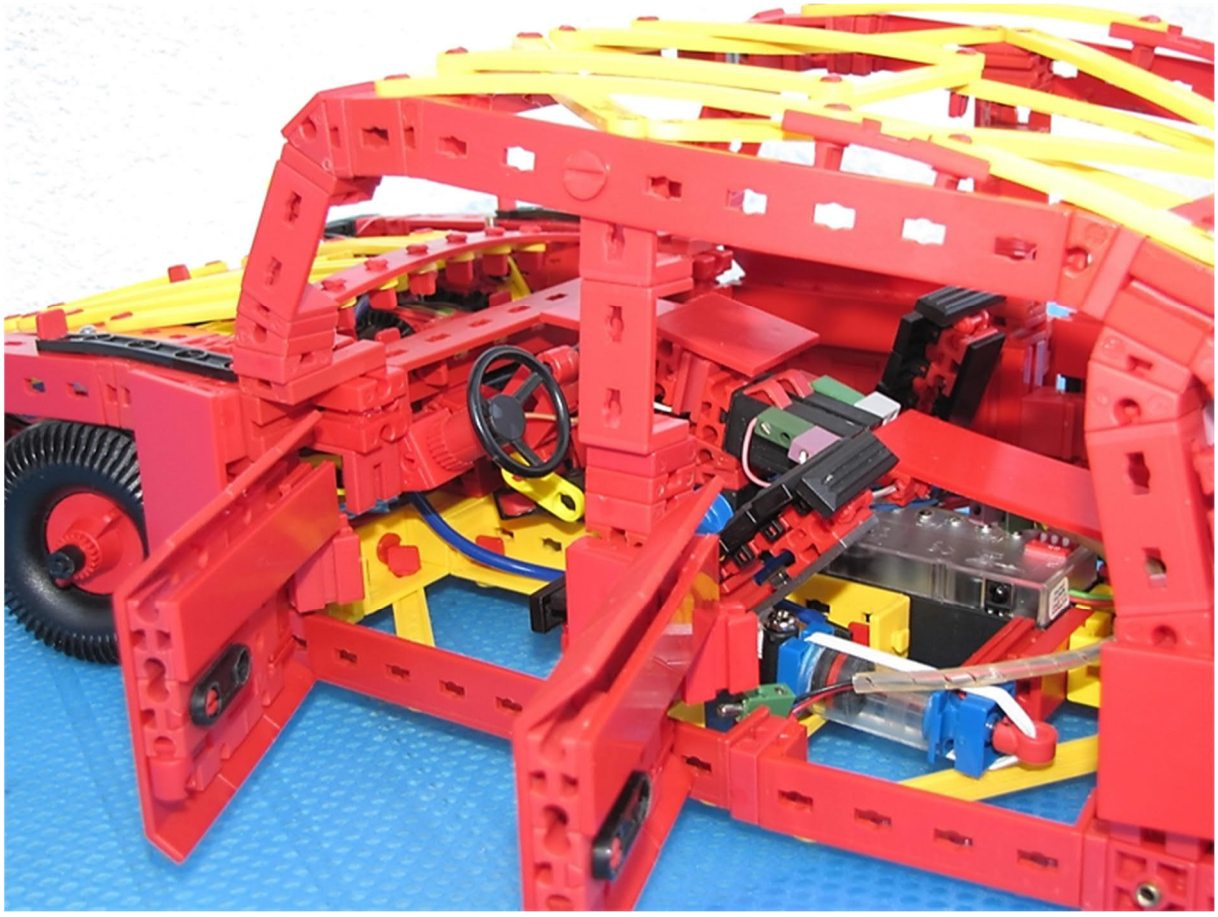


Abb. 8: Innenraum

Pneumatik Platz gefunden: ein P-Betätiger arbeitet gegen einen Gummizug und drückt einen Minitaster, wenn der Abschaltdruck erreicht wird. Die Scheinwerfer lenken noch nicht mit, die Nylonseile wollen nicht recht passen. Die Rücksitze fehlen auch noch.

Das Heck (Abb. 7) kann richtig weit angehoben werden. Das Dumme daran ist, dass dadurch die Nase tiefer gedrückt wird, und die vordere Hubmechanik nicht genug Arbeitsweg hat, um das wettzumachen. Die Rückleuchten sind liegend im gelben Winkelträger verbaut. Die Blinker sind da, wo sie hingehören: an der Dachkante (die hier etwas verzogen ist). Der Kofferraumdeckel ist sehr weit oben gelagert.

Die Türen (Abb. 8) haben Streben-15 zur Verriegelung und werden durch BSB-Grundplatten zusammengehalten. In der Mittelkonsole sieht man die Taster für den Blinker, das P-Handventil für die Fahrwerksabstimmung und die IR-Fernbedienung. Der P-Zylinder mit dem Gummizug dient nur als Anzeige für den Druck in der Pneumatik. Die Türen sind in schwarzen Knubbeln (163204 D4 mm Schlauchanschluss gerade) gelagert. Unter dem Lenkrad befindet sich der Lichtschalter.

Die beiden Platten 30 · 30 der Mittelkonsole bilden das Ende des Hebels, mit dem die Frontpartie mit Hilfe zweier kurzer P-Zylinder angehoben wird. Der Hebel ist vor der Fahrgastzelle gelagert und wird dann zu einem Parallelogramm ergänzt, an dem der ganze Antriebsblock befestigt ist.

Die IR-Fernbedienung (im schwarzen Gehäuse in Abb. 9 mittig) ist Teil des Längsträgers. Daneben ist der Hauptschalter angebracht. In der Mittelkonsole stecken die beiden kurzen P-Zylinder, die die Front anheben und absenken.

Die Bowdenzüge für die mitlenkenden Scheinwerfer sind an den Teilen der Lenkung angebunden.

Ganz hinten in Abb. 10 sieht man die roten Leuchten in der hinteren Stoßstange.

Die Bowdenzüge für die mitlenkenden Scheinwerfer sind noch nicht verbaut, die Nylonschnur baumelt noch „nur so“ herum.

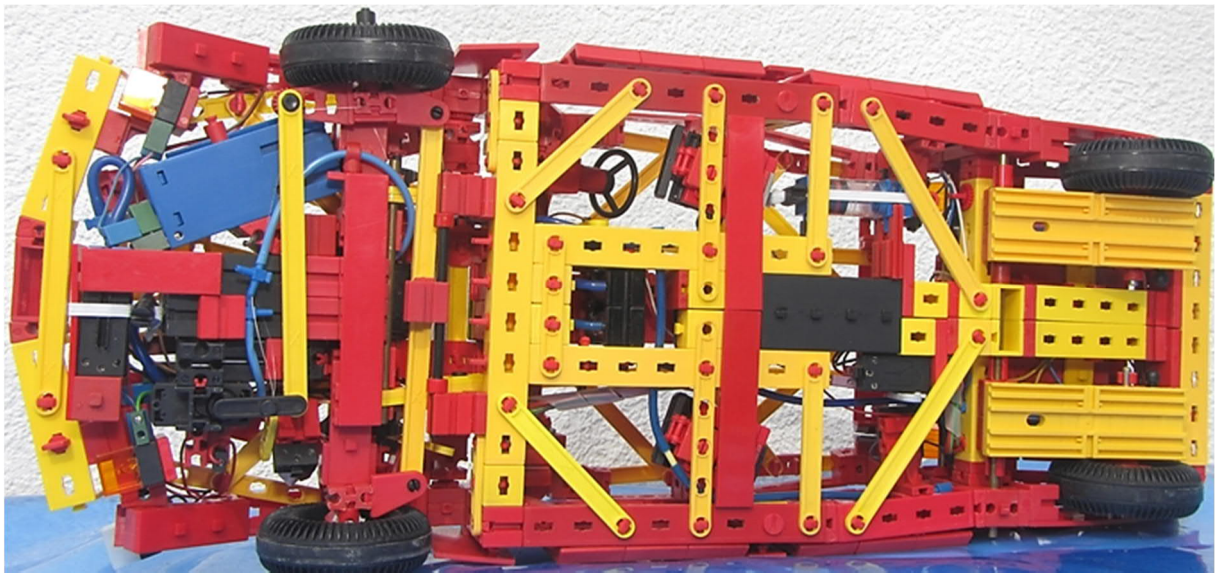


Abb. 9: Von unten (1)

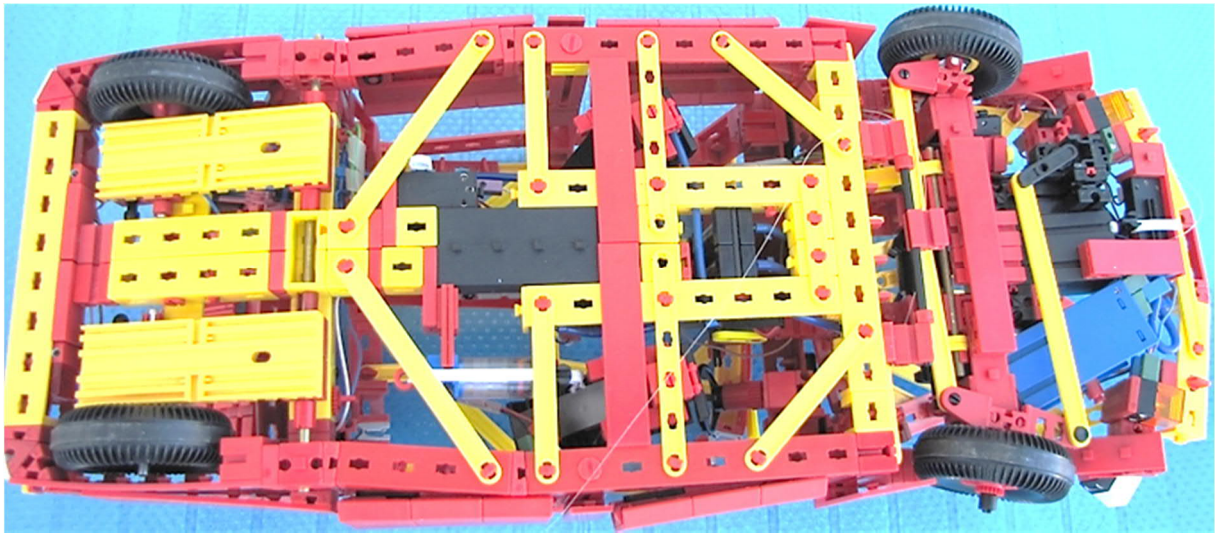


Abb. 10: Von unten (2)

Die Spurstange musste geteilt werden (die Länge ging nicht anders auf) und hat einen BS7,5, der als Schlitten auf einer schwarzen K-Achse hin und her läuft. Hier dient er auch als Endpunkt für die Nylonseile für die mitlenkenden Scheinwerfer, aber das musste nochmal geändert werden. Die Messingachse oberhalb der Spurstange gehört zur vorderen Hubvorrichtung.

In Abb. 12 ist das Fahrwerk drucklos und abgesenkt. In der Mittelkonsole liegt die vordere Platte 30 · 30 tiefer als die hintere.

In Abb. 13 ist das Fahrwerk voll angehoben (die vordere Platte 30 · 30 in der Mittelkonsole zeigt es an). Hinten sieht man einen deutlichen Unterschied zur Lage „unten“; vorne nicht so sehr.

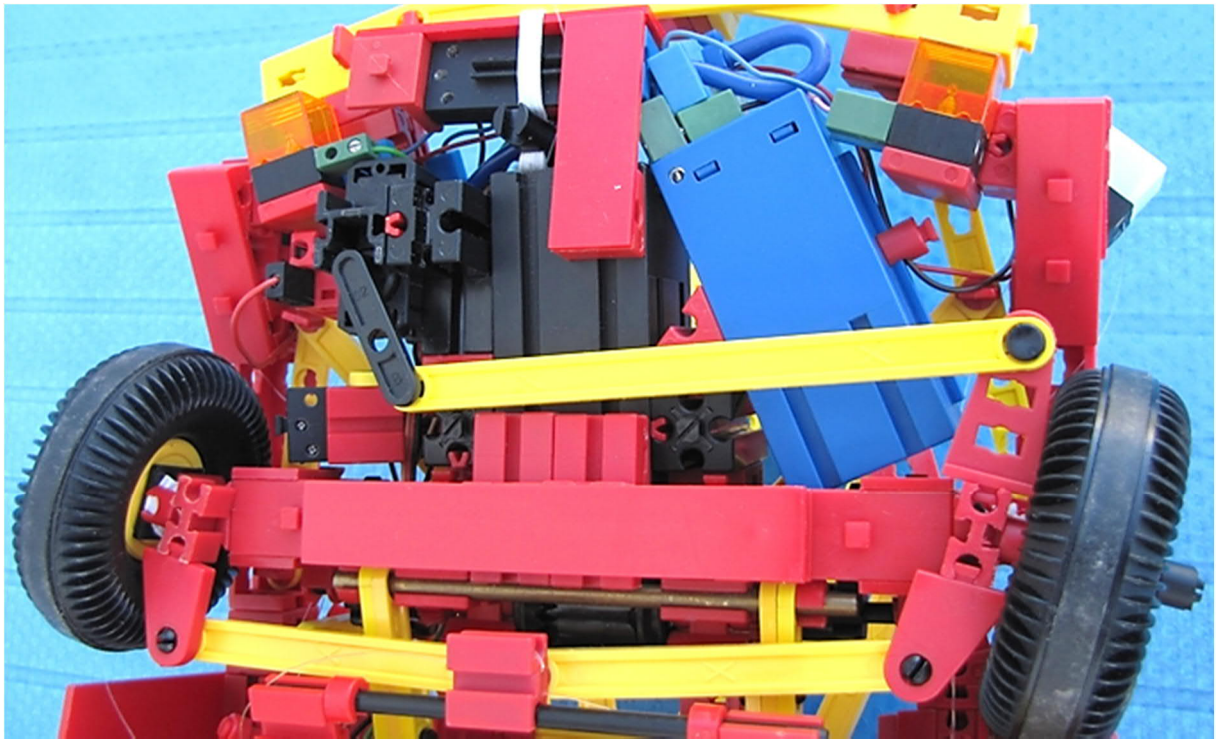


Abb. 11: Lenkung

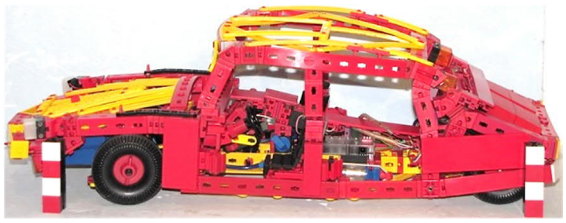


Abb. 12: Hubvorrichtungen abgesenkt

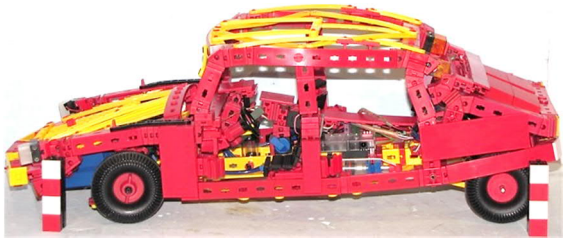


Abb. 13: Hubvorrichtungen angehoben

Die vordere Hubvorrichtung ist ein wenig „fake“: die P-Zylinder in der Mittelkonsole arbeiten über ein Parallelogrammgestänge, das die Karosserie (Kotflügel, Stoßstange, Motorhaube) gegenüber der gesamten Antriebseinheit (Motor, Vorderachse, Lenkung, Kompressor) hebt oder senkt. Die Vorderachse gewinnt damit also kein bisschen Bodenfreiheit.

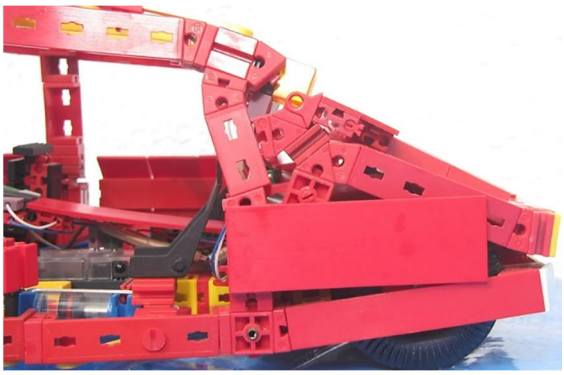


Abb. 14: Schiefe Winkel

Durch die ungleich weiten Hubwege (hinten viel, vorne wenig) bleibt die Schnauze trotz „Anheben“ in ihrer Höhe „über Grund“ nahezu unverändert. Der

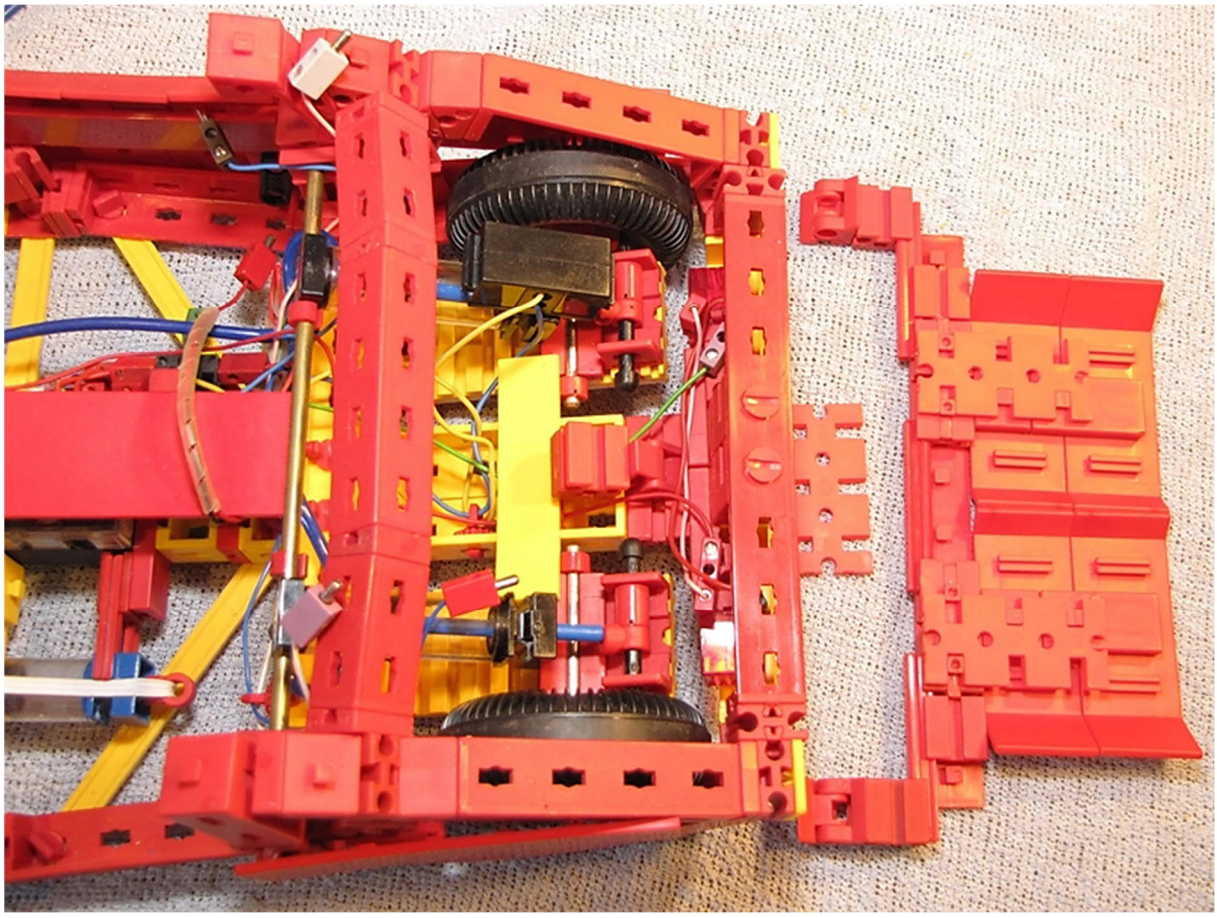


Abb. 15: Heckklappe und Raum für den Akku

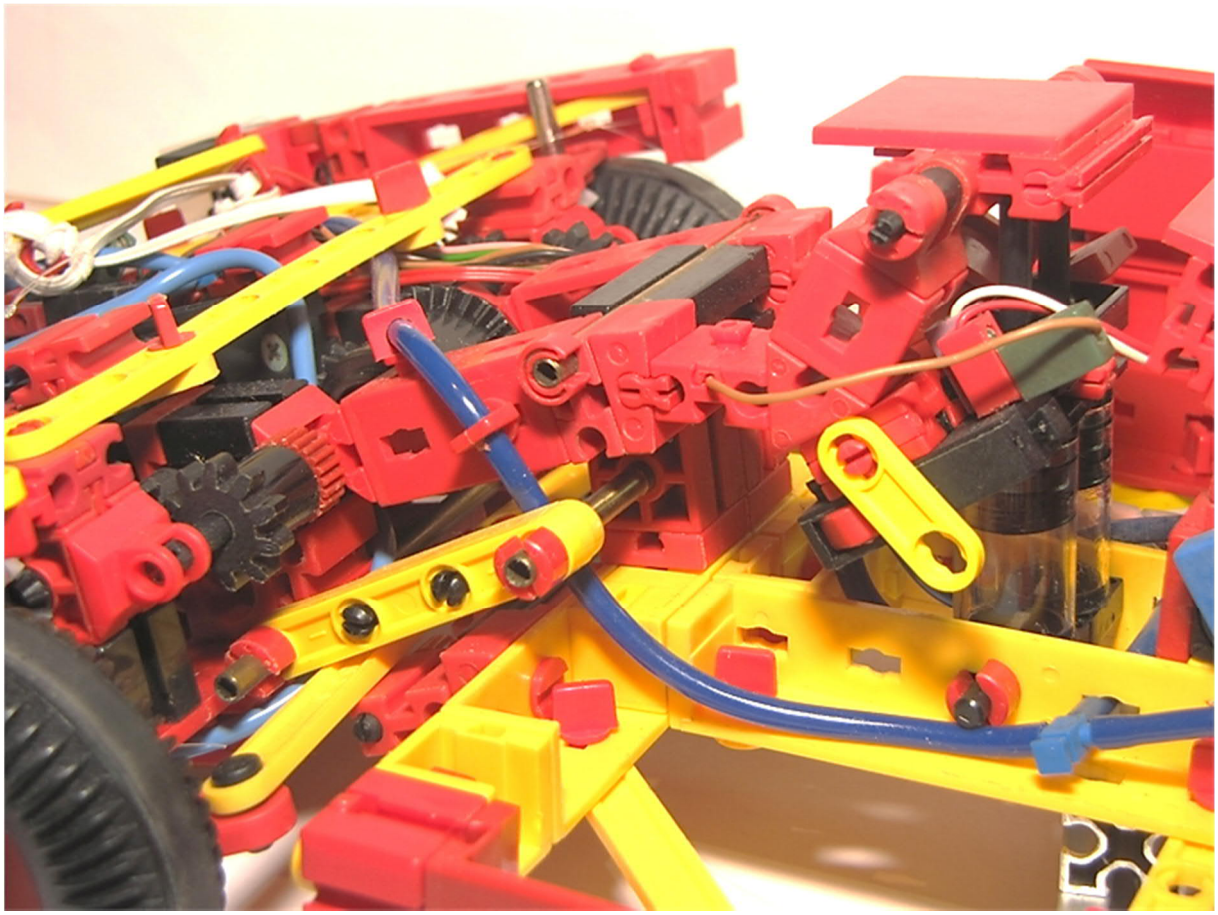


Abb. 16: Parallelogrammgestänge vorn

Stolz des Erbauers hat aber verhindert, dem hinteren Hubweg engere Grenzen zu setzen.

Dachschräge, Heckklappe, Radkasten, C-Säule – lauter Winkel, die irgendwie zusammenpassen müssen (Abb. 14). Der Sitzbaustein ist wohl da, aber in den Maßstab passt er nicht. Ganz rechts ist ein Leuchtstein im gelben Winkelträger erkennbar.

Den Kofferraum (Abb. 15) füllt der ft-Akku komplett aus. Die rechte Radschwinge trägt auch die ft-Blinkelektronik, die von den Minitastern in der Mittelkonsole wahlweise auf die linken oder rechten Blinker geschaltet wird. Die Heckklappe wird von BSB-Grundplatten Spur N (36093) zusammengehalten.

Die vordere Hubmechanik (Abb. 16) ist ein Parallelogrammgestänge, das je Seite oben einen roten Winkelträger 30 enthält, und unten ein Päckchen aus drei Streben I-45-Loch. Der obere Teil ist nach hinten verlängert, wo er zur Mittelkonsole wird, die von den beiden P-Zylindern angehoben wird.

Die Frontscheinwerfer in Abb. 17 werden von der Lenkung über Bowdenzüge [5] angesteuert. Die Nylonseile (124875 oder beliebige Angelschnur) verlaufen in den beiden hellblauen P-Schläuchen und arbeiten gegen Gummizüge (in Weiß, im Supermarkt unter „Haushaltswaren“).

Das Differential, die Messingachse rechts davon und die schwarze K-Achse vor der Mittelkonsole sind Enden und Drehpunkt des Hebels, mit dem die vordere Karosserie angehoben wird. Diese Mechanik verbraucht hier eine ganze Menge Platz. Beim Original sieht das ganz anders aus.

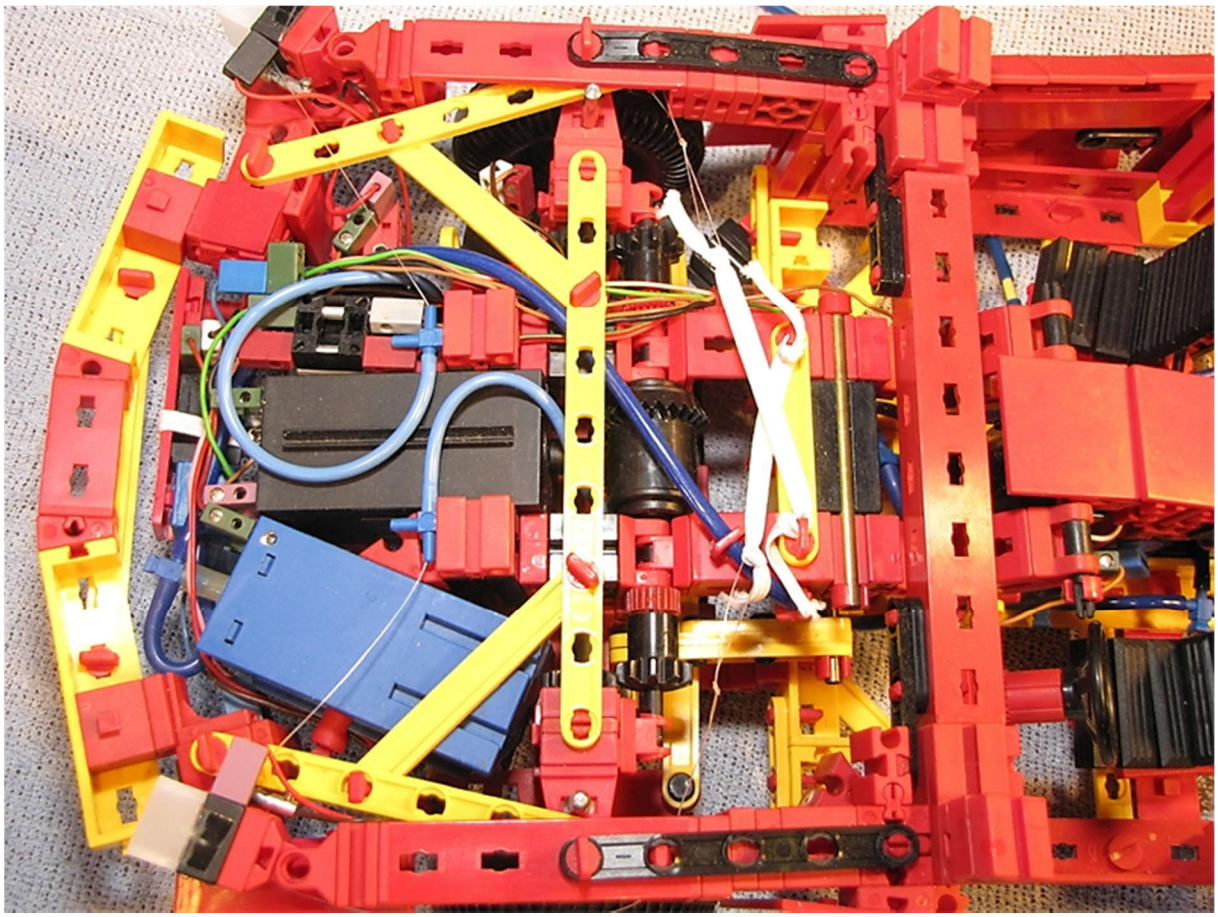


Abb. 17: Fertiger Motorraum, mitlenkende Scheinwerfer

Ein ft-Frontantrieb mit gefederter Einzelradaufhängung steht noch aus.

Quellen

[1] Harald Steinhaus: *Citroen DS (Déesse, die Göttin)*. Im Bilderpool der ft Community, 2012.

[2] Wikipedia: *Citroën Traction Avant*.

[3] Wikipedia: *Hydropneumatik*.

[4] Harald Steinhaus: *Fendt GTA380 reloaded*. Im Bilderpool der ft Community, 2015.

[5] Ralf Geerken: *Bowdenzug*. ft:pedia 1/2017, S. 13–16.

Uhren

Synchronuhr mit Stirnradgetriebe

Dirk Fox

In der ft:pedia und im Bilderpool der fischertechnik-Community wurden schon zahlreiche großartige Synchronuhren – von einem 50-Hz-Wechselstrom-Motor angetriebene Uhrengetriebe – vorgestellt. Hier kommt eine weitere Variante mit einem reinen Stirnradgetriebe, ausschließlich mit (ungemoddeten) fischertechnik-Zahnrädern.

Hintergrund

Synchronuhren üben eine ganz besondere Faszination aus. Wenn sie gelingen, surren sie leise vor sich hin und zeigen die Zeit mit großer Präzision an [1, 2] – jedenfalls soweit und solange die Netzfrequenz im Mittel bei 50 Hz liegt [3] und man ein fischertechnik-Netzteil mit Wechselstromausgang sein eigen nennt.

Ihre Konstruktion birgt eine faszinierende Herausforderung: das passende Getriebe. Die Netzfrequenz von 50 Hz, die am Synchronmotor über Elektro- und Permanent-Magnete die Welle (Antrieb) in Drehung versetzt, muss in eine Umdrehungsgeschwindigkeit der Minutenachse (Abtrieb) von $\frac{1}{3600}$ Hz gewandelt werden, damit sich der Minutenzeiger genau einmal pro Stunde dreht.

Wir benötigen also eine Übersetzung von 1:180000 ins Langsame. Eine erste Untersetzung nimmt der die Uhr antreibende Elektromotor vor; der Faktor wird von der Zahl der verwendeten Permanent-Magnete am Rotor bestimmt [1, 4, 5].

Für meine Synchronuhr mit Schrittschaltwerk, die den Minutenzeiger alle 60 Sekunden um ein 60stel weiterdreht, habe ich sechs Neodym-Magnete verwendet [6]. Bis zum Schrittschaltwerk benötigte ich eine Untersetzung von 1:500, die ich über drei Schneckengetriebe 1:10 und eine Übersetzung 2:1 realisierte. Zusammen mit dem Motor (1:6) entsprach das einer Untersetzung von 1:3000; das Schrittschaltwerk untersetzte 1:6 und ein nachfolgendes Schneckengetriebe ein weiteres Mal 1:10.

Schneckengetriebe an einem Z10 haben aber leider viel Spiel und lassen sich nicht besonders elegant verbauen. Sie haben außerdem etwa den dreifachen Reibungsverlust im Vergleich mit einem Zahnradgetriebe [7]. Außerdem ändern sie die Achsausrichtung, was nach meinem Empfinden die Ästhetik der Uhr beeinträchtigt.

Ein Stirnradgetriebe, das – ohne Schrittschaltwerk – eine Übersetzung ins Langsame von 1:30.000 bildet, ist jedoch mit fischertechnik-Zahnrädern nicht konstruierbar – es fehlen Zahnräder, mit denen ein durch fünf teilbares Übersetzungsverhältnis gelingt. Dachte ich jedenfalls.

Getriebe mit Differential

Da fiel mir Thomas Püttmanns genialer Beitrag zur Konstruktion kompakter Getriebe mit Differentialen ein [8]. Durch die Koppelung von zwei der drei An- bzw. Abtriebsachsen eines Differentials über ein Getriebe lassen sich mit etwas Geschick beliebige Übersetzungsverhältnisse konstruieren – das hat Thomas bereits mit seinem Planetarium beispielhaft gezeigt [2, 9].

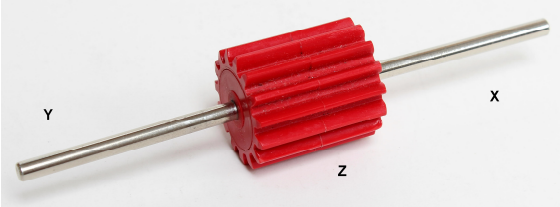


Abb. 1: fischertechnik-Differential mit Z15 als Differentialkäfig (31043)

Im Folgenden bezeichnen wir die Zahl der Umdrehungen der einen Achse des Differentials mit x, die der anderen mit y und die des Differentialkäfigs in der Mitte mit z. Damit gilt die von Thomas in [8] hergeleitete folgende Gleichung für das Verhältnis der Umdrehungen der drei An-/Abtriebe zueinander:

$$x + y = 2z$$

Der Trick besteht nun darin, zwei der drei möglichen An-/Abtriebe x, y und z in ein festes Übersetzungsverhältnis zueinander zu zwingen. Wenn wir beispielsweise den Differentialkäfig (z) über ein Getriebe mit der y-Achse koppeln, dann können wir in obiger Gleichung die Umdrehungen des Differentialkäfigs z durch die Umdrehungen der y-Achse ausdrücken. Abb. 2 zeigt einen solchen Getriebeentwurf mit dem Differential aus Abb. 1.

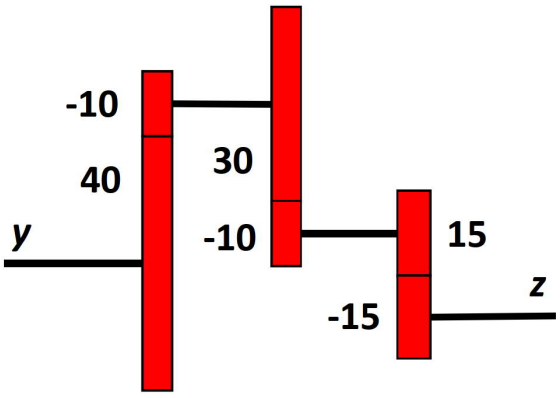


Abb. 2: Getriebeentwurf

Man erkennt, dass eine Umdrehung der y-Achse zu vier Umdrehungen des Z10 in umgekehrter Drehrichtung führt (1:4).

Jede Umdrehung dieser Achse sorgt über das Z30 für drei Umdrehungen des Z10 in der Mitte von Abb. 2 (1:3). Mit dem Z15 auf derselben Achse entspricht diese Umdrehungszahl bis auf die Drehrichtung der Zahl der Umdrehungen des Differentialkäfigs (z-Achse, 1:1). Damit gilt:

$$z = -\big(-3(-4y)\big) = -12y$$

Das negative Vorzeichen bezeichnet jeweils eine Umkehr der Achsdrehrichtung. Wenn sich die y-Achse einmal dreht, dreht sich der Differentialkäfig also zwölfmal in entgegengesetzter Richtung.

Setzen wir diese Relation in Thomas' Differentialgetriebegleichung ein, ersetzen wir also z durch $-12y$, so erhalten wir:

$$x + y = -24y$$

Damit gilt für die Übersetzung zwischen der Antriebsachse x und der Abtriebsachse y des Getriebes:

$$y = -\frac{1}{25}x$$

Abb. 3 zeigt den Getriebeentwurf aus Abb. 2 als fischertechnik-Modell. (Es ist nicht die einzige mögliche Konstruktion – auch mit anderen Zahnradkombinationen erreichen wir eine Übersetzung 1:12 zwischen y-Achse und Differentialkäfig.)

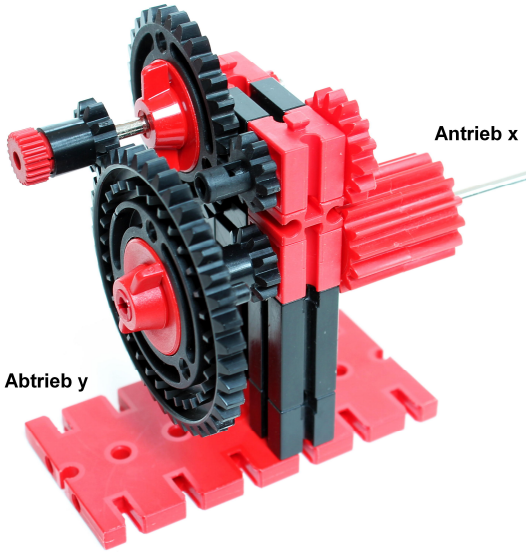


Abb. 3: Getriebe 1:25

Wenn wir nun das Z10 auf der *y*-Achse ein Z40 antreiben lassen, dann erhalten wir eine Übersetzung von 1:100 ins Langsame (mit gleicher Drehrichtung).

Schalten wir ein zweites solches Getriebe 1:25 dahinter, liefert uns das ein Stirnradgetriebe, das 1:2500 ins Langsame übersetzt. Mit drei weiteren Übersetzungen 2:3, 1:2 und 1:4 (zusammen also 1:12) erhalten wir das gewünschte Getriebe mit einer Untersetzung von 1:30000 – ausschließlich mit fischertechnik-Stirnrädern (Abb. 4).

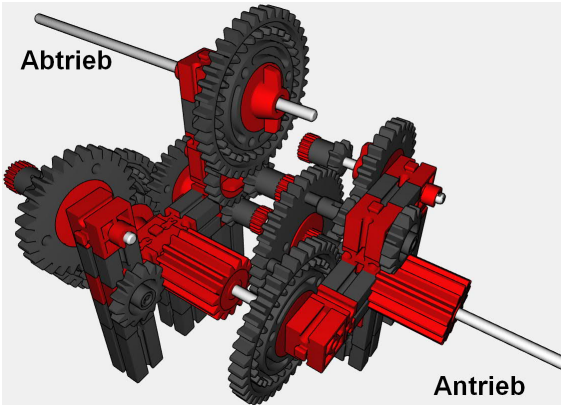


Abb. 4: Stirnradgetriebe 1:30000

Das ginge grundsätzlich noch etwas einfacher mit zwei (statt drei) Übersetzungen 1:3 und 1:4 ins Langsame. Die zusätzliche Übersetzung habe ich gewählt, damit die gesamte Getriebekonstruktion möglichst kompakt bleibt.

Insgesamt besteht das Getriebe aus 10 Stirnradgetrieben und zwei Differentialen. Die Reibungsverluste sollten geringer sein als bei meiner ursprünglichen Konstruktion mit vier Schnecken-, einem Stirnrad- und einem Schrittschaltgetriebe – dessen Reibung entsprach etwa der von mindestens 14 Stirnradgetrieben. Tatsächlich ist das Getriebe ganz ohne spezielle Achslager überraschend leichtgängig – auch dank der verwendeten alten Differentiale – und hat nur minimales Spiel: perfekt für eine Synchronuhr.

Synchronuhr

Jetzt müssen wir nur noch den Synchronmotor aus [6] auf die Antriebsachse sowie Zeiger und Zifferblatt mit einer Untersetzung 1:12 auf die Abtriebsachse montieren, und schon haben wir eine fischertechnik-Synchronuhr mit Stirnradgetriebe.

Bei der Konstruktion habe ich eine etwas größere Bauhöhe in Kauf genommen, damit Antrieb (Synchronmotor) und Abtrieb (Zeiger) der Uhr in einer Flucht liegen und die Uhr exakt auf eine Experimentierplatte 500 (32985) passt.

Das Resultat zeigen Abb. 5 und 6. Auf der Webseite zu dieser ft:pedia-Ausgabe findet ihr eine fischertechnik-Designer-Datei der Uhr zum Download.

Quellen

[1] Dirk Fox: *Der Elektromotor*. ft:pedia 3/2013, S. 4–8.

[2] Dirk Fox, Thomas Püttmann: *Technikgeschichte mit fischertechnik*. dpunkt-Verlag, 2015; Kapitel 10: Der Elektromotor.

[3] Dr. Gobmaier: *Netzfrequenzmessung*.

[4] Matthias Dettmer: *Synchronmotoren*. ft:pedia 2/2016, S. 48–52.

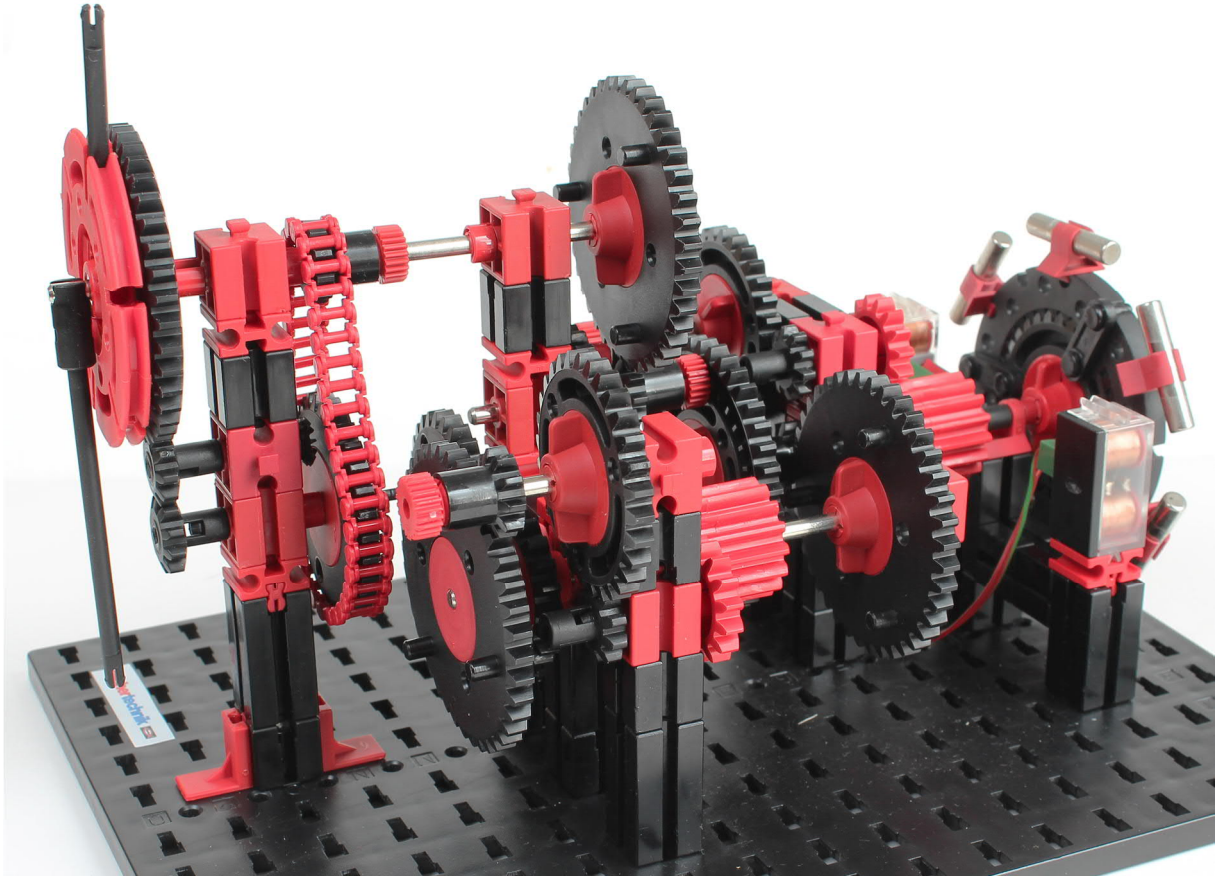
[5] Rüdiger Riedel: *Neue Synchronmotoren*. ft:pedia 2/2017, S. 25–31.

[6] Dirk Fox: *Synchronuhr mit Schrittschaltwerk*. ft:pedia 1/2017, S. 48–53.

[7] Artur Fischer: *fischertechnik Hobby – Experimente und Modelle*. Hobby 2 Band 1 (39521), S. 41 und 44.

[8] Thomas Püttmann: *Zahnräder und Übersetzungen (Teil 2)*. ft:pedia 3/2011, S. 25–28.

[9] Thomas Püttmann: *Planetarium*. ft:pedia 4/2011, S. 39–51.



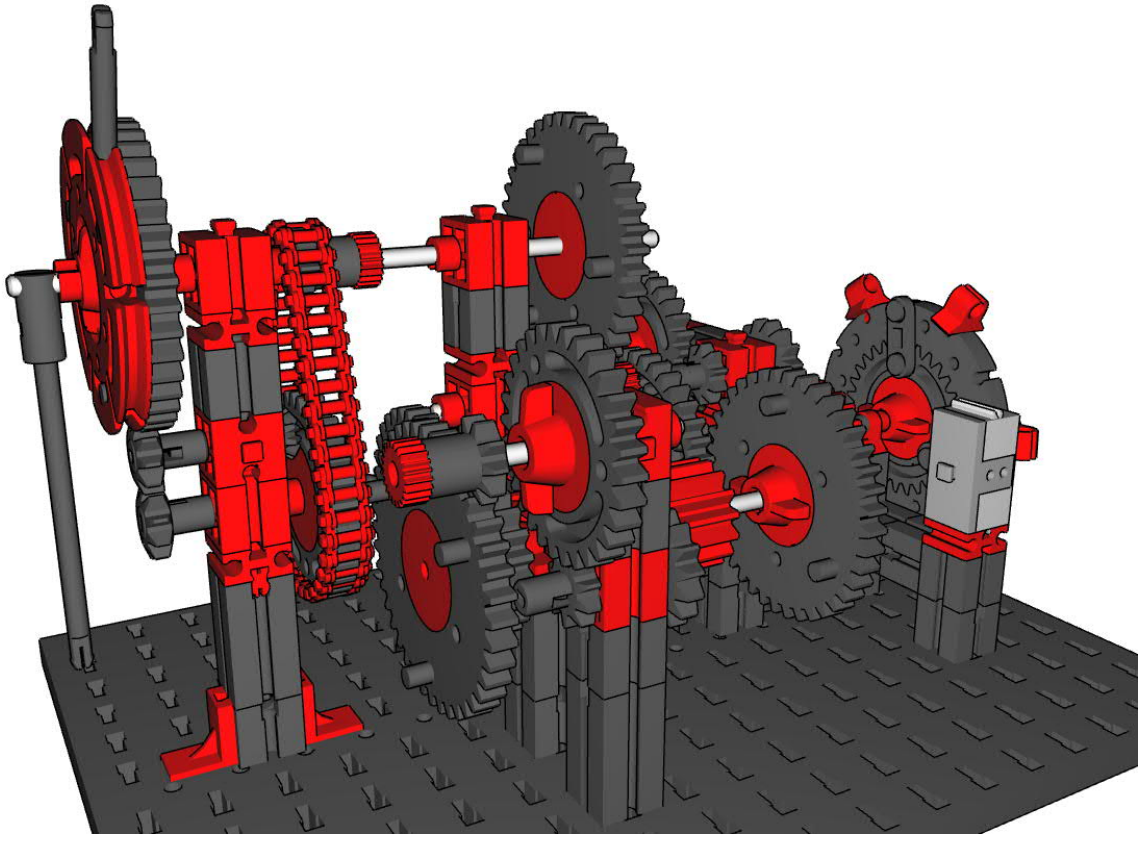


Abb. 5: Synchronuhr mit Stirnradgetriebe; Abb. 6: 3D-Konstruktion (fischertechnik-Designer)

Modell

Parallele Roboter – Tripteron und Agile Eye

Florian Bauer

In diesem Beitrag möchte ich ein paar Modelle von parallelen Robotern aus fischertechnik vorstellen. Unter einem Roboter soll hier – wie in der einschlägigen Fachliteratur üblich – ein mechanisches System verstanden werden, das über eine kinematische Kette, also einer Gruppe von Armen und Gelenken, einen End-Effektor (z. B. einen Greifer) in der Ebene oder im Raum bewegen kann.

Um zu erklären, was ein paralleler Roboter ist, möchte ich zuerst seinen Verwandten, den seriellen Roboter vorstellen, der oft in der industriellen Fertigung anzutreffen ist.

Serieller Roboter

Bei einem seriellen Roboter ist der End-Effektor über eine sequenzielle Kette von Armen und Gelenken mit der Basis, dem festen Bezugssystem, verbunden. Die Aktuatoren sitzen jeweils an den Gelenken und bestimmen die Orientierung der Arme zueinander. Jeder Aktuator bewegt auch alle nachfolgenden Aktuatoren bis zum End-Effektor. Über die Geometrie ergeben sich dadurch Position und Orientierung des End-Effektors. Zur Modellierung und Visualisierung von virtuellen Robotermodellen kann ich die *Robotics Library* [1] empfehlen.

Der serielle Aufbau erfordert eine hohe Steifigkeit der Arme und Spielfreiheit in den Gelenken, da sich ansonsten Ungenauigkeiten entlang der kinematischen Kette verstärken können.

Um diesen Effekten entgegenzuwirken, hat man parallele Roboter entworfen, die eine hohe Steifheit mit geringeren Massen der Arme vereinen und bei denen schnellere Bewegungen mit hoher Präzision ausgeführt werden können.

Paralleler Roboter

Bei einem parallelen Roboter ist der End-Effektor gleichzeitig mit mehreren Armen über Gelenke mit Aktuatoren verbunden, die sich an der Basis befinden. Dadurch können kräftige schwere Aktuatoren eingesetzt werden.

Dies ist meines Ermessens ein entscheidender Vorteil gegenüber seriellen Robotern, bei denen die Aktuatoren auf den bewegten Armen montiert werden müssen und dadurch große Drehmomente auf die vorhergehenden Gelenke ausüben.

Die Aktuatoren bestimmen die Orientierung der sogenannten *proximalen Arme*, welche jeweils über ein weiteres Gelenk (*distaler Arm*) mit dem End-Effektor verbunden sind. Der End-Effektor wird von mehreren der distalen Links gehalten. Im Allgemeinen führt eine Verstellung nur eines Aktuators zur gleichzeitigen Veränderung mehrerer distaler Links und der Position des End-Effektors.

Parallele Roboter können sogenannte *Singularitäten* aufweisen, bei denen die Position des End-Effektors nicht mehr eindeutig bestimmt ist. Solche Singularitäten gilt es beim Betrieb über die Steuerungs-Software unbedingt zu vermeiden, da dort die Stabilität des End-Effektors nicht mehr sichergestellt ist.

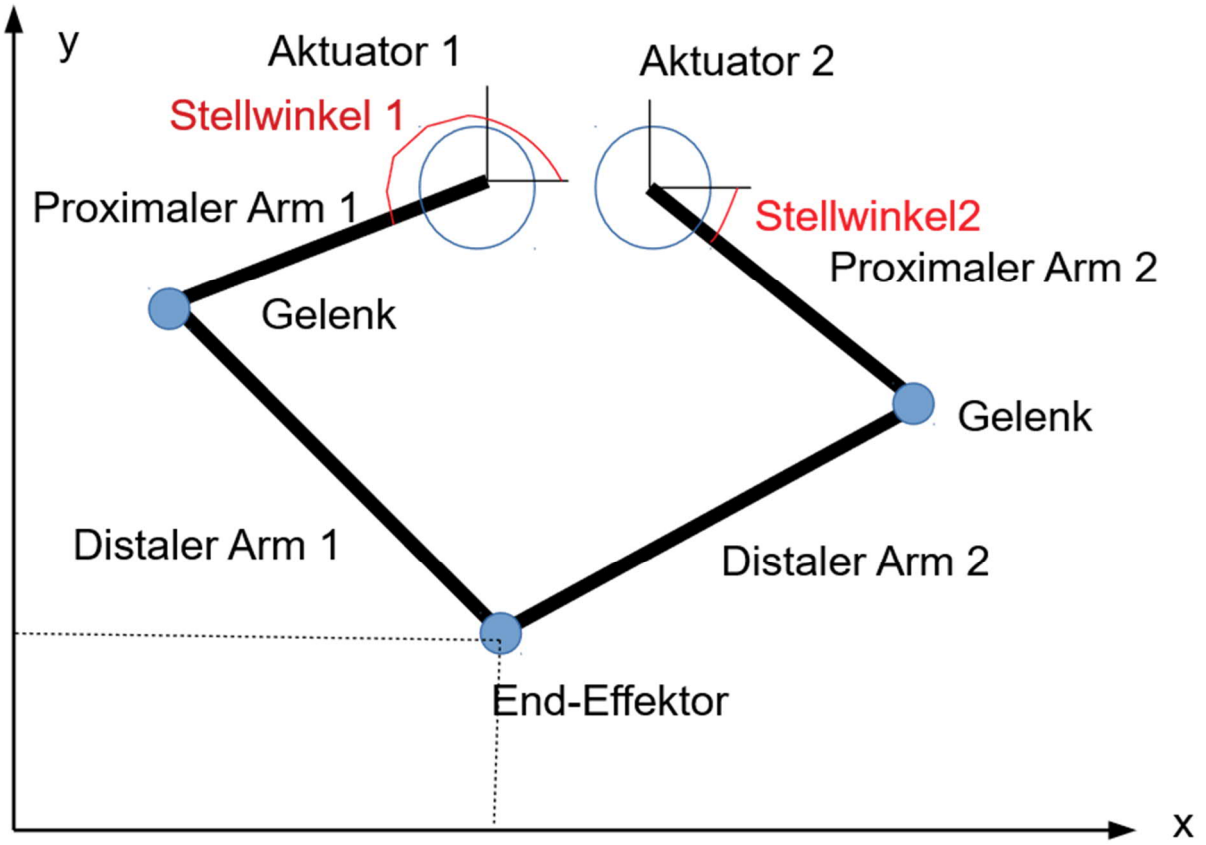


Abb. 1: SCARA, ein planarer paralleler Roboter

Der Arbeitsbereich eines parallelen Roboters kann durch solche Singularitäten eingeschränkt sein.

Ein Beispiel für ein System mit hohen Steifigkeiten ist der *Hexapod*, ein paralleler Roboter mit sechs Freiheitsgraden. Die Verstellung muss präzise synchron über mehrere Aktoren erfolgen, da sonst Spannungen oder Verklemmungen auftreten können.

Ein einfacher, schneller, planarer, paralleler Roboter für Pick-und-Place-Aufgaben ist der sogenannte *SCARA*, bei dem zwei Rotations-Aktoren einen End-Effektor innerhalb einer Ebene bewegen (siehe Abb. 1). Die zwei Stellwinkel der Basis-Aktuatoren des SCARA bestimmen die X-Y-Position des End-Effektors. In der ft-Datenbank ist ein Schachroboter beschrieben, der auf diesem Prinzip basiert [2].

Die üblichen Aktuatoren in einem Roboter sind Dreh-Aktuatoren (*rotational joints:* R) oder Linear-Aktuatoren (*prismatic joints:* P).

Tripteron

Wenden wir uns nun der ersten Konstruktion eines parallelen Roboters mit drei translatorischen Freiheitsgraden zu, der mit drei Linear-Aktuatoren betrieben wird: dem *Tripteron* (3 DOF-3PPP Manipulator, [3]).

Beim Tripteron werden drei Linear-Aktuatoren (*Prismatic Joints,* daher PPP) verwendet, die jeweils zwei mit einem Gelenk verbundene Arme verschieben. Die Drehachsen aller Gelenke eines jeden Armes liegen parallel zur jeweiligen Aktuator-Achse. Am anderen Ende des Doppel-Gelenks ist der End-Effektor über ein weiteres Scharnier verbunden.

Die Kinematik dieses kartesischen Mechanismus‘ ist einfach und selbsterklärend: Eine Verschiebung in jeder Richtung (X, Y oder Z) wird 1:1 auf den End-Effektor übertragen. Das erste Modell eines Tripterons mit fischertechnik ist in Abb. 2 zu sehen.

Als Linearantrieb habe ich Mini-Motoren mit Zahnstangengetriebe gewählt, die entlang einer Metallachse geführt werden. Die Ansteuerung der Motoren erfolgt mittels einer einfachen Umpolschalter-Taster-Kombination.

Für die Steuerung mit einem Mikrocontroller bräuchte man neben Motortreibern noch eine Positionsmessung. Hier könnte ich mir eine miniaturisierte Reflex-Lichtschranke vorstellen, die die feinen Zähne der Zahnstangen zählt. Oder besser noch recycelte Decoder-Streifen („Glasmaßstab des armen Mannes“) und Lichtschranken aus alten Officejet-Druckern. End-Schalter für die Referenz-Fahrt wären dann auch nötig. Alternativ könnte man den Abstand absolut über einen Reflex-Abstandssensor messen.

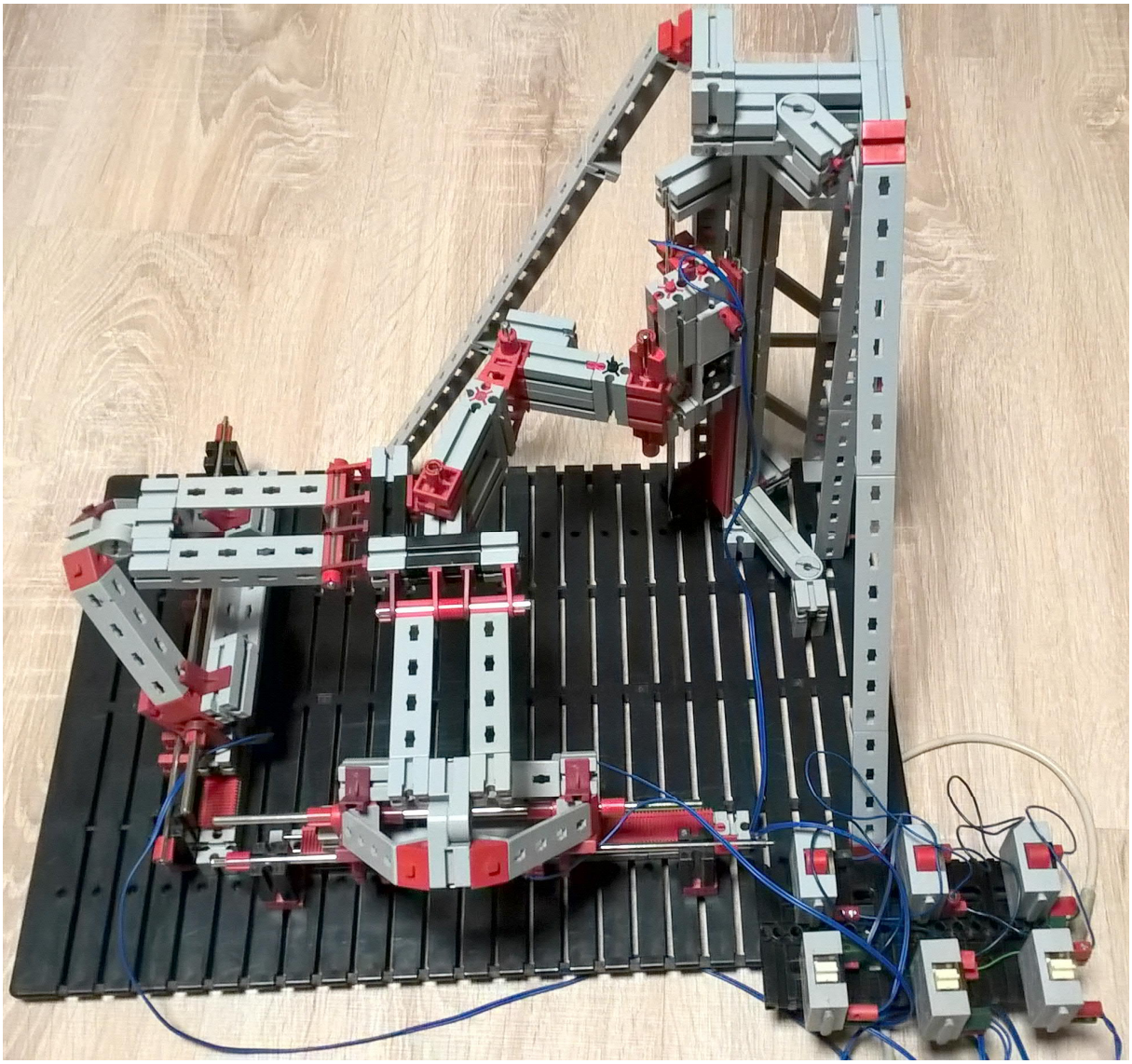


Abb. 2: fischertechnik-Modell eines Tripterons

Bei der mechanischen Konstruktion gibt es zwei wesentliche Herausforderungen mit fischertechnik:

1. *Stabilität und Verwindungssteifigkeit der Arme und der Vertikal-Achse:* Hier habe ich für den proximalen Arm eine Dreiecks-Konstruktion eingesetzt. Die Vertikal-Achse wurde mit fischertechnik-Statik-Elementen aufgebaut und verstrebt.
2. *Spielfreiheit der Lager:* Als Lager zwischen proximalen und distalen Armen mussten Gelenksteine herhalten, die ein geringes laterales Spiel aufweisen. Für das Scharnier am End-Effektor sind mehrere Lagersteine eingesetzt.

Das zweite Modell (Abb. 3) ist eine leichtgewichtigere Variante des ersten. Als Arme habe ich hier kleine Bauplatten verwendet, die eine gute Steifigkeit aufweisen.

Es gibt eine Vielzahl von kartesischen parallelen Robotern, von denen einige im Wikipedia-Artikel [3] beschrieben sind.

Ein nicht-kartesischer „Bruder“ des Tripterons ist der Delta-Robot, bei dem die Linear-Aktuatoren alle parallel sind, was die Kinematik etwas verkompliziert. Ein schönes fischertechnik-Modell findet sich in der fischertechnik-Datenbank [4] und in [5].

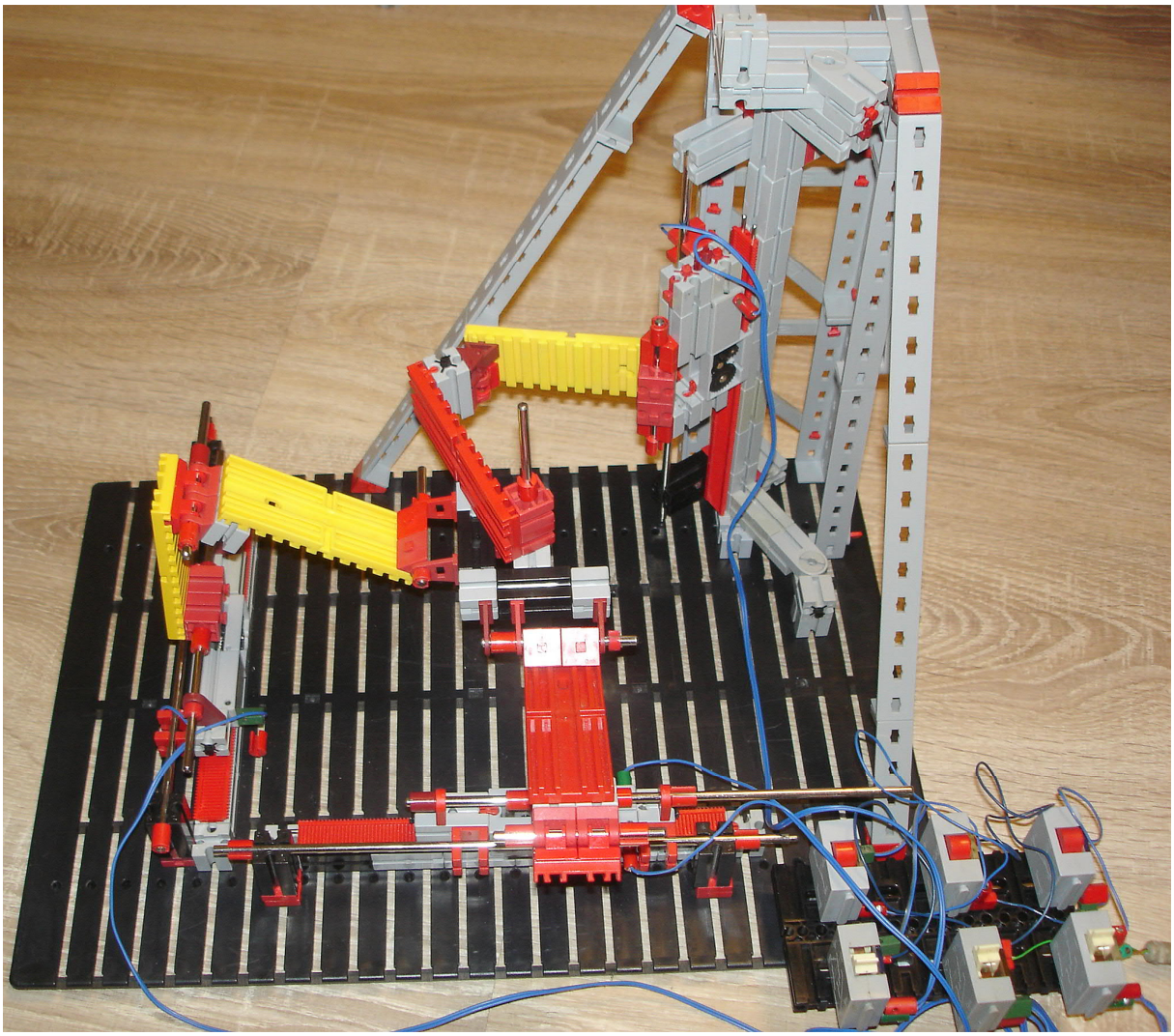


Abb. 3: Tripteron-Modellvariante mit Bauplatten

Agile Eye

Das Agile Eye ist ein paralleler Roboter mit drei Winkel-Freiheitsgraden (3 DOF-3RRR Manipulator), der 2002 am Robotik-Labor der Laval-Universität entwickelt wurde, um eine Kamera sehr schnell ausrichten zu können [6, 7]. Die Ansteuerung erfolgt über drei basisfeste Winkel-Aktuatoren.

Folgende Herausforderungen gilt es zu meistern:

1. *Gewicht, Stabilität und Verwindungssteifigkeit der Arme:* Hier müssen gebogene Arme gebaut werden. Durch die große Zahl von fischertechnik-Bauteilen ist dies schwer zu erreichen und die Arme sind nicht besonders verwindungssteif. Arme in doppelreihiger Bauweise sind zwar steifer, aber dafür auch schwerer.
2. *Antrieb:* Durch das hohe Gewicht der Arme, die ein großes Drehmoment auf die Antriebsachsen ausüben, muss eine große Übersetzung mit Schneckenantrieb verwendet werden, was dann im Betrieb zu geringen Geschwindigkeiten führt.
3. *Komplizierte Kinematik:* Will man den End-Effektor in eine bestimmte Richtung steuern, muss man über die sogenannte *inverse Kinematik* die Stellwinkel der Aktuatoren berechnen, um die gewünschte Ausrichtung anzufahren.

   Die Kinematik für das Agile Eye ist nicht trivial und in [8] beschreiben. Die dort

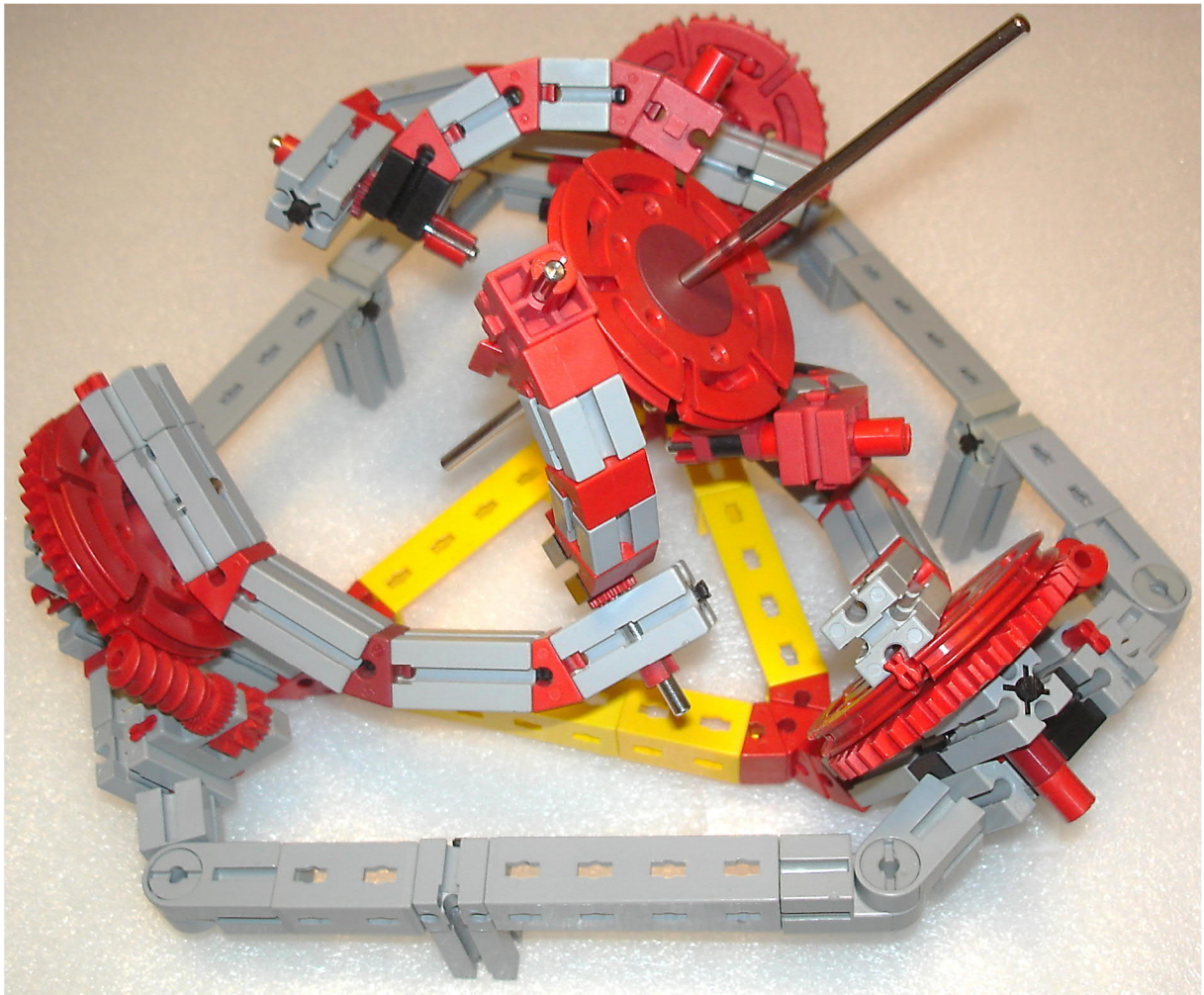


Abb. 4: fischertechnik-Modell eines Agile Eyes

aufgeführten Berechnungen sind für die vorliegende fischertechnik-Modell leider nicht geeignet, da sie davon ausgehen, dass benachbarte Achsen senkrecht zueinander stehen und sich alle Achsen in einem Punkt treffen, was bei diesem Modell nicht der Fall ist. (Streng genommen handelt es sich beim vorliegenden Modell um einen *Spherical Parallel Manipulator* (SPM). Für die verallgemeinerte SPM-Geometrie mit nicht senkrechten Achsen finden sich in [9] Lösungen).

4. *Stabile Unterkonstruktion:* Es ist wichtig, dass die gesamte Unterkonstruktion stabil ist, damit sich die Aktuator-Achsen zueinander nicht verstellen können.

Zudem muss man bei der Ansteuerung die singulären Bereiche meiden, die man im fischertechnik-Modell wunderbar „erfühlen“ kann: In den singulären Stellungen wird die Konstruktion plötzlich „weich“ und der End-Effektor kippt dann plötzlich ab.

Vereinfachtes Agile Eye

Wenn man auf einen Freiheitsgrad verzichten kann, kann man die Konstruktion sehr stark vereinfachen und kommt zu einem kompakten Aufbau wie in Abb. 5, der als Spiegel-Halter gedacht war. Hier kann man mit zwei ortsfesten Aktuatoren zwei Winkel des End-Effektors (hier: des Spiegels) einstellen. Dadurch, dass die beiden Aktuatoren mit der stabilen Basis verbunden sind, ist der Mechanismus besser entkoppelt als bei einem seriellen Aufbau, bei dem der zweite Aktuator im bewegten Bezugssystem ist. In diesem Fall übt man beim Verstellen unweigerlich ein Drehmoment auf die andere Achse aus.

Der Spiegel dreht sich bei Verstellung um die eigene Normale, was bei Abbildung von Laserstrahlen kein Problem darstellt, bei

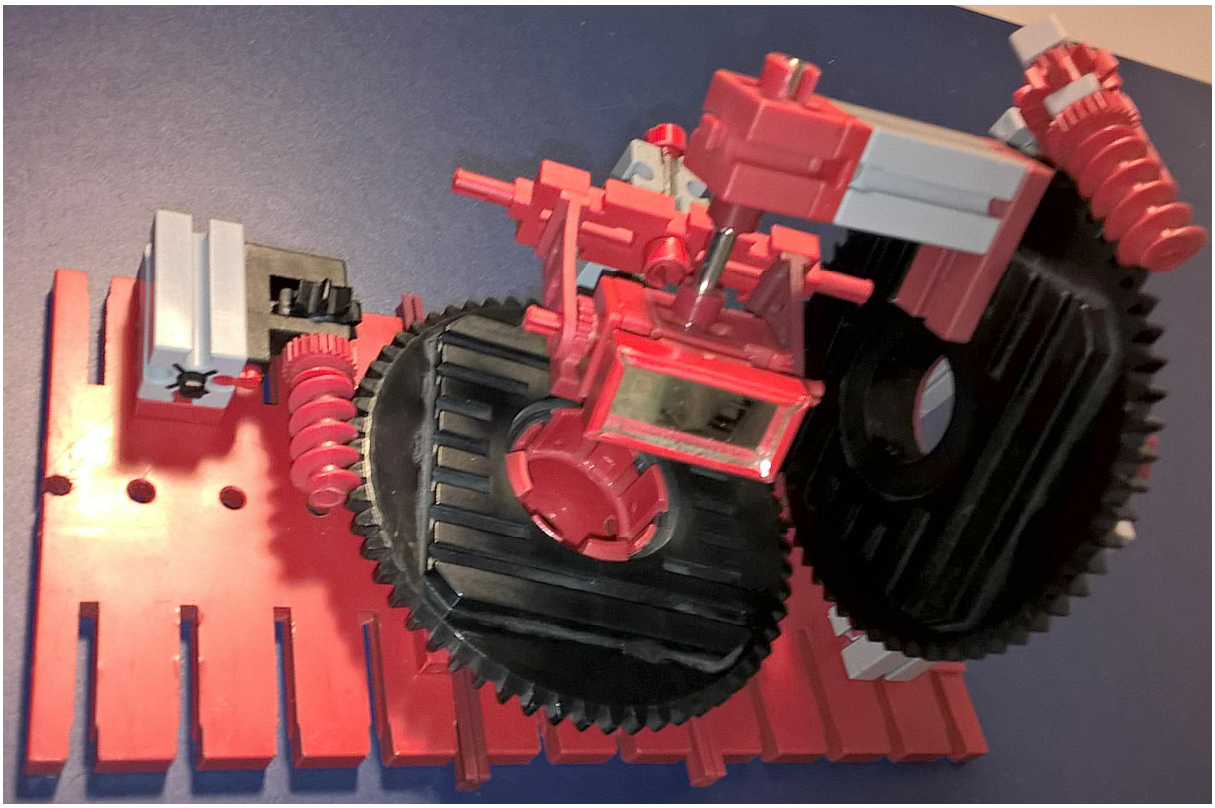


Abb. 5: Modell des vereinfachten Agile Eyes. Auf den ersten Blick ist der Zusammenhang mit dem „normalen“ Agile Eye nicht zu erkennen.

Typ	Arme	Aktuatoren	2D/3D	Freiheitsgrade	Gelenke
SCARA	4	3 × rot.	2D	3 × Position	5
TRIPTERON	6	3 × linear	3D	3 × Position	9
HEXAPOD	6	6 × linear	3D	3 × Position + 3 × Orientierung	12
AGILE EYE / SPM	6	3 × rot.	3D	3 × Orientierung	9
Simplified AGILE EYE	3	2 × rot.	3D	2 × Orientierung	5

Tab. 1: Überblick über parallele Mechanismen

Bildern allerdings schon. Diese Konstruktion ist für eine Kamera-Führung wegen der damit verbundenen Bild-Kippung nicht geeignet.

Die Kinematik ist selbsterklärend und trivial: Die Winkel der Aktuatoren übertragen sich 1:1 auf die Neigungswinkel des Spiegels.

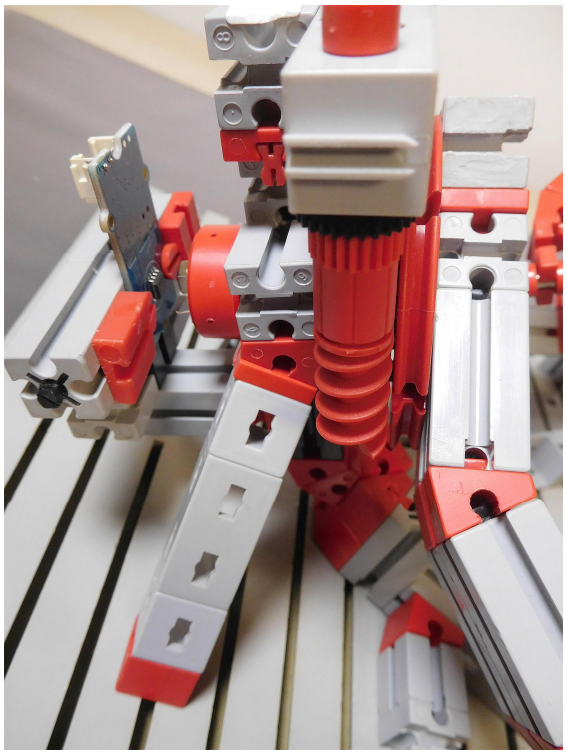


Abb. 6: Grove (AS5600) 12-bit Magnetencoder mit Würfelmagnet

Ausblick

Die in diesem Artikel erwähnten parallelen Mechanismen sind in Tab. 1 kurz zusammengestellt.

Die Schwierigkeiten liegen wie so oft in Gewicht und Stabilität des Aufbaus. Geringe Verwindungssteifigkeit und Lagerspiel summieren sich entlang der kinematischen Ketten. Für das Agile Eye kann man wie in Abb. 8 gezeigt speziell gebogene fischertechnik-kompatible Arme entwerfen und mit dem 3D-Drucker herstellen, die leichter und verwindungssteifer sind und mit denen die Geometrie besser abgebildet werden kann. Verbesserungen sind sicher möglich und ich würde mich über diesbezügliche Vorschläge freuen.

Eine Computer-Ansteuerung lässt sich mit PID-positionsgeregelten fischertechnik-Motoren und absoluten Magnet-Gebern realisieren (Abb. 6, 7). Für das Agile Eye wären noch eine geeignete Inverse Kinematik herzuleiten und die Berechnungen im Mikrocontroller zu implementieren.

Quellen

[1] Robotics Library: https://www.roboticslibrary.org/

[2] Dirk Uffmann: *Schachroboter mit Scara-Arm und Zugerkennung durch Bildverarbeitung*. Im Bilderpool der fischertechnik-Community, 2018.

[3] Wikipedia: *Cartesian Parallel Manipulators*.

[4] Thomas Püttmann: *Delta-Roboter*. Im Bilderpool der fischertechnik-Community, 2017.

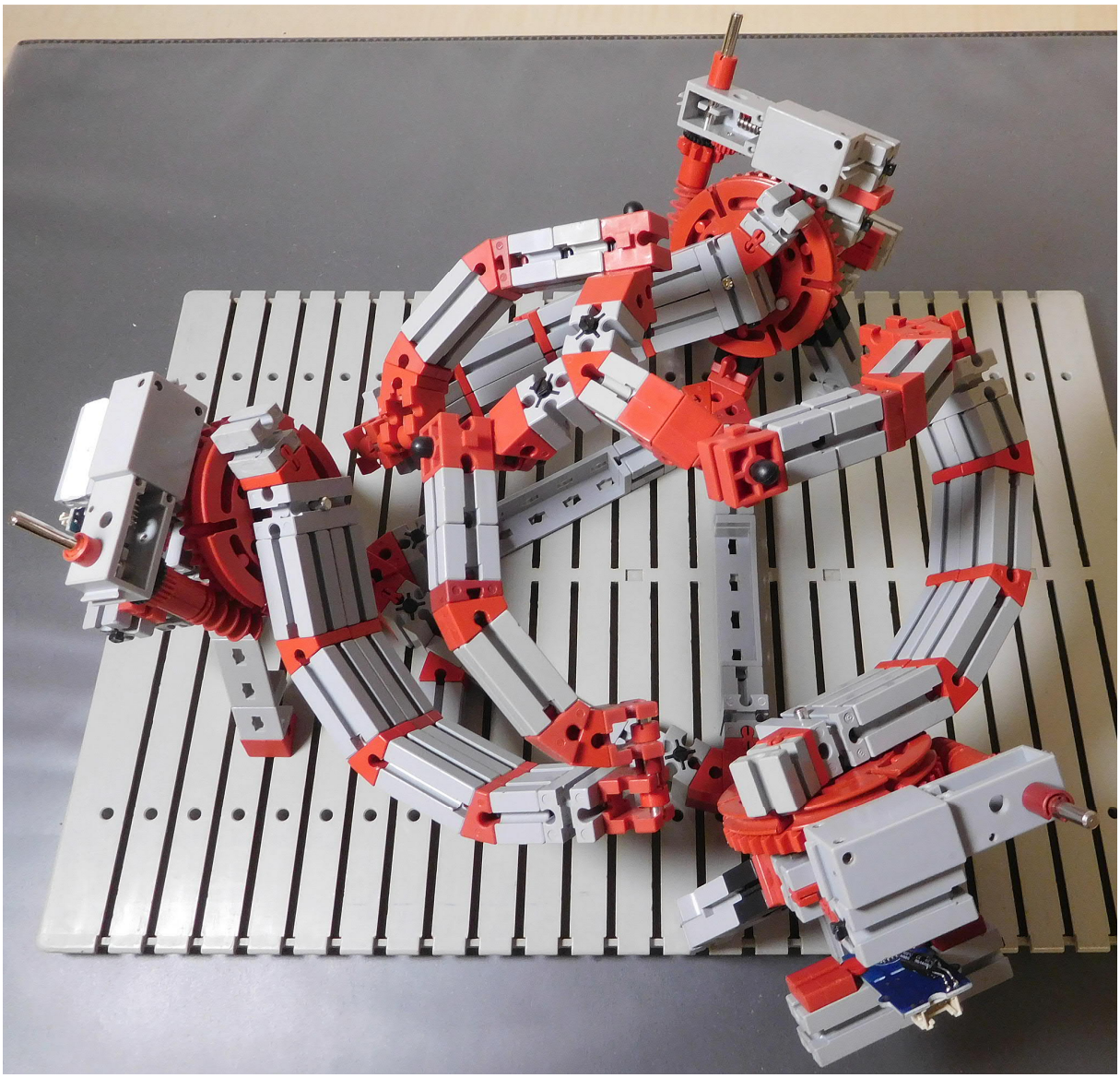


Abb. 7: Stabilisiertes Agile Eye mit Minimot-Antrieb

[5] Dirk Fox, Thomas Püttmann: *fischertechnik-Roboter mit Arduino*. dpunkt.Verlag, 2020.

[6] Laboratoire de robotique: *The Agile Eye*. Auf der Website der Université Laval, 2002-2015.

[7] Laboratoire de robotique de l'université Laval: *2-DOF Agile eye (simplified version)*. Auf Youtube, 2015.

[8] Ilian A. Bonev, Damien Chablat and Philippe Wenger: *Working and Assembly Modes of the Agile Eye*. In: Proceedings IEEE International Conference on Robotics and Automation, Juni 2006.

[9] Aibek Niyetkaliyev, Almas Shintemirov: *An Approach for Obtaining Unique Kinematic Solutions of a Spherical Parallel Manipulator*. 2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Besançon, Fankreich, 8.-11.07.2014.

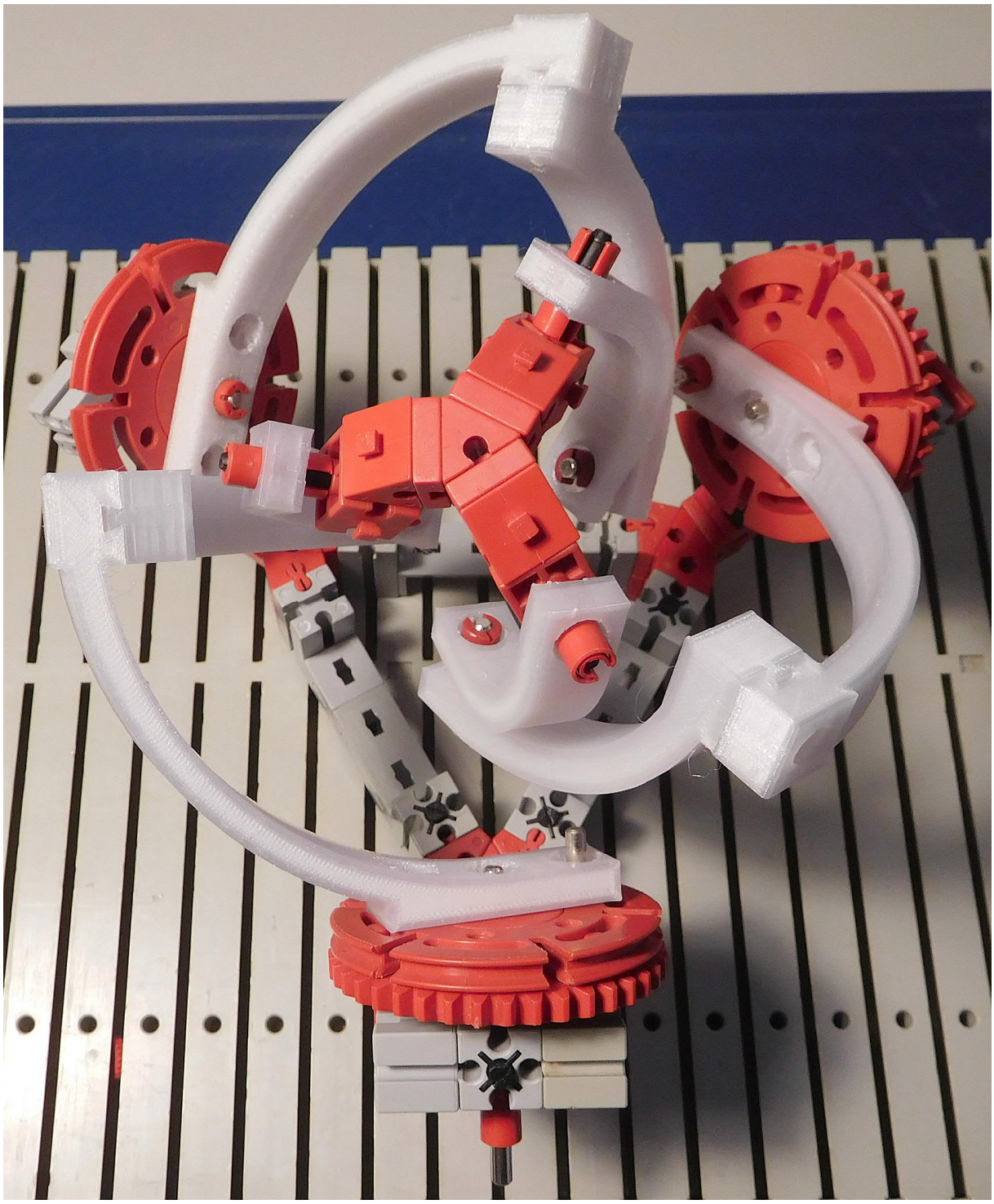


Abb. 8: Modifiziertes Agile Eye mit Armen aus dem 3D-Drucker

Computing

Sensoren am TXT: IR-Dioden und -Transistoren

Kurt Mexner

Mit verschiedenen Sensoren werden die Möglichkeiten des TXT Discovery Sets erheblich erweitert. Mit dieser Reihe möchte ich einige dieser Sensoren vorstellen. Heute beschäftigen wir uns mit Infrarotstrahlung.

Infrarot-Strahlung

Jeder von uns kommt täglich mit unsichtbarem Licht in Berührung, und zwar dann, wenn er die Fernseh-Fernbedienung in die Hand nimmt. Diese sendet Infrarot-Strahlen aus (IR-Strahlung), die das Auge nicht erkennt.

Ihr könnt sie aber sichtbar machen: Nehmt einmal eure Digitalkamera oder euer Handy zur Hand und richtet die Austrittslinse der Fernbedienung auf das Kameraobjektiv. Bei eingeschalteter Kamera (Handy) erkennt ihr auf dem Kameramonitor ein bläuliches, pulsierendes Licht an der Austrittslinse eurer Fernbedienung.

ROBO Pro erkennt unsichtbares Licht

Auch ROBO Pro kann IR-Licht erkennen. Benötigt werden hierzu IR-Fototransistoren (z. B. PT333-3C). Dabei ist auf die Polung zu achten: Der Pluspol hat ein längeres Anschlussbein, der Minuspol eine abgeflachte Seite. Beim Anschluss an einen Eingang des TXT kommt der Minuspol an den Anschluss, der zum Anzeigebildschirm zeigt; der Pluspol kommt an den äußeren Anschluss.

Das Eingangselement wird auf ‚Fototransistor, analog 5 kOhm' eingestellt. Es werden Werte zwischen 0 und 15.000 angezeigt. Je geringer die IR-Strahlung, desto höher ist der (Widerstands-)Wert; je stärker, desto geringer.

ROBO Pro verfügt über einen Interface-Test. Mit ihm lassen sich die Eingänge abfragen. Er wird in der oberen Menüreihe über „Interface testen" aufgerufen. Neben der Meldung, ob die Interface-Anbindung funktioniert, erscheinen auch die acht Eingänge mit ihren dort aktuell anliegenden Werten.

Zielt nach dem Anschließen des IR-Transistors mit eurer Fernbedienung auf ihn und beobachtet, was am Eingang angezeigt wird. Mit hoher Wahrscheinlichkeit (abhängig vom Typ eurer Fernbedienung) wird weiterhin 15.000 angezeigt oder ein sehr stark schwankender Wert. Warum ist das so?

Die Fernbedienung sendet eine pulsierende, kodierte Strahlung aus. ROBO Pro kann sie zwar erkennen, aber nicht dekodieren. Außerdem senden Glühlampen, LED-Leuchten oder Leuchtstoffröhren impulsweise ebenfalls IR-Strahlung aus. Auch diese Störstrahlung wird erfasst und muss ggf. herausgefiltert werden.

ROBO Pro sendet unsichtbares Licht

Da die Strahlung der Fernbedienung nicht gut erkennbar ist und sie auch von ROBO Pro nicht angesteuert werden kann, müssen wir uns einen eigenen Sender bauen. Dies

ist mit handelsüblichen IR-Sendedioden möglich (z. B. Osram SFH 415, Sanken SID1050M, Kyosemi KED862M51E).

Diese haben die gleiche Form und ähnliche Daten wie die bekannten LEDs. Sie können mit 3 bis 6 Volt angesteuert werden und verbrauchen ca. 20 bis 100 mA. Wichtig ist, auf die richtige Polung zu achteten und einen Vorwiderstand zu verwenden. Da der Ausgang des TXT mit 9 Volt arbeitet, sind wir mit einem Widerstand von 390 Ohm und 0,25 Watt auf der sicheren Seite.

Im ROBO Pro-Programm ‚IR Sender und Empfänger.rpp' aktiviert ein Mausklick auf den „Knopf" den Sender. Da der Knopf als Schalter eingestellt ist, wird nun kontinuierlich IR-Strahlung gesendet. Ein erneuter Mausklick beendet die Aussendung der IR-Strahlung. Eine Kontroll-Lampe am Monitor zeigt an, ob das Programm ordnungsgemäß funktioniert. Als einfache Anzeige dient nun nochmals der Interface-Test. Dort müssten nun Werte unterhalb von 15.000 angezeigt werden.

ROBO Pro empfängt unsichtbares Licht

Im TXT Controller ist bereits ein IR-Empfänger eingebaut. Dieser ist jedoch auf die Fernbedienung von fischertechnik ausgelegt [2]. Es ist mir bisher nicht gelungen, ihn mit anderen Mitteln zum Leben zu erwecken.

Auch der im TXT Discovery Set enthaltene Fototransistor reagiert auf IR-Strahlung. Er ist jedoch nicht sehr empfindlich.

Daher lohnt der Rückgriff auf handelsübliche Fototransistoren. Für die Nutzung am TXT Controller eignen sich nur Fototransistoren und Fotodioden mit zwei Anschlüssen. Transistoren mit drei Beinen benötigen weitere Bauteile zur Ansteuerung.

IR-Empfang mit Fototransistoren

Um die IR-Strahlung deutlicher zu erfassen ist ein kleines Programm notwendig. Für den Empfänger im ROBO Pro-Programm ‚IR Sender und Empfänger.rpp' ist das Eingangselement auf ‚Fototransistor' und ‚analog 5 kOhm' eingestellt. Im analogen Verzweigungselement ist ein Schwellenwert eingestellt (<12.000), der je nach örtlicher Gegebenheit und der verwendeten Bauteile angepasst werden sollte. Die empfangenen IR-Werte werden in der Textanzeige dargestellt und erleichtern die Einstellung des Schwellenwertes.

Wenn der Schwellenwert überschritten wird, leuchtet das Lampen-Element auf dem Monitor.

Ab und zu leuchtet das Lampen-Element auf, obwohl der IR-Sender nicht betätigt wurde. Dies liegt an der erwähnten Störstrahlung von Glühlampen, Leuchtstoffröhren und LED-Leuchten. Eine Abdeckung oder Beschattung des IR-Transistors kann hier Abhilfe schaffen.

IR-Empfang mit Fotodioden

Eine andere Möglichkeit, die Störstrahlung zu vermeiden (und die Strahlung der Fernseh-Fernbedienung sichtbar zu machen) besteht darin, eine Fotodiode zu verwenden (z. B. Pollin, Bestellnummer 121752). Bei Fotodioden muss das Eingangselement als ‚Farbsensor, analog 10 V' betrieben werden.

Der Schwellenwert sollte bei ca. 12 liegen. Bei zunehmender IR-Strahlung steigen die Werte, bei abnehmender IR-Strahlung verringern sie sich. Fotodioden reagieren schneller auf IR-Strahlung, Fototransistoren sind langsamer, dafür aber empfindlicher.

Mit dem Programm ROBO Pro-Programm ‚IR Sender und Empfänger.rpp' hat man nun zwei Möglichkeiten, IR-Strahlung zu erkennen und kann im Einzelfall entscheiden, für welche Anwendung sich welcher Empfänger besser eignet.

Beim Betreiben müsst ihr auf die unterschiedlichen Einstellungen achten:

Typ	Interface-Test	Verzweigungs-element	Eingangs element
Foto-transistor	Analog 5 kOhm (NTC)	Foto-transis-tor, z. B. < 12.000	Foto-transistor, Analog 5 kOhm
Foto-diode	Analog 10 V (Farb-sensor)	Farb-sensor, z. B. > 12	Farb-sensor

Tab.1: ROBO Pro-Einstellungen (Fototransistor/Fotodiode)

Erhöhung der Empfindlichkeit

Durch eine Parallelschaltung von Fotodioden und Fototransistoren kann man die Empfindlichkeit erheblich erhöhen.

Fototransistoren und Fotodioden reagieren auch auf Tageslicht und Kunstlicht, jedoch nicht so stark. Wenn die Gehäuse dunkel eingefärbt sind, werden Tageslicht und Kunstlicht zusätzlich herausgefiltert und die IR-Empfindlichkeit steigt.

Praktische Anwendungen

Zunächst lässt sich damit wunderbar experimentieren. Wie empfindlich ist das Gerät? Welche Entfernung kann mit IR-Strahlung überbrückt werden? Wie kann man die Reichweite erhöhen?

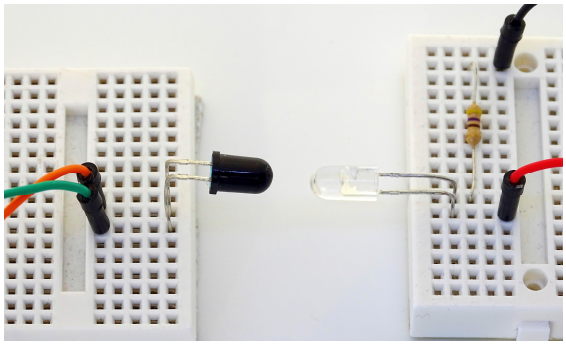


Abb. 1: IR-Sender und IR-Empfänger

Abb. 1 zeigt die Gegenüberstellung von Sender und Empfänger. Abstände bis 30 cm können damit überbrückt werden.

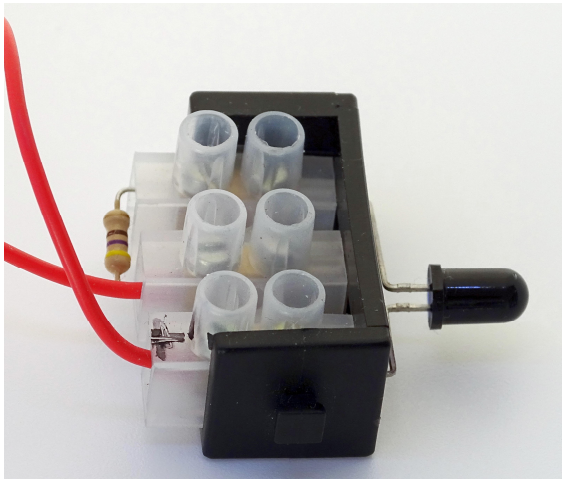


Abb. 2: IR-Empfängerbaustein

Abb. 2 und 3 zeigen, wie die IR-Bauteile in fischertechnikteile „integriert“ werden können. Bei Abb. 2 konnte auch der Vorwiderstand eingebaut werden.

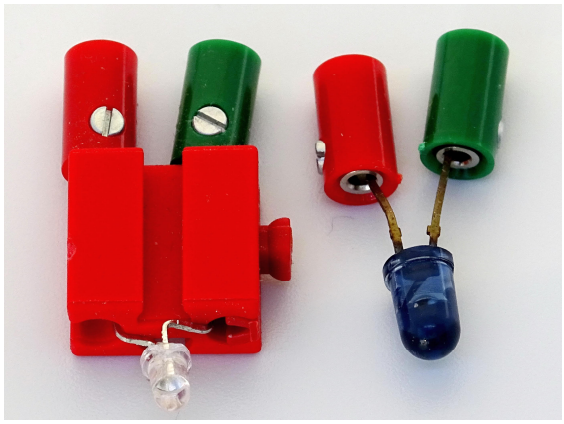


Abb. 3: IR-Senderbausteine

Die Schaltung ließe sich erweitern, um mit einer handelsüblichen Fernbedienung auch ein fischertechnik-Modell ein- und auszuschalten.

Wenn Sender und Empfänger einander gegenüberstehen, ergibt sich eine Lichtschranke. Wenn sie eng nebeneinander stehen, wären ein Näherungsschalter oder ein Kollisionswarner denkbar.

Das ROBO Pro-Programm ‚IR Sender und Empfänger.rpp‘ (sowohl mit Fotodioden als auch mit Fototransistoren) kann von der Download-Seite der ft:pedia heruntergeladen werden.

Tipp: Internetseiten für den Elektronik-Einstieg

Wer im Umgang mit den elektronischen Bauteilen unsicher ist, dem empfehle ich folgende Einstiegsseiten:

- Der Elektronik-Experimentier-Server
- Strippenstrolch.de
- Elektronik-Kompendium

Auf Google und Youtube finden sich zahlreiche weitere Einstiegsseiten und Einführungen.

Saarländische fischertechniker gesucht

Am Schluss noch ein persönliches Anliegen. Falls es im Saarland noch mehr fischertechniker gibt, würde ich mich über eine Kontaktaufnahme zum Erfahrungsaustausch freuen. Mein Schwerpunkt ist der Robotics TXT Controller mit der ROBO Pro-Software.

Referenzen

[1] Kurt Mexner: *ROBO Pro wird sensibel – Lichtempfindliche Widerstände*. ft:pedia 3/2021, S. 67–69.

[2] David Holtz: *Alternative Controller (2): Infrarot-Empfänger*. ft:pedia 2/2016, S. 60–67.

Elektronik

Silberlinge: Original oder Nachbau (Teil 5)

Peter Krijnen

Diese Folge dreht sich um den Nachbau des h4-Flip-Flop-Bausteins.

Zuerst wusste ich nicht, was ich über das Flip-Flop 36479 erklären sollte. Ich könnte natürlich beschreiben, was das Flip-Flop (Abb. 120) macht (Speichern von Signalen) und kann (Teilen von Signalen) und wofür es verwendet werden kann. Ich könnte mich auch nur an einen Schaltplan und ein paar Fotos halten. Ich werde das trotzdem machen, aber ich fand es etwas wenig.

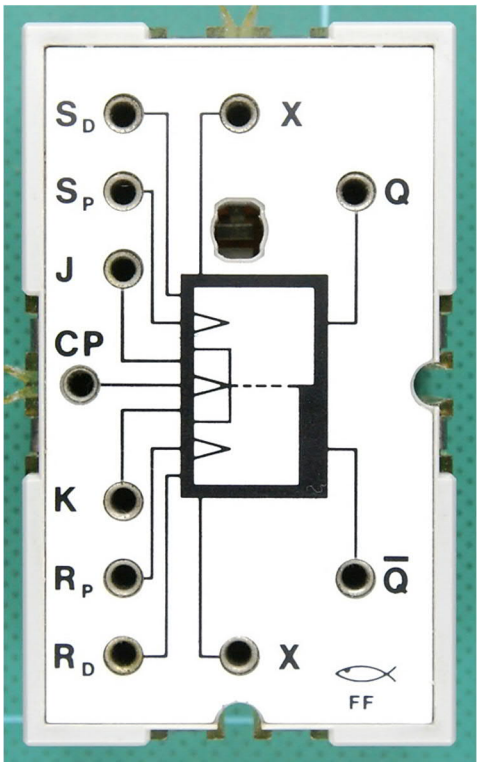


Abb. 120: h4 FF Flip-Flop-Baustein

Zur Bedienung und Verwendung des Flip-Flops verweise ich deshalb auf das Kapitel zum Flip-Flop, Seite 63 bis 74, in „Experimente + Modelle hobby 4 Band 3“ [1]. Es macht wenig Sinn, ganze Teile dieses Kapitels zu kopieren. Nachdem ich es selbst mehrmals gelesen hatte, gingen mir die Themen „elektronische Orgeln“ und „Grenzfrequenz“ immer wieder durch den Kopf.

Auf Seite 65 von hobby 4 Band 3 heißt es, dass in elektronischen Orgeln Frequenzteilung verwendet wird. Für Orgeln beträgt die Frequenz für C_{high} 8,36 kHz. Für jede tiefere Oktave wird diese Frequenz durch zwei geteilt. Ich weiß, dass dies der Fall ist, weil ich selbst zwei Jahre bei Emminent Orgels in Bodegraven Niederlande gearbeitet habe.

Als ich 1974 eine Führung durch die Fischerwerke bekommen konnte, stand im Musterzimmer eine kleine Orgel – eine Tastatur mit 49 Tasten, aufgebaut aus grauen Bausteinen. Das umfasst also vier Oktaven. Die zwölf Töne wurden mit zwölf Grundbausteinen erzeugt und mittels drei Flip-Flops (insgesamt 36) pro Ton aufgeteilt. Soweit ich mich erinnern kann, gingen die Ausgänge dann zu Schaltern, die sich unter den Tasten befanden. Die Schalter wurden dann so angeschlossen, dass immer nur ein Ton zu hören war. Ob das der höchste oder der tiefste Ton der gedrückten Tasten war, weiß ich allerdings nicht mehr. Als Lautsprecher fungierte ein Mikrofon/Lautsprecher-Baustein.

Auf Seite 65 von hobby 4 Band 3 wird auch die Grenzfrequenz des Flip-Flops besprochen. Wenn man nur Steuerungen für alle Arten von Maschinen baut, ist das nicht wirklich wichtig. Für diejenigen, die sich

auch mit Audio, also Klängen, beschäftigen, ist es hilfreich zu wissen, welche Frequenzen die Silberlinge verarbeiten können.

Grenzfrequenz

Für die folgenden Messungen habe ich Ausgang A_2 mit „5“ und „7“ eines Grundbausteins verbunden und einen 100-nF-Kondensator zwischen „-“ und E_1 gelegt. Die maximal einstellbare Frequenz beträgt 22,7 kHz. Ausgang A_1 ist mit CP des Flip-Flops verbunden.

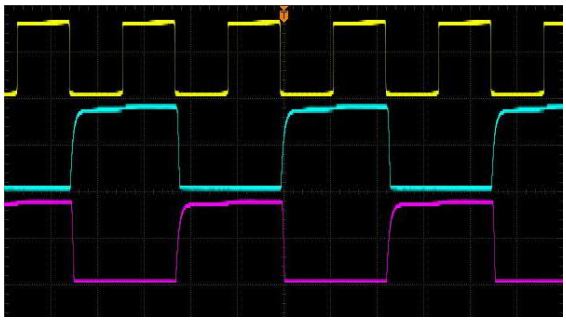


Abb. 121: Zwei-Teilung des Flip-Flops: A_1 (gelb) schwingt mit 2,21 kHz, Q (blau) mit 1,11 kHz

Wenn wir auf Abb. 121, meinen Oszilloskop-Bildschirm, schauen, sehen wir auf Kanal 1 (gelb) den Ausgang A_1 des Grundbausteins. Die Frequenz beträgt 2,21 kHz. Auf Kanal 2 (blau) sehen wir den Q-Ausgang des Flip-Flops und Kanal 3 (violett) zeigt $\overline{Q}$. Die Frequenz beträgt jetzt 1,11 kHz.

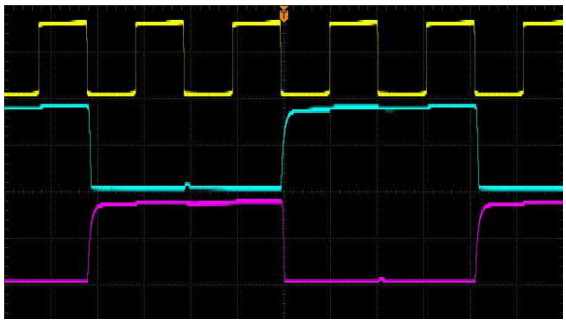


Abb. 122: A_1 = 2,40 kHz, Q = 600 Hz: Teilung durch 2

In Abb. 122 sehen wir wieder Ausgang A_1 auf Kanal 1. Die Frequenz ist diesmal etwas höher: 2,40 kHz. Kanal 2 zeigt den Ausgang Q des Flip-Flops. Wir sehen jedoch jetzt, dass die Frequenz nicht mehr die Hälfte beträgt. Das sollte 2,40 kHz ÷ 2 = 1,20 kHz sein, ist aber jetzt nur noch 600 Hz. Dies bedeutet, dass dieses Flip-Flop eine Grenzfrequenz von 1,11 kHz hat.

In Abb. 121 sehen wir, dass die Breite (Periode) der auf Q sichtbaren Rechteckwelle das Zweifache der Breite der Rechteckwelle auf A_1 beträgt. In Abb. 122 sieht man deutlich, dass die Breite jetzt 4 mal A_1 geworden ist: 2,40 kHz ÷ 4 = 600 Hz.

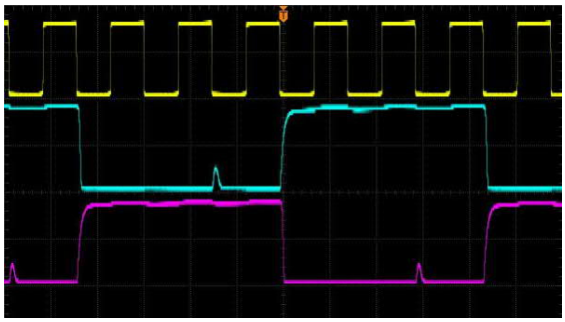


Abb. 123: A_1 = 3,38 kHz, Q = 564 Hz: Teilung durch 6

Für Abb. 123 habe ich die Frequenz auf 3,38 kHz erhöht. Die Rechteckwelle ist jetzt 6-mal so breit wie A_1. Das bedeutet, dass die Frequenz von Q auf 3,38 kHz ÷ 6 = 564 Hz abfällt.

Wenn ich die Frequenz für Abb. 124 weiter erhöhe, sehen wir, dass die Rechteckwelle auf Q nicht mehr symmetrisch ist. Wir sehen jetzt eine Rechteckwelle, deren unterer Teil 3-mal und der obere Teil 4-mal so breit wie A_1 ist. A_1 ist hier 4,03 kHz. Q hat eine Frequenz von 4,03 kHz ÷ (3 + 4) = 576 Hz.

Die maximale Frequenz, die das für diesen Test verwendete Flip-Flop verarbeiten kann, liegt bei etwa 4,17 kHz. In Abb. 125 ist zu erkennen, dass die Rechteckwelle nun wieder symmetrisch ist und die Teilung 8 beträgt: 4,17 kHz ÷ 8 = 513 Hz. Auf Abb. 126 hat sich das Flip-Flop komplett „verirrt“. 4,38 kHz sind also zu viel des Guten.

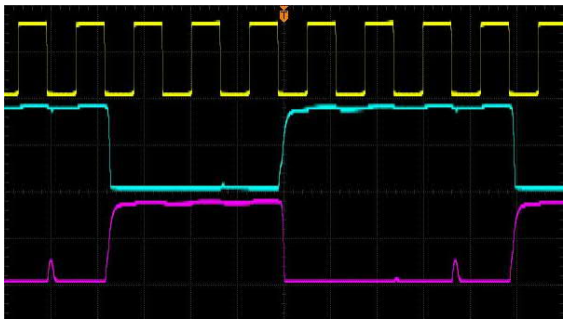


Abb. 124: A_1 = 4,03 kHz, Q = 576 Hz: Die Teilung geht jetzt durch 3 und 4 und ist nicht symmetrisch

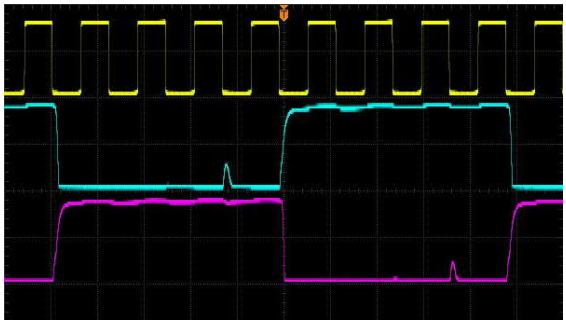


Abb. 125: A_1 = 4,17 kHz, Q = 513 Hz: Teilung durch 8

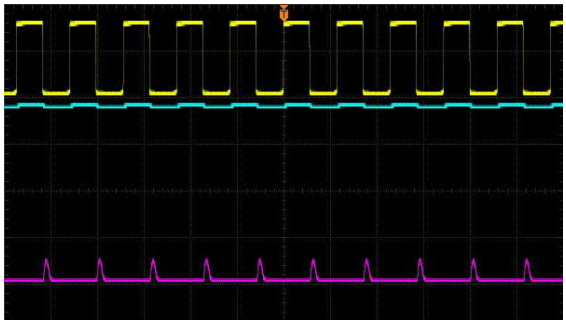


Abb. 126: A_1 = 4,38 kHz: Das Flip-Flop kann nicht mehr folgen

Was für den CP-Eingang gilt, gilt auch für die S_P- und R_P-Eingänge. S_P ist mit A_1 und R_P mit A_2 verbunden. In Abb. 127 sehen wir, dass das Flip-Flop ab 2,5 kHz nicht mehr richtig folgen kann. Die meiste Zeit kann es das zwar noch, beginnt aber bereits, Impulse zu überspringen. Bei 7,35 kHz überspringt das Flip-Flop bereits drei Impulse (siehe Abb. 128). Oberhalb von 7,35 kHz ist hier Schluss.

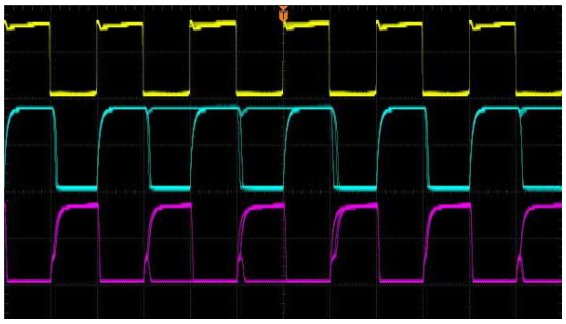


Abb. 127: A_1 = 2,50 kHz: Beide S_P- und R_P-Eingänge können nicht mehr richtig folgen

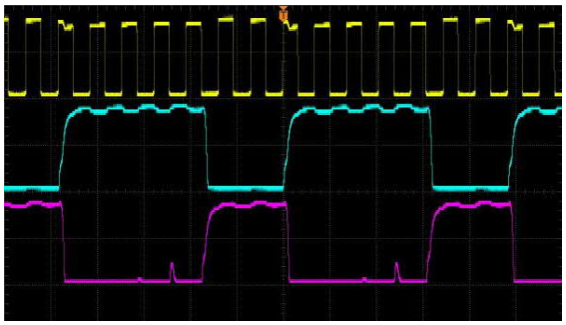


Abb. 128: A_1 = 7,35 kHz: Das Flip-Flop verliert mehrere Impulse

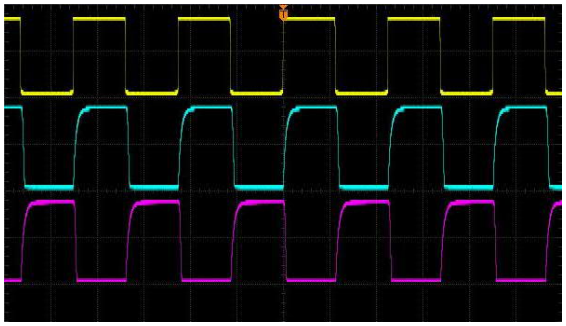


Abb. 129: A_1 = 2,21 kHz: An beiden Ausgängen sieht das Signal aus wie eine Rechteckwelle

Für die Eingänge S_D und R_D sieht es besser aus. In Abb. 129 sehen wir, dass die Rechteckwellen an den beiden Q-Ausgängen bei 2,21 kHz immer noch wie Rechteckwellen aussehen. Abb. 130 zeigt deutlich, dass sich bei einer Frequenz von 22 kHz die Form der Rechteckwelle zu einer Art Sägezahn verändert hat, dessen Amplitude auch noch kleiner geworden ist.

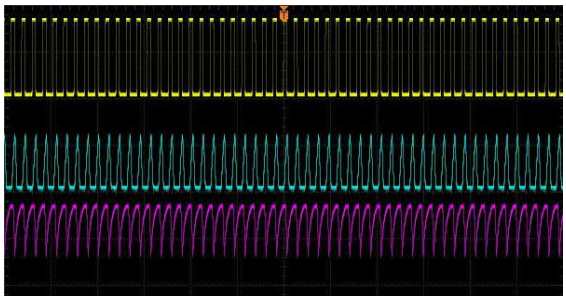


Abb. 130: A_1 = 22 kHz:
An beiden Ausgängen sieht das Signal nicht mehr aus wie ein Rechteck

Noch ein paar Zusatzinfos

Wird der oben erwähnte 100-nF-Kondensator am Grundbaustein auf 1 nF geändert, ist die höchste einzustellende Frequenz 96 kHz. Wird der Kondensator komplett weggelassen, beträgt die Frequenz 117 kHz. Das kann jedoch bei jedem Exemplar anders sein. Um eine stabile Frequenz zu erhalten, ist es jedoch wichtig, dass die Versorgungsspannung für den Grundbaustein von einem 9-V-Stabilisator gespeist wird. Wenn die Spannung durch einen Motor oder ein anziehendes Relais variiert, variiert auch die Frequenz. Ich habe dies mit mehreren Grundbausteinen getestet. Mit dem Grundbaustein mit einem 25-kΩ-Potentiometer, mit einem 1-nF-Kondensator zwischen „-“ und E_1, A_2 an „6“ und „7“ messe ich eine Frequenz von 133 kHz bei 3 V. Bei 6 V ist die auf 99 kHz gefallen, bei 9 V auf 77 kHz und bei 12 V sind es nur noch 68 kHz. Bei der Version mit einem 50-kΩ-Potentiometer sind dies nacheinander: 118 kHz bei 3 V, 71 kHz bei 6 V, 64 kHz bei 9 V und 63 kHz bei 12 V.

Der Mono-Flop kommt nur auf magere 100 Hz.

Die Z_{OR}- und Z_{AND}-Ausgänge von OR-NOR und AND-NAND sind bis 10 kHz noch zuverlässig. Z_{NOR} und Z_{NAND} gehen jedoch nur bis 7,35 kHz. Wird die Frequenz höher, verschlechtern sich Form und Amplitude so stark, dass von einem zuverlässigen Betrieb nicht mehr gesprochen werden kann.

Gedanken

Zwei Tage nach den oben genannten Messungen kam mir die Idee, dass etwas dafür verantwortlich sein muss, dass das Flip-Flop eine so niedrige Grenzfrequenz hat. Normalerweise wird eine Frequenz teilweise durch Kondensatoren in Kombination mit Widerständen bestimmt, in der Elektronikwelt besser bekannt unter dem Namen RC-network.

Wenn wir uns die Schaltung des Flip-Flops ansehen (Abb. 131), sehen wir insgesamt sechs Kondensatoren von jeweils 22 nF. Es dauert eine gewisse Zeit, einen Kondensator vollständig aufzuladen. Diese Zeit wird bestimmt durch das Produkt der Größe der Kapazität und der des Widerstandes und, wie wir oben gesehen haben, auch durch die Größe der Spannung. Es versteht sich, dass sich ein Kondensator mit geringer Kapazität schneller auflädt als einer mit hoher Kapazität. Ist der Widerstandswert niedrig, wird schneller geladen als bei einem hohen Widerstandswert. Das Aufladen auf beispielsweise 3 V geht schneller als auf 12 V. Das gleiche gilt fürs Entladen. Genau das wird beim Mono-Flop verwendet.

Ich habe mich jedoch auf die Kondensatoren konzentriert. Um zu überprüfen, ob meine Gedanken richtig waren, baute ich ein weiteres Flip-Flop. Ich hatte noch ein paar Platinen herumliegen. Ich habe jedoch alle sechs 22-nF-Kondensatoren auf 10 nF gesenkt. Das Ergebnis davon ist in Abb. 132 zu sehen.

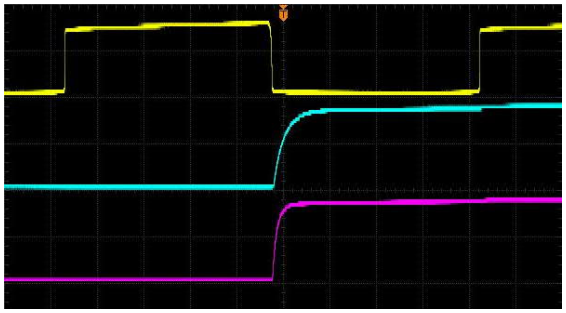


Abb. 132: A_1 = 2,21 kHz:
Rise Time 26 µs für Q mit 22 nF (blau) und 12,5 µs für Q mit 10 nF (violett)

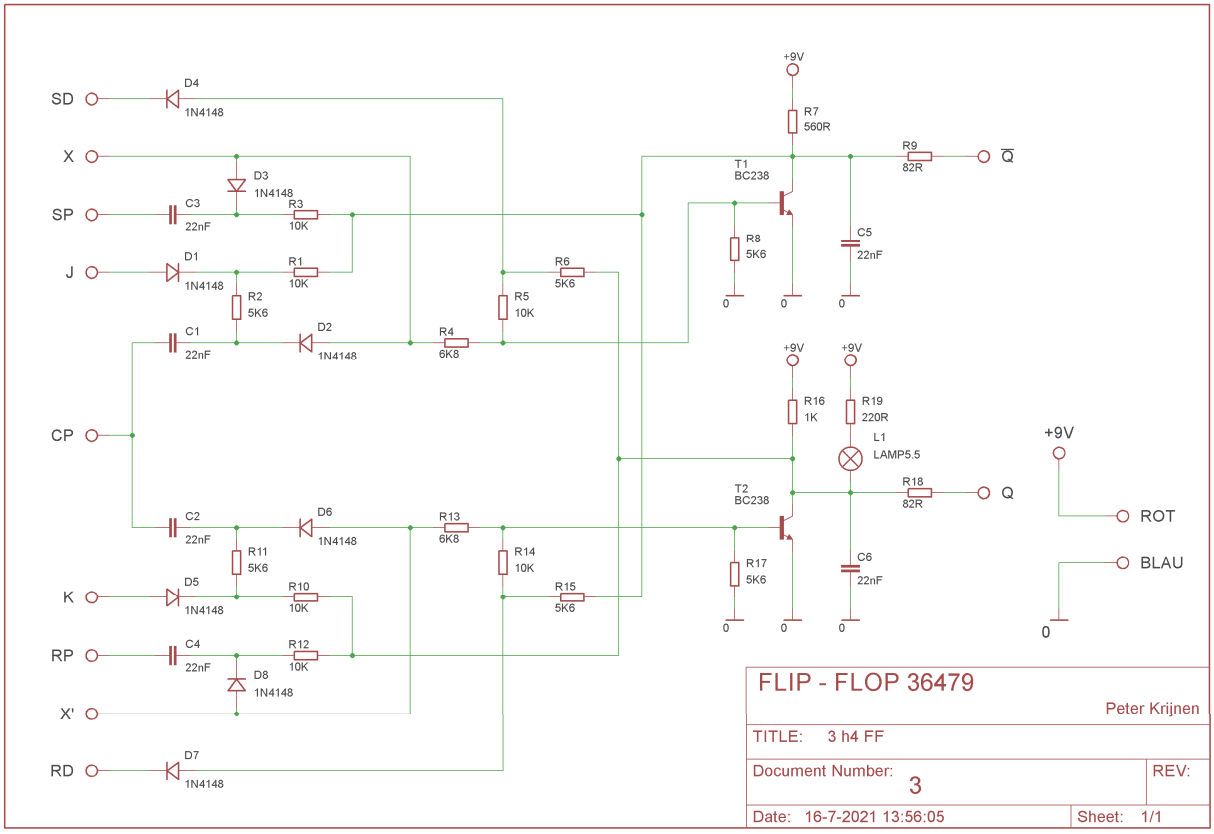


Abb. 131: Schaltbild des Flip-Flops

Kanal 1 (gelb) ist wieder A_1 des Grundbausteins, Kanal 2 (blau) ist Q des Flip-Flops mit den 22-nF-Kondensatoren und Kanal 3 (violett) ist Q der Version mit 10 nF. Mein Oszilloskop kann die Ladezeit, besser bekannt als Anstiegszeit oder auf Englisch „Rise Time“, selbst messen. Um den Unterschied der Anstiegszeiten der beiden Flip-Flops genauer messen zu können, musste ich auf 50 µs pro Teilung „hineinzoomen“. Es wurde deutlich, dass die Anstiegszeit des Grundbausteins am Ausgang A_1 bei einer Frequenz von 2,21 kHz weniger als 2 µs beträgt. Abb. 132 zeigt deutlich, dass die Anstiegszeit der 10-nF-Version mit einer Zeit von 12,5 µs geringer ist als die 26 µs der Version mit 22 nF.

Für Abb. 133 habe ich das Oszilloskop wieder auf 200 µs pro Teilung eingestellt. Damit sich die beiden Flip-Flops nicht gegenseitig beeinflussen, habe ich das zweite Flip-Flop für weitere Messungen an A_2 des Grundbausteins angeschlossen. Auf Abb. 133 sehen wir dasselbe wie auf Abb. 122, wobei Kanal 3 um einen halben Impuls verschoben ist.

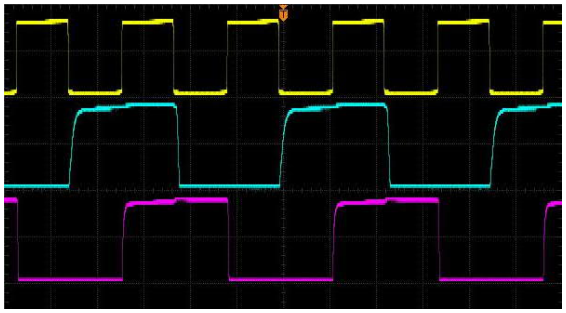


Abb. 133: A_1 = 2,21 kHz: An den Q-Ausgängen beider Flip-Flops sieht das Signal gut aus

Wenn wir nun die Frequenz auf 2,4 kHz erhöhen, sehen wir auf Abb. 134, dass das Flip-Flop mit den 22 nF (Kanal 2) bereits wieder Impulse überspringt, während das Flip-Flop mit den 10 nF (Kanal 3) noch ordentlich durch zwei teilt. Bei weiterer Erhöhung der Frequenz kann die 10-nF-Version bis zu 6,41 kHz ordentlich folgen,

siehe Abb. 135. Bei 6,58 kHz geht auch die 10-nF-Version in die Knie (Abb. 136). Bei 11,9 kHz am CP-Eingang ist auch die 10-nF-Version am Ende (Abb. 137).

An die S_P- und R_P-Eingänge angeschlossen, läuft das Flip-Flop bis zu 5 kHz reibungslos. Oberhalb von 5 kHz endet es auch hier. An den S_D- und R_D-Eingängen sieht es genauso aus wie in Abb. 130. Die Amplitude ist jedoch dieselbe wie bei A_1.

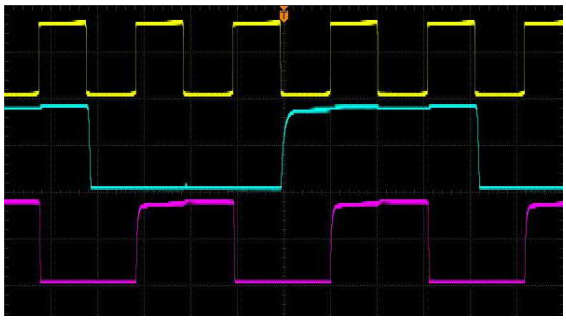


Abb. 134: A_1 = 2,40 kHz:
Q mit 22 nF (blau) verliert Impulse,
Q mit 10 nF (violett) nicht

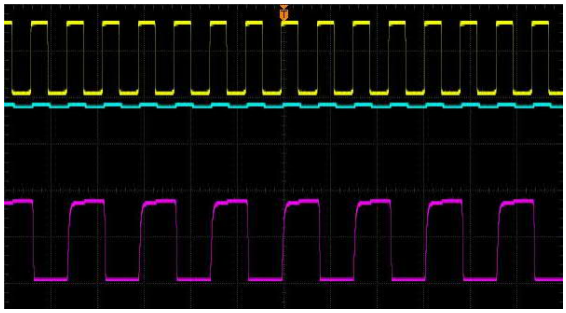


Abb. 135: A_1 = 6,41 kHz:
Q mit 22 nF (blau) ist am Ende,
Q mit 10 nF (violett) noch nicht

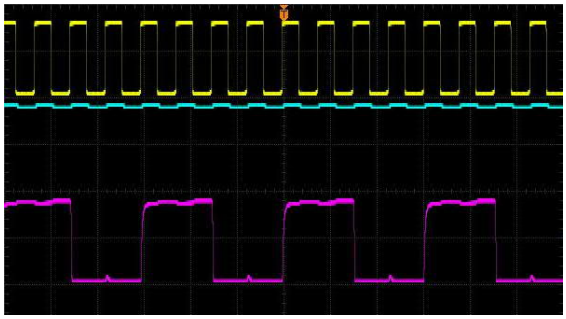


Abb. 136: A_1 = 6,58 kHz:
Q mit 22 nF (blau) ist am Ende,
Q mit 10 nF (violett) verliert nun auch Impulse

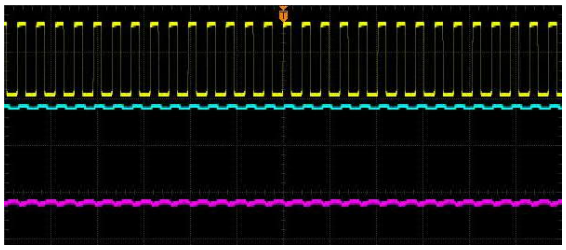


Abb. 137: A_1 = 11,9 kHz:
Q mit 10 nF (violett) ist nun auch am Ende

Nachbau

Ich habe auch das Flip-Flop auseinandergenommen, um den Schaltplan und das Layout zu übernehmen. Abb. 138 zeigt die Komponentenanordnung und Abb. 139 zeigt die Leiterbahnseite. Das Layout zeigt Abb. 140. Für dieses Modul habe ich auch ein alternatives Layout entwickelt (Abb. 141).

Abb. 138: Platine des Flip-Flops

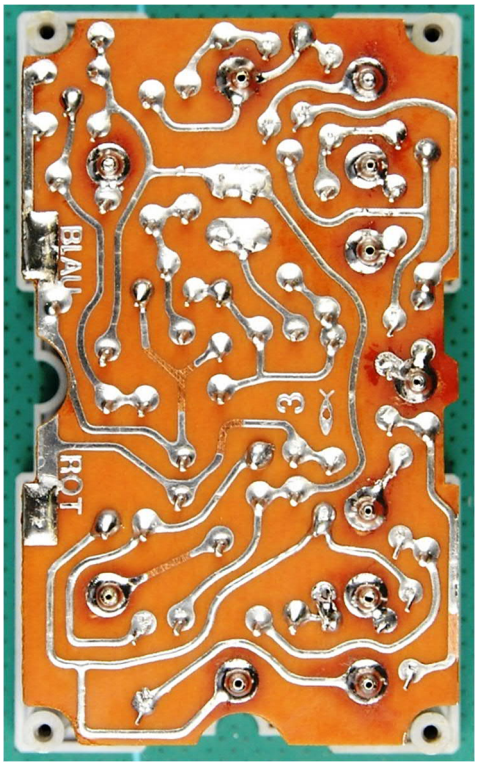


Abb. 139: Leiterbahnseite des Flip-Flops

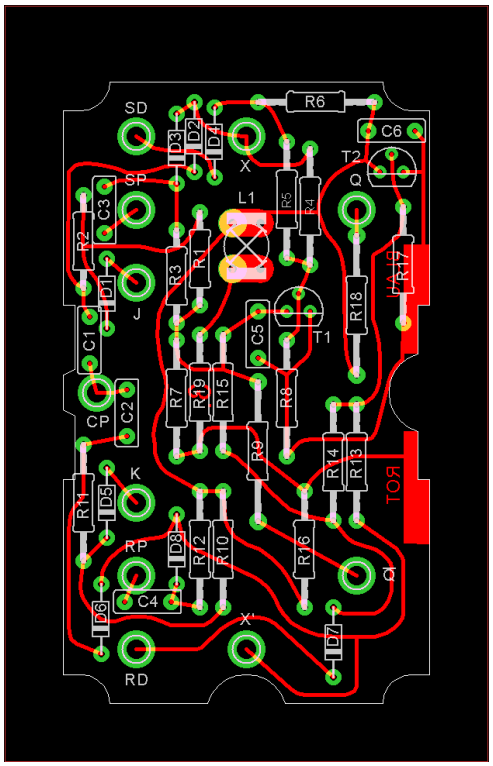


Abb. 140: Layout des Flip-Flops

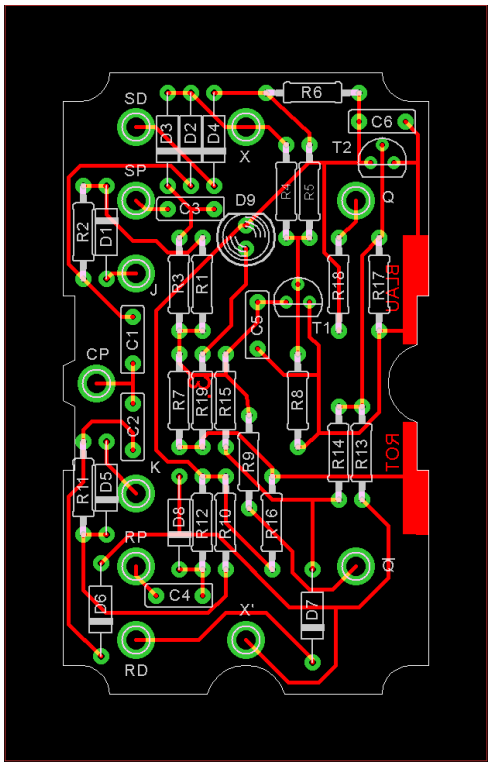


Abb. 141: Nachbau 1: Ersatz-Layout zum Einbau in das Original-Silberlinggehäuse

Das Ergebnis des Nachbaus gefällt mir sehr gut. Schauen wir uns noch einmal den Schaltplan in Abb. 131 an. Wenn wir eine Linie von CP aus ziehen, sehen wir, dass die untere Hälfte mit der oberen Hälfte identisch ist. Das Bild wird jedoch um die Linie von CP gespiegelt. Lässt man das Lämpchen mit Widerstand R19 weg, sind nur noch die beiden Widerstände R7 und R16 unterschiedlich. Dies ist in Abb. 143 viel besser zu sehen. Hätte ich die LED in der Mitte platziert und die Widerstände R7 und R19 vertauscht, wäre die Symmetrie perfekt gewesen. Trotzdem habe ich die LED am Originalplatz belassen.

Die Abb. 142, 143 und 144 zeigen meine Version für mein alternatives 45 × 75-Gehäuse.

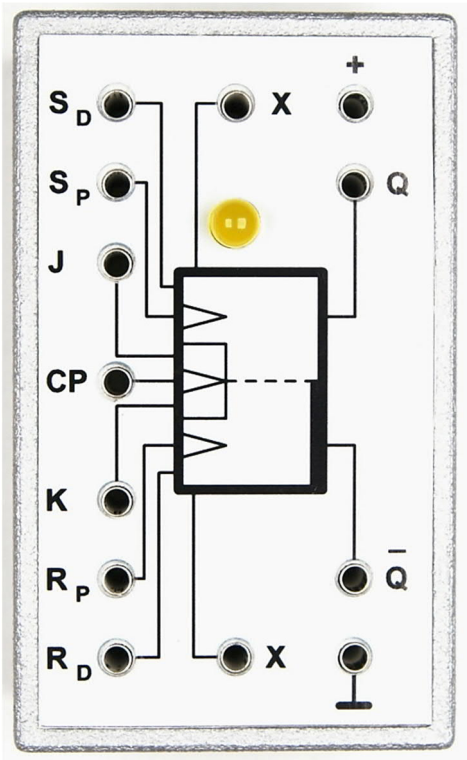


Abb. 142: Nachbau 2: Frontplatte für das 45 × 75-Gehäuse

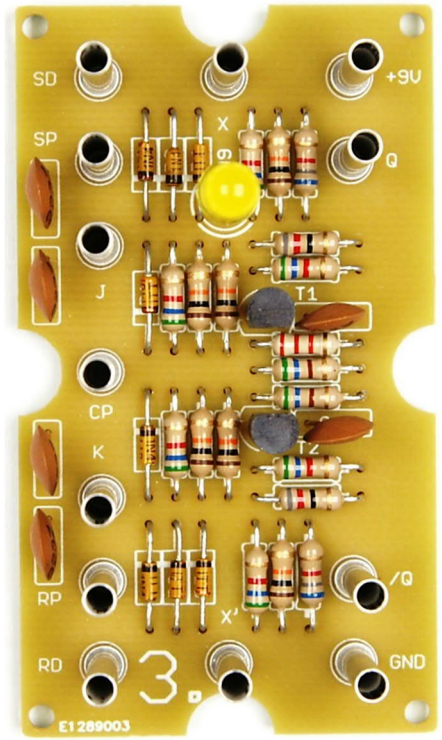


Abb. 143: Nachbau 2: Platine in das 45 × 75-Gehäuse

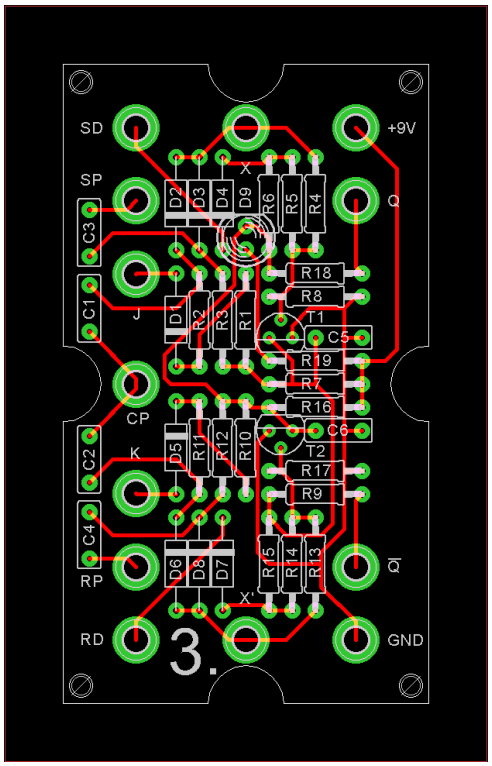


Abb. 144: Nachbau 2: Layout für mein 45 × 75-Gehäuse

Die Abb. 145, 146 und 147 zeigen eine Ausführung für die 60 × 60-Kassette (32076).

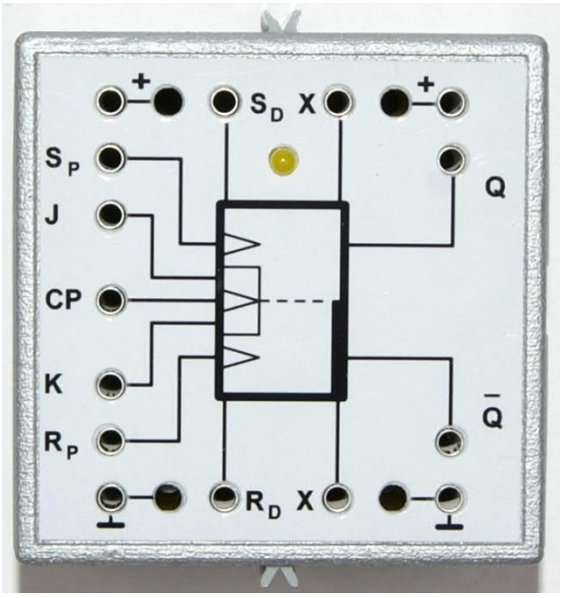


Abb. 145: Nachbau 3: Frontplatte für das 60 × 60-Gehäuse

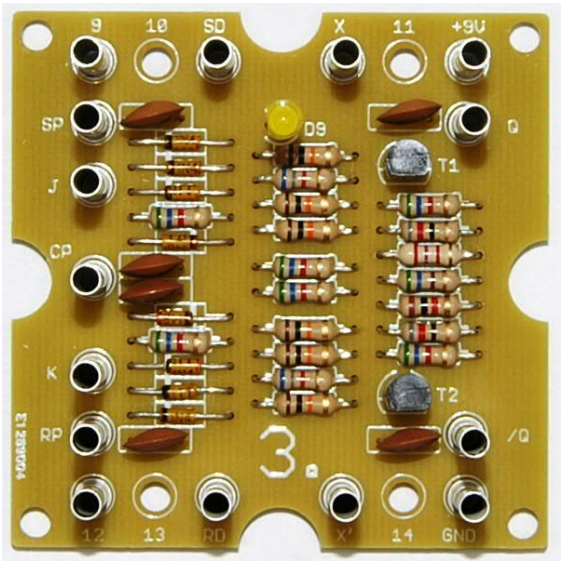


Abb. 146: Nachbau 3: Platine für das 60 × 60-Gehäuse

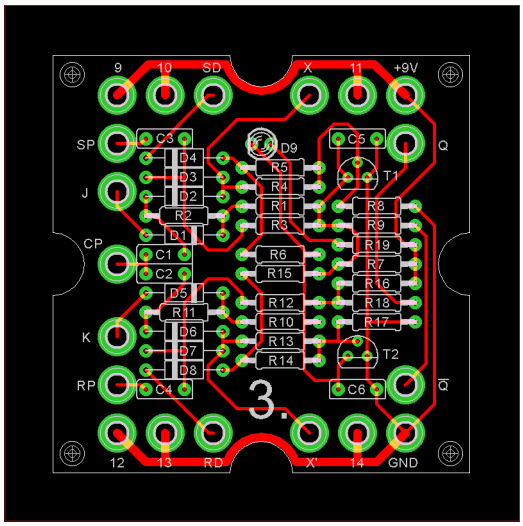


Abb. 147: Nachbau 3: Layout für 60er Kassette (32076)

Abb. 148, 149 und 150 zeigen die Version für das Batteriegehäuse (32263). Auf den beiden Abb. 151 und 152 ist die Shield-Version zu sehen.

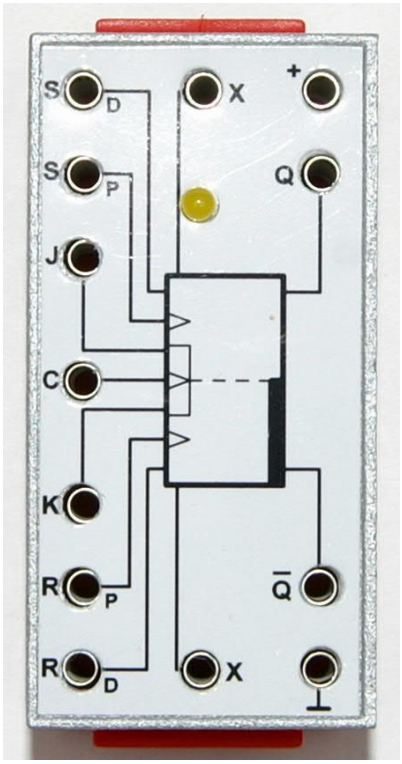


Abb. 148: Nachbau 4: Frontplatte für das Batteriegehäuse (32263)

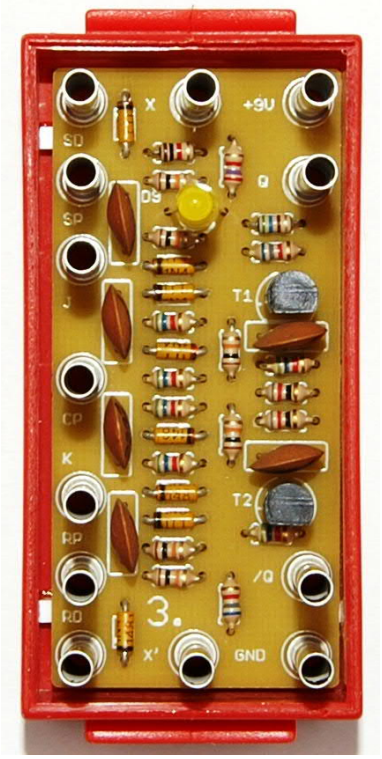


Abb. 149: Nachbau 4: Platine in das 30 × 60-Gehäuse

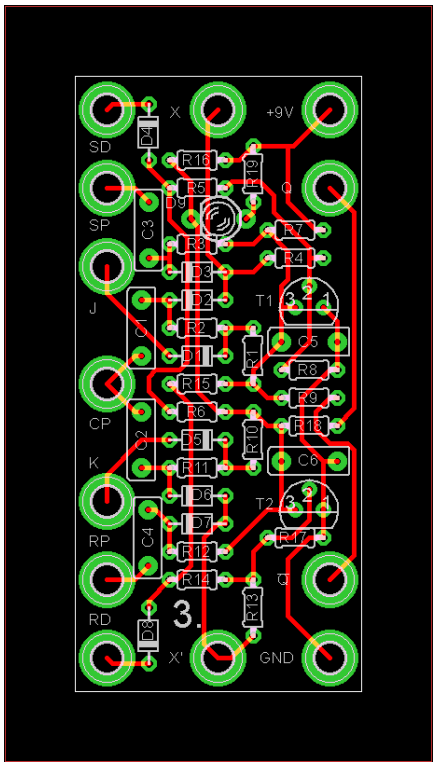


Abb. 150: Nachbau 4: Layout für das Batteriegehäuse (32263)

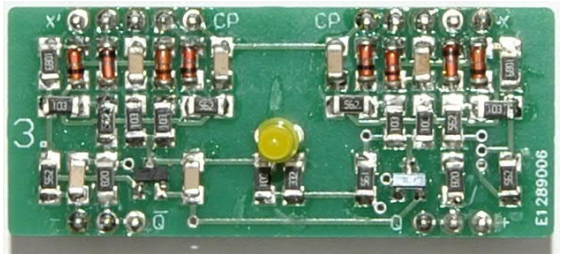


Abb. 151: Nachbau 5 für ein Breadboard: Flip-Flop als Shield in SMD Technik

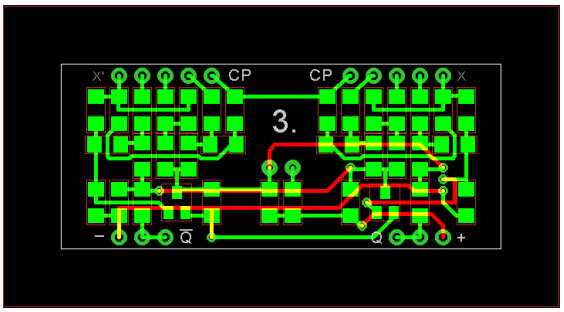


Abb. 152: Nachbau 5: Doppelseitiges Layout des Shields

Gibt es noch etwas zu erwähnen? Ja!

Wenn wir uns den Schaltplan noch einmal ansehen, sehen wir, dass der Widerstand R19 in Reihe mit der Lampe liegt. Diese Kombination liegt dann parallel zum Widerstand R16. Beim Nachbau wurde die Lampe durch eine LED ersetzt. Da eine LED aber einen höheren Vorwiderstand benötigt als eine Lampe, wird der Endwiderstand von LED + R19 parallel zu R16 zu hoch. Das heißt, im Gegensatz zum Original leuchtet die LED bereits in dem Moment, in dem das Modul mit Spannung versorgt wird. Um zur ursprünglichen Ausgangslage zu gelangen, müssen wir den Widerstand R16 auf 560 Ω absenken.

Die beiden X-Anschlüsse können beim Nachbau entfallen. Diese werden nur in Verbindung mit dem Dyn.-AND-Baustein verwendet und haben dann grundsätzlich die gleiche Wirkung wie die S_P- und R_P-Eingänge.

Damit komme ich zum Ende meines Beitrags. In der nächsten Folge werde ich auf das Mono-Flop eingehen.

Quellen

[1] fischertechnik: *Experimente + Modelle hobby 4 Band 3*. Auf der Website des fischertechnik-Clubs NL.

Elektronik

Der Zauberling (Teil 2): Das Zauberbuch

Arnoud van Delden

Ein erfolgreicher Zauberer muss einen guten Zauberstab, bezaubernde Kleidung und wundervolle magische Gegenstände haben. Die „Hardware“ muss in Ordnung sein. Aber um die wahren Wunder zu vollbringen, ist die Neugier an Zaubersprüchen und Beschwörungen entscheidend. Der Zauberspruch ist komplett mit der richtigen Zauberformel und genau die Kombination aus Zauberstab und Zaubersprüchen enthält die wahre Magie. Höchste Zeit also, einen Blick in das große experimentelle Zauberbuch zu werfen: die Software des Zauberlings.

Das Schöne am Arduino Pro Mini Atmega328P ist, dass er einfach aus der Arduino IDE heraus programmiert werden kann, sobald die Brücke zur TTL-Schnittstelle der kleinen Platine hergestellt ist. Auf diese Weise kann die Software grundsätzlich immer in Entwicklung bleiben und einfach erweitert oder angepasst werden. Nachdem ich einige Testprogramme zur Überprüfung der Ansteuerung der Motor- und Servoausgänge erstellt hatte, begann ich, die allgemeine Funktionalität der E-Tec- und Elektronik-Module nachzubilden. Da die Software des Zauberlings so einfach einzustellen ist, habe ich das „Grundprogramm“ gleich in einigen Punkten angepasst. In Teil 3 dieser Artikelreihe wird dabei eine Anwendung betrieben. In diesem Teil lest ihr, welche anderen Tricks der Zauberling bereits beherrscht.

Programme und Funktionen

Auf der Vorderseite des Zauberlings kann über DIP-Schiebeschalter eingestellt werden, welches „Programm“ bzw. welche Steuerfunktion ausgeführt werden soll. Damit ist wie beim E-Tec-Modul ein schnelles Umschalten zwischen verschiedenen Programmen möglich. Dies ist für ein programmierbares Modul wie den Zauberling während der Softwareentwicklung besonders nützlich. Während beispielsweise an einer verbesserten Version einer bestimmten Funktionalität gearbeitet wird, kann eine frühere stabile Version dieser Funktionalität auch einfach „unter der Schaltfläche“ gehalten werden.

Der Zauberling ist noch ein erster Prototyp und die Software befindet sich daher noch in der Entwicklung. Die folgende Tabelle zeigt das bereits Erreichte in verschiedenen Perfektionsstufen: Das „Basisprogramm“ ist unter anderem aus dem E-Tec-Modul bekannt und benötigt drei Schalter an den Eingängen. Wenn Schalter 1 gedrückt wird, beginnt der Motor sich im Uhrzeigersinn zu drehen. Wenn Schalter 2 gedrückt wird, beginnt der Motor gegen den Uhrzeigersinn zu drehen. Durch Drücken des Schalters 3 wird der Motor gestoppt. Eine kleine Änderung, die ich vorgenommen habe, ist, dass der Motor durch erneutes Drücken des Schalters 3 die letzte Drehrichtung nach dem Stopp wieder aufnimmt.

Prog	DIP1	DIP2	DIP3	DIP4	I1 Logik	I2 Logik	I2 Logik	Funktion	Potentiometer
0	-	-	-	-	Negativ	Negativ	Negativ	Grundprogramm mit gleich-mäßiger Drehrichtungs-änderungen	Zeit 0~2500ms
1	-	-	-	ON	Positiv	Positiv	Positiv	Grundprogramm mit gleich-mäßiger Drehrichtungs-änderungen	Zeit 0~2500ms
2	-	-	ON	-	Auto	Auto	Auto	AND/NAND−Logikglied	Schwellwert Sensoren
3	-	-	ON	ON	Auto	Auto	Auto	OR/NOR−Logikglied	Schwellwert Sensoren
4	-	ON	-	-	Auto	Auto	Auto	XOR/XNOR−Logikglied	Schwellwert Sensoren
5	-	ON	-	ON	Auto	Auto	Auto	SR-Flip Flop - SET=IN1, CLK=IN2, RESET=IN3 (R ist dominant/Ausgangszustand)	Schwellwert Sensoren
6	-	ON	ON	-	Auto	Auto	Auto	JK-Flip Flop - J=IN1, CLK=IN2, K=IN3	Schwellwert Sensoren
7	-	ON	ON	ON	Auto	Auto	Auto	D-Flip Flop mit Reset	Schwellwert Sensoren
8	ON	-	-	-	Auto	Auto	Auto	Monoflop mit Reset (Zeit mit Potentiometer einstellbar)	Pulszeit 10~2500ms
9	ON	-	-	ON	Auto	Auto	Auto	Blinklicht (Duty-Cycle 50%, Geschwindigkeit mit Potentio-meter einstellbar)	Frequenz
10	ON	-	ON	-	Auto	Auto	Auto	Blitzlicht (mehrere kurze Blitze hintereinander, Geschwindigkeit einstellbar)	Frequenz
11	ON	-	ON	ON	-	-	-	Zukünftige Projekte :-)	-
12	ON	ON	-	-	-	-	-	Zukünftige Projekte :-)	-
13	ON	ON	-	ON	-	-	-	Zukünftige Projekte :-)	-
14	ON	ON	ON	-	-	-	-	Demo: Drehrichtungen der Motoren (Und für zukünftige Projekte)	-
15	ON	ON	ON	ON	-	-	-	Demo: Servobewegungen (Und für zukünftige Projekte)	-
	bit3	bit2	bit1	bit0					

- Positive Logik: LOW → HIGH-Übergang ist binär 1 (für aktive Sensoren oder Signale: Eingang DIP-Schalter OFF)
- Negative Logik: HIGH → LOW-Übergang ist binär 1 (für passive Sensoren Eingang DIP-Schalter ON, für aktive Sensoren OFF)
- Auto: Zustand beim Einschalten ist Normalzustand, Änderung ist Schaltzustand. Schaltpegel & Hysterese mit Poti einstellbar

Tab. 1: Programme und Funktionen des „Zauberlings“

Positive und negative Logik

In der Regel ist es wichtig zu wissen, ob ein Schalter oder Sensor am Eingang in Ruhestellung geöffnet oder geschlossen ist und somit das Erfassungsverhalten zu interpretieren. Bei einem Programm wie dem „Grundprogramm“ im E-Tec-Modul ist dies beispielsweise sinnvoll, weil beim Einschalten des Moduls möglicherweise schon einer der Schalter gedrückt wurde.

Schließlich muss die Bewegungsrichtung dann so gewählt werden, dass dieser Schalter losgelassen wird, ohne den Bewegungsbereich zu verlassen.

Andererseits ist es schön, sowohl Spannungsabfälle als auch Spannungsanstiege als logische Erkennungspegel an den Eingängen verwenden zu können. Beispiele für Sensoren, die einen Spannungsanstieg verursachen, wenn sie melden müssen, sind beispielsweise die Fototransistoren aus einem Leuchtkasten oder ein Schalter, der in Ruhestellung geerdet ist.

Manche Sensoren reagieren jedoch umgekehrt, wie z. B. das Ausgangssignal eines aktiven IR-Hindernissensors im Ruhezustand 5 V beträgt und dieses bei Erkennen eines Hindernisses auf 0 V abfällt. Man könnte dies „negative Logik“ nennen.

Aus diesem Grund hatte ich mir die Herausforderung gestellt, die Verarbeitung der

```
void setup() {
  …
  in1Default = analogRead(IN1);
  in2Default = analogRead(IN2);
  in3Default = analogRead(IN3);
  …
  …
}

void loop() {
  …
  …
  potValue = analogRead(POTMETER);
  sensorTreshold = map(potValue, MINREG, MAXREG, 1, TRESHOLD_MAX);
  …
  …
  // Read inputs ,take hysteresis into account
  in1Value = analogRead(IN1);
  if (abs(in1Value-in1Default) > sensorTreshold) {
    in1ValueTrigger = in1Value;
    in1Active = true;
  } else {
    if (abs(in1Value-in1ValueTrigger) > sensorTreshold/2) { // Hysteresis...
      in1Active = false;
      in1TriggerUsed = false;
    }
  }
  in2Value = analogRead(IN2);
  if (abs(in2Value-in2Default) > sensorTreshold) {
    in2ValueTrigger = in2Value;
    in2Active = true;
  } else {
    if (abs(in2Value-in2ValueTrigger) > sensorTreshold/2) { // Hysteresis...
      in2Active = false;
      in2TriggerUsed = false;
    }
  }
  in3Value = analogRead(IN3);
  if (abs(in3Value-in3Default) > sensorTreshold) {
    in3ValueTrigger = in3Value;
    in3Active = true;
  } else {
    if (abs(in3Value-in3ValueTrigger) > sensorTreshold/2) { // Hysteresis...
      in3Active = false;
      in3TriggerUsed = false;
    }
  }
  …
  …
}
```

Listing 1: Messung der Grundlinienwerte und Umgang mit Schwellenwert und Hysterese

Eingangssignale des Zauberlings möglichst einheitlich zu halten. Das Modul lässt sich flexibler einsetzen, wenn (fast) alle Arten von Sensoren und Logik an den Eingängen austauschbar verwendet werden können. Denn als Schaltimpuls reicht meist nur die Signalisierung der abweichenden Ausgangsstellung.

Es ist möglich, beim Einschalten (oder nach einem Reset) des Zauberlings die gemessenen Eingangswerte als Referenzwerte zu speichern. Mit dem frontseitigen Potentiometer kann ein Schwellenwert eingestellt werden. Ein Eingang wird „aktiv“, sobald der entsprechende Referenzwert um mehr als den Schwellenwert überschritten wird. Dabei spielt es keine Rolle, ob der Wert nach oben oder unten vom Referenzwert abweicht. Der eingestellte Schwellenwert dient auch als Maß für die Hysterese bei Sensoren, die nach der Erkennung erst allmählich auf ihren Ausgangswert zurückkehren. Dadurch wird verhindert, dass der Eingang „flattert“, wenn sich der Ausgangswert des Sensors in den Bereich um die Erkennungsschwelle bewegt. Auf diese Weise wird das Erfassungsverhalten eines langsam zum Referenzwert zurückkehrenden Sensors verbessert.

Ich habe die Software, die wie gesagt noch eine Weile weiterentwickelt wird, auf meiner Webseite veröffentlicht. Der Code-Schnipsel in Listing 1 zeigt (stark vereinfacht) die Messung der Grundlinienwerte und wie mit Schwellenwert und Hysterese umgegangen wird.

Dieser Trick macht es möglich, die Eingänge des Zauberlings mit den unterschiedlichsten Sensoren zu nutzen. Auch aktive Sensoren wie der IR-Hindernissensor, der PIR-Bewegungssensor oder ein Hall-Effekt-Sensor können verwendet werden. Einzige Bedingung ist, dass der Referenzwert zu Beginn mit einem „nicht aktivierten“ Sensor gemessen werden kann. Da die meisten Steuerungen die Sensoren hauptsächlich als Detektoren verwenden, wird diese Forderung in der Praxis oft erfüllt.

Aber natürlich kann es spezielle Ausnahmen geben. Bei einigen Steuerungen (wie zum Beispiel dem „Basisprogramm“) ist es besser, wenn die Art des Sensors und das Verhalten fest gewählt werden können. Die Software bietet diese Auswahl auch explizit für das „Basisprogramm“ in zwei Varianten an, so dass es sowohl mit konventionellen Sensoren mit positiver Logik (wie der bekannten Lichtschranke mit Fototransistor) als auch mit Sensoren mit negativer Logik verwendet werden kann.

Logikprüfung

Bei beiden Varianten des „Grundprogramms“ wird geprüft, ob die Sensoren beim Start ihre Kennlinie erfüllen (also z. B. eine Unterspannung am jeweiligen Eingang gemessen wird, wenn das „Grundprogramm“ in positiver Logik verwendet wird). Es darf nur ein Sensor/Eingang abweichen, von dem angenommen wird, dass er sich bereits beim Start meldet und nicht in der nicht aktivierten Ruheposition startet. Sobald sich die Eingangspegel des Sensors ändern, wird der Sensor im Laufe des Programms neu initialisiert, wobei der Referenzwert gemessen und gespeichert wird.

Weichen bei Verwendung des „Grundprogramms“ mehr als ein Sensor beim Start ab, erfolgt eine Fehlermeldung durch gleichzeitiges Blinken der drei LEDs auf der Frontseite. In diesem Fall sollte die Verkabelung überprüft werden. In den meisten Fällen reicht es aus, den mechanischen Schalter 3 (Start/Stopp) von „Öffner“ auf „Schließer“ (oder umgekehrt) zu stecken, um das Modul zurückzusetzen.

Schaltrauschen

Was in Software erledigt werden kann, muss nicht in Hardware erledigt werden. Beim Studium des Schaltplans in Teil 1 wäre dem aufmerksamen Elektroniker aufgefallen, dass an den Eingängen keine Entstörkondensatoren eingebaut sind. Für jeden Eingang wird in der Software ein Semaphor *inxTriggerUsed* zurückgesetzt (siehe Codefragment oben), der in der Verarbeitungsroutine eingestellt werden kann, um zu signalisieren, dass die auslösende Flanke des Signals „bearbeitet“ wurde. Dies ignoriert alle falschen zusätzlichen Schaltimpulse oder Rauschen um den mechanischen Schaltpunkt, beispielsweise eines Schalters, herum.

Graduelle Geschwindigkeitskurve

Einer der kleinen Ärgernisse des Standard-Grundprogramms war für mich, dass der Motor beim Einschalten nicht gleichmäßig anspringt oder beim Ausschalten eine sanfte Rampe hat. Und gerade an den Stellen, an denen sich die Drehrichtung abrupt ändert, wäre ein allmählicher Verlauf angenehmer, um Verschleiß am Motor und unnötige Kräfte am Modell zu vermeiden.

Statt einer harten Umpolung wie in Abb. 1 wäre ein gradueller Verlauf gemäß Abb. 2 wünschenswert.

Der Zauberling ist nun dazu in der Lage und das „Basisprogramm“ wurde entsprechend angepasst. Allerdings erweist sich das Drehmoment des Motors bei niedrigen Drehzahlen und um den Stillstand in der Praxis als so gering, dass die Kurve nicht exakt verfolgt werden kann. Daher konnte die Kurve problemlos durch eine Gerade ersetzt werden. Dies hat die Software offensichtlich noch einfacher gemacht.

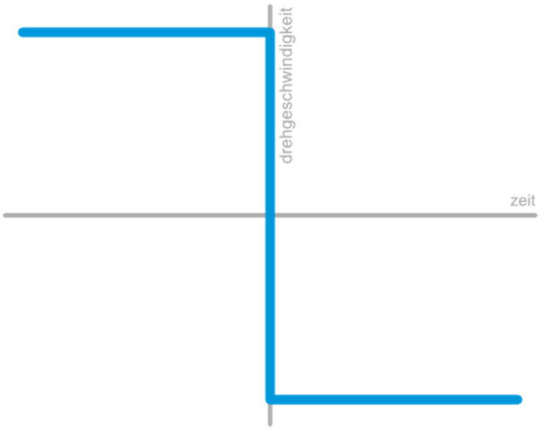


Abb. 1: Einfache harte Polaritätsumkehr

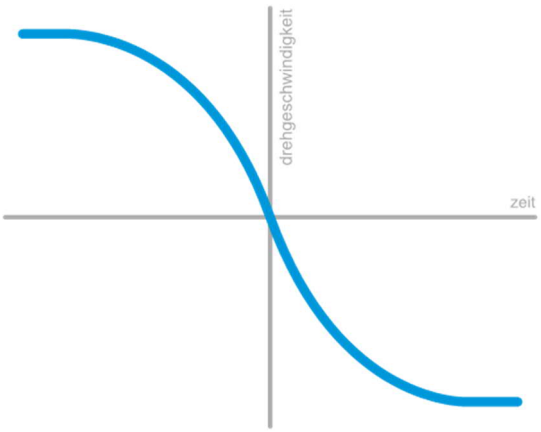


Abb. 2: Allmählicher Drehzahlverlauf

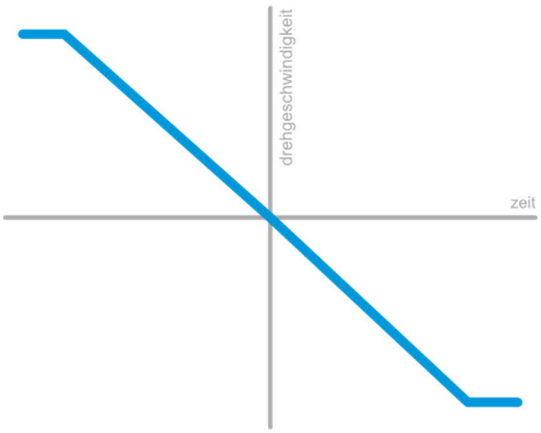


Abb. 3: Vereinfachte allmähliche Rotationsrate

Im (vereinfachten) Codefragment in Listing 2 wird zunächst die Anzahl der Schritte bestimmt, in denen die Drehzahl und/oder die Drehrichtung erfolgen müssen. Dies erfolgt anhand des Wertes, der mit dem Potentiometer auf der Frontseite eingestellt werden kann. Wird dieser ganz nach links (auf null) gedreht, verhält sich der Zauberling wie vom E-Tec Modul bekannt. Je weiter das Potentiometer nach rechts gedreht wird, desto langsamer werden die Geschwindigkeits- und Richtungsänderungen. Die Zeit dafür ist von 0 bis 2,5 Sekunden einstellbar. In dem von mir erstellten Video seht ihr, wie butterweich der Motor ausgeht und wieder anspringt.

```
void setMotorSpeed(bool rotateDir, int motorSpeed, bool smooth) {
  // rotateDir = true means O1/O2 CW rotation
  // rotateDir = false is O1/O2 CCW, output O3/O4 is reversed
  // motorSpeed is a value between 0 and 512
  // smooth = potmeter setting is value of gracefully descend and ascend
  int tempSpeed;
  int speedStep;

  if (rotateDir!=currentMotorDir || motorSpeed!=currentMotorSpeed) {
    int smoothDelay = timeDelay/25;
    if (currentMotorDir == rotateDir) { // No directional changes...
      speedStep = (motorSpeed-currentMotorSpeed)/SPEED_STEPS;
      tempSpeed = currentMotorSpeed;
      do {
        tempSpeed += speedStep;
        if (abs(motorSpeed-tempSpeed) <= abs(speedStep)) tempSpeed = motorSpeed; // Last
                                                                                 // step...
        delay(smoothDelay);
        if (rotateDir) {
          Out1.drive(tempSpeed);
          Out2.drive(-tempSpeed);
        } else {
          Out1.drive(-tempSpeed);
          Out2.drive(tempSpeed);
        }
      } while (abs(motorSpeed-tempSpeed) > abs(speedStep));
    } else { // Towards and through zero...
      for (tempSpeed=currentMotorSpeed; tempSpeed>0;
           tempSpeed-=(currentMotorSpeed/SPEED_STEPS)) {
        // Slow down to zero...
        delay(smoothDelay);
        if (currentMotorDir) {
          Out1.drive(tempSpeed);
          Out2.drive(-tempSpeed);
        } else {
          Out1.drive(-tempSpeed);
          Out2.drive(tempSpeed);
        }
      }
      speedStep = motorSpeed/SPEED_STEPS; // Always positive...
      for (tempSpeed=0; tempSpeed<=motorSpeed; tempSpeed += (motorSpeed/SPEED_STEPS)) {
        // Build up to motorSpeed again...
        if (abs(motorSpeed-tempSpeed) <= speedStep) tempSpeed = motorSpeed; // Last step...
        delay(smoothDelay);
        if (rotateDir) {
          Out1.drive(tempSpeed);
          Out2.drive(-tempSpeed);
        } else {
          Out1.drive(-tempSpeed);
          Out2.drive(tempSpeed);
        }
      }
    }
    currentMotorDir = rotateDir;
    currentMotorSpeed = motorSpeed;
  }
}
```

Listing 2: Code für die Umschalt-Rampe

Denkt daran, dass diese Motorabschaltung nach der Sensorerkennung erfolgt. Die bewegliche Konstruktion muss dies natürlich ermöglichen und darf nicht hängen bleiben oder kollidieren. In der Praxis kann dies jedoch bei der Wahl der Position der Sensoren leicht berücksichtigt werden. Ein zusätzlicher Vorteil könnte sogar darin bestehen, dass diese Eigenschaft genutzt werden kann, um die maximale Auslenkung eines sich bewegenden Schlittens einzustellen, ohne die Sensoren zu bewegen.

Der Zauberling kann noch nicht mit Schrittmotoren umgehen, daher wird die zurückgelegte Strecke momentan nur anhand der eingestellten Zeit ermittelt. Das Ergebnis wird bei verschiedenen Getrieben und Lastzuständen des Motors variieren. Sollte eine nächste Inkarnation des Zauberlings mehr Inputs erhalten, könnten diese für die Rückmeldung eines Encoder-Motors verwendet werden. Aber im Moment hielt ich das alles nicht für nötig.

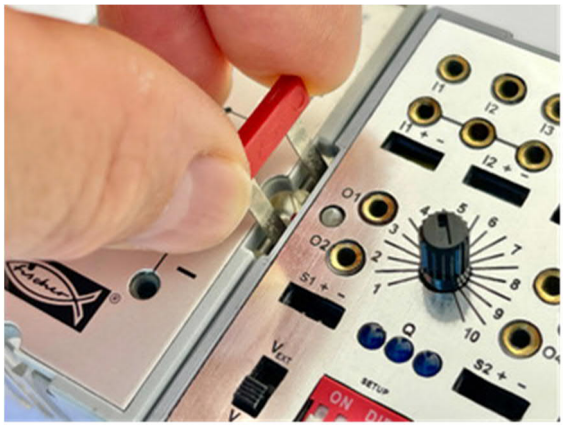


Abb. 4: Der Zauberling ist voll „Silberling-kompatibel“

Fazit

Die Software für den Zauberling wird wohl in naher Zukunft weiter ausgebaut und verfeinert. Bitte zögert nicht, mir eine E-Mail zu senden [1] oder mich im Forum [2] zu kontaktieren, wenn ihr Anregungen, Kommentare, Fragen oder Ideen für zukünftige Funktionen des Zauberlings habt. Denn für einen wahren Zauberer wie den Zauberling ist das scheinbar Unmögliche die ultimative Herausforderung ☺.

Eine Beschreibung, weitere Informationen und einen Link zu einem Video zu diesem Projekt findet ihr unter [3].

Quellen

[1] E-Mail-Adresse des Autors: arnoud@whizzbizz.com

[2] Nickname des Autors auf https://forum.ftcommunity.de: Arnoud-Whizzbizz

[3] Zauberling-Website in niederländischer oder englischer Sprache

[4] Zauberling-Software auf GitHub

[5] Arnoud van Delden: *Der Zauberling (Teil 1): Der Zauberstab*. ft:pedia 3/2021, S. 56–66.

Elektronik

Der Zauberling (Teil 3): Ein erster Trick

Arnoud van Delden

Der neumodische Magier ist in den Startlöchern. Der Zauberstab wurde poliert. Die ersten Zaubersprüche sind gelernt. Es ist noch nicht perfekt, aber das große Zauberbuch kann jederzeit ergänzt oder verbessert werden. In wenigen Sekunden sieht er einen Raum voller alter Bekannter, mit denen er zusammenarbeiten kann. Dann geht der Vorhang auf. Es ist an der Zeit, seine magischen Künste zu zeigen!

Vor und zurück, auf und ab

Eine immer wiederkehrende Herausforderung für den Bauherren von bewegten fischertechnik-Modellen ist die Hin- und Her- bzw. Auf- und Abbewegung. Ein Aufzug muss hoch und runter fahren, ein Karren von links nach rechts, ein Schieber auf und zu oder eine Schubstange von vorne nach hinten. Obwohl bei kleineren Bewegungen heutzutage der Einsatz eines Servos die naheliegende Wahl ist, kommt man bei einem größeren Hub nicht um eine mechanische Konstruktion oder einen Drehrichtungswechsel eines Motors an den Extrempunkten herum.

Am besten ist eine Lösung mit berührungslosen Sensoren und bei der der Motor an seinen Extrempunkten nicht abrupt umgekehrt wird, sondern eine allmähliche Drehzahländerung stattfindet. In diesem Artikel beschreibe ich meine Suche nach Lösungen und der Rolle, die die Magie des Zauberlings dabei spielen könnte.

Traditionell: rein mechanisch

Eine Lösung ohne jegliche Hilfe von Elektronik ist das rein mechanische Uhrwerk. Mit einem Exzenter, einer azentrisch rotierenden Stangen- oder Kurbelwellenkonstruktion kann eine rotierende Bewegung in eine lineare umgewandelt werden. Dieses Prinzip ist auch in den fischertechnik *hobby*-Büchern schön erklärt. Die Bewegungen einer Nähmaschine oder einer Stichsäge sind dafür gute Beispiele. Wenn die bewegliche Stange eine andere Stange mit Drehpunkt antreibt, entsteht eine Schwingbewegung, wie wir sie beispielsweise von unseren Scheibenwischern kennen. Weitere Hintergrundinformationen finden Interessierte in den Büchern „Elemente der Technik“ in der Bibliothek auf der Website des niederländischen fischertechnik-Clubs [1].

Ein rein mechanisches Verfahren ist zuverlässig und einfach. Die Geschwindigkeit um den „Wendepunkt“ der Gleitbewegung folgt einem allmählichen Verlauf. Die Bewegung an den Extrempunkten ist nicht abrupt oder ruckartig. Dadurch wird die Konstruktion durch die Trägheit des beweglichen (gleitenden) Teils nur minimal belastet. Die Drehbewegung hat eine feste Geschwindigkeit und der Motor kann sich einfach in eine Richtung weiterdrehen. Und weil sie ohne störanfällige Kontakte oder Sensoren arbeitet, müssen keine

Schutzvorrichtungen eingebaut werden, um zu verhindern, dass die Gleitbewegung unerwartet ihre Grenzen überschreitet und sich die Konstruktion „zerreißt“. Für ein mechanisches Verfahren ist jedoch eine aufwändige Hilfskonstruktion für das Getriebe etc. erforderlich. Außerdem ist der Bauraum im Vergleich zum begrenzten Hub der Schubbewegung groß.

Elektromechanisch

Die einfachste elektromechanische Lösung ist die Verwendung eines Polwendeschalters. Wenn die Verkabelung genügend Bewegungsfreiheit hat und der Verfahrweg relativ klein ist, kann der Polwendeschalter am beweglichen Teil der Struktur montiert werden. Bei längeren Trajektorien können Nocken abwechselnd den entlang der Trajektorie positionierten Schalter betätigen (Abb. 1). Eine andere Lösung besteht darin, eine am beweglichen Element befestigte Schnur durch das kleine Loch im Hebel des Polwendeschalters zu führen und diesen durch Knoten in der Schnur umschalten zu lassen.

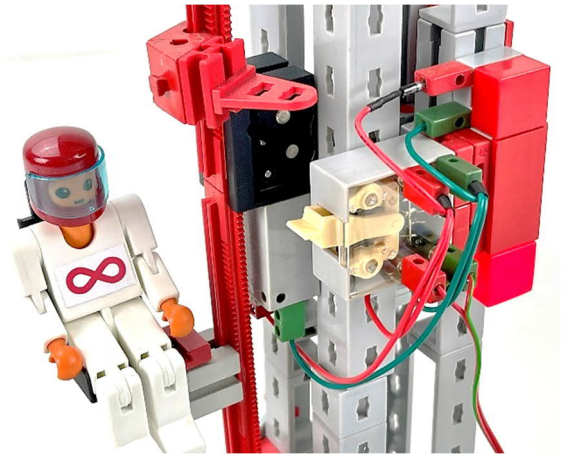


Abb. 1: Lösung mit Polwendeschalter

Gegenphasig geschaltete LEDs können ohne großen zusätzlichen Verdrahtungsaufwand direkt abwechselnd mit der Bewegungsrichtung als Anzeige aufleuchten. Dieses Verhalten können wir auch durch den Einsatz von Dioden in den Anschlüssen herkömmlicher Glühbirnen erreichen.

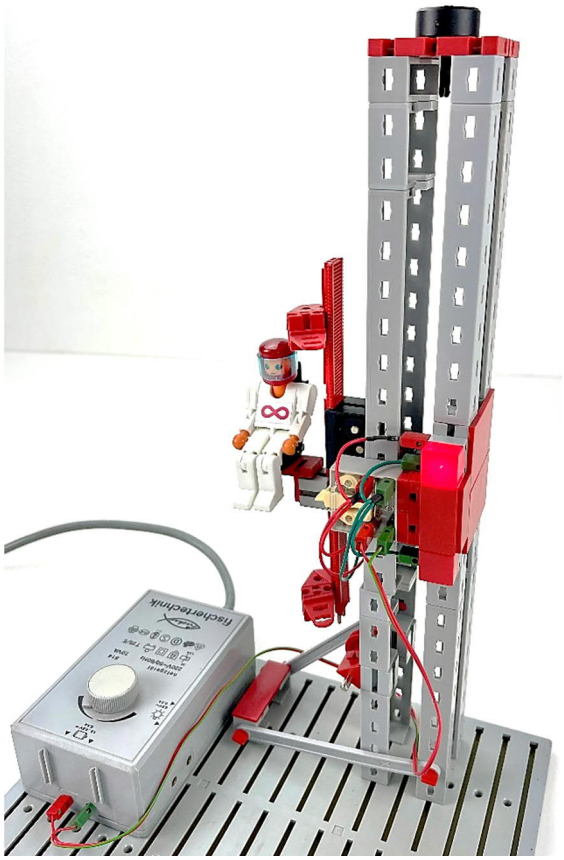


Abb. 2: Lift mit Polwendeschalter

Kontrolle mit Silberlingen

Nostalgiker können die obige Lösung mit dem traditionellen Silberling Flipflop h4 FF (30815) aufbauen, bei dem das Relaismodul h4 RB (30812) immer die Versorgungsspannung des Motors umkehrt. Schalter können auf diese Weise an die Flipflop-Impulseingänge angeschlossen werden. Bei passiven Sensoren wie Fotowiderständen oder Transistoren, bei denen die Detektionsschwelle einstellbar sein muss, werden diese Signale über Grundbausteine h4 G (30813) geführt. Theorie und zahlreiche Baubeispiele findet ihr im Heft des *IC Digital Praktikum* oder hobby 4, Band 3.

Im Experiment in Abb. 3 wurden Fotowiderstände (32698) als Grenzsensoren verwendet, aber natürlich können auch Fototransistoren (36134) als Lichtsensoren verwendet werden. Eine weitere berührungslose Art der Endpunkterkennung ist mit dem Reedkontakt (36120) möglich.

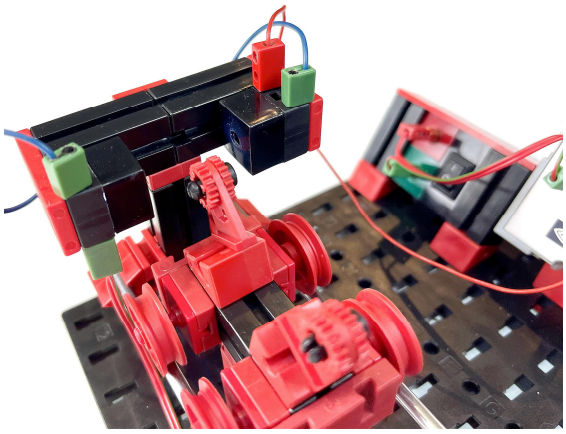


Abb. 4: Detailblick auf die Lichtschanke

Mit mechanischen Impulsschaltern oder Magnetschaltern an den Extrempunkten des Hubwegs kann mit dem Flip-Flop-Silberling eine elektronische Lösung geschaffen werden. Dazu können wir die flankensensitiven Eingänge S_P und R_P verwenden. Vergesst jedoch nicht die Entstörung mit einem 100-nf-Kondensator gegen Schaltrauschen und Störimpulse, die bei mechanischen Stromstoßschaltern auftreten können (siehe auch hobby 4, Band 3, Abb. 66.3 und 66.4). Bei Verwendung von berührungslosen Sensoren wie dem Foto-Widerstand (32698) oder dem Foto-Transistor (36134) kann die Schwellenwerterkennung des Lichtsensors mit dem Grundbaustein (36391) eingestellt werden. Die beiden Ausgangssignale der Grundbausteine können über ein ODER-Gatter mit dem CP verbunden werden, so dass die Sensoren als Umschalter arbeiten.

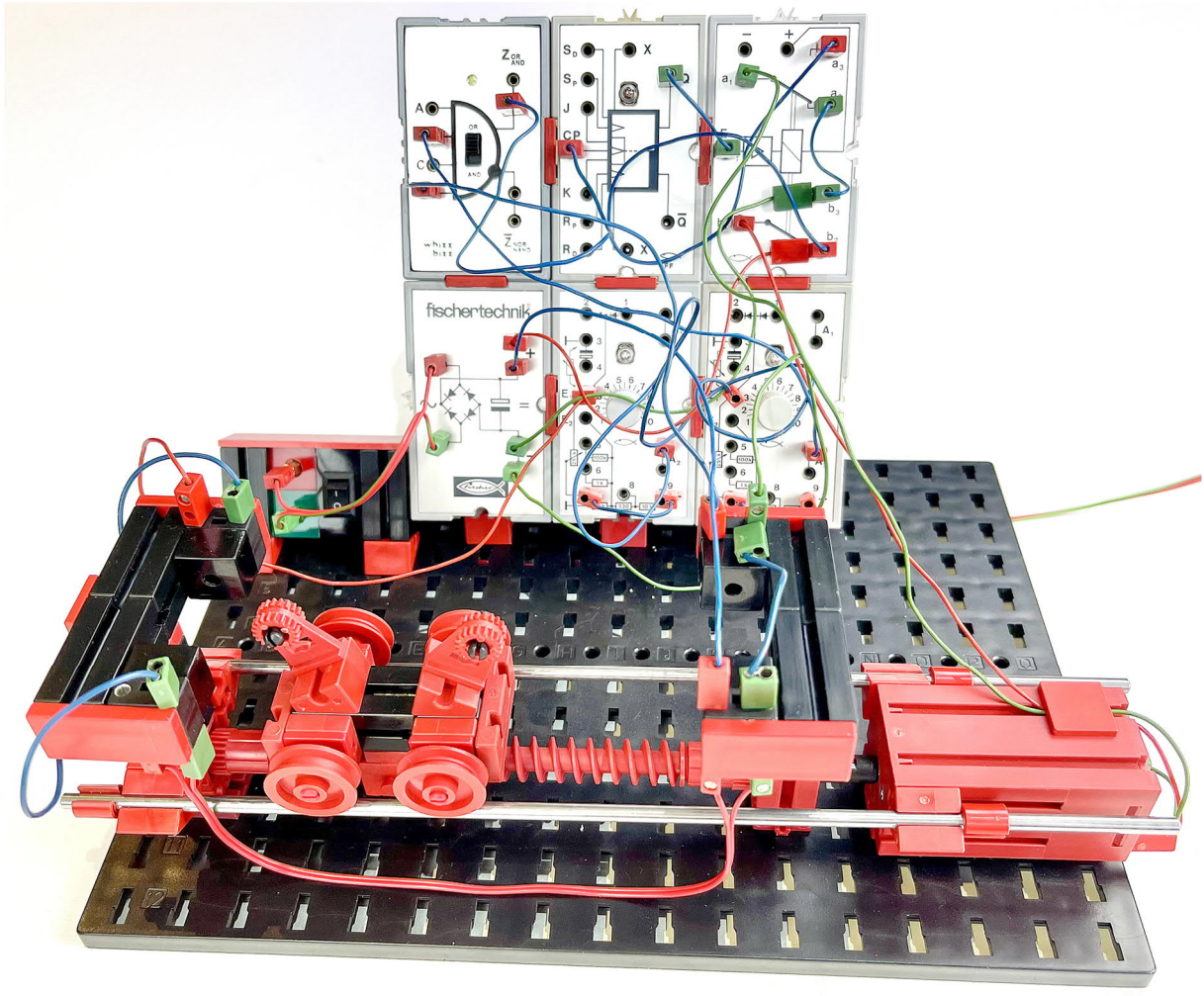


Abb. 3: Eine Lösung mit Lichtschranken und Silberlingen

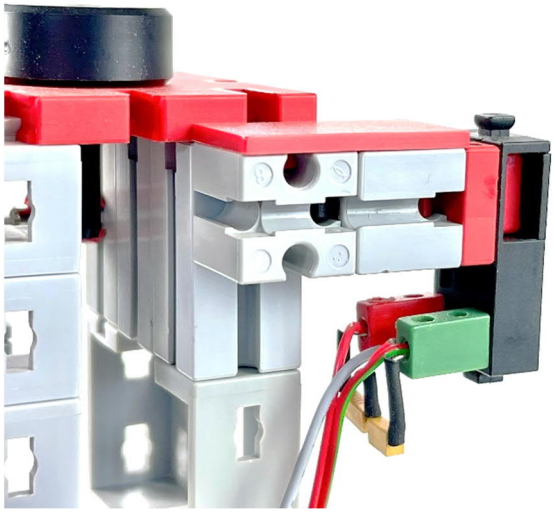


Abb. 5: Störschutz mit 100-nf-Kondensator gegen das Prellen des Tasters

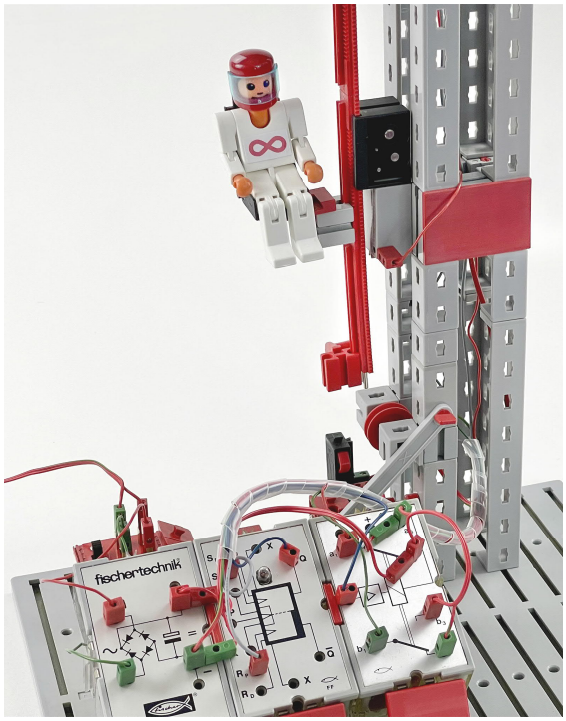


Abb. 6: Ansteuerung des h4 FF direkt mit Tastern

Aktive Sensoren

Anstelle eines Leuchtsteins schien es ein lustiges Experiment zu sein, zwei IR-Hindernissensoren zu verwenden, um die Endpunkte zu erkennen. Dies sind „aktive“ Sensoren, die eine Versorgungsspannung von 5 V benötigen und im Ruhezustand ihre Versorgungsspannung am Ausgang ausgeben.

Sobald innerhalb der am Modul einstellbaren Entfernung ein Objekt erkannt wird, sinkt diese Ausgangsspannung auf null. Dies ist im Grunde dieselbe negative Logik, die die Silberlinge verwenden.

Ein zusätzlicher Vorteil ist, dass diese Module bei einer höheren Versorgungsspannung, wie den 9 V der Silberlinge, ohne größere Probleme betrieben werden können. Der auf dem Modul eingesetzte Spannungskomparator LM393 hält laut Datenblatt sogar bis zu 36 V problemlos aus. Das einzige Problem ist, dass die beiden SMD-LEDs auf der Platine auf 5 V eingestellt sind und bei einer Spannung von 9 V sehr hell leuchten. Und dieses visuelle Erkennungsfeedback ist tatsächlich sehr nützlich, daher wäre es schade, wenn sie deswegen durchbrennen.

Um diese Sensoren für die Verwendung mit den Silberlingen (oder dem TXT-Controller) geeignet zu machen, habe ich zwei verschiedene Modifikationen ausprobiert, die beide gut funktionierten:

- Bei einem Modul habe ich die 1 kΩ SMD-Vorwiderstände der LEDs durch (leider nicht-SMD) 2,2 kΩ ersetzt, sodass die LEDs bei 9 V einfach schwächer leuchten. Die Versorgungs- und die Signalausgangsspannung dieses Moduls entsprechen daher voll und ganz den Silberlingen.
- Bei einem zweiten Modul bin ich einen anderen Weg gegangen und habe einen 78L05 Spannungsregler hinzugefügt, sodass die gesamte Schaltung mit 9 V versorgt werden kann, aber intern nur mit 5 V läuft. Zum Glück wird die Signalspannung von nur 5 V von den Silberlingen noch ordentlich als LOW erkannt, daher scheint dies eine sinnvolle Idee für diejenigen zu sein, die direkt mit 9 V arbeiten möchten (mit dem TXT-Controller oder den Silberlingen).

Abb. 7: Original und zwei modifizierte IR-Hindernissensoren

Mit dem Zauberling und dem „Basisprogramm“

Nach einem kurzen Test, bei dem ich die Fotowiderstände des Vorgängermodells durch die Silberlinge mit den modifizierten IR-Sensoren ersetzt habe, kam der Zauberling auf den Plan. Mit ihm konnten einige Punkte verbessert werden:

- Da der Zauberling den direkten Anschluss von 5-V-betriebenen aktiven Sensoren unterstützt, ist es nicht notwendig, Sensoren zu modifizieren.
- Auch bei Verwendung mechanischer Druckschalter ist eine Entstörung gegen Schaltgeräusche nicht mehr erforderlich.
- Das modifizierte „Basisprogramm“ mit den schrittweisen Drehrichtungswechseln kann unnötige(n) Motorbelastung und -verschleiß vermeiden und verhindert, dass bei jedem abrupten Polaritätswechsel eine Stoßwelle durch das Modell geht.

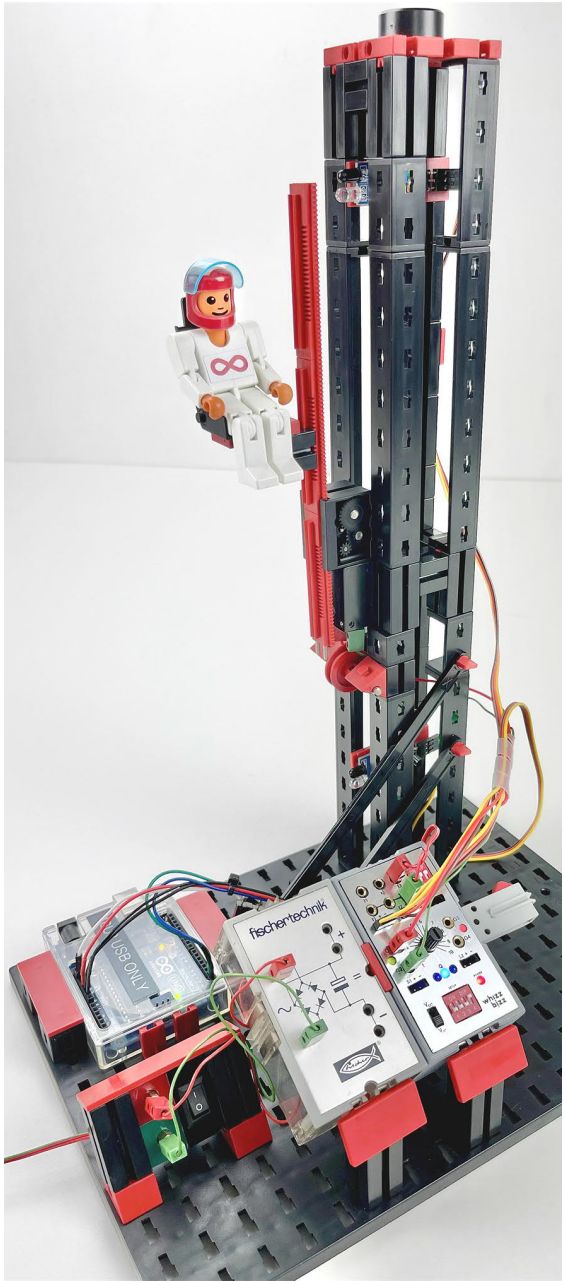


Abb. 8: Auf und ab mit dem Zauberling

Fazit

Der Zauberling löst einige praktische Probleme wie von Zauberhand. Seine flexiblen Eingänge ermöglichen das Experimentieren mit praktischen Sensoren außerhalb des bekannten fischertechnik-Sortiments. Die allmählichen Drehrichtungswechsel werden von fischertechnik-Motoren sehr geschätzt und wären mit diskreter Elektronik nicht so einfach zu realisieren. Auf den Projektseiten des Zauberlings [2] findet ihr weitere Informationen und ein Video, das die Motorsteuerung in Aktion zeigt. Mr. Möbius [3] bestand darauf, dass er auf dem so wunderbar allmählich „schwebenden“ Stuhl sitzen darf.

Abb. 8: Der Zauberling von außen

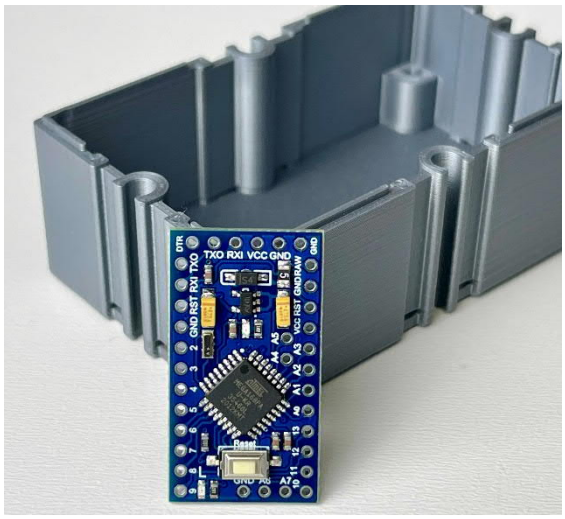


Abb. 9: Das Herz des Zauberlings

Es ist ein schöner erster Trick des Zauberlings. Ich bin gespannt, was er in Zukunft noch aus dem Hut zaubern wird. Er arbeitet gut mit den Silberlingen zusammen und wurde bei meinen Experimenten bereits liebevoll in diese traditionsreiche fischertechnik-Familie aufgenommen.

Quellen

[1] Fischer-Werke: *Elemente der Technik*. Teil 1, S. 15. Auf der Website des fischertechnik-Club NL.

[2] Arnoud van Delden: Website zu diesem Artikel auf Niederländisch und Englisch.

[3] Arnoud van Delden: *Die unendliche Lemniskate*. ft:pedia 2/2021, S. 36–44.

[4] Arnoud van Delden: *Der Zauberling (Teil 1): Der Zauberstab*. ft:pedia 3/2021, S. 56–66.

[5] Arnoud van Delden: *Der Zauberling (Teil 2): Das Zauberbuch*. In dieser Ausgabe der ft:pedia.

Elektronik

Ampelsteuerung

Hans-Christian Funke

Schon mit den ersten em-Baukästen von fischertechnik konnte man mit Hilfe von Schleifringen und Abnehmern eine Ampel-Schaltung herstellen. In späteren Varianten konnte man mit Nockenscheiben und Tastern die Umsetzung etwas eleganter lösen. Mit den Silberlingen war eine erste komplett elektronische Umsetzung möglich. Daran anknüpfend zeige ich hier verschiedene Umsetzungen mit meinen Elektronikmodulen.

Die meisten würden dieses Problem eh mit einem TXT und einem kleinen Programm lösen, aber hier geht es ums Tüfteln und Knobeln mit Gattern und Speichern.

Es gibt viele Möglichkeiten, zum Ziel zu gelangen. Hier stelle ich euch zwei Varianten vor – eine sehr einfache Version, die wirklich nur den Ablauf nachstellt (ohne Realitätsbezug) und eine zweite Variante, bei der man sämtliche Lichtphasen einzeln zeitlich auf den „Verkehr“ abstimmen kann.

Variante 1

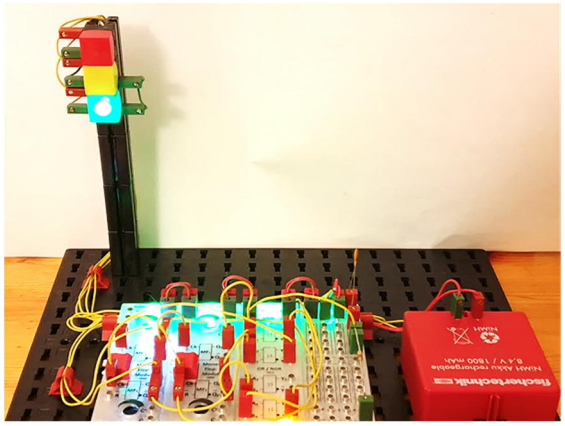


Abb. 1: Das Ampel-Modell in Variante 1

In Abb. 2 werden zwei Taktungen abgebildet, wie diese z. B. ein Taktgenerator (70004) erzeugt. Ist ein Takt auf 1 (high), dann leuchtet die LED. Beim längeren Takt ist es die rote LED („Anhalten“), der kurze Takt passt genau dazu, wie die gelbe LED leuchten muss – einmal mit der roten LED und einmal ohne die rote LED. Die Grünphase dazwischen, wenn die gelbe und rote LED aus (low) sind, wird mit Hilfe eines AND-Gatters abgefragt – allerdings benötigen wir ein logisch 1 (high), damit das AND-Gatter am Ausgang ebenfalls high wird, darum müssen beide Eingangssignale (gelb und rot) vorher invertiert werden.

Variante 2

Mit der Variante 1 ist zwar schon einmal die grundsätzliche Umsetzung zur Nachbildung einer Ampel-Schaltung erbracht, aber fernab von der Realität. Um eine realitätsnahe Simulation zu erhalten, benötigen wir individuelle Zeitintervalle. Dafür eignet sich ein Mono-Flop (70005) ideal, weil hier die Schaltzeiten über den eingebauten Regler schnell individuell eingestellt werden können.

Die Umsetzung der Schaltung kann auf mehreren Wegen erfolgen, aber wenn man die Schaltzeit für jede Phase separat bestimmen möchte, braucht man pro Lichtphase ein Mono-Flop (MF).

Hier wird erst einmal nur die Schaltung für eine Ampel-Schaltung (Fußgängerüberweg) vorgestellt, um so das Prinzip besser veranschaulichen zu können. Für den Aufbau werden 4 MFs benötigt (2 × 70005), ein AND-Gatter und zwei OR-Gatter.

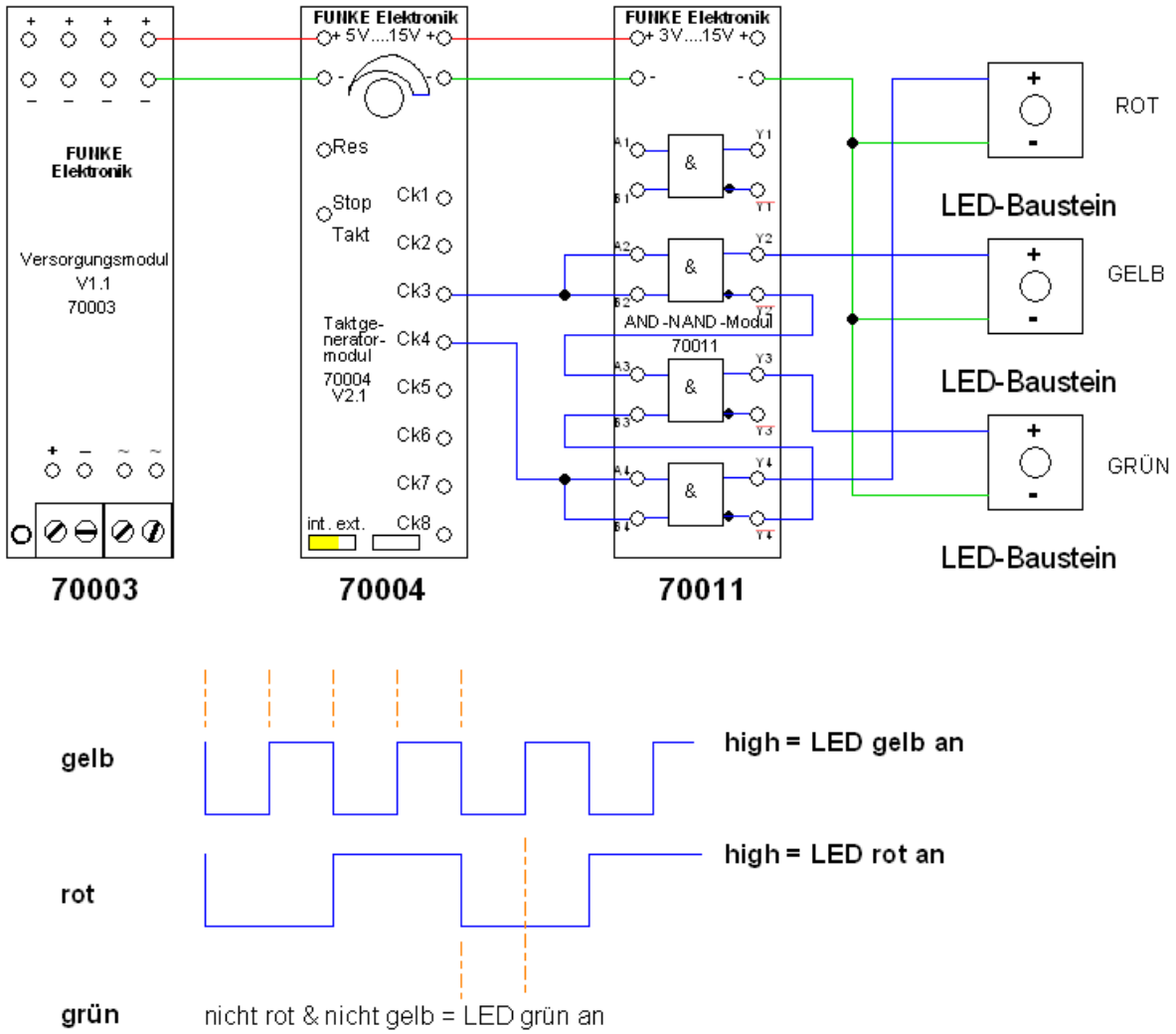


Abb. 2: Sehr einfache Ampelschaltung (deutsche Norm)

Damit für das AND-Gatter nicht extra ein Elektronikmodul eingesetzt werden muss, habe ich das AND-Gatter mit einem OR-Gatter simuliert (Abb. 3).

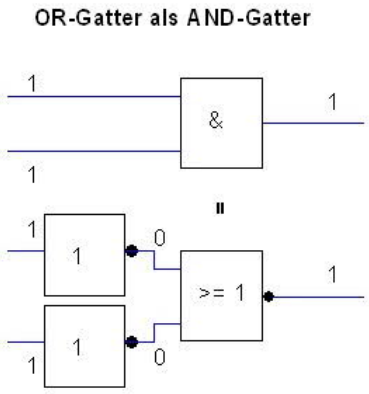


Abb. 3: AND-Gatter-Ersatzschaltung

Abb. 4: Blick auf die Starter-Schaltung

Somit wird nur noch ein OR-NOR-Gatter (70013) und ein Experimentiermodul (70021) benötigt. Das Experimentiermodul wird nicht zwingend gebraucht, es beherbergt eine Starter-Schaltung aus einem Widerstand und einem Kondensator (der Widerstand ist links und der Kondensator rechts in Abb. 4 zu sehen), die theoretisch auch einfach so an den Eingang vom OR-Gatter gesteckt werden könnten.

In der Schaltung ist ersichtlich, dass die vier MFs alle zu einem Ring zusammen geschaltet werden. Nach dem Einschalten der Versorgungsspannung befinden sich daher alle MFs im Ruhezustand, also benötigt eines der MFs einen Stups, damit die Schaltung losläuft.

Gehen wir durch die Schaltung (siehe Abb. 5): Die Start-Schaltung aktiviert MF1 und es beginnt mit der roten LED (nach dem Einschalten sinnvoll – erst einmal alles anhalten). Das MF2 (gelb_1) wartet die Zeit ab, in der die gelbe LED nicht leuchten soll. Die gelbe LED wird über ein ODER-Gatter gesteuert und leuchtet nur, wenn MF2

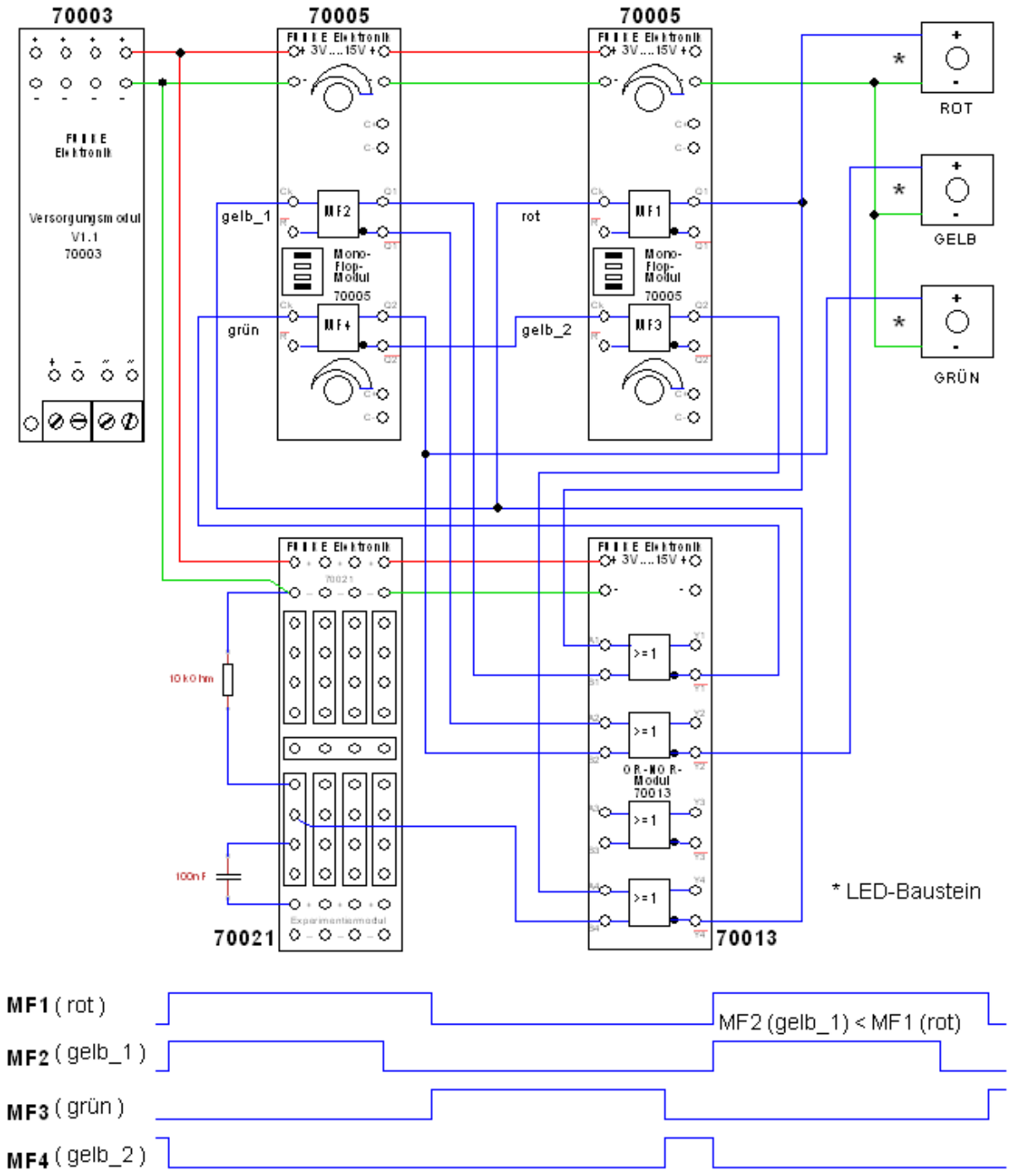


Abb. 5: Ampel-Schaltung mit Mono-Flops

(gelb_1 = nicht gelb) und MF3 (grün) beide low sind.

Sobald MF1 und MF2 abgelaufen sind (beide auf low), wird MF3 gestartet – die grüne LED. Ist MF3 abgelaufen, sind MF2 und auch MF3 low: Die gelbe LED leuchtet. Das MF4 ist somit die Leuchtdauer der gelben nach der grünen Phase. Damit ist der Kreis geschlossen; MF1 und MF2 werden wieder angestoßen.

Abschluss

Für eine Ampel-Schaltung mit kreuzendem Verkehr erhöht sich die Anzahl der erforderlichen Zeitglieder. Die Erweiterung erfolgt nach dem gleichen Prinzip.

Man kann allerdings überlegen, ob man die MFs mit gleichen Zeiten z. B. für die gelben Phasen durch Steuerung mit Logik-Gattern mehrfach verwenden kann und so MFs einspart. Viel Spaß beim Tüfteln!

Übrigens sind die verwendeten LEDs weiße 1-mA-LEDs. Sie sind in die fischertechnik-Leuchtsteine eingesetzt, auf die man die bunten neuen und alten Kappen aufsetzen kann. Ihr könnt aber auch die farbigen LED-Bausteine verwenden (Abb. 6):

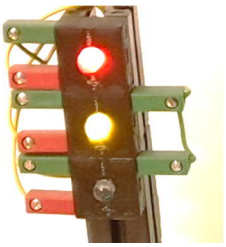


Abb. 6: Ampel mit farbigen LEDs

Mit dem „Labor für digitale Elektronik“ [11] werden auch diese LED-Bausteine im Online-Shop bei Franz Santjohanser erhältlich sein.

Alles rund um die Elektronikmodule erhaltet ihr beim offiziellen fischertechnik-Servicepartner [1]. Im Online-Shop sind die Elektronikmodule, die Zusatzmodule, LED-Bausteine, Baukästen, Labore und Zubehör erhältlich.

Elektronikmodul-Videos

Auf YouTube hat „Zerobrain“ Videos unter Verwendung der Elektronikmodule erstellt mit interessanten Themen aus der digitalen Elektronik [2, 3, 4].

Quellen

[1] santjohanser Spielen. Lernen. Technik.

[2] „Zerobrain“: *Grundkurs Digitaltechnik 1*. Auf YouTube, 2021, ca. 34 min.

[3] „Zerobrain“: *Grundkurs Digitaltechnik 2: Geschwindigkeit mit einfachen Gattern.* Auf YouTube, 2021, ca. 17 min.

[4] „Zerobrain“: *Keiner versteht Flip-Flops | 4 Bit Digitalzähler erklärt*. Auf YouTube, 2021, ca. 23 min.

[5] Hans-Christian Funke: *Elektronikmodule (Teil 1): Einleitung*. ft:pedia 4/2019, S. 40–46.

[6] Hans-Christian Funke: *Elektronikmodule (Teil 2)*. ft:pedia 1/2020, S. 52–61.

[7] Hans-Christian Funke: *Elektronikmodule (Teil 3)*. ft:pedia 2/2020, S. 68–81.

[8] Hans-Christian Funke: *Elektronik-Module (Teil 4)*. ft:pedia 3/2020, S. 59–70.

[9] Hans-Christian Funke: *Elektronik-Module (Teil 5)*. ft:pedia 4/2020, S. 80–94.

[10] Hans-Christian Funke: *Elektronik-Module (Teil 6)*. ft:pedia 1/2021, S. 94–99.

[11] Hans-Christian Funke: *Labor für digitale Elektronik*. ft:pedia 3/2021, S. 49–55.

Regelungstechnik

PID-Regler – eine experimentelle Einführung

Dirk Fox

Regler finden wir in zahlreichen technischen Geräten. Und sie verbreiten sich immer weiter: Sie sind das Herzstück vieler (teil-)autonomer Systeme und zählen inzwischen zu den wichtigsten Anwendungen von Mikrocontrollern [1]. Eine zentrale Rolle spielen darunter PID-Regler [2]. Mit der Oszilloskop-Funktion von ROBO Pro lässt sich deren Funktionsweise am TXT-Controller wunderbar anschaulich zeigen.

Steuerung und Regelung

Die meisten Mikrocontroller-Anwendungen unserer fischertechnik-Modelle sind *Steuerungen*. Dabei stellen wir einen Aktor auf einen vorgegebenen Wert ein, beispielsweise eine LED auf eine bestimmte Helligkeit oder einen Motor auf eine feste Geschwindigkeit. Für viele solcher Steuerungen ist der TXT Controller bei weitem überdimensioniert – das könnten auch ein paar simple ICs.

In der Praxis viel wichtiger und technisch auch erheblich spannender sind *Regler*. Im Unterschied zu einer Steuerung sorgt ein Regler dafür, dass eine vorgegebene „Führungsgröße“ (wie z. B. die Temperatur eines Kühlfachs, die Helligkeit eines Raums oder die Geschwindigkeit eines Fahrzeugs) auch bei plötzlichen Störungen (wie dem Öffnen der Kühlschranktür, abnehmendem Tageslicht oder hügeligem Gelände) möglichst schnell wieder erreicht wird.

Ein Regler muss solche Störungen erkennen und die Steuerung des Aktors so anpassen, dass das System die Vorgabe wieder einhält.

Dazu benötigt der Regler einen „Feedback“-Mechanismus, der ihn informiert, wenn es zu Abweichungen der Regelgröße von der vorgegebenen Führungsgröße kommt. Aus der gemessenen Abweichung berechnet der Regler eine neue Stellgröße für den Aktor, um die Abweichung der Regelgröße von der Führungsgröße zu verringern (Abb. 1).

Beispiel: Will man bei einer motorgetriebenen Achse die Zahl der Umdrehungen pro Zeiteinheit auch bei variabler Last konstant halten, muss man die Winkelgeschwindigkeit der Achse messen und bei Abweichungen von der Sollvorgabe die Motorspannung erhöhen oder senken.

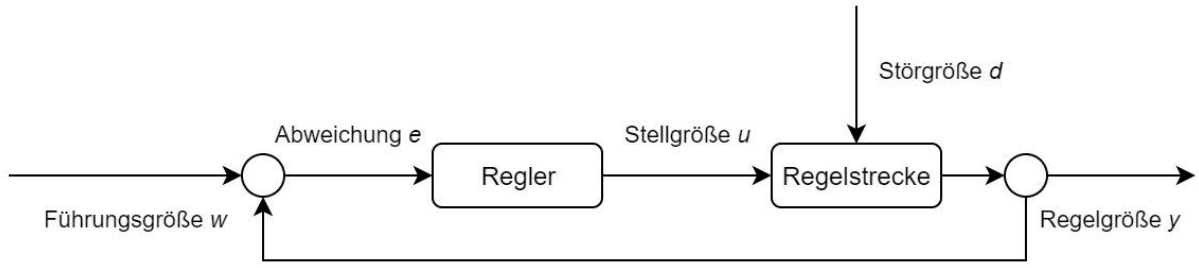


Abb. 1: Schema eines Reglers

Einfache Regelungsaufgaben mit einem Aktor, der nur an- oder ausgeschaltet werden kann, löst man mit einer so genannten Zweipunktregelung [3]: Sinkt die Regelgröße unter (bzw. steigt sie über) einen vorgegebenen Schwellenwert, wird der Aktor (bspw. ein Kühl- oder Wärmeaggregat) zugeschaltet, bis ein zweiter, höherer Schwellenwert erreicht ist. Solche einfachen Regler finden sich in Backöfen, Heizungen, Kühlschränken oder Klimaanlagen. Die Reglerkurve entspricht einem unregelmäßigen „Rechtecksignal“. Der Bereich, in dem die Regelgröße zwischen den Schwellenwerten „pendelt“, nennt man *Hysterese*. („Nachwirkung“). Soll die Regelung in beide „Richtungen“ erfolgen, um beispielsweise einen festen Abstand einzuhalten [4, 5], verwendet man meist eine Dreipunktregelung („zu nah“, „zu fern“, „Abstand ok“).

Komplizierter wird es, wenn ein Regler die Regelgröße nach einer Störung so schnell wie möglich wieder auf die vorgegebene Führungsgröße einstellen soll. Voraussetzung dafür ist ein möglichst stetig steuerbarer Aktor (bspw. ein Motor), über den das System geregelt werden kann.

Bei einer solchen Regelung können verschiedene unerwünschte Effekte auftreten:

- Der Regler stellt die Führungsgröße zu langsam wieder ein;
- Der Regler schwingt über, d. h. aus einer zu kleinen wird eine zu große Regelgröße;
- Der Regler fängt an zu oszillieren und schaukelt sich schlimmstenfalls sogar auf, anstatt in die gewünschte Ruhelage zurückzukehren.

Um einen Regler für das gewünschte Regelungsziel zu optimieren, müssen zunächst einige Randbedingungen festgelegt oder bestimmt und verschiedene Annahmen getroffen werden. Dazu zählen insbesondere die maximale Größe einer Störung, auf die der Regler reagieren soll, die Auflösung und Genauigkeit der Messung der Regelgröße oder die Verzögerung, mit der eine Änderung der Stellgröße sich auf die Regelgröße auswirkt.

Schließlich hat der gewählte Reglertyp maßgeblichen Einfluss auf die Wirkung. Besonders leistungsfähig und flexibel (und daher sehr verbreitet) sind P-, PI-, PD- und PID-Regler [2]. Die Funktionsweise und Wirkung dieser Regler werden wir nun am Beispiel eines „analogen Spurfolgers“ veranschaulichen.

Regelungsaufgabe

In [6] habe ich vorgestellt, wie man einen „digitalen“ Spurfolger mit dem fischertechnik-Spursensor sehr elegant als Endlichen Automaten in ROBO Pro programmieren kann.

Der Schönheitsfehler dieser Lösung: Der Spurfolger folgt einem Zick-Zack-Kurs. Es ist offensichtlich, dass das Ergebnis nicht optimal ist, denn der vom Spurfolger zurückgelegte Weg ist deutlich länger als die Spur; außerdem muss der Spurfolger bei jedem Richtungswechsel verlangsamen.

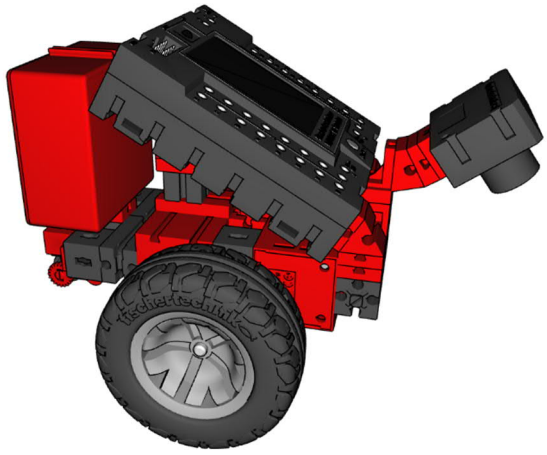


Abb. 2: Analoger Spurfolger mit TXT Controller und Kamera

Man kann die Spurfolger-Aufgabe aber auch als ein *Regelungsproblem* betrachten und mit Hilfe eines Reglers lösen. Dazu ersetzen wir an unserem kleinen „Buggy“ zunächst einmal den Spursensor durch eine Kamera (Abb. 2).

Eine fischertechnik-Designer-Datei des Modells findet ihr im Downloadbereich der ft:pedia. Die Konstruktion beschränkt sich auf das Wesentliche; aber natürlich darf der kleine Buggy um Stoßstangen, Beleuchtung etc. erweitert werden (Abb. 3) – oder auch ganz anders aussehen.

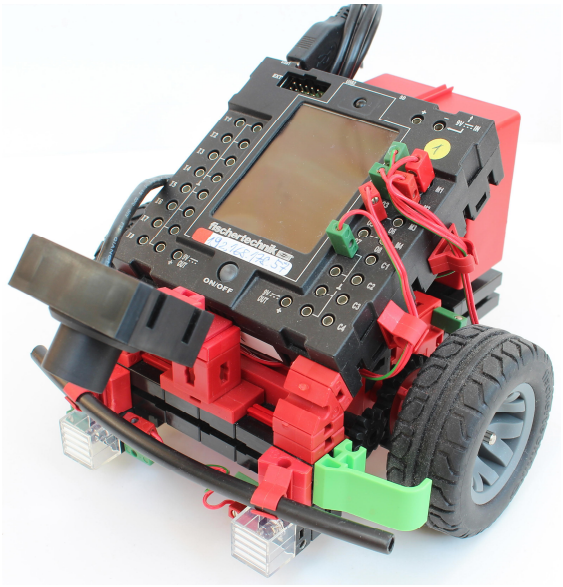


Abb. 3: „Gepimpter" Buggy

Der Regler muss nun dafür sorgen, dass der Mittelpunkt der von der Kamera erkannten Spur in der Mitte des Bildausschnitts der Kamera liegt (*Führungsgröße w*). Die Linienerkennung von ROBO Pro liefert uns den Abstand vom Mittelpunkt (*Abweichung e*, Abb. 4) als „analogen" Wert, aus dem der Regler die erforderliche Korrektur *u* der Motorspannung berechnet.

Die Konfiguration der Linienerkennung ist einfach (Abb. 5): Sie sollte die gesamte Breite des Kamerabilds im oberen Drittel umfassen; die minimale und die maximale Spurbreite müssen ggf. an den verwendeten Parcours angepasst werden.

Die Kamera hat eine horizontale Auflösung von 320 Punkten; daher sollten die Koordinaten auf -160 bis +160 festgelegt werden, um eine möglichst hohe Genauigkeit der Messungen zu erreichen. Bei der Bildrate (*frames per second*, fps) habe ich mit 25 fps und hoher Genauigkeit gute Ergebnisse erzielt (Abb. 5). Damit wird zugleich die maximal mögliche Zahl der Richtungskorrekturen pro Sekunde (alle 40 ms) festgelegt. Diese Wahl beeinflusst die Wirkung des Reglers erheblich.

Um die Abweichung von der Spur zu korrigieren, muss der Buggy nach links oder rechts gesteuert werden. Das erreichen wir, indem wir die Geschwindigkeit des einen Motors erhöhen und die des anderen verringern.

Dafür müssen wir zunächst eine „Basisgeschwindigkeit" für die beiden Motoren definieren. Mit der Basisgeschwindigkeit legen wir zugleich den maximalen „Lenkeinschlag" des Buggy fest, nämlich die Differenz zur Maximalgeschwindigkeit (512). Hier haben wir einen weiteren Optimierungsparameter unseres Reglers: Je höher die Basisgeschwindigkeit, desto schneller der Buggy – aber desto schwächer die Wirkung der Steuerung.

Wir beginnen mit einer Basisgeschwindigkeit von 400; später können wir sie bei Bedarf noch anpassen. Damit der Buggy geradeaus fährt, müssen die Geschwindigkeiten der beiden Motoren in der Regel etwas unterschiedlich gewählt werden.

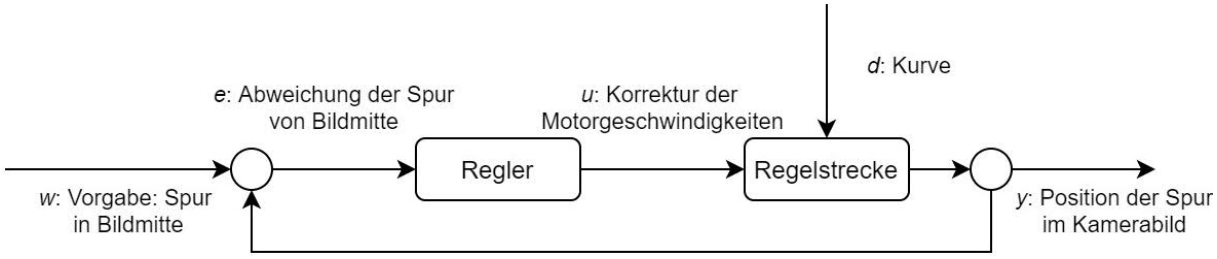


Abb. 4: Komponenten des Spurfolger-Reglers

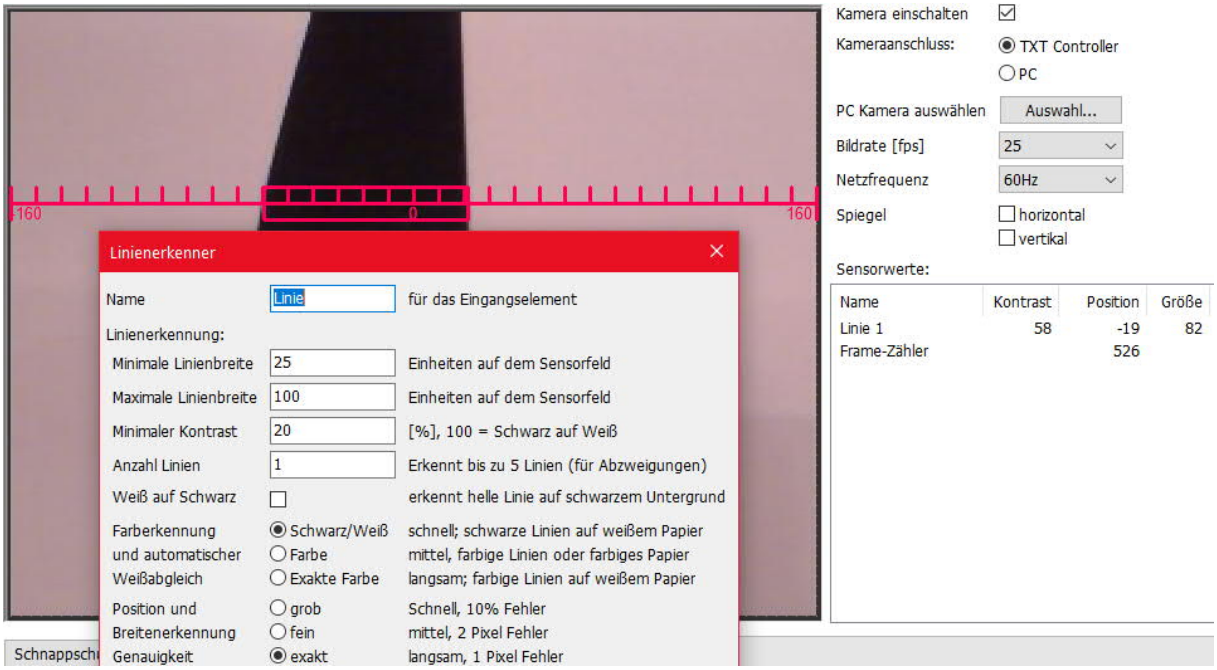


Abb. 5: Konfiguration der Linienerkennung

Mit dem simplen ROBO Pro-Programm in Abb. 6 könnt ihr die Geradeausfahrt über einer schwarzen Spur testen und die Basisgeschwindigkeit des schnelleren Motors anpassen (senken). Wenn ihr das Programm im Onlinemodus startet, wird in ROBO Pro die Abweichung von der Spur angezeigt.

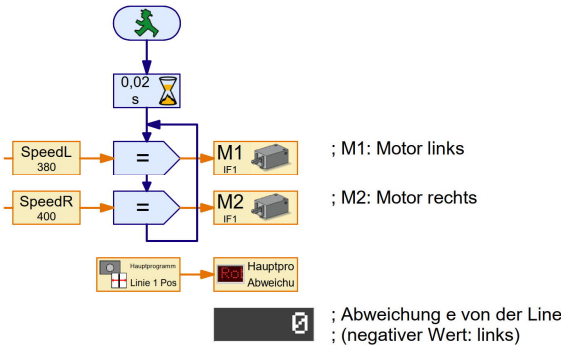


Abb. 6: ROBO Pro-Programm ‚Spurfolger Kamera.rpp' (Geradeausfahrt)

Dreipunkt-Regler

Für eine simple Dreipunktregelung können wir nun eine maximale Abweichung von der Spur nach links bzw. rechts als Schwellenwerte festlegen. Überschreitet der Buggy den einen, muss er nach rechts steuern, überschreitet er den anderen, nach links.

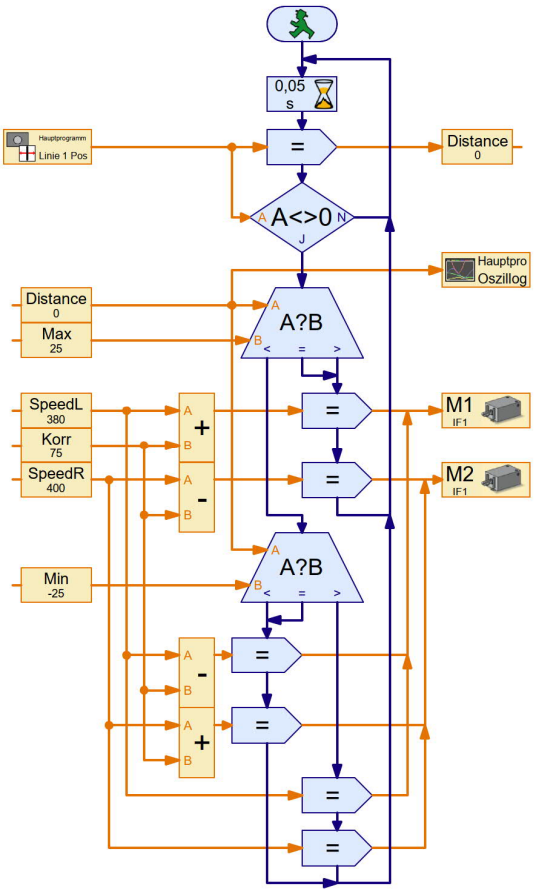


Abb. 7: ROBO Pro-Programm ‚Spurfolger Dreipunkt-Regler.rpp'

Abb. 7 zeigt den Dreipunktregler als ROBO Pro-Programm. Das Ergebnis ist ein Fahrtverlauf mit mehr oder weniger demselben Zick-Zack-Kurs, dem unser Buggy auch mit einem IR-Spursensor folgen würde. Abb. 8 zeigt den Verlauf des Spurabstands. Man erkennt, dass der Regler stark überschwingt: Es kommt zu Abweichungen von mehr als 100.

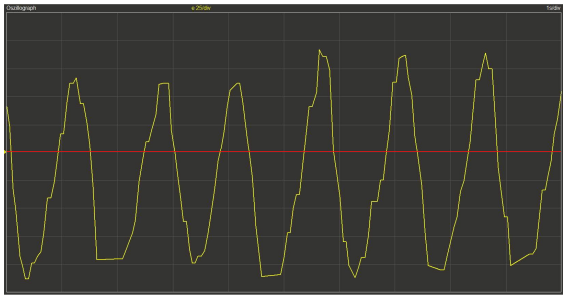


Abb. 8: Dreipunktregelung [Skala: 1s/25]

Immerhin können wir hier nicht nur die Stärke des Lenkeinschlags, sondern auch die Schwellenwerte, bei denen der Regler reagieren soll, sehr fein anpassen. Damit lassen sich die Amplitude und die Frequenz der „Schwingung“ in Abb. 8 ändern und das Überschwingen etwas dämpfen. Viel besser wird die Regelung dadurch aber nicht.

P-Regler

Kann man die Größe der Abweichung der Regelgröße von der Führungsgröße messen, so verwendet man üblicherweise einen *Proportionalregler*, der die Änderung der Stellgröße abhängig von (oder *proportional zu*) der Größe der Abweichung vornimmt. Ein solcher Regler wird auch kurz P-Regler genannt. Die Güte des Reglers hängt dabei (auch) von der Auflösung und Genauigkeit dieser Messung ab. Von einem Zwei- und Dreipunktregler unterscheidet er sich durch seinen stetigen Verlauf.

Um den Korrekturwert u für die Stellgröße zu erhalten, multiplizieren wir die Abweichung mit einem *Proportionalitätsfaktor* K_p. Zur Bestimmung dieses Faktors stellen wir ein paar Überlegungen an:

- Die maximale Motorgeschwindigkeit ist 512. Daher sollte der Korrekturwert bei der von uns gewählten Motor-Basisgeschwindigkeit (400) den Wert 110 nicht überschreiten.
- Die Abweichung von der Spur kann bis zu ±130 betragen. Die Korrektur sollte aber möglichst schon bei kleinen Abweichungen (< 40) erfolgen.

Der Proportionalitätsfaktor sollte also zwischen 0 und 2 liegen. Wir können uns an einen geeigneten Wert herantasten, indem wir mit 0,1 beginnen und den Wert schrittweise um 0,1 erhöhen, bis der Regler zu oszillieren beginnt, ohne sich aufzuschaukeln.

Die Abweichung von der Spur e und die berechnete Stellgröße u (Korrekturwert der Motorgeschwindigkeiten) mit

$$u = K_p \cdot e$$

könnt ihr euch im Online-Mode in ROBO Pro mit der Oszilloskop-Funktion anzeigen lassen. Das ROBO Pro-Programm für den P-Regler zeigt Abb. 10.

Um den P-Regler zu testen, starten wir den Buggy parallel zur Spur, und zwar so, dass die Linienerkennung die Spur gerade noch rechts (oder links) im Bild findet. Abb. 9 zeigt die Abweichung von der Spur über die ersten 10 Sekunden der Testfahrt mit einem P-Regler mit Proportionalitätsfaktor 0,4.

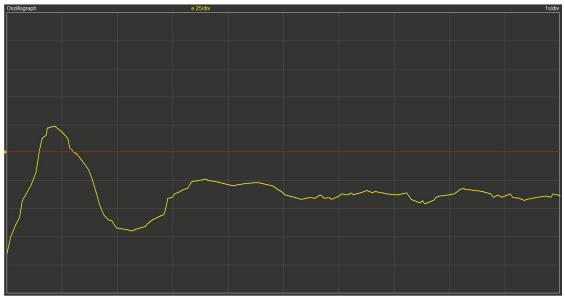


Abb. 9: P-Regler mit $K_p = 0{,}4$ (Abweichung e [Skala: 1s/25])

Man erkennt, dass der Regler zunächst kurz überschwingt und sich dann um einen Wert von etwa -35 einpendelt.

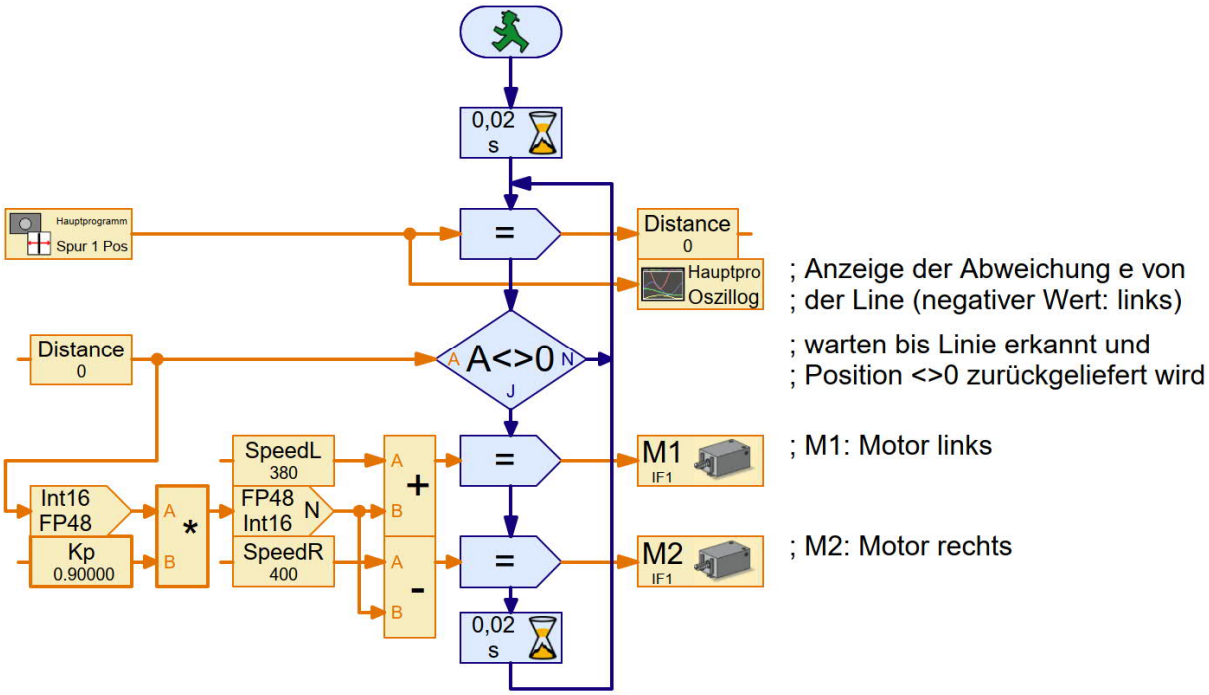


Abb. 10: ROBO Pro-Programm ‚Spurfolger P-Regler.rpp'

Erhöhen wir den Proportionalitätsfaktor, so reagiert der Regler schneller, schwingt dabei aber deutlich über. Bei einem Wert von 0,9 beginnt der Regler zu oszillieren (Abb. 11) und der Buggy folgt wieder einem Zick-Zack-Kurs.

Ungefähr hier liegt also die obere Grenze für den Proportionalitätsfaktor. Die Stellgröße *u* liegt dabei immer noch im Bereich von ±100; wir müssen die Basisgeschwindigkeit der Motoren also nicht senken, um den Regelungsbereich zu vergrößern.

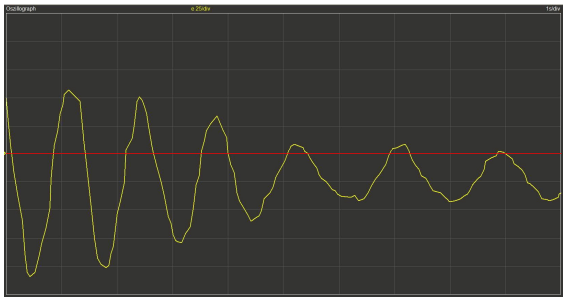


Abb. 11: Mit $K_p = 0{,}9$ beginnt der P-Regler zu oszillieren [Skala: 1s/25]

Zusammengefasst: Der P-Regler korrigiert bei großen Abweichungen stärker als bei kleinen und passt sich so an eine Störung an. Allerdings schwingt er leicht über und beginnt bei zu großem K_p irgendwann zu oszillieren. Außerdem schwingt er sich auf einen festen Abstand zur Spur ein. Diese Nachteile versuchen wir nun zu mildern.

D-Glied

Das Überschwingen des P-Reglers können wir dämpfen, indem wir den Regler beim Zurückschwingen „abbremsen": Wenn die Größe der Abweichung zunimmt, soll die Richtungskorrektur des Buggy verstärkt, wenn sie abnimmt, soll die Korrektur reduziert werden.

Dazu ergänzen wir die Berechnung der Stellgröße um einen Korrekturwert, der sich aus der *Veränderung* der Abweichung berechnet, also der ersten Ableitung des Verlaufs der Regelgröße. Dieser Korrekturwert wird daher auch *Differentialglied* (oder D-Glied) genannt.

Dabei muss die Veränderung immer über dieselbe Zeiteinheit gemessen werden. In unserem Programm erzwingen wir das mit einem Warte-Element von 20 ms in der Schleife (Abb. 12). Noch besser wäre, die Zeit zwischen zwei aufeinander folgenden Linienerkennungen zu messen.

Auch hier benötigen wir einen Faktor (K_d), mit dem wir die Veränderung der Abweichung multiplizieren, bevor wir sie zur Stellgröße addieren.

$$u = K_p \cdot e + K_d \cdot (e - e_{last})$$

Abb. 12 zeigt das resultierende ROBO Pro-Programm.

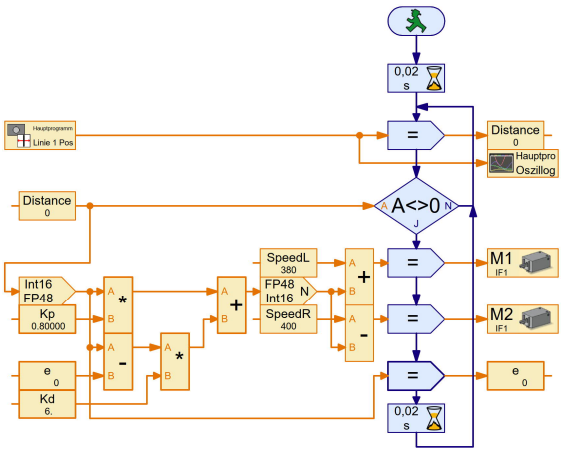


Abb. 12: ROBO Pro-Programm ‚Spurfolger PD-Regler.rpp'

Es gibt verschiedene Ansätze, um einen geeigneten Wert für K_d zu finden. Eine bewährte praktische Methode ist, zunächst einen möglichst großen Proportionalitätsfaktor zu wählen, bei dem der P-Regler gerade noch nicht oszilliert (hier z. B. $K_p = 0{,}8$). Dann startet man mit einem kleinen Wert für K_d (z. B. $K_d = 1$) und erhöht K_d so lange schrittweise, bis der Regler sich schnell der Führungsgröße nähert, ohne überzuschwingen. Bei unserem Buggy erreichen wir das z. B. mit $K_d = 6$ (Abb. 13).

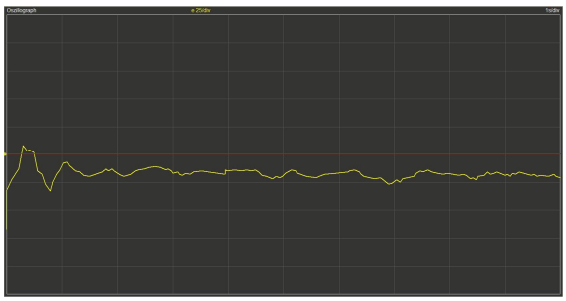


Abb. 13: PD-Regler mit $K_p = 0{,}8$ und $K_d = 6{,}0$ [Skala: 1s/25]

Mit etwas Aufwand kann man den PD-Regler nun noch feiner justieren, bis das Überschwingen kaum noch zu erkennen ist. Die stabile Dauerabweichung von der Mitte der Spur liegt jedoch immer noch bei etwa 20 – diesen Nachteil des P-Reglers schwächt das D-Glied zwar ab, gleicht ihn jedoch nicht vollständig aus.

I-Glied

Die stabile „Dauerabweichung" des Buggys von der exakten Mitte der Spur können wir mit einem weiteren Regelglied korrigieren, das die Abweichungen summiert und in die Berechnung der Stellgröße einfließen lässt. Dieses „integrierende" Reglerglied wird auch als I-Regler (oder I-Glied) bezeichnet.

Damit der I-Anteil des Reglers sich nicht auf einen beliebig großen Wert aufaddiert, müssen wir den Wert „kappen" oder auf andere Art das Wachstum begrenzen, z. B. indem wir vor dem Aufaddieren der aktuellen Abweichung den Wert verkleinern (bspw. halbieren) oder beim Erreichen der Führungsgröße den Wert auf Null zurücksetzen.

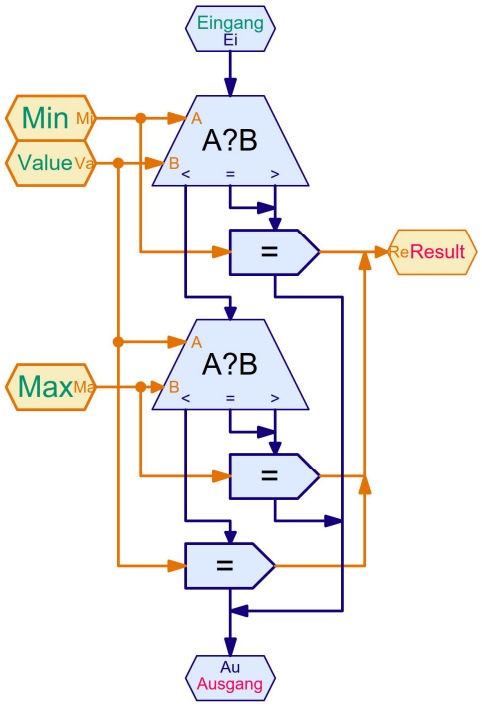


Abb. 14: Unterprogramm Constrain

In unserem Unterprogramm *Constrain* (Abb. 14) wird der Wert der Fehlersumme sum_e auf ±500 begrenzt. Die Grenze kann nahezu beliebig gewählt werden – der Integralfaktor K_i, mit dem wir den aufsummierten Fehler sum_e bei der Stellgrößenberechnung multiplizieren, bestimmt den Einfluss auf die Stellgröße.

$$u = K_p \cdot e + K_d \cdot (e - e_{last}) + K_i \cdot sum_e$$

K_i müssen wir (abhängig vom Maximalwert der Fehlersumme) relativ klein wählen: Er soll ja wirken, wenn Proportional- und Differentialanteil klein sind und der Buggy mehr oder weniger geradeaus fährt.

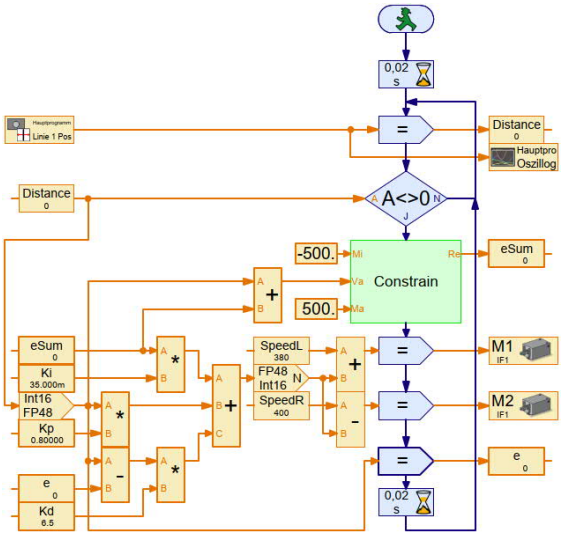


Abb. 15: Programm ‚Spurfolger PID-Regler.rpp'

Abb. 15 zeigt den PID-Regler als ROBO Pro-Programm. Die Faktoren K_p und K_d übernehmen wir von unserem PD-Regler. Den Faktor K_i müssen wir so lange vergrößern, bis der Buggy sich auf der Spur einschwingt. Da das Integralglied das Überschwingen verstärkt, müssen wir zum Ausgleich K_d leicht vergrößern.

In Abb. 16 sehen wir das Ergebnis eines Testlaufs mit $K_i = 0{,}035$, in dem die Annäherung an die Mitte der Spur recht gut gelingt. Das ist ein Reglerverlauf, wie wir ihn uns wünschen: Eine schnelle Reaktion in ca. 100 ms, minimales Überschwingen und kein dauerhafter Abstand zur Spur.

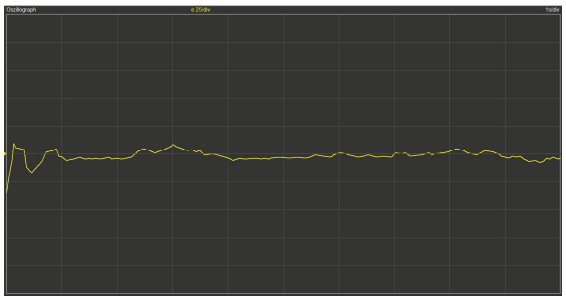


Abb. 16: PID-Regler mit $K_p = 0{,}8$, $K_d = 6{,}5$ und $K_i = 0{,}035$ [Skala: 1s/25]

Offline-Mode

Die ROBO-Pro-Programme zeigen (im Online-Mode) bei jeder Linienerkennung den gemessenen Abstandswert im Oszilloskop an – 25 Mal pro Sekunde. Wenn wir die Daten stattdessen innerhalb der Mess-Schleife ausgeben lassen, dann wird der Verlauf „eckiger": Es werden nur noch etwa 10-12 Messwerte je Sekunde angezeigt.

Unser Programm schafft also statt 25 nur 10-12 Schleifendurchläufe pro Sekunde. Da der Aufwand der Rechenoperationen im PC nicht ins Gewicht fällt, muss es die Übermittlung der Daten sein, die die Regelstrecke verlängert. Das können wir ändern, indem wir die Oszilloskopanzeige durch eine Ausgabe des Abstandswerts auf dem TXT-Display ersetzen und das Programm im Download-Mode starten.

Allerdings sind nun unsere Regelungsparameter nicht mehr optimal: Durch die häufigeren Messungen reagiert der P-Regler öfter, und die Korrektur durch das D-Glied fällt geringer aus, da die Abstandsänderungen bei kürzeren Messintervallen natürlich betragsmäßig kleiner sind. Der Effekt wird noch verstärkt, wenn wir die Bildrate der Kamera auf den Maximalwert von 60 fps erhöhen.

Das Nachjustieren des Reglers gelingt aber auch ohne Oszilloskop. So können wir den Proportionalfaktor K_p in der Download-Variante des P-Reglers (Programm ‚Spurfolger P-Regler (download).rpp') auch mit bloßem Auge einstellen: Wir passen dessen

Wert (wie oben) so lange an, bis der P-Regler gerade noch nicht oszilliert.

Den Faktor K_d können wir übernehmen, müssen ihn aber durch den Quotienten der beiden Messfrequenzen (fps) teilen: 60/25 = 2,4. Durch Änderungen in kleinen Schritten können wir das Ergebnis erforderlichenfalls anpassen, bis das Überschwingen des Reglers kaum noch zu erkennen ist (Programm ‚Spurfolger PD-Regler (download).rpp').

Unseren Faktor K_i sollten wir überhaupt nicht anpassen müssen. Wenn es zu mehr Messungen kommt, „zieht" das I-Glied den Buggy schneller an die Linie – das hat höchstens Auswirkungen auf das D-Glied (Programm ‚Spurfolger PID-Regler (download).rpp').

Optimierungen

Gleich mehrere Parameter bestimmen die Wirkung des PID-Reglers und beeinflussen sich auch gegenseitig. Mit diesen Parametern könnt ihr insbesondere die Download-Version des Reglers experimentell weiter optimieren. Beachtet aber, dass ihr sie nur in kleinen Schritten ändert, da es sonst passieren kann, dass ihr den Regler komplett unbrauchbar macht.

Diese Parameter sind:

- Die *Basisgeschwindigkeit*: Sie begrenzt den Regelungsbereich zu Gunsten der Geschwindigkeit des Buggy.
- Der *Winkel der Kamera* und die *Position der Linienerkennung* im Bild: Je weiter entfernt vom Buggy die Spur erkannt wird, desto früher reagiert die Richtungskorrektur. Dadurch „schneidet" der Buggy allerdings engere Kurven (Abb. 17). Umgekehrt kann die Spur bei kurzer Voraussicht in einer scharfen Kurve das Bild verlassen, bevor der Buggy seine Fahrtrichtung ausreichend korrigiert hat.

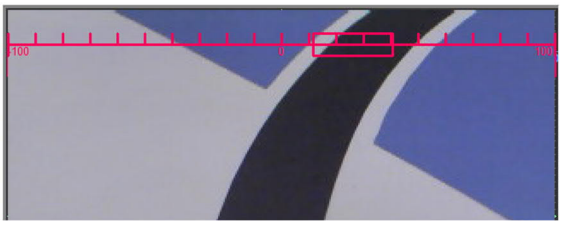


Abb. 17: Linienerkennung am oberen Bildrand

- Die *Bildrate* (fps), das Zeitglied des Reglers, bestimmt die Länge der Regelungsstrecke: Je kürzer, desto häufiger erfolgt eine Fahrtrichtungskorrektur, und desto geringer die gemessene Änderung der Abweichung.
- Die *Faktoren* K_p, K_d *und* K_i des PID-Reglers (mit der oben jeweils beschriebenen Wirkung).
- Die Art der *Begrenzung der Fehlersumme*, damit dessen Wert nicht ins Unkontrollierte wächst.

Den größten Effekt erzielt ihr mit einer Erhöhung der Basisgeschwindigkeit und der anschließenden Anpassung aller drei Faktoren. Dabei könnt ihr wie im vorangegangenen Abschnitt vorgehen – K_i sollte konstant bleiben, K_p bis kurz vor dem Oszillieren vergrößert werden und K_d nur eine geringe Anpassung benötigen.

Kreisparcours

Der Regler funktioniert natürlich nicht nur auf einer Geraden – schließlich soll er die Fahrt des Buggy ja gerade an kurvige (also „gestörte") Spurverläufe anpassen. Wir können ihn daher z. B. mit dem ROBO Explorer Parcours (128962) oder dem ROBO TX Training Lab Parcours 1 (Rückseite, 137652) oder Parcours 2 (Vorderseite, 137676) testen. In der fischertechnik-Datenbank gibt es alle drei Parcours' im A1-Format zum Ausdrucken.

Im Download-Bereich der ft:pedia habe ich euch einen Parcours in A0 (ca. 1020 x 720 mm) ohne Farbflächen bereitgestellt – darauf lässt sich der Spurfolger hervorragend testen (Abb. 18).

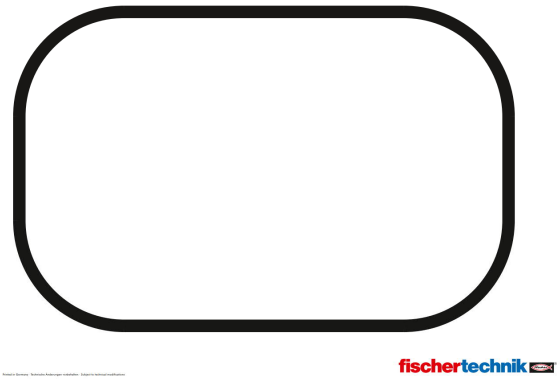


Abb. 18: Kreisparcours zum Download

Damit die Regelung bei sehr engen Kurven noch „mitkommt“ und die Kamera die Spur nicht verliert, müsst ihr ggf. die Basis-Geschwindigkeit ein wenig senken. Eine geringfügig überhöhte Geschwindigkeit gleicht der Buggy aus: Verliert er die Spur, sorgt die Warteschleife im Programm dafür, dass die Einstellung der Motoren erhalten bleibt. So findet der Buggy die Spur normalerweise auch bei sehr scharfen Kurven kurz darauf wieder.

Wer es probieren möchte: Mit meiner (leidlich optimierten) Download-Version des PID-Reglers mit einer Basisgeschwindigkeit von 485 benötigt mein Buggy knapp 10 Sekunden für eine Runde auf dem Parcours in Abb. 18. Wer schafft das schneller?

Fazit

Der 1922 von *Nicolas Minorsky* (1985-1970), dem Erfinder des Gyrometers, für die US-Navy entwickelte PID-Regler ist heute Standard bei anspruchsvollen Regelungsaufgaben. Mit diesem äußerst mächtigen Werkzeug lassen sich – ganz ohne KI oder andere „Kanonen“ – beeindruckend autonome Systeme entwickeln.

Seine volle Leistungsfähigkeit erreicht der PID-Regler, wenn er, wie beschrieben, Schritt für Schritt an das System angepasst wird. Abb. 19 zeigt den Verlauf der Regler-Varianten bei optimaler Wahl der Parameter im Vergleich.

Dabei ist zu beachten, dass jede Änderung von Komponenten des Systems mit Einfluss auf die Regelung eine neue Kalibrierung des PID-Reglers (also die Bestimmung der Faktoren K_p, K_d und K_i) erforderlich machen kann.

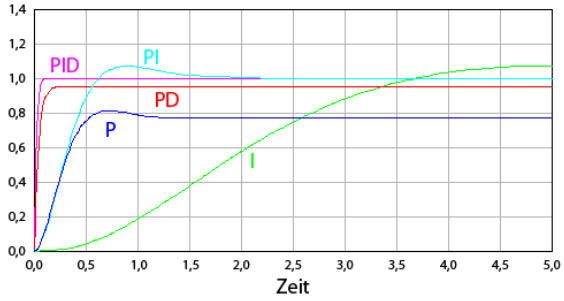


Abb. 19: Verlauf der Regler-Varianten im Vergleich (Bild: rn-wissen.de)

Das Experimentierbeispiel „analoger Spurfolger“ zeigt, dass das kein Hexenwerk ist, sondern – sofern man dabei systematisch vorgeht – mit begrenztem Aufwand gemeistert werden kann. Es ist auch faszinierend, die Wirkung des Reglers, die mit bloßem Auge kaum erkennbar ist, in der Oszilloskopanzeige von ROBO Pro zu verfolgen. Für komplexere Systeme als unseren Buggy gibt es außerdem heuristische Faustformelverfahren zur Bestimmung der Parameter eines PID-Reglers [7, 8].

Beachten muss man allerdings, dass die wichtigste Grenze eines jeden (PID-) Reglers das Zeitglied ist: Da die Wirkung der Regelung fast immer erst etwas verzögert eintritt, kann der Regler nur Störungen bis zu einer bestimmten Größe ausgleichen. Wird bei unserem Buggy beispielsweise die Kurve zu eng, verliert die Kamera die Spur, bevor die Richtungskorrektur abgeschlossen ist. Diese Grenze sollte man zuvor abschätzen, um zu wissen, ob der Regler ein vorliegendes Regelungsproblem überhaupt lösen kann.

Mit dem fischertechnik TXT Controller und ROBO Pro kann man die Wirkung eines PID-Reglers über die Oszilloskop-Funktion auch hervorragend veranschaulichen – z. B. im Informatik- oder Technik-Unterricht durch eine Projektion des Grafen via Beamer oder Wand-Display.

Alle in diesem Beitrag verwendeten ROBO Pro-Programme findet ihr als Download zu dieser Ausgabe der ft:pedia.

Referenzen

[1] Honeywell: *Grundlagen der Reglerparametrierung*. Einführung, 2007.

[2] Tim Wescott: *PID without a PhD*. 2018.

[3] Artur Fischer: *Zweipunktregelung*. Hobby 4 Band 2 (39052), Fischer-Werke, 1971, S. 48-51.

[4] Stefan Falk: *Dreipunktregelung*. ft:pedia 2/2019, S. 32–38.

[5] Artur Fischer: *Dreipunktregelung*. Hobby 4 Band 5 (39545), Fischer-Werke, 1975, S. 10-13.

[6] Dirk Fox: *Endliche Automaten in Robo Pro*. ft:pedia 3/2014, S. 42–50.

[7] Wikipedia: *Faustformelverfahren (Automatisierungstechnik)*.

[8] Jim Sluka: *A PID Controller For Lego Mindstorms Robots*. 2009-2018.

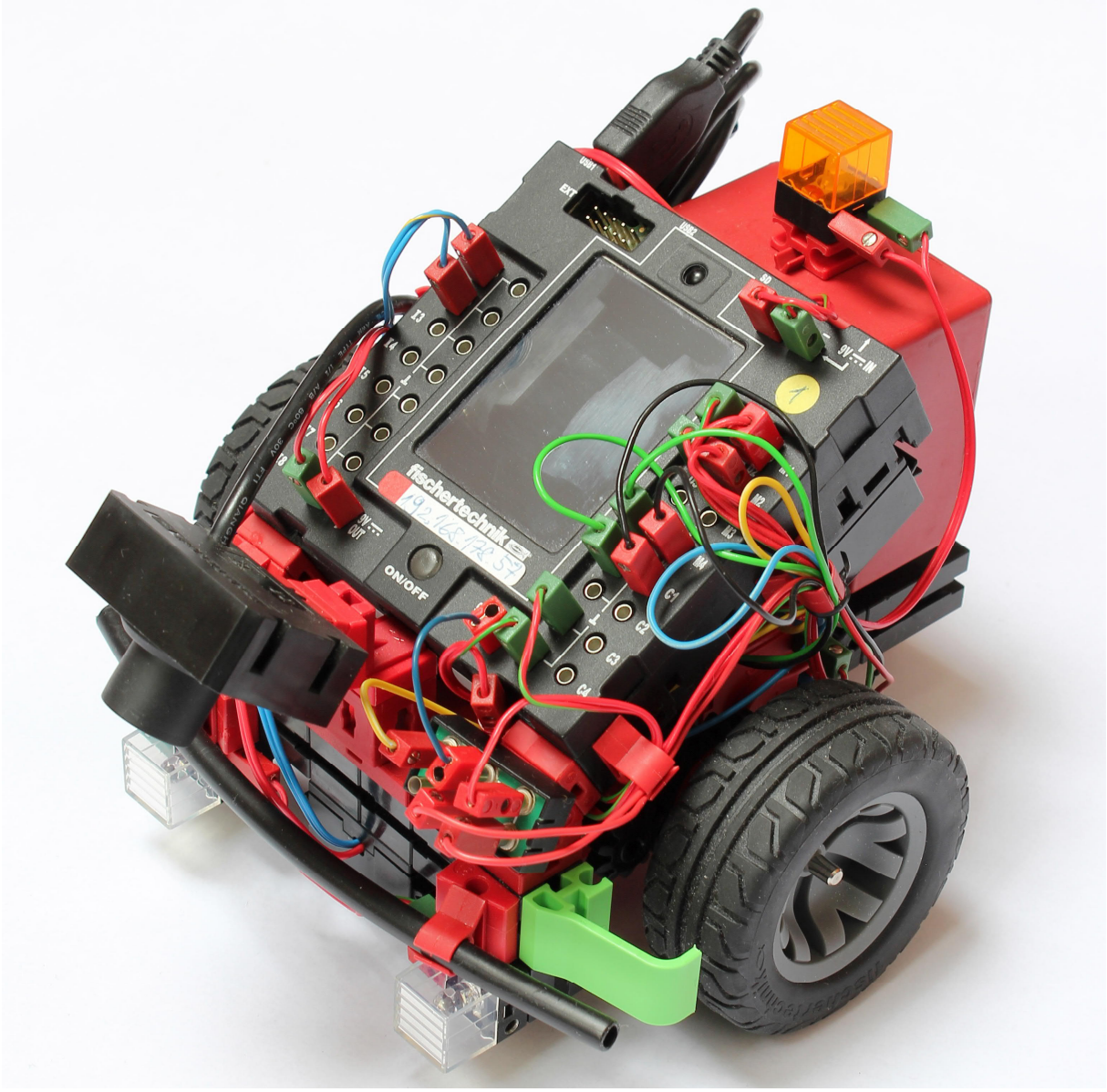


Abb. 20: Buggy mit Blinklicht, Frontscheinwerfern, Kamera und Spursensor

Computing

fischertechnik-Roboter mit Arduino (Teil 4): Buggy-Steuerung mit dem ftDuino

Helmut Jawtusch

Die Roboter aus dem Buch „fischertechnik-Roboter mit Arduino“ [1] haben zahlreiche Leser zu weiteren Modellvarianten angeregt. In loser Folge werden in dieser Serie einige dieser Modelle vorgestellt. Diesmal wird das Herz des „Buggy“ durch einen ftDuino ersetzt.

Hintergrund

Im Buch „fischertechik-Roboter mit Arduino“ [1] beschreibt Dirk Fox den Bau eines Buggys mit zwei S-Motoren und der Steuerung über den Arduino Uno. Zum Buch gibt es die Webseite: fischertechnik-roboter-mit-arduino.de, auf der sich Sketche und ergänzendes Material zum Download befinden.

Mich haben vor allem die Anwendungen interessiert, bei denen alternativ der ftDuino verwendet werden kann.

IR-Fernsteuerung

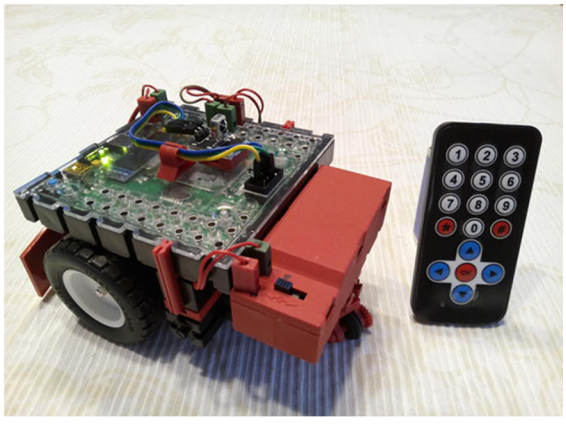


Abb. 1: Buggy mit IR-Fernsteuerung

Ich habe mir für ca. 5 € die *Infrarot IR Drahtlose Fernbedienung mit Sensor VS1838B für Arduino 100* gekauft. Beim Arduino wird die Datenleitung der IR-Diode am Pin D7 angeschlossen. Der Anschluss des IR-Sensors am ftDuino erfolgt über den I2C-Port: GND an 1, Vcc (+5V) an 2 und die Daten an 5 (SDA). Zur Decodierung der Signale der Fernsteuerung wird (wie am Arduino auch) das Skript IRremote.h eingebunden. Das folgende Skript zeigt den ersten Teil des Sketches (~.ino) mit den Anpassungen an den ftDuino:

```
#include <adc_table.h>
#include <Ftduino.h>
#include <IRremote.h>

//#define DEBUG
#ifdef DEBUG
#define Baudrate 115200
#endif
//Tastencodes der Fernsteuerung
#define IRreceive 2
#define Arduino     0xFF
#define Vorwaerts   0x18E7
#define Rueckwaerts 0x4AB5
#define Rechts      0x5AA5
#define Links       0x10EF
#define Schneller   0xB04F
#define Langsamer   0x6897
#define Stopp       0x38C7

#define MaxSpeed 64 //
Maximalgeschwindigkeit der Motoren
#define Step 4 // Schrittweite der
Geschwindigkeitsaenderung

IRrecv irrecv(IRreceive);
unsigned int mode = 1;
decode_results results;
byte motorspeed = 32;
```

```
void rechts(uint16_t speed)
  { mode = 1;
    ftduino.motor_set(Ftduino::M2,
Ftduino::LEFT, speed );
    ftduino.motor_set(Ftduino::M3,
Ftduino::RIGHT, speed );
   }
…
void set (uint16_t m) {
  switch(m) {
    case 1: rechts(motorspeed);
      break; ...
```

Listing 1: Sketch-Anpassungen der IR-Fernsteuerung des Buggy für den ftDuino

Die Datenabfrage erfolgt hier über den Pin 2 (`IRreceive`) des im ftDuino ‚verbauten' Arduino Leonardo, der direkt mit dem SDA-Pin des I2C-Ports verbunden ist.

Achtung: Beim Hochladen des Skriptes über die USB-Verbindung erscheint merkwürdigerweise eine Fehlermeldung, wonach die IDE am COM-Port keinen ftDuino finden kann. Wenn man beim Hochladen aber kurz eine Taste der IR-Fernsteuerung drückt, wird der ftDuino ohne Fehler gefunden und das Skript wird hochgeladen.

Die angegebenen Tasten-Codes passen zu der kleinen Fernbedienung, die in Abb. 1 zu sehen ist. Bei Verwendung anderer IR-Fernbedienungen kann man deren Tasten-Codes mit dem *Seriellen Monitor* der Arduino-IDE ermitteln. Mit der Flag-Variablen `mode` und der Prozedur `set(…)` wirkt sich eine Veränderung der Geschwindigkeit unmittelbar auf die aktuelle Bewegung aus.

Pixy-Folger, Hinderniserkenner und Smartphone-Steuerung

Alle Anwendungen aus meinem Artikel in der ft:pedia 1/2021 [2] lassen sich auf den Buggy übertragen, da ich dort die Counterfunktion der Encodermotoren nicht verwende. Ich möchte hier nur auf den Hinderniserkenner eingehen. Die Abfrage der Entfernungen mit dem Ultraschall-Abstandssensors ist für das Umfahren eines Hindernisses zu langsam (zwei Messungen pro Sekunde). Für diesen Zweck eignet sich hervorragend der IR-Abstandssensor Sharp OA41SK mit einem Messbereich von 4 cm bis 30 cm (Abb. 2).

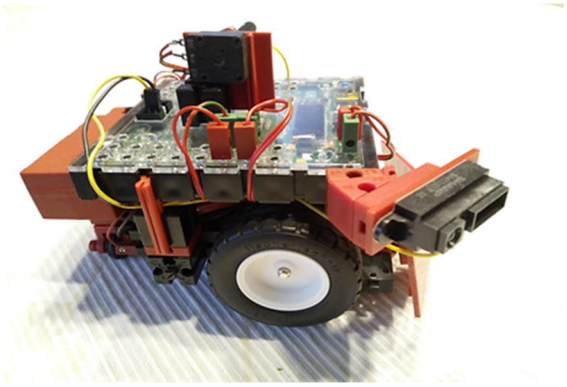


Abb. 2: Buggy mit IR-Abstandssensor

Die Stromversorgung erfolgt wieder über Pin 1 und 2 am I2C-Port. Die Datenleitung kann an einen der I-Eingänge angeschlossen werden. Der Sensor liefert je nach Entfernung ohne Verzögerung Spannungswerte. Daher wird der Eingang auf VOLTAGE gestellt:

```
ftduino.input_set_mode(Ftduino::I4
, Ftduino::VOLTAGE)
```

Um die Geschwindigkeit bei Betrieb leicht an die Verhältnisse anpassen zu können habe ich noch ein 5 kOhm-Potentiometer angeschlossen.

Die ftDuino-Sketche zur IR-Fernsteuerung und zum Hinderniserkenner sowie der Treiber IRremote können als Anlage zu diesem Beitrag auf ftpedia.de heruntergeladen werden.

Quellen

[1] Dirk Fox, Thomas Püttmann: *fischertechnik-Roboter mit Arduino*. dpunkt-Verlag, 2020.

[2] Helmut Jawtusch: *TX Controller und ftDuino im Vergleich*. ft:pedia 1/2021, S. 114–118.

Computing

Experimente mit dem Kombisensor – Teil 2

Dirk Fox

Vor drei Jahren brachte fischertechnik den „Kombisensor“ (158402) auf den Markt – einen IMU-Sensor (Inertial Measurement Unit), der via I²C-Protokoll an den TXT angeschlossen werden kann. Der etwas unglückliche Name verrät allerdings nicht, was der Sensor alles kann. Im ersten Teil des Beitrags [3] habe ich den Sensor und einige Einsatzmöglichkeiten vorgestellt. Jetzt geht es ein wenig in die Tiefe – und heraus kommt ein verbesserter ROBO Pro-Treiber.

Einführung

Der im Kombisensor von fischertechnik [1] werkelnde Bosch-IMU-Sensor BMX055 [2] umfasst einen Beschleunigungs-, einen Kompass- und einen Gyrosensor. Der ROBO Pro-Treiber von fischertechnik reizt allerdings die Möglichkeiten des Sensors bei weitem nicht aus – und enthält zudem einige kleine und ein paar größere Bugs. Im ersten Teil des Beitrags habe ich einige davon gefixt und gezeigt, wie man den Kompass-Sensor kalibriert.

In diesem Teil gehen wir nun ein wenig tiefer auf die Möglichkeiten des Sensors ein, korrigieren ein paar weitere Fehler und ergänzen den Treiber um neue Funktionen.

Beschleunigungssensor

Rauschen

Das Rauschen des Beschleunigungssensors in der ROBO Pro-Implementierung von fischertechnik lässt sich, wie im ersten Teil des Beitrags gezeigt [3], unter anderem darauf zurückführen, dass die Rohwerte der X-, Y- und Z-Achse nicht arithmetisch um vier Bits geshiftet werden. Nur die 12 höchstwertigen Bits der Rohdaten sind signifikant; die niederstwertigen vier können beliebige Daten enthalten.

Das verbleibende Rauschen hängt vor allem von der gewählten *Output Data Rate* (ODR) ab. Bei maximaler ODR (2000 Samples/s) liegt es an allen drei Achsen bei ca. ±0,01 g (g = 9,81 m/s², Abb. 1). Reduziert man die ODR, sinkt auch das Rauschen: Wenn die Anwendung mit 16 Messungen pro Sekunde auskommt, ist kein Rauschen mehr festzustellen (Abb. 2).

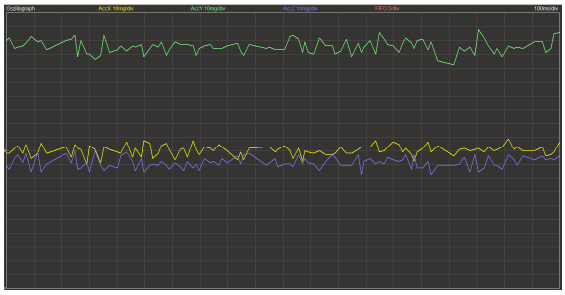


Abb. 1: Acc-Messwerte bei ODR = 2000 Hz [Skala: 0,1s/0,01g]

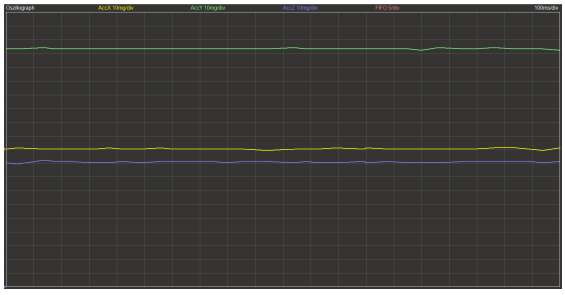


Abb. 2: Acc-Messwerte bei ODR = 16 Hz [Skala: 0,1s/0,01g]

Obwohl sich das Rauschen im Bereich von ±0,01 g bewegt, wirkt sich der Unterschied deutlich auf die Winkelberechnungen aus (Pitch und Roll, siehe [3]).

Abb. 3 zeigt Pitch und Roll in Ruhelage bei 2.000 Samples/s, Abb. 4 bei 16 Hz: Das Rauschen sinkt auf ein 20stel – von ±1° auf ±0,05°.

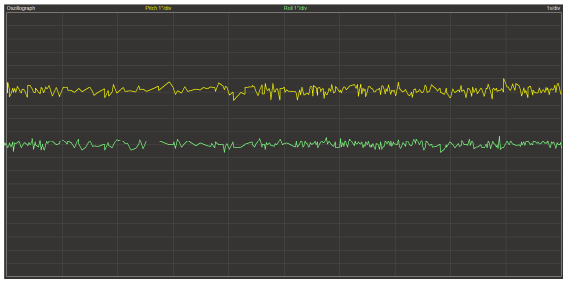


Abb. 3: Pitch (gelb) und Roll (grün) in Ruhelage bei 2.000 Samples/s [Skala: 1s/1°]

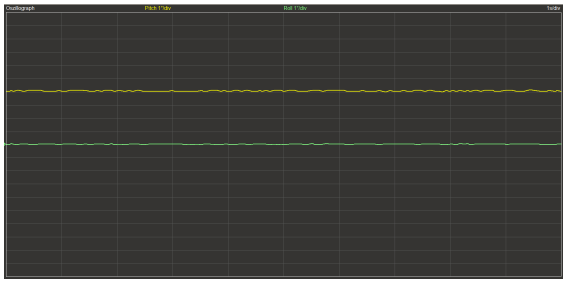


Abb. 4: Pitch (gelb) und Roll (grün) in Ruhelage bei 16 Samples/s [Skala: 1s/1°]

Wenn die Genauigkeit der Beschleunigungswerte wichtig ist, sollte daher eine möglichst niedrige Messfrequenz gewählt werden.

Offset

Sensoren haben oft einen kleinen konstanten Fehler (Offset). Wenn wir den Sensor auf einer mit einer Wasserwaage ausgerichteten Plattform montieren (Abb. 5, 6), können wir diesen Fehler messen (Abb. 7).

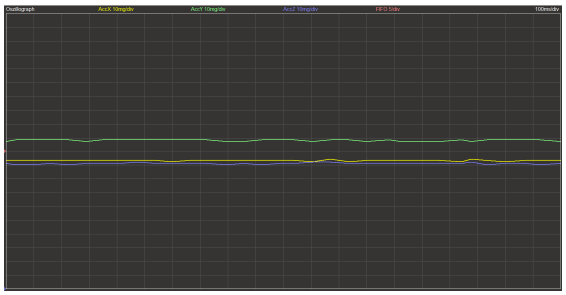


Abb. 7: Konstanter Fehler von etwa 0,01 g in X- (gelb), Y- (grün) und Z-Richtung (blau) [Skala: 0,1s/0,01g][1]

Der Sensor bietet mehrere integrierte Verfahren zur Kompensation solcher Fehler oder Störungen. Mit der *Fast Compensation* wird über 16 Messungen der mittlere Fehler der Messwerte einer Achse zu einem Vorgabewert bestimmt. Als Vorgabe kann in Register 0x37 für jede der drei Achsen einer der Werte +1g, -1g oder 0 gewählt werden. Bei allen weiteren Messungen wird der Fehler dann automatisch abgezogen.

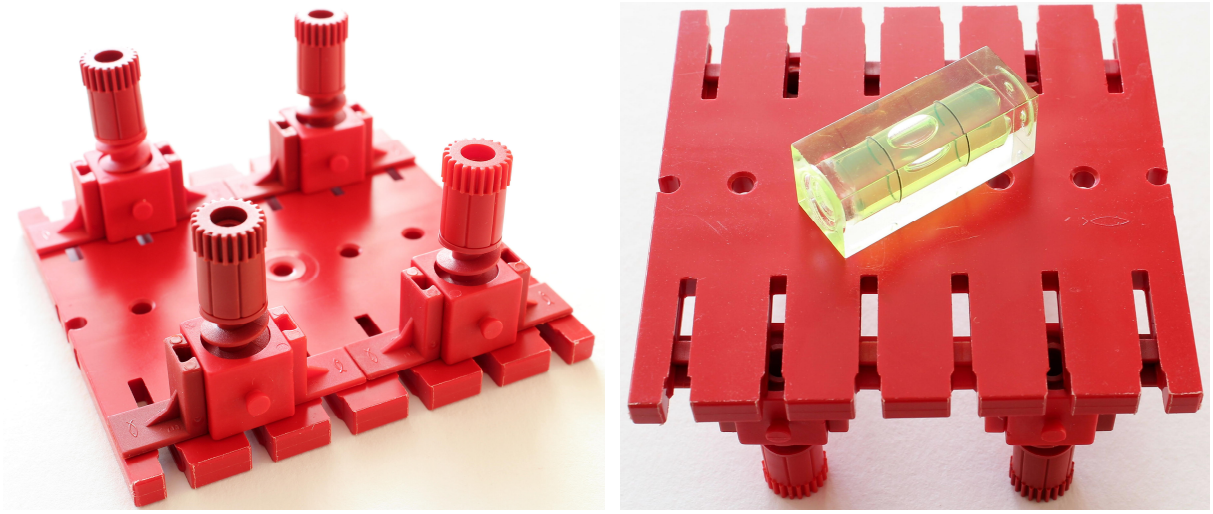


Abb. 5, 6: Messplattform für den Kombi-Sensor, ausgerichtet

[1] Ein Tipp für die Nutzung des Oszilloskops: Ein Druck auf die „alt“-Taste hält die Anzeige an.

Die Funktion *BMX055_AccFastComp*, um die ich den Treiber ergänzt habe, führt die Kompensation nacheinander für jede der drei Achsen durch. Dazu wird in Register 0x36 jeweils ein Trigger für die entsprechende Achse gesetzt. Anschließend stehen die Offset-Werte in den Registern 0x38 (X-Offset), 0x39 (Y-Offset) und 0x3A (Z-Offset) und können dort ausgelesen werden.

Die Bestimmung des Offset muss im Range ±2g in horizontaler Ruhelage erfolgen. Die Wirkung der Korrektur veranschaulicht der Graf in Abb. 8.

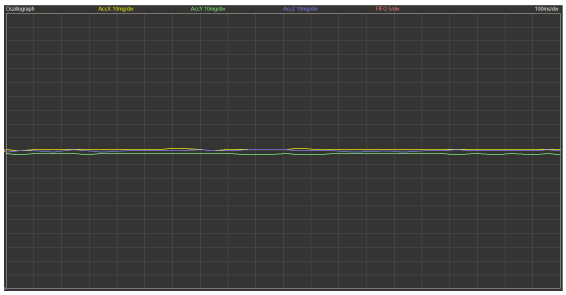


Abb. 8: Messwerte nach Fast Compensation [Skala: 0,1s/0,01g]

In Ruhelage darf der Sensor nur die Erdbeschleunigung messen. Also muss die Wurzel aus der Summe der Quadrate der drei vom Sensor bestimmten Beschleunigungswerten genau 1 g entsprechen [3]. Im Programm *Acc-Oszilloskop.rpp* wird die Abweichung von diesem Wert in % angezeigt. Ohne Kompensation liegt der Fehler bei bis zu ±0,9%; nach der Kompensation sinkt der Fehler unter ±0,1% – die Messwerte sind also noch einmal fast um den Faktor 10 genauer.

Die *Slow Compensation*, ein Hochpassfilter, filtert niedrige Frequenzen aus und „zieht“ die Mittelwerte der Beschleunigungswerte kontinuierlich gegen Null. Sie kann in Register 0x37 wahlweise auf ein oder 10 Hz eingestellt und in Register 0x36 für jede der drei Achsen einzeln aktiviert werden. Das kann z. B. für die X- und Y-Achse sinnvoll sein, wenn bei einer gleichförmigen Bewegung die relativen Beschleunigungen gemessen werden sollen. Im Treiber übernimmt das die Funktion *BMX055_AccSlowComp*.

Die Einstellungen und berechneten Kompensationswerte bleiben bis zum Ausschalten des Sensors erhalten. Damit sie nicht manuell zurückgesetzt werden müssen (Offset-Reset in Register 0x36), habe ich in der Funktion *BMX055_AccInit* einen Softreset (über Register 0x14) ergänzt.

Pufferung

Für die Nutzung des Sensors in einer Echtzeitanwendung ist eine weitere Korrektur im fischertechnik-Treiber erforderlich.

Der Sensor bietet die Möglichkeit, bis zu 32 Messwerte in einem FIFO-Puffer (*First In, First Out*) zwischenzuspeichern. Für ausgewählte Anwendungen kann das hilfreich sein, wenn der Zugriff auf die Daten *temporär* langsamer erfolgt als die gewählte *Output Data Rate* (ODR) des Sensors und man keine Messwerte verlieren möchte. Ist das Auslesen der Messwerte jedoch *generell* langsamer als die Bandbreite, führt die Verwendung des FIFO-Puffers dazu, dass man beim Auslesen veraltete Messwerte erhält. Schon bei der relativ niedrigen ODR von knapp 31 Hz läuft der Puffer z. B. beim Acc-Oszilloskop ruck-zuck voll: In Abb. 9 sieht man die Zahl der im Puffer gespeicherten Messwerte (rote Kurve) in weniger als einer Sekunde auf 32 ansteigen.

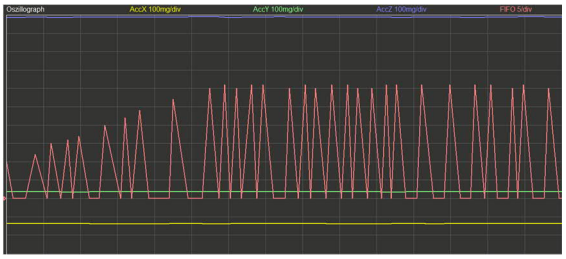


Abb. 9: Volllaufen des FIFO-Puffers (rot: Anzahl Messwerte im Puffer [Skala: 0,1s/5])

fischertechnik nutzt diesen STREAM-Modus des FIFO-Stack bei der Initialisierung des Sensors standardmäßig, weil sich die Verfügbarkeit neuer Messdaten über den FIFO-Status im Register 0x0E elegant abfragen lässt. Das ist allerdings unnötig:

Jeder Messwert wird vom Sensor immer auch im FIFO abgelegt und seine Verfügbarkeit über 0x0E signalisiert. Der Default-Modus BYPASS macht den FIFO zu einem Stack der Tiefe 1 – und schon liest man nur noch den jüngsten Wert aus.

Läuft der FIFO bei einer ODR von 31 Hz im STREAM-Modus voll, liest der TXT regelmäßig Werte des Beschleunigungssensors aus, die fast eine Sekunde früher gemessen wurden. Das kann man mit bloßem Auge erkennen: Wenn man den Sensor bewegt, erfolgt die Anzeige im Oszilloskop mit dem fischertechnik-Treiber sichtbar verzögert. Für eine Echtzeitanwendung sind solche veralteten Sensorwerte vollkommen unbrauchbar.

Die verbesserte Version des ROBO Pro-Treibers verwendet daher den FIFO-Modus BYPASS mit Stack-Tiefe 1. Die Wirkung erkennt man unmittelbar an der Reaktionszeit des Acc-Oszilloskops auf Bewegungen des Sensors. Zusätzlich habe ich eine Funktion ergänzt, mit der man den FIFO-Modus wählen kann. Die Messwerte, die in den FIFO eingestellt werden, können dabei auch auf einzelne Achsen des Beschleunigungssensors beschränkt werden. Damit verkürzt man sowohl die Messung als auch den Auslesevorgang, wenn man nur einen Achsenwert auswerten möchte. Diese Funktion *BMX055_AccSetFIFOMode* zeigt Abb. 10.

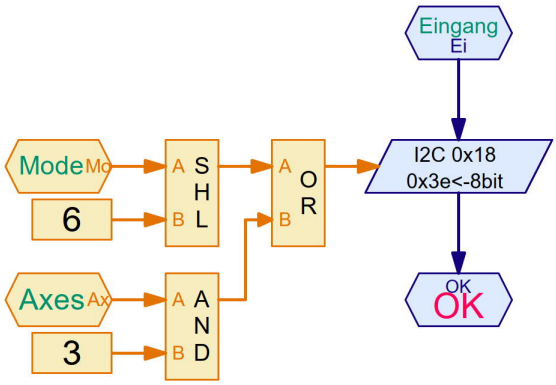


Abb. 10: Einstellung FIFO-Modus (0: BYPASS, 1: FIFO, 2: STREAM), zu berücksichtigende Achsen (0: alle, 1: X, 2:Y, 3: Z)

Gyrosensor

Rauschen

Auch das Gyroskop rauscht stark (bis zu 0,01°/s), wenn die Rohdaten nicht gefiltert werden (Abb. 11). Da alle 16 Bit der Messwerte signifikant sind, wird es nicht durch undefinierte Bits verursacht. Auch wirkt sich die Wahl des Messbereichs nicht auf das Rauschen aus.

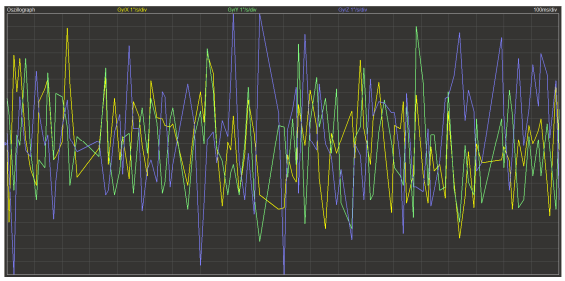


Abb. 11: Rauschen des Gyro-Sensors ohne Filterung [Skala: 0,1s/0,001°/s])

Das Rauschen lässt sich durch die Wahl eines Filters in Register 0x10 bei der Initialisierung senken: Ein kaskadiertes Integrator-Differentiator-Filter (CIC) verwirft bis zu 95% der Rohdaten und gibt die verbleibenden Messwerte an ein nachfolgendes *Infinite Impulse Response* Filter (IIR) weiter. Die ODR sinkt dabei auf 100 Hz – für viele Anwendungsfälle immer noch genug.

Die Wirkung der Filterung ist beeindruckend: Abb. 12 und 13 zeigen die gefilterten Daten bei einer Dezimierung auf die Hälfte (ODR: 200 Hz) resp. ein 20stel der Rohdaten (ODR: 100 Hz).

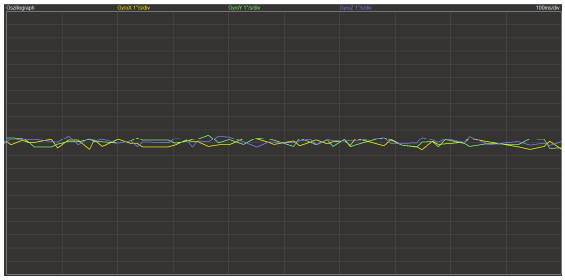


Abb. 12: Filter-Mode 2: Sensor-Werte mit Decimation Filter 2 [Skala: 0,1s/0,001°/s])

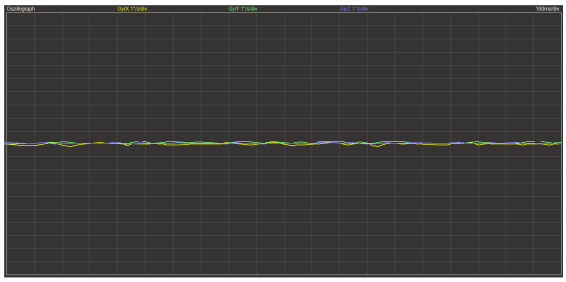


Abb. 13: Filter-Mode 5: Sensor-Werte mit Decimation Filter 20 [Skala: 0,1s/0,001°/s])

Vergrößert man die Skala des Oszilloskops kann man erkennen, dass das Rauschen bei ODR = 100 Hz auf etwa ±0,0002°/s sinkt (Abb. 14) – sensationell.

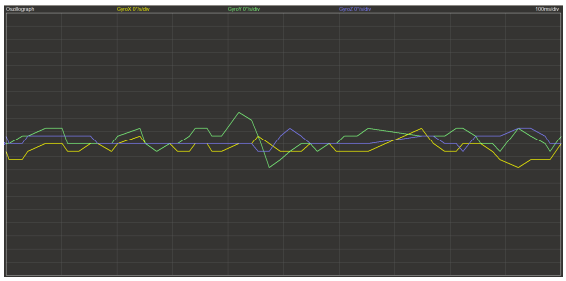


Abb. 14: Verbleibendes Rauschen bei maximaler Filterung [Skala: 0,1s/0,0001°/s]

Offset

Jeder Sensor, auch das Gyroskop, kann einen konstanten Fehler (Offset) haben. Bei meinem Sensor konnte ich allerdings keinen solchen Fehler messen (siehe oben).

Der Sensor bietet ähnliche integrierte Verfahren zur Kompensation von Fehlern wie der Beschleunigungssensor, die ebenfalls per Voreinstellung deaktiviert sind. Für die *Fast Compensation* kann man die Zahl der Messungen, deren Mittelwert als Offset bestimmt wird, zwischen 32, 64, 128 und 256 wählen und die Offsetbestimmung für jede der drei Achsen separat aktivieren. Um eine möglichst hohe Genauigkeit zu erzielen, wird die Einstellung des Messbereichs ±125 empfohlen.

Eingestellt, konfiguriert und aktiviert wird die Fast Compensation in Register 0x32. Das übernimmt im Treiber die neue Funktion *BMX055_GyroFastComp* (Abb. 15). Dabei sollte der Sensor in Ruhelage sein.

Die berechneten Offset-Werte werden in den Registern 0x36 bis 0x3A abgelegt und bei allen folgenden Messungen automatisch von den Rohwerten abgezogen.

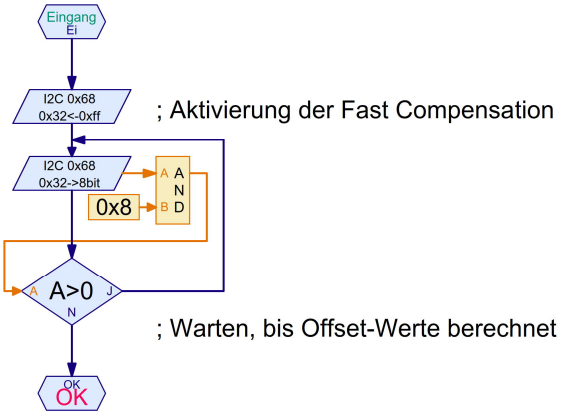


Abb. 15: BMX055_GyroFastComp

Bei der *Slow Compensation* misst der Sensor kontinuierlich, ob die Werte der selektierten Achsen des Gyroscops über einen einstellbaren Zeitraum (40 bis 1280 ms) um einen wählbaren Schwellenwert (0,1°/s bis 1°/s) von 0 abweichen. Tritt dieser Fall ein, wird der Offset vom Rohwert abgezogen. Die Rohdaten konvergieren anschließend gegen 0. Konfiguriert und aktiviert wird die Slow Compensation über Register 0x31. Das übernimmt die neue Treiber-Funktion *BMX055_GyroSlowComp*.

Auch beim Gyrosensor bleiben die Einstellungen und berechneten Offsets bis zum Ausschalten des Sensors erhalten. Damit sie nicht manuell zurückgesetzt werden müssen (Offset-Reset in Register 0x21), habe ich auch in der Funktion *BMX055_GyroInit* einen Softreset (Register 0x14) ergänzt.

Pufferung

Die Messungen des Gyrosensors können ebenfalls in einem FIFO-Stack vorgehalten werden. Hier passen sogar bis zu 100 Messungen in den Stack. Aber auch dieser Stack läuft in rund zwei Sekunden voll (Abb. 16). Bei einer Messfrequenz von 100 Samples/s führt das zu einer Verzögerung von einer Sekunde zwischen Messung und Auslesen des Werts.

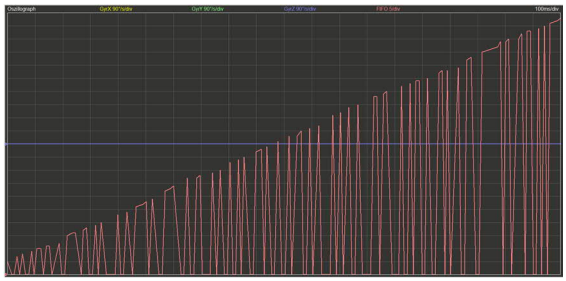


Abb. 16: Volllaufen des FIFO-Puffers (rot: Anzahl Messwerte im Puffer [Skala: 0,1s/5])

Auch bei der Initialisierung des Gyro-Sensors wird im fischertechnik-Treiber der FIFO-Stack im STREAM-Mode aktiviert – was unweigerlich zu veralteten Messwerten führt. Das Problem lässt sich wie beim Beschleunigungssensor lösen: Per Default ist der FIFO-Mode BYPASS im Register 0x3E eingestellt, der einen FIFO mit Tiefe 1 verwendet; im korrigierten Treiber wird dieser Modus beibehalten.

Mit der zusätzlichen neuen Treiber-Funktion *BMX055_GyrSetFIFOMode* kann jeder andere FIFO-Mode aktiviert werden, auch hier bei Bedarf für ausgewählte Achsen (Abb. 17).

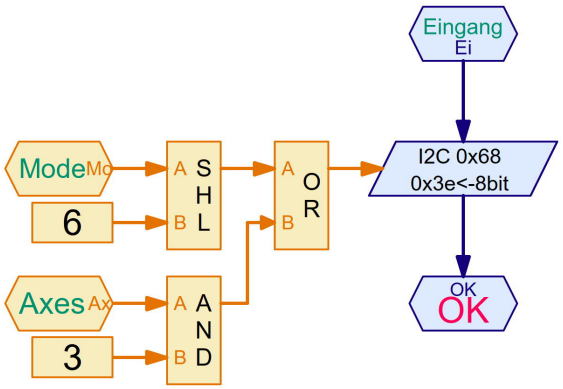


Abb. 17: Wahl des FIFO-Modus (0: BYPASS, 1: FIFO, 2: STREAM) und der zu berücksichtigenden Achsen (0: alle, 1: X, 2:Y, 3: Z)

Magnetometer

Rauschen

Wer sich die Magnetometer-Daten ansieht, die der ROBO Pro-Treiber liefert, wird sich über das starke Rauschen der Werte wundern. Eine Ursache dafür ist, wie in Teil 1 des Beitrags gezeigt [3], dass der fischertechnik-Treiber wie bei den Werten des Beschleunigungssensors keinen arithmetischen Rechts-Shift der X- und Y-Werte um die drei niederstwertigen Bits und beim Z-Wert um das niederstwertige Bit vornimmt, denn bei diesem Sensor sind nur die höchstwertigen 13 bzw. 15 Bits der Rohdaten signifikant. Aber auch nach der Korrektur der Werte enthalten die ungefilterten Daten ein deutliches Rauschen, liegen aber zumindest in dem in horizontaler Lage zu erwartenden „Range“ von 0-40 µT in X- und Y-Richtung und ziemlich genau 50 µT in Z-Richtung (Abb. 18). Insbesondere die Rohdaten der Z-Achse (blau) rauschen ganz erheblich.

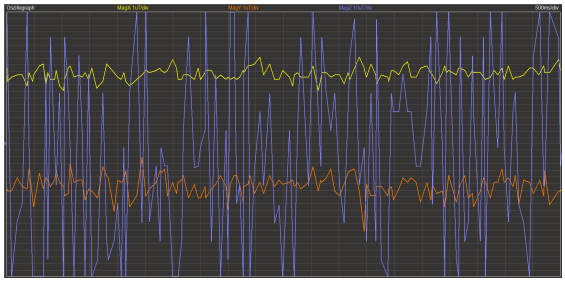


Abb. 18: Rauschen der ungefilterten Rohdaten [Skala: 0,5s/1µT bzw. 10µT (Z-Achse)]

Im ersten Teil des Beitrags habe ich gezeigt, wie sich das Rauschen durch eine Bestimmung des arithmetischen Mittelwerts über eine vorgegebene Anzahl von Messwerten verringern lässt [3]. Diese Lösung ist sehr flexibel, da die Anzahl der für die Mittelwertbestimmung herangezogenen Messwerte beliebig gewählt werden kann.

Wesentlich schneller ist es allerdings, diese Aufgabe dem Sensor zu übertragen. Denn der BMX055 bringt eine Funktion mit, die den Mittelwert aus einer in Register 0x51 (X-/Y-Achse) und 0x52 (Z-Achse) vorgegebenen Anzahl von Messungen ermittelt.

Prinzipiell kann jeder Wert aus {0, …, 255} konfiguriert werden; für die X- und Y-Achse ergibt sich die Zahl der Messungen daraus durch Multiplikation mit zwei. Dabei sinkt allerdings die ODR. Im Datenblatt werden vier „typische“ Varianten empfohlen (Tab. 1) [2].

Preset	Rep. X/Y *nXY*	Rep. Z *nZ*	recommended ODR [Hz]
Low power preset	3	3	10
Regular preset	9	15	10
Enhanced regular preset	15	27	10
High accuracy preset	47	83	20

Tab. 1: Empfohlene Wiederholraten für das Magnetometer

Der fischertechnik-Treiber verwendet in der Initialisierungsfunktion die Regular-Einstellung. Das Ergebnis zeigt Abb. 19.

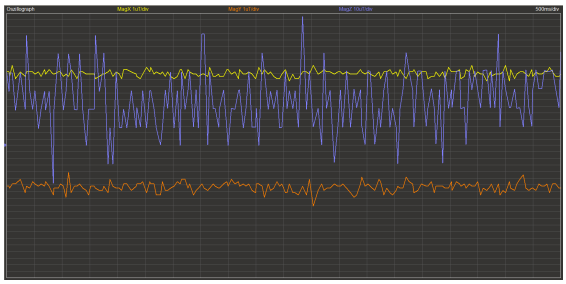


Abb. 19: Rauschen im Regular Mode [Skala: 0,5s/1µT bzw. 10µT (Z-Achse)]

Noch einmal deutlich reduzieren lässt sich das Rauschen mit der High-Accuracy-Filterung (Abb. 20).

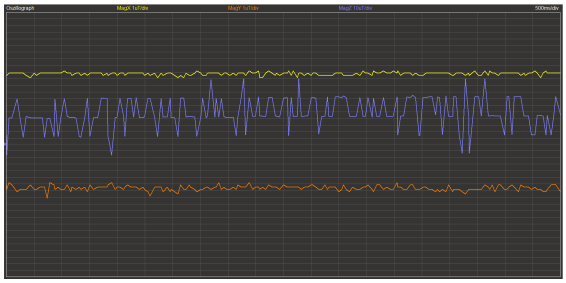


Abb. 20: Messdaten im High Precision Mode [Skala: 0,5s/1µT bzw. 10µT (Z-Achse)]

Nach Datenblatt lässt sich der Rauschanteil (*Root Mean Square*, RMS) damit gegenüber dem *Regular Mode* von 0,6 auf 0,3 µT halbieren. Das deckt sich mit meinen Messergebnissen.

Wählt man die maximal mögliche Anzahl an Messungen für die Mittelwertbildung (511 für X und Y, 256 für Z), so verschwindet das Rauschen in X- und Y-Richtung vollständig; in Z-Richtung bleibt ein „Restrauschen“ von ±30 µT. Die ODR sinkt auf ca. 6 Hz (Abb. 21).

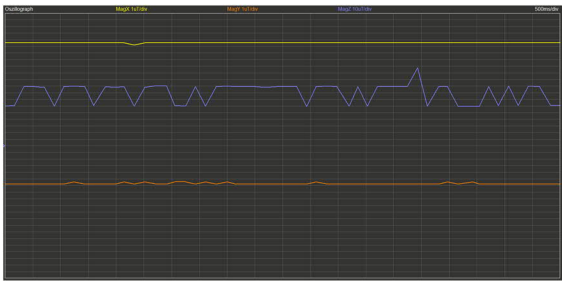


Abb. 21: Maximale Mittelwertbildung [Skala: 0,5s/1µT bzw. 10µT (Z-Achse)]

Ganz so weit muss man nicht einmal gehen: Auch eine Mittelwertbildung über 129 Messungen in X-, Y- und Z-Richtung lässt bei einer ODR von 15 Hz das Rauschen praktisch verschwinden. Um den Filtermodus wählbar zu machen, habe ich im Treiber die Funktion *BMX055_MagSetPrecMode* ergänzt, die auch einen „Mode 0“ (keine Filterung) und einen „Mode 5“ (*Ultra High Precision*) kennt (Abb. 22).

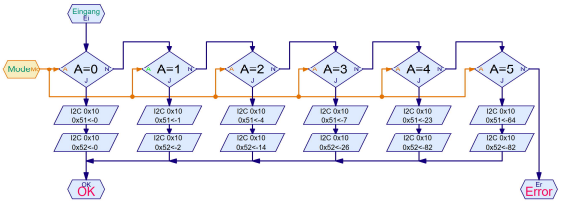


Abb. 22: Unterprogramm BMX055_MagSetPrecMode

Offset

Die Kalibrierung des Magnetometers zum Ausgleich eines Hard-Iron-Fehlers habe ich im ersten Teil des Beitrags beschrieben [3]. Die Variablen *CalX* und *CalY* müssen für die jeweilige Anwendungsumgebung gemessen und gesetzt werden.

Temperaturkompensation

Für die Kompensation der von der Temperatur des Sensors verursachten Störungen sind im fischertechnik-ROBO Pro-Treiber zwei Funktionen für die X- und die Y-Achse vorgesehen.

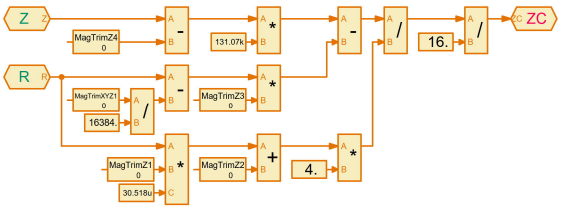


Abb. 23: Temperaturkompensation Z-Achse

Damit auch die Werte der Z-Achse genutzt werden können, die z. B. für die Berechnung der Neigungskompensation der Kompasswerte benötigt werden (siehe unten), habe ich die Initialisierungsfunktion um das Auslesen der erforderlichen Konstanten des Sensors erweitert und auf der Grundlage der Originaltreiber für den BMM050 von Bosch eine entsprechende Funktion für die Z-Achse ergänzt (*BMX055_MagCompZ*) [4].

Heading

Die im ersten Teil des Beitrags vorgestellte Berechnung der Abweichung von der Nord-Ausrichtung (Winkel ψ, „Heading“) können wir nun mit dem korrigierten und verbesserten Treiber vornehmen [3].

Abb. 24 zeigt die Ergebnisse der Messung mit dem fischertechnik-Treiber. Die Schwankungen sind mit bis zu ±20° erheblich – vor allem aber ist der Wert selbst fehlerhaft.

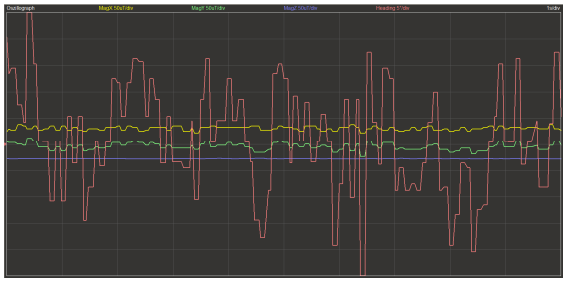


Abb. 24: Heading ψ (rot) mit fischertechnik-Treiber [Skala: 1s/50µT bzw. 1s/5° (Heading)]

Im *High Precision Mode* sieht das schon ganz anders aus. Zwar „springt“ das Ergebnis auch hier noch um rd. ±4°, aber in Abb. 25 ist zu erkennen, dass wir das mit einem Filter leicht korrigieren können.

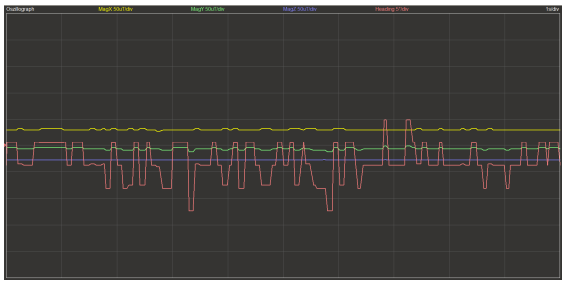


Abb. 25: Heading ψ (rot) mit verbessertem Treiber [Skala: 1s/50µT bzw. 1s/5° (Heading)]

Eine Möglichkeit zur Filterung der Ausschläge ist die Bestimmung des arithmetischen Mittels aus mehreren Messwerten (wie in Teil 1 des Beitrags beschrieben [3]). Das wirkt wie ein Tiefpass-Filter: Kurze Ausschläge werden herausgemittelt, beständige Änderungen hingegen (verzögert) durchgelassen.

Effizienter als die Speicherung der n letzten Messwerte und eine jeweils neue Mittelwertberechung ist es, den neuen Heading-Wert anteilig aus dem zuvor bestimmten und dem gerade berechneten Winkel ψ abzuleiten:

$$Head_{neu} = a \cdot Head_{alt} + (1 - a) \cdot \psi$$

mit $a \in [0,1]$.

Diese Mittelwertbestimmung (oder Filterung) wird auch als *Exponentional Moving Average* oder *Infinite Impulse Response* (IIR-Filter) bezeichnet, weil sich abrupte Änderungen des Winkels erst mit „unendlicher“ Verzögerung vollständig auf den Heading-Wert auswirken.

Die Reduktion des Rauschens „erkaufen“ wir uns dabei also mit einer verlangsamten Wirkung einer Winkeländerung über eine größere Zahl von Messungen – je nach gewählter ODR und der Stärke der Winkeländerung kann die Verzögerung einige Sekunden betragen.

Das Ergebnis der IIR-Filterung zeigt Abb. 26 für $a = 0{,}98$: Das Rauschen sinkt auf unter ±0,5°.

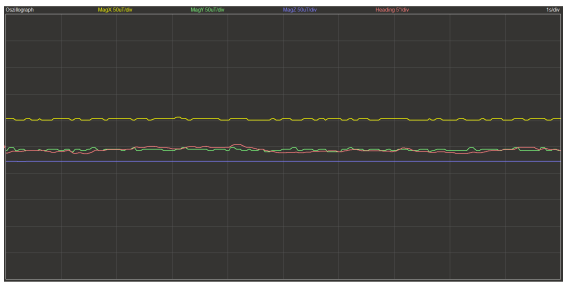


Abb. 26: Heading ψ (rot) mit verbessertem Treiber und Mittelwertbildung [Skala: 1s/50µT bzw. 1s/5° (Heading)]

Fazit

Im Kombisensor von fischertechnik steckt ein äußerst präziser IMU-Hochleistungssensor von Bosch, der über zahlreiche Funktionen verfügt.

Der von fischertechnik ausgelieferte Treiber für ROBO Pro enthält jedoch einige die Qualität der Messdaten deutlich beeinträchtigende Einstellungen und nutzt einige wichtige Sensorfunktionen unvollständig oder gar nicht: Die vom Treiber gelieferten Daten rauschen stark, sind ungenau und veraltet.

Daher habe ich die entsprechenden Fehler korrigiert, alle für anspruchsvolle Anwendungen nach meiner Einschätzung wichtigsten Funktionen im Treiber ergänzt und einige wenig sinnvolle Konfigurationen angepasst.

Die korrigierte und erweiterte Version 2.0 des Treibers (*Kombisensor-BMX055*) sowie einige Testprogramme, die ich im Beitrag verwendet habe, findet ihr im Downloadbereich der ft:pedia.

Der Sensor kennt außerdem noch Selbsttest-Funktionen und solche, die helfen, den vom Sensor verbrauchten Strom zu minimieren. Sie sind wichtig für Smartphones, in denen der Sensor ständig im Hintergrund arbeitet, aber für die in fischertechnik-Modellen vorstellbaren Einsatzzwecke sicherlich weniger relevant, daher habe ich sie im Treiber nicht ergänzt.

Quellen

[1] fischertechnik: *Kombisensor Art.-Nr. 158402 3-in-1 Orientierungssensor*. Kurzanleitung, 09.06.2017.

[2] Bosch: *BMX055. Small, versatile 9-axis sensor module*. Data Sheet. Revision 1.1, 07.11.2014.

[3] Dirk Fox: *Experimente mit dem Kombisensor*. ft:pedia 3/2021, S. 78–91.

[4] Bosch Sensortec: *bmm050.c*, v2.0.3, 12.12.2014, github.com


TOP 10
Spielzeug
2021
Bundesverband des Spielwaren-Einzelhandels e.V.
Deutscher
SPIELZEUG
PREIS 2021
DAS GOLDENE SCHAUKELPFERD
Sieger in der Kategorie
„Spiel & Technik“