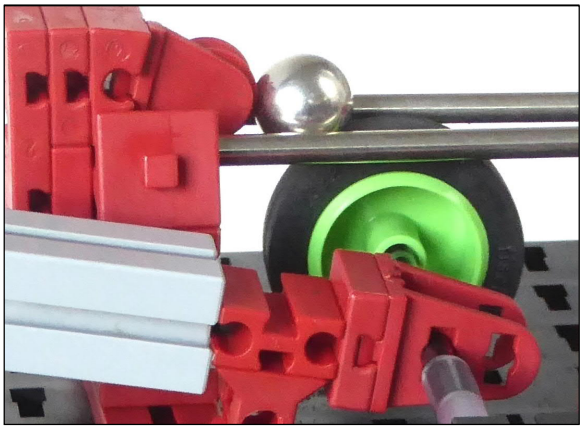
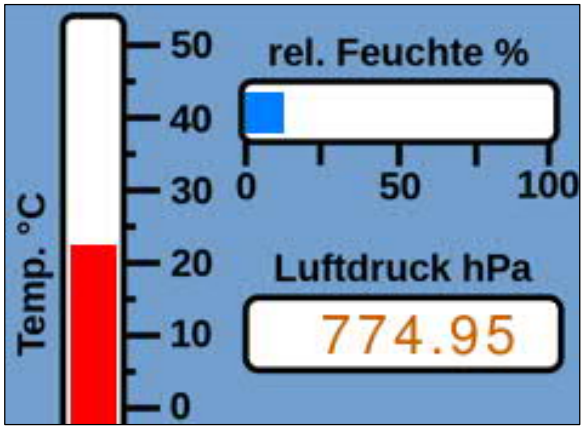
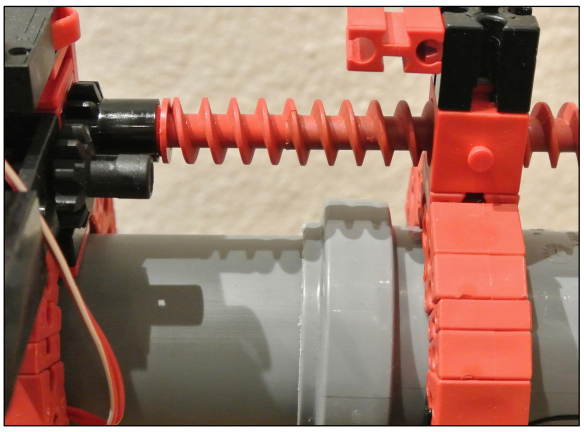
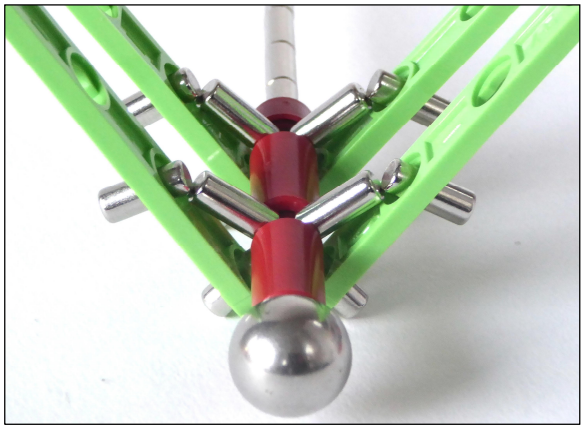
ft:pedia

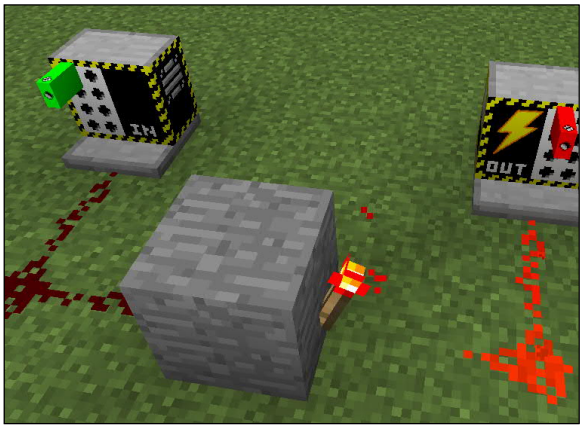
Heft 1/2019











Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

„Muss man das verstehen?“

Dirk Fox, Stefan Falk

Nein, muss man nicht - nicht jeder muss Technik verstehen. Was aber, wenn sie niemand mehr versteht? Auch kein Problem – schließlich wissen wir alle: *Creativity is King*. Wir benötigen schließlich disruptive Innovationen und keine drögen Produktvariationen.

Wie aber funktioniert eigentlich Innovation – oder, in trockenem Deutsch, „Erneuerung“? Gelegentlich wirkt sie wie ein vom Himmel gefallenes Gottesgeschenk, besonders dann, wenn sie Altbekanntes mit aufmischender Wirkung auf den Kopf zu stellen scheint. Bei genauer Betrachtung ist Innovation aber tatsächlich fast immer etwas anderes – nämlich vor allem harte Arbeit. Oder, wie *Thomas Alva Edison* (1847-1931, nach oder mit Artur Fischer der wohl erfolgreichste Erfinder aller Zeiten) 1903 gesagt haben soll: „*Genius is one per cent inspiration, ninety-nine per cent perspiration*.“ (Erfindergeist ist 1% Inspiration und 99% Transpiration.)

Und wenn man ganz genau hinschaut, erkennt man, dass Innovation meist in einer von drei Formen daherkommt: als Variation, Adaption oder Optimierung. Eine innovative *Variation* gelingt durch den Austausch einzelner Komponenten oder Gestaltungsaspekte – Material, Methode oder Mechanismus einer bestehenden Lösung. Dabei kann das Ergebnis zur Erneuerung werden, und sei es nur durch eine plötzlich intuitive Benutzeroberfläche.

Eine innovative *Adaption* nutzt Erkenntnisse, Erfahrungen oder Einsichten aus einem gänzlich anderen Zusammenhang zur neuartigen Lösung einer gegebenen Problemstellung – und beschreitet dabei einen bislang unbekannten Weg. Sie ist im Kern also vor allem eine Transferleistung.

Und schließlich ist da die oft gering geschätzte *Optimierung*, die sich meist in Gestalt einer kontinuierlichen, schrittweisen Verbesserung anschleicht, aber endlich in eine neue, innovative Qualität umschlagen kann: Die Summe vieler kleiner Besserungen mutiert dabei zur innovativen Leistung.

Jede dieser Innovationsarten erfordert zwar etwas Abstand zum Detail, um „das große Ganze“ zu sehen, aber eben auch eine von tiefer technischer Kenntnis einerseits (wie ist etwas realisierbar?) und umfangreichem, idealerweise erfahrungsgetränktem Breitenwissen andererseits getragene Kompetenz. In keinem Fall entspringt Innovation allein einem „kreativen Talent“ oder plötzlicher Eingebung. Wer sich mit dieser Hoffnung der Gewinnung tieferer und differenzierter (Technik-) Kenntnisse verweigert, die Kenntnisfreiheit gar zur kreativen Tugend erhebt, sollte sich über sein Scheitern nicht wundern. Schließlich wird Wasser auch nicht durch Rotfärben zu Wein.

Nein, nicht jeder muss Technik verstehen. Aber wer das nicht will, sollte auch nicht von Innovationen träumen. Für alle anderen gibt es fischertechnik – spielerischer und beglückender kann man sich wohl kaum auf die Innovationen von morgen vorbereiten.

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?
Nordconvention	27.04.2019	Wedemark
Clubdag	11.05.2019	Twello (NL)
FanClubTag	13.07.2019	Waldachtal
Südconvention	21.09.2019	Sinsheim

Impressum

http://www.ftcommunity.de/ftpedia

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

Autoren: Christian Bergschneider, Stefan Falk, Dirk Fox, Stefan Fuss, Martin Giger, Peter Habermehl, Till Harbaum, Wilhelm Lichtenberg, Rolf Meingast, Rüdiger Riedel, Leon Schnieber

Wissenschaft

Minimodelle schreiben Geschichte

Christian Bergschneider, Stefan Fuss

Die Höhlenmalerei im südfranzösischen Lasceaux enthält einige der ältesten bildlichen Darstellungen der Menschheitsgeschichte. Aufgrund der jüngsten Auswertungen dieser Bilder müssen wahrscheinlich wesentliche Kapitel der Geschichte überarbeitet und teilweise neu geschrieben werden.

Wissenschaftler aus aller Welt nutzen regelmäßig *fischertechnik* für ihre Forschung. So erregte 2018 besonders der Beitrag zur schwarzen Materieforschung aus [1] ein hohes Interesse in der wissenschaftlichen Community. In diesem Jahr zeigen die Veröffentlichungen der Ergebnisse zweier archäologischer Forscherteams, dass *fischertechnik* ein weitaus größerer Beitrag zur Menschheitsgeschichte eingeräumt werden muss.

Die Höhle von Lasceaux

Avril le Premier und sein Team erforschen seit langem die Höhlenmalerei im französischen Lasceaux. In den Bildern haben prähistorische Künstler im Zeitraum von 36.000 bis ca. 15.000 v. Chr. ihre Umwelt dargestellt. Viele Tier- und Jagdszenen geben uns eine Vorstellung der Lebensbedingungen am Ende der Steinzeit.

Abb. 1: Tierszene aus der Höhle von Lascaux [2]

Die Atemluft der Besucher führte zur Bildung von Schimmel. Die Neubildung von schwarzem Schimmel konnten die Archäologen in den letzten Jahren sehr gut verhindern, allerdings war es nicht möglich, stark betroffene Flächen wieder zu reinigen [3].

Mit einer neuen Methode gelang es dem Team um Avril nun den schwarzen Schimmel zu beseitigen, und es erlaubt uns Einblicke in bislang verlorene Details.

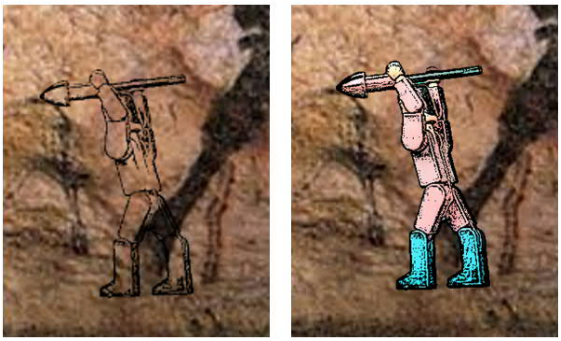


Abb. 2: Links das neu entdeckte Detail, rechts zur besseren Erkennbarkeit farbig markiert

Die beiden Bilder in Abb. 2 zeigen einen erstaunlichen Ausschnitt aus der Tierszene nach der Reinigung. Ganz deutlich zu sehen ist ein fischertechnik-Männchen auf der Jagd.

Waldachtal

Ein deutsches Archäologenteam forscht in Waldachtal an der europäischen Industriegeschichte. Bislang datierte das Team die Geburtsstunde von *fischertechnik* in die

Vorweihnachtszeit des Jahres 1964. In diesem Zeitraum lieferte *Artur Fischer* den ersten fischertechnik-Baukasten an einige seiner Kunden aus. Er galt deshalb bislang als Erfinder der *fischertechnik*-Bausteine [4].

Bei Grabungen auf dem Firmengelände konnte das deutsche Team nun ein Gegenstück zur Wandmalerei in Lascaux und weitere Modelle in erstaunlich guten Zustand bergen.

Abb. 3: Fundstück aus Waldachtal

Die frappierende Ähnlichkeit des Fundstücks aus Abb. 3 lässt die deutschen Forscher die Fundstücke auf den Zeitraum am Ende der Steinzeit datieren – auf ca. 25.000 v. Chr. Sie gehen fest davon aus, dass die noch anstehende Untersuchung über die Radiokarbonmethode ihre Datierung bestätigen wird.

Noch ist offen, ob *fischertechnik* die neuesten Erkenntnisse aufgreifen und endlich einen Baukasten mit Minimodellen zur Evolution und Menschheitsgeschichte zum Weihnachtsgeschäft auf den Markt bringen wird (Abb. 4).

Quellen

[1] H.A.R.R.Y.: *Letztes Physikrätsel beinahe gelöst – Dank fischertechik*, Forumsbeitrag ftcommunity.de

[2] Prof. Saxx: *Lascaux painting.jpg*, wikimedia.org

[3] Wikipedia. *Die Höhle von Lasceaux*

[4] Wikipedia: *fischertechnik*

Abb. 4: Designstudie für die Verpackung des neuen Baukastens

Modell

Spielereien mit Magneten

Rüdiger Riedel, Stefan Falk

Durch die Dynamic-Kästen haben wir Stahlkugeln, seit der „Technikgeschichte mit fischertechnik“ [1] haben wir kleine Stabmagnete. Was lässt sich damit noch anfangen?

Die Neodym-Magnete haben 4 mm Durchmesser und 10 mm Länge (diese Magnete gibt es im Internet z. B. bei fischerfriendsman.de; sie sind kein fischertechnik-Produkt).

Kleine Autos

Naheliegend ist die Verwendung der Magnete als Achsen und der Kugeln als Räder von Fahrzeugen. Diese Fahrzeuge rollen frappierend leicht, weil die Kugel kaum an den Magneten reibt – unbedingt ausprobieren!

Abb. 1: Der kleine Wagen

Abb. 2: Einzelteile des kleinen Wagens

Vier Statik-Streben 15 I (36914) ergeben gerade die richtige Breite des Chassis.

Mit 2 S-Streben 30 I und 4 S-Streben 15 I sowie einem Klemmstift (107356) bauen wir ein Rennauto.

Abb. 3: Rennwagen

Die Räder nicht vergessen!

Fetzige Autos

Die folgenden Abbildungen geben weitere Anregungen für Fahrwerke und Fahrzeuge:

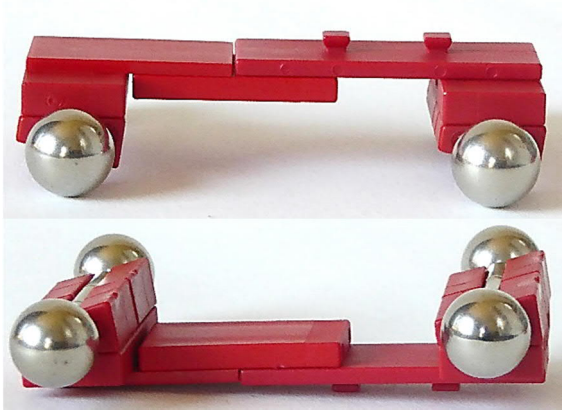


Abb. 4: Flaches Fahrwerk

Wenn wir für etwas Bodenfreiheit sorgen, können die Kugelräder auch tatsächlich bei Unebenheiten einfedern.

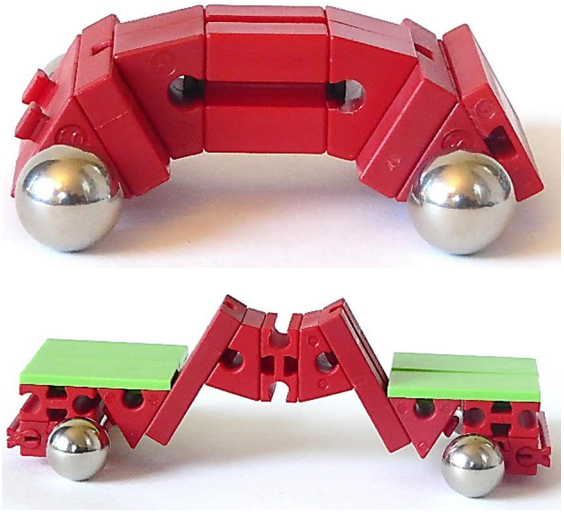


Abb. 5: Autos

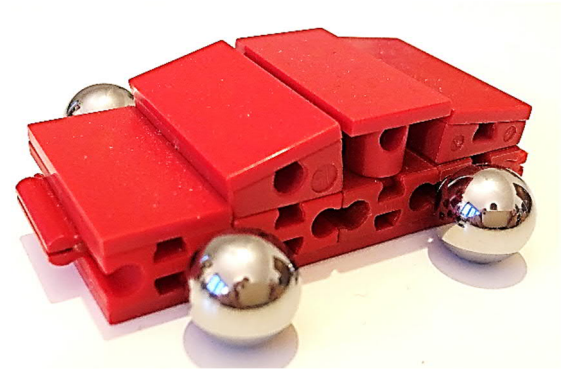


Abb. 6: Der Kugelräder-Audi aus dem Film "I, Robot"

Abb. 7: Magnet-o-Cart 1

Abb. 8: Magnet-o-Cart 2

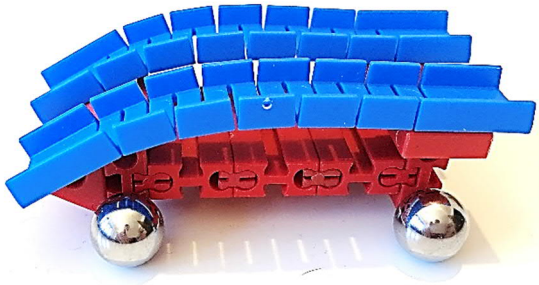


Abb. 9: Überlegt euch einen Namen!

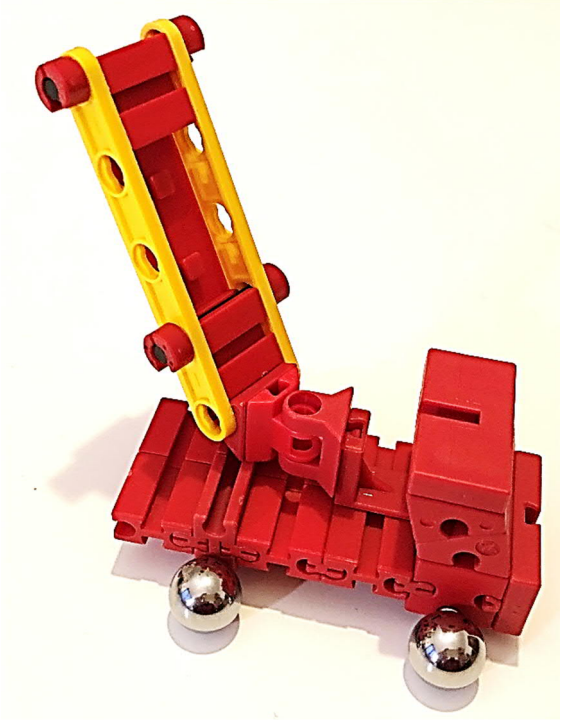


Abb. 10: LKW mit drehbarer Feuerwehrleiter

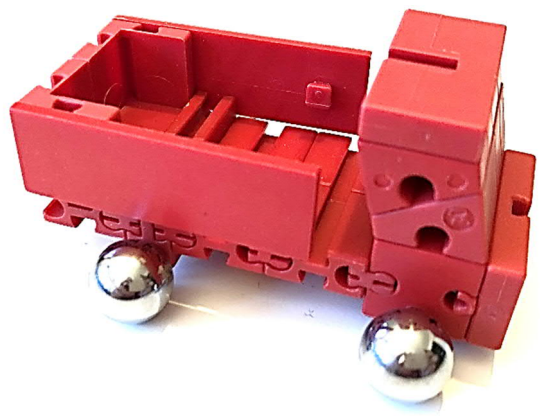


Abb. 11: Lastwagen mit Pritsche

Die Libelle

Eine Besonderheit der Libelle ist die Klemmbefestigung ihrer Flügel mit Magneten.

Abb. 4: Die Libelle

Der Kopf, also die Kugel, wird durch einen Magneten an einer Achse 30 gehalten.

Abb. 5: Haltemechanik der Flügel

Der Körper wird von zwei Hülsen 15 (31983) gebildet.

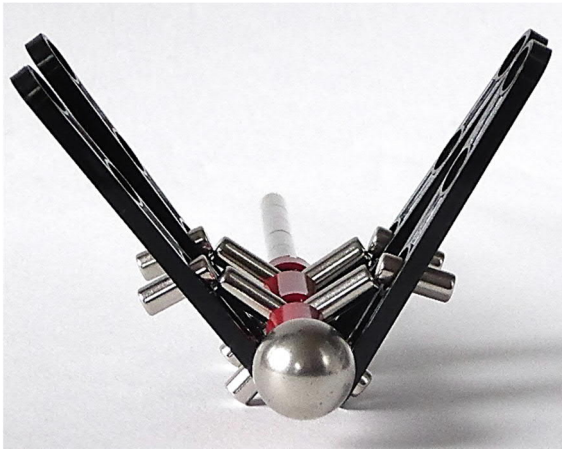


Abb. 6: Libelle von vorne

Mit abwechselnder Polung setzen wir die zweimal vier Stützmagnete an die Achse, wie in Abb. 8 gezeigt wird.

Abb. 7: Das Grundgerüst, Schritt 1

In jeden der nach unten zeigenden Magnete wird eine S-Strebe 45 I eingehängt und durch Einschieben eines weiteren Magneten durch das zweite Loch fixiert.

Abb. 8: Das Grundgerüst, Schritt 2

Noch das Hinterteil ansetzen aus sechs Magneten und einer Klemmbuchse, fertig ist die Libelle!

Abb. 9: Dragon-fly

Die englische Bezeichnung „dragon-fly“ für Libelle klingt so schön dramatisch, die muss hier unbedingt verwendet werden.

Der fischertechnik-Mann ist der Stärkste

Enorme Gewichte kann der heben, und richtig standfest ist er dabei auch.

Abb. 10: Sooo stark bin ich!

Für das kleine Podest verwenden wir eine V-Grundplatte und schieben in eins der Seitenlöcher einen Magneten hinein. In die Stiefel kommt ebenfalls je ein Magnet, beide mit unterschiedlicher Orientierung.

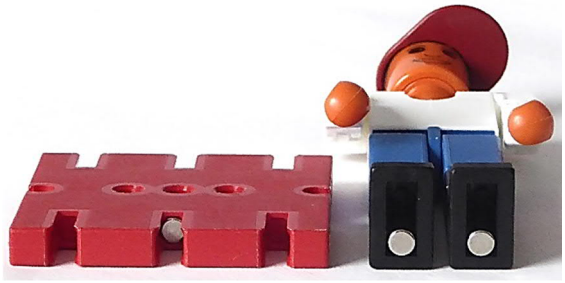


Abb. 11: Magnete für die Standfestigkeit

Den fischertechnik-Mann auf die Platte stellen, evtl. um 180° drehen, und dann steht er sicher und aufrecht.

Abb. 12: Seht her, wie ich die Hanteln schwinge!

Quellen

[1] Dirk Fox, Thomas Püttmann: *Technikgeschichte mit fischertechnik.* dpunkt-Verlag, 2015, S. 218.

Tipps & Tricks

fischertechnik refurbishment (Teil 1)

Wilhelm Lichtenberg

Wer kennt das nicht: Viele alte Bausteine (z. B. BS30, BS15, alle Farben) sind nur noch eingeschränkt zu gebrauchen, weil der Zapfen zwar noch vorhanden ist, aber die Steine meistens nur noch sehr locker miteinander verbunden werden können. Diese wandern dann meistens in die große Gemischtkiste, wo sie auf ihre nächste Bestimmung wie z. B. das „modding“ warten oder sogar entsorgt werden. Aber das muss nicht sein, weil es eine sehr einfache Methode gibt die Steine wieder „fit“ zu machen.

Hintergrund

Anlass der Idee war meine Bereitschaft, die ca. 20 etwas verwahrlosten ut-1 Kästen aus der Grundschule meiner Frau in einen ordentlichen Zustand zu versetzen, um sie im Sachunterricht nutzen zu können. Mein ursprünglicher Gedanke war, beschädigte Steine durch funktionstüchtige aus meiner Sammlung zu ersetzen. Bei der ersten Sichtung stellte ich fest, dass fast alle grauen Steine (400 Stück BS30, 200 Stück BS15 etc.) nicht mehr richtig zu gebrauchen waren. Nur wenige Steine waren wirklich defekt, was mal wieder die Qualität von fischertechnik unter Beweis stellt.

Dabei fiel mir als erstes ein Baustein in die Hände, bei dem der Zapfen ca. 1 mm hervorstand. Für mich lag nahe, diesen mittels Schraubstocks wieder in die richtige Position zu drücken. Als Zufallsergebnis kam dabei heraus, dass sich der Baustein danach wieder wie neu anfühlte, weil auch der Zapfen ein wenig deformiert wurde. Somit war die Idee zu diesem Artikel geboren.

Umsetzung

Ein kleiner Schraubstock war schnell zur Hand. Zur Schonung der Bauteile wurden aus einem Aluwinkelprofil ein paar Schonbacken zurecht gesägt, um nicht die Riffelung meiner Schraubstockbacken in den Bausteinen zu verewigen.

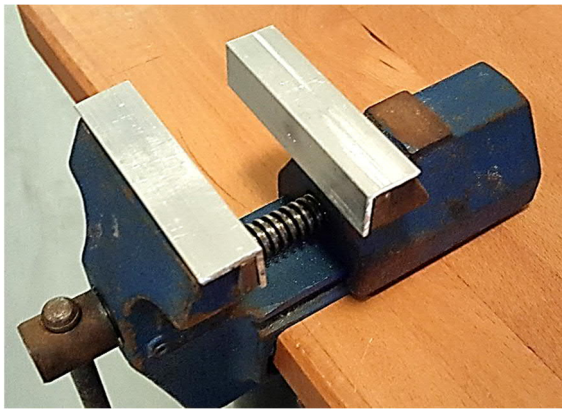


Abb. 1: Schraubstock mit Schonbacken

Bei der Art des Schraubstockes sollte unbedingt auf eine Parallelführung der Backen geachtet werden, damit die Bausteine nicht deformiert werden. Kleine Feinmechaniker-Schraubstöcke gibt es schon für unter 20 € zu kaufen.

Um ein Gefühl für die richtige Stauchung der defekten Bausteine zu bekommen, sollte man vorab das Verfahren an einigen Steinen ausprobieren. Eines sollte natürlich nicht verschwiegen werden: Man kann mit zu viel Schwung oder zu wenig Gefühl die Steine endgültig zerstören. Lieber mehrmals den Zapfen leicht stauchen als einmal zu viel.

Nach meiner Einschätzung sind die Bauteile nicht mehr funktionsfähig, weil eine Deformation von Feder und Nut durch unzähliges Montieren und Demontieren durch eine hohe Biegebeanspruchung der Zapfen vorliegt. Die Zapfenkontur hat werksseitig nach der Herstellung nur wenige Hundertstel Übermaß, sowie die Nuten entsprechend Untermaß, um eine einwandfreie Funktion sicherzustellen. Leider gibt es manche Bauteile bei fischertechnik, die bereits als Neuteil schon recht locker sitzen, aber das ist ein anderes Thema.

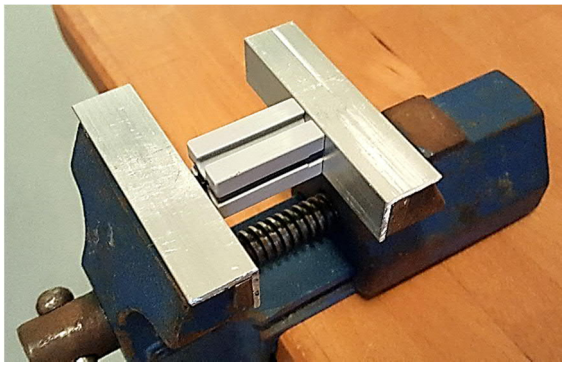


Abb. 2: BS30 im Schraubstock eingespannt

Sind die ersten Versuche erfolgreich absolviert, kann man sich an eine größere Stückzahl heranmachen. Aber auch nur einzelne Bausteine, die beim Bauen von Modellen auffällig sind, lassen sich geschwind wieder reparieren.

Ich selbst habe viel Freude daran, wenn die „alten“ Modelle aus der grauen Ära wieder einwandfrei funktionieren und bei den Kids Begeisterung hervorrufen. Mit „schlabbrigen“ Modellen, die bereits beim Spielen auseinanderfallen, ist Frust bereits vorprogrammiert.

Qualitätssicherung

Um jedem Stein wieder ein neues fischertechnik-Leben zu geben, sollte man diesen vorher an einem „guten“ Stein getestet haben. Hierzu wird der „refurbished“ Stein in die Nut und Feder des jeweils anderen guten Steins geschoben, um die Funktion zu testen. Ich denke, dass jeder fischertechniker ein Gefühl hat, wann ein Baustein die optimale Füge- bzw. Haltekraft besitzt.

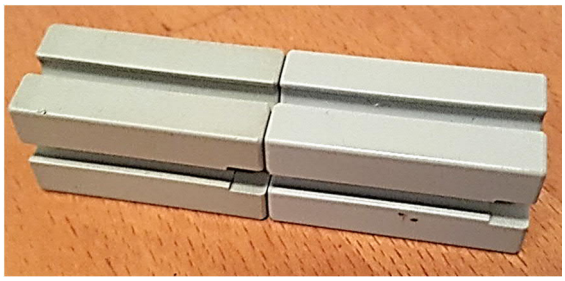


Abb. 3: Funktionstest

Bei genauer Betrachtung der Steine ist mir noch ein weiteres Phänomen aufgefallen: Die stirnseitige Nut des Bausteins ist oftmals nach übermäßigem Nutzen nach außen gewölbt. Meine Theorie dazu ist, dass viele unbedarfte Nutzer von fischertechnik nicht den Zusammenbau verstehen, wenn man es ihnen nicht von Anfang an richtig zeigt. Ich vermute, dass unbedarfte Nutzer die Handhabung der fischertechnik-Bauteile oftmals mit Lego verwechseln und versuchen, die Steine ineinander zu stecken statt sie ineinander zu schieben. Um dies zu vermeiden ist es wichtig, den Kindern die richtige Handhabung von fischertechnik-Bauteilen von Anfang an zu vermitteln. Um auch hier eine Lösung anzubieten, lassen sich die Deformationen auf der stirnseitigen Nut ebenfalls mittels Schraubstocks beseitigen.

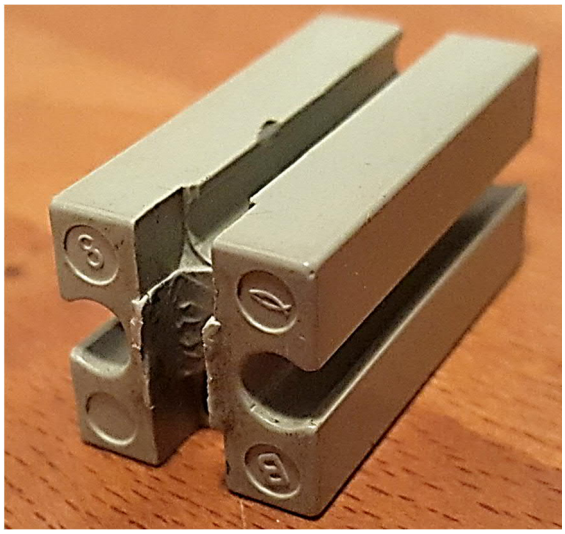


Abb. 4: Deformation an der stirnseitigen Nut

Hierzu muss man den Stein im Schraubstock schon etwas genauer positionieren, wobei der schwarze Zapfen nur auf den Schonbacken aufliegt. Wird der Baustein jeweils vier Mal um 90° um die Längsachse gedreht und gestaucht, erhält man am Ende einen funktionstüchtigen Baustein.

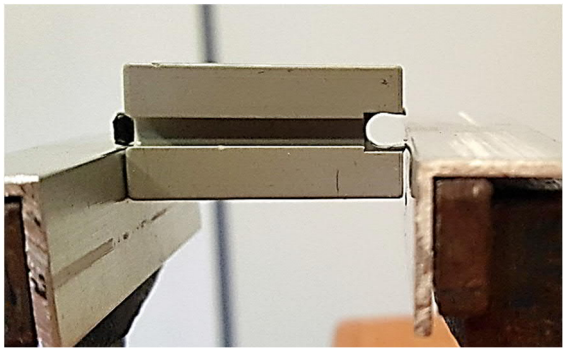


Abb. 5: Einspannung bei deformierter Nut

Auch hier ist Vorsicht geboten, weil es Steine gibt, die sehr alt und dadurch spröde geworden sind.

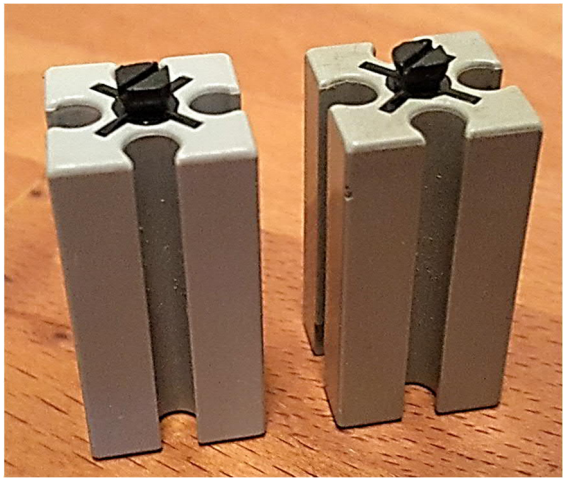


Abb. 6: Unterscheidung nur Anhand von Schönheit ist schwierig für die Funktionsfähigkeit

Die Sprödigkeit ist nicht unbedingt eine Frage des Alters, sondern der Art, wie sie über Jahrzehnte gelagert wurden. Das zusammengebaute Modell, das jahrelang auf der Fensterbank stand, ist meistens schlimmer dran als über 30 Jahre verpackte Einzelteile im Keller. Auf die genauen Zusammenhänge wird hier nicht weiter eingegangen. Spröde Bausteine erkennt man leider nicht am Äußeren, sondern nur durch ausprobieren. Nur durch Unterscheidung von „schöner Stein“ erkennt man die Funktion leider noch nicht. Es empfiehlt sich, bei der Stauchung der Steine eine entsprechende Schutzbrille zu tragen.

Die oben gezeigte Methode lässt sich natürlich auch auf andere Bausteine übertragen, wie z. B. die alten Gelenksteine (Abb. 7).

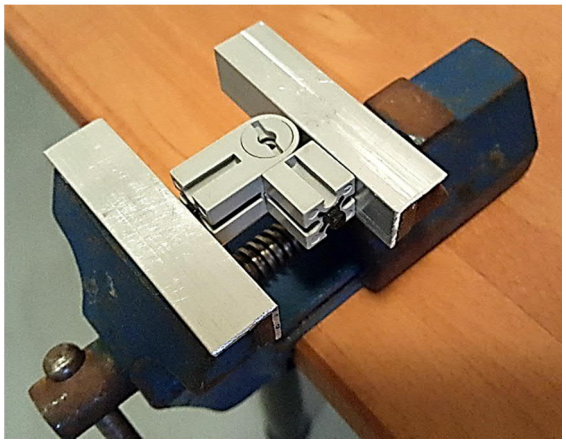


Abb. 7: Die Methode ist auf andere Bausteine übertragbar

Fazit

Wie viele von uns wissen, ist es kaum mehr möglich, graue Steine in einem sehr guten Zustand zu erwerben. Mit etwas Glück findet man welche in einem ebay-Konvolut oder einer wohlbehüteten Sammlung. Sollte jemand seine alten defekten Steine mittels der „Refurbishment-Methode“ veräußern wollen, sollte dies der Fairness halber gekennzeichnet sein.

Derzeit liegen mir noch keine Langzeitergebnisse vor, wie lange der Zustand anhält. Erste Versuche haben aber gezeigt, dass sich nach unzähligem richtigem Aufschieben und Auseinanderbauen der Steine keine Veränderungen zeigen.

Mit meiner Übung der Aufarbeitung von 20 ut-1 Kästen nahm das Ganze natürlich etliche Stunden in Anspruch. Der Aufwand war es mir aber wert und ich hoffe, einen positiven Beitrag zur Technikbegeisterung für die Schüler meiner Frau geleistet zu haben.

Tipps & Tricks

Die Welt der ft-Winkelbausteine (Teil 3): UFO-Ringe

Rüdiger Riedel

Wie ich in der ft:pedia 1/2017 beschrieben habe, lassen sich mit den fischertechnik-Winkelsteinen auch dreidimensionale Gebilde, Schrauben und Wellenringe bauen. Das will ich hier ergänzen um Ringe mit scharfem Rand, die UFO-Ringe.

Es ist nur ein kleiner Dreh

Setzen wir die Winkelsteine immer abwechselnd um 90° gedreht und dann zurückgedreht aufeinander, ergibt sich ein neuer effektiver Winkel, der zu Kreisringen (genauer: Vielecken) mit höherer Bausteinzahl führt. Bisher galt für Winkelsteinringe:

360° / 7,5° = 48 Steine WS7,5

360° / 15° = 24 Steine WS15

360° / 30° = 12 Steine WS30

360° / 60° = sechs Steine WS60

Für UFO 1, das nur aus Winkelsteinen 7,5° zusammengesetzt ist, benötigen wir jetzt 68 Steine.

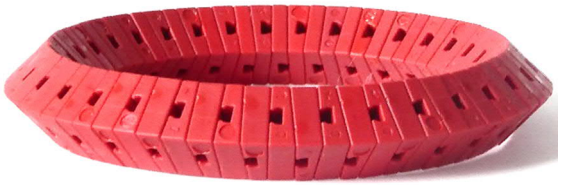


Abb. 1: UFO 1 aus 68 WS7,5

Die breitere Seite des Winkelsteins liegt abwechselnd an der oberen und der unteren Schräge des Ringes. Beim UFO 2 in Abb. 4 sieht man es deutlicher.

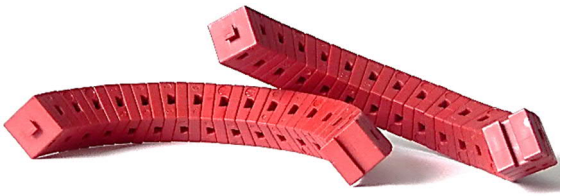


Abb. 2: Aufbau von UFO 1

Erste Erkenntnis: Da wir immer einen Stein nach oben und den nächsten nach unten drehen, muss die Gesamtzahl an Steinen eines Ringes gerade sein.

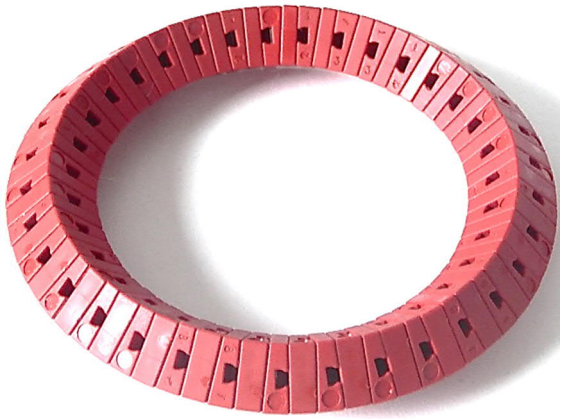


Abb. 3: UFO 1 von oben

Der Ring UFO 2 aus 34 Winkelsteinen 15° gelingt genauso gut.

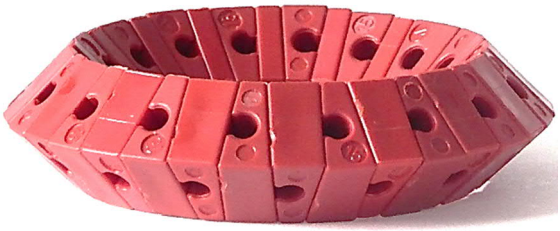


Abb. 4: UFO 2 aus 34 WS15

Die äußere Kontur ist ebenfalls scharf und gleichmäßig.

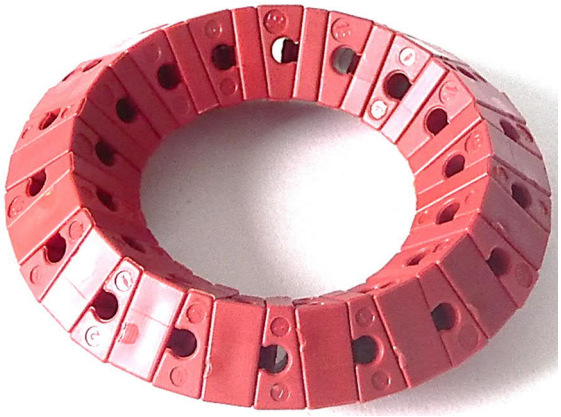


Abb. 5: UFO 2 von schräg oben

Bis jetzt haben wir die Winkelsteine WS7,5 (7,5°) und WS15 (15°) verwendet. Als nächster käme der WS30 dran, aber wir können auch die Steine kombinieren und dadurch haben wir auch die Winkel 22,5°, 30°, 37,5°, 45°, 52,5° und 60°.

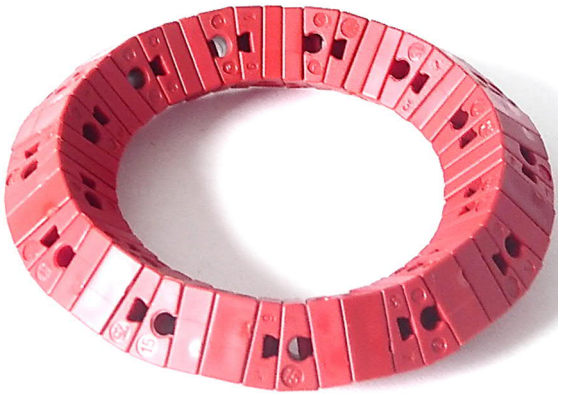


Abb. 6: 24 WS(15+7,5) entsprechend 24 WS22,5

Wir setzen also die WS22,5 aus je einem WS7,5 und einem WS15 zusammen und fügen 24 dieser Kombisteine zu einem Ring zusammen (Abb. 6). Dieser Ring ist etwas sperrig. Ohne Kraftaufwand lassen sich 23 der Kombisteine einfügen, wodurch der Ring aber nicht geschlossen wird.

Nach der oben gewonnenen Erkenntnis und dem Winkelverhalten müssen wir einen vierundzwanzigsten einfügen. Mit etwas Nachdruck gelingt es, die fischertechnik-Bausteine machen das problemlos mit!

Noch widerspenstiger zeigt sich der Ring aus Winkelsteinen 30° (Abb. 7)

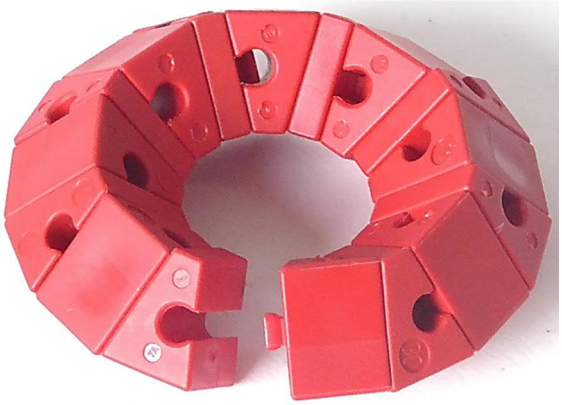


Abb. 7: 16 WS30

Mit einer Kombination aus WS15 und WS7,5 kann er auf etwas unschöne Weise geschlossen werden (Abb. 8).

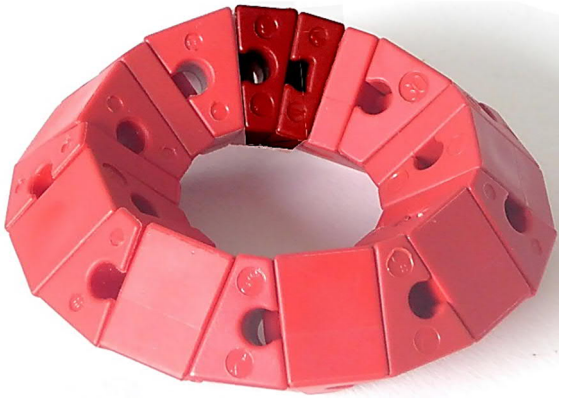


Abb. 8: 16 WS30 plus WS15 plus WS7,5, Variante 1

Ein etwas besseres Aussehen erhält man durch diagonales Anbringen der beiden Zusatzsteine (Abb. 9).

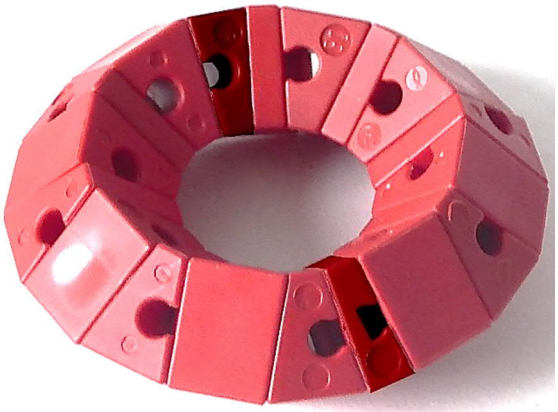


Abb. 9: 16 WS30 plus WS15 plus WS7,5, Variante 2

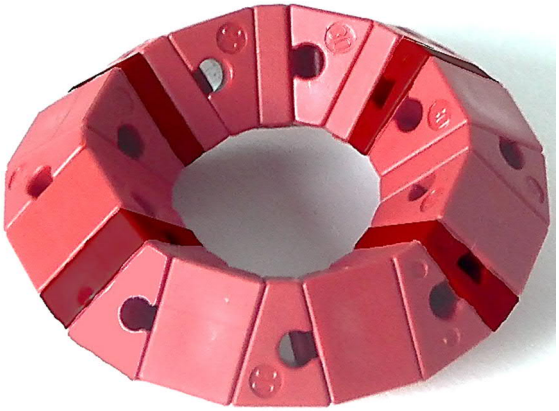


Abb. 10: 16 WS30 plus 4 WS7,5

Fügen wir statt eines WS15 und eines WS7,5 vier Stück WS7,5 ein, ergibt sich eine etwas veränderte Ringstruktur (Abb. 10). Der effektive Einzelwinkel ist jetzt 30° + 7,5°/4 = 31,875° oder ungefähr 31,9°.

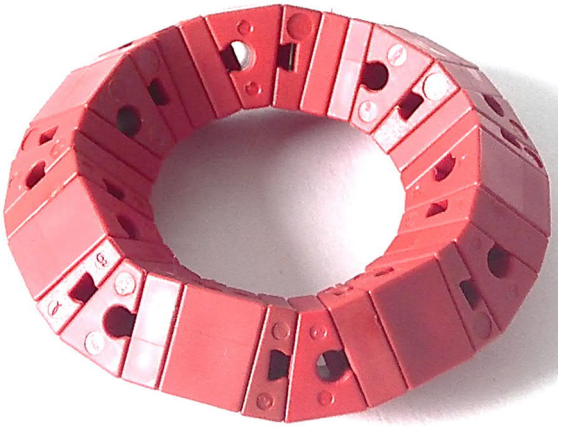


Abb. 11: 14 WS(30+7,5) entsprechend 14 WS37,5

Für den Ring aus Kombisteinen WS37,5 nehmen wir 14-mal WS30 + WS7,5 (Abb. 11).

Zwölf der aus je einem WS30 und einem WS15 zusammengesetzten WS45 lassen sich problemlos zu einem Ring zusammenfügen (Abb. 12).

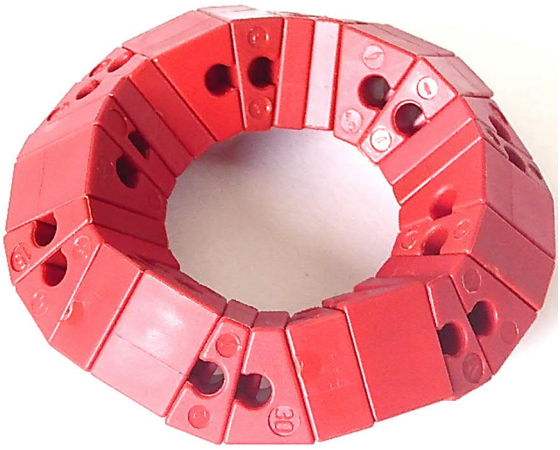


Abb. 12: 12 WS(30+15) entsprechend 12 WS45

Auf entsprechende Weise erhalten wir einen Ring aus zehn zusammengesetzten WS52,5 (Abb. 13). Hier machen wir uns negative Winkel zunutze wie in [1] auf Seite 20 beschrieben, also WS60 – WS7,5.

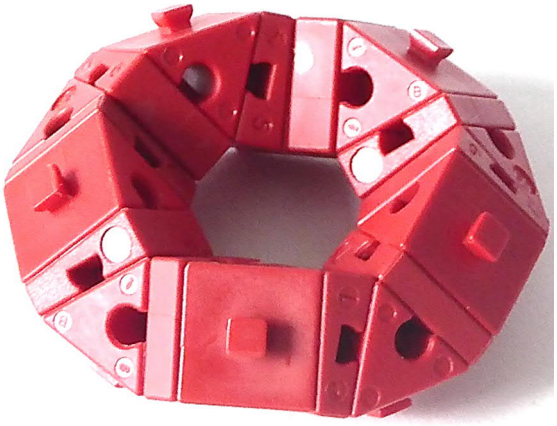


Abb. 13: Zehn WS(60-7,5) entsprechend zehn WS52,5

Der Ring aus acht WS60 schließt sich nicht (Abb. 14), wir müssen also Ergänzungsbausteine einfügen. Mit zusätzlichen vier WS7,5 lässt sich der Ring vervollständigen.

Bezieht man die Zusatzsteine auf die Hauptbausteine, dann wären das acht WS63,75.

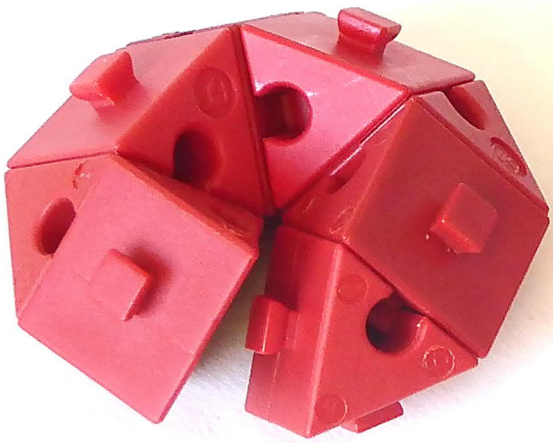


Abb. 14: Acht WS60

Für ein besseres Aussehen habe ich vier Winkelsteine 60° 3N (31918) benutzt, so dass keine Zapfen überstehen (Abb. 15).

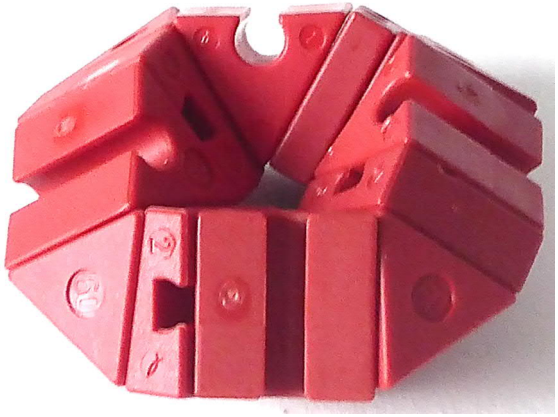


Abb. 15: Acht WS63,75

Die bisher vorgestellten Ringe habe ich in Abbildung 16 zu einem Turm zusammengestellt. Den Fuß bilden zwei Speichenräder 90x15 (36916) und dazwischen eingeklemmt UFO 1. Dann folgen nach oben die Ringe aus WS22,5, WS15, WS37,5, WS30 Variante 1, WS45, WS30 Variante 2. WS52,5 und WS63,75.

Abb. 16: Turm

Erweiterung

Es gibt noch mehr Möglichkeiten für UFO-Ringe, indem die „Original"-Winkelsteine 15°, 30° und 60° ersetzt werden durch Winkelsteinkombinationen:

15° → 2 · 7,5°

30° → 2 · 15°
1 · 15° + 2 · 7,5°
4 · 7,5°

60° → 2 · 30°
1 · 30° + 2 · 15°
1 · 30° + 1 · 15° + 2 · 7,5°
1 · 30° + 4 · 7,5°
4 · 15°
3 · 15° + 2 · 7,5°
2 · 15° + 4 · 7,5°
1 · 15° + 6 · 7,5°
8 · 7,5°

Abb. 17: 34 WS(7,5+7,5) entsprechend 34 WS15

Der Ring 34 WS15 in Abb. 17 hat etwa die gleichen Abmessungen wie UFO 1 in Abb. 3: $D_a \approx 127$ mm, $D_i \approx 85$ mm.

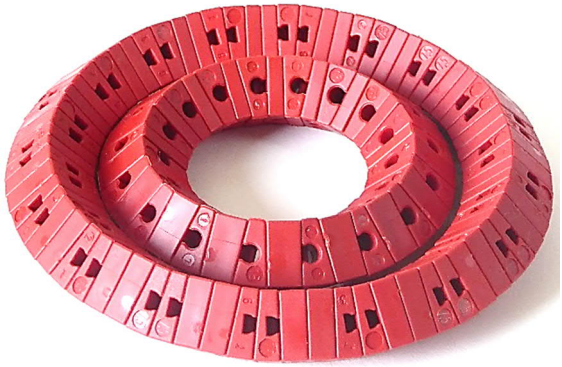


Abb. 18: 34 WS15 in 34 WS(7,5+7,5)

Abb. 19: 24 WS(7,5+7,5+7,5) entsprechend 24 WS22,5

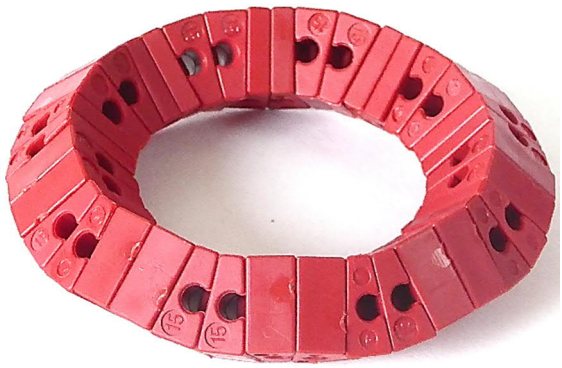


Abb. 20: 18 WS(15+15) entsprechend 18 WS30

Abb. 21: 18 WS(7,5+15+7,5) entsprechend 18 WS30

Einige der bisher vorgestellten Ringe lassen sich recht gut zu einem Kegel stapeln:

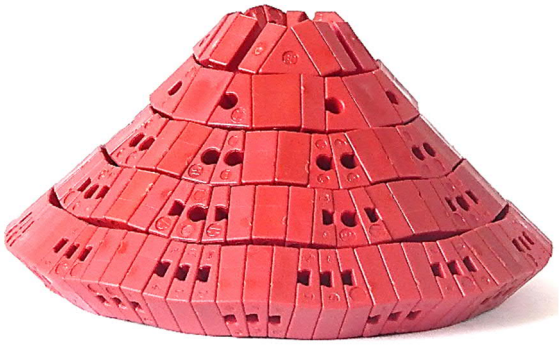


Abb. 22: Der Kegel

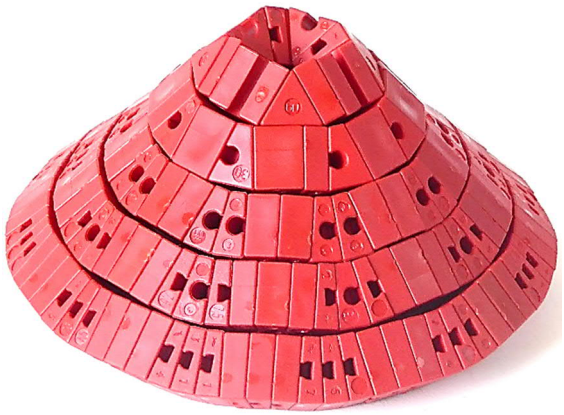


Abb. 23: Der Kegel

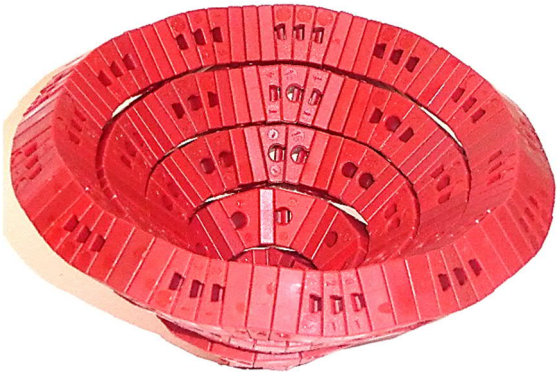


Abb. 24: Das Innere des Kegels

Verwenden wir wieder negative Winkel wie oben beschrieben, bekommen wir eine besondere Variante des Ringes aus 18 WS30. Wir bilden 18-mal die Kombination WS60 – WS30 und bekommen den zackigen Ring nach Abb. 25:

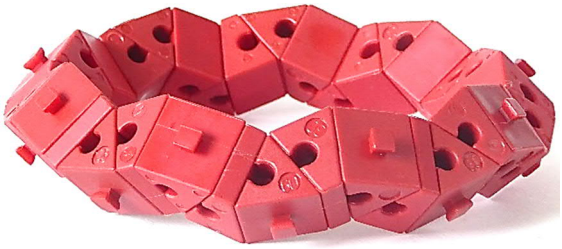


Abb. 25: 18 WS(60-30) entsprechend 18 WS30

In den Abb. 26 und 27 seht ihr eine Variante des Ringes von Abb. 13, jetzt zusammengesetzt aus zehn WS(15+30+7,5) entsprechend zehn WS52,5.

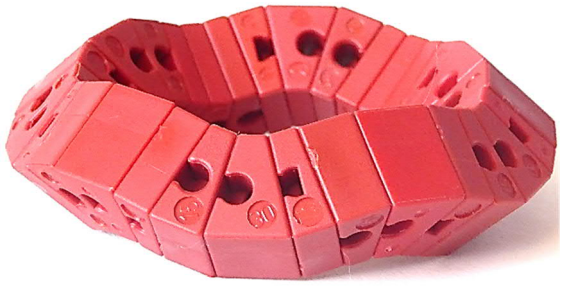


Abb. 26: Zehn WS52,5 Variante

Wie bei allen zusammengesetzten Winkelbausteinen verändert sich die Seitenkontur girlandenförmig.

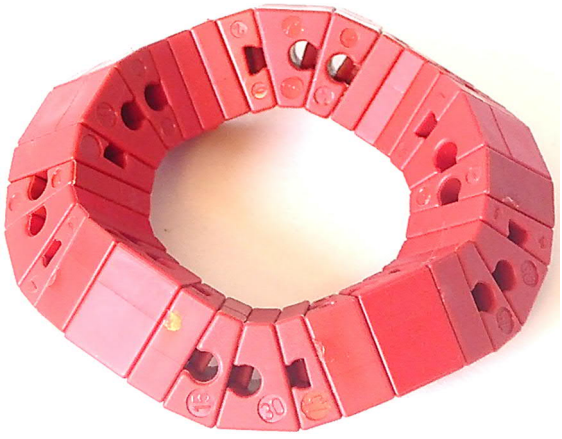


Abb. 27: Variante zehn WS52,5 von oben

Zwei weitere Beispiele:

Die wirksamen Winkel der Bausteingruppen betragen 67,5° in Abb. 28 und 90° in Abb. 29.

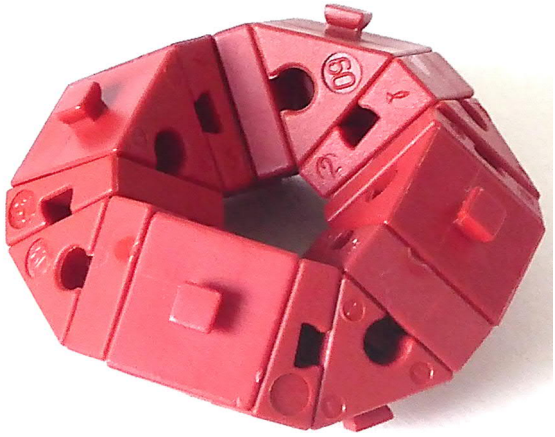


Abb. 28: Acht WS(60+7,5) entsprechend acht WS67,5

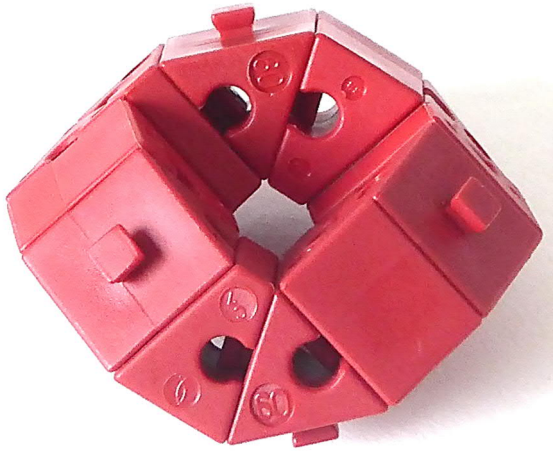


Abb. 29: Sechs WS(60+30) entsprechend sechs WS90

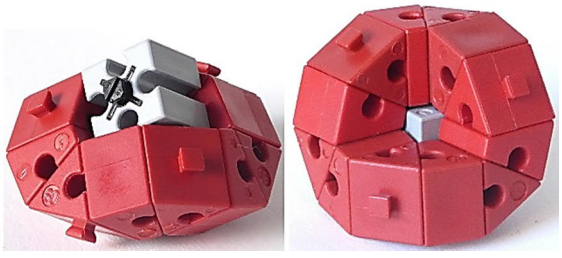


Abb. 30: Ein BS15 passt in die Mulde

Bis auf den kleinen Knick in der Mulde des Ringes sechs WS90, verursacht durch die WS30, sitzt der BS15 genau darin. Rechts in Abb. 30 sieht man ein kleines Stück davon aus der Rückseite des Ringes herausschauen.

In Abb. 29 haben wir eine Bausteinkombination mit einem Gesamtwinkel von 90° eingesetzt, aber da war doch noch was…?

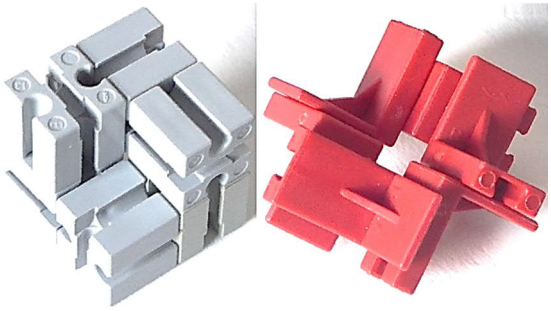


Abb. 31: Zweimal sechs WS90

Der „Winkelstein 90°“ wurde in [2] erläutert und so können wir den Ring sechs WS90 auch aus je sechs Bausteinen 15 oder Winkelsteinen 10x15x15 bauen.

Anwendung

Versehen wir UFO 2 (Abb. 4 und 5) mit Speichen und Achse, erhalten wir ein Rad für ein einachsiges Modell mit O-Beinen. Zur Zentrierung und Fixierung werden auf beiden Seiten der Räder je vier Federnocken in die BS7,5 eingeschoben (siehe Abb. 34).

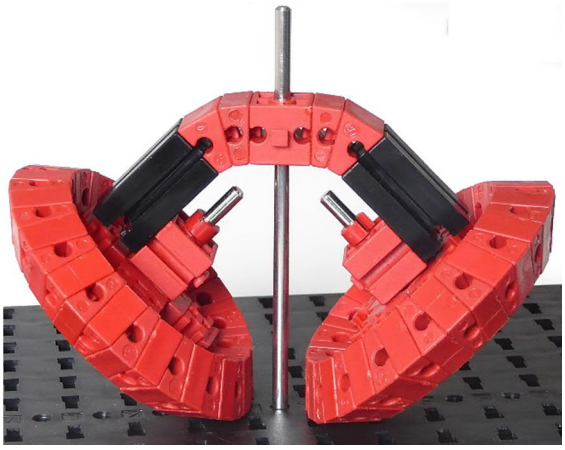


Abb. 32: Rundherum

Das Modell in Abb. 32 bleibt auch ohne die Stange aufrecht stehen!

Will man die Räder antreiben, werden Rastaufnahmeachsen 22,5 (130593) als Basis der Achse genommen, mit der Fixierung aus je einer Bauplatte 15·30·3,75 1N (32330) und zwei V-Achsen 20 (31690) mit Klemmbuchsen 5 (37679) (Abb. 34).

Mit weiteren Bauteilen und einer Fernsteuerung wird daraus das M-Zeug nach Abb. 33. Und es fährt hierhin, dorthin, rückwärts und im Kreis, zur Freude des Erbauers.

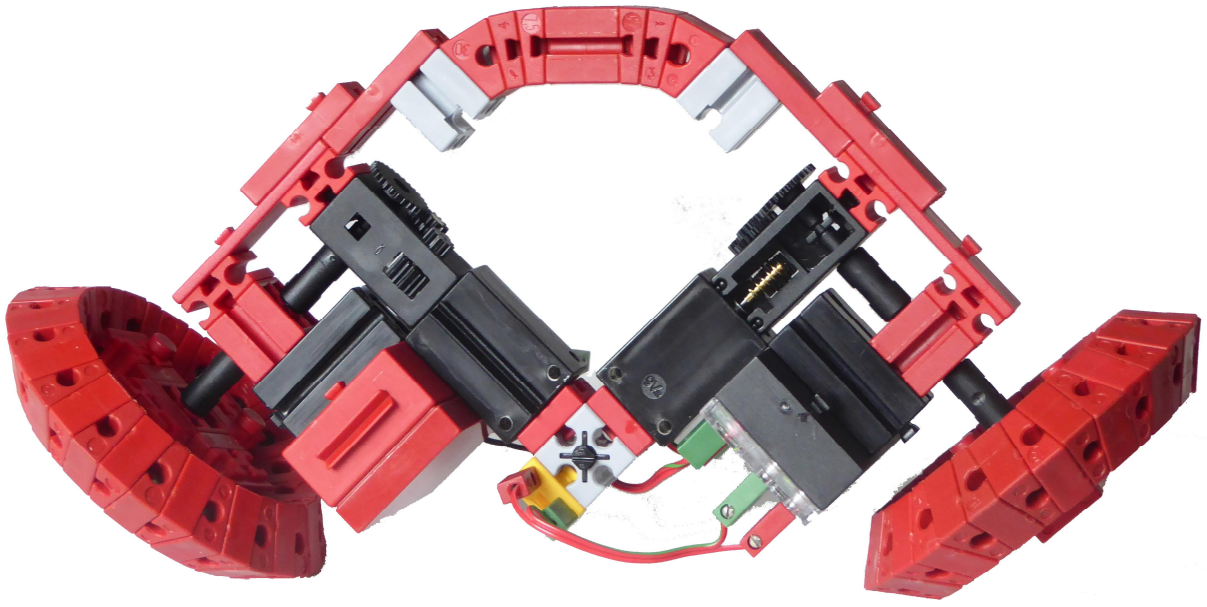


Abb. 33: Das M- (Fahr-) Zeug

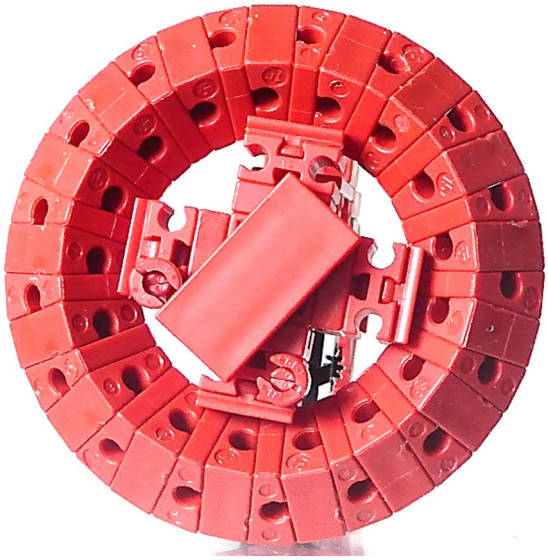


Abb. 34: UFO 2 mit Speichen und Achsbefestigung

Über die Ringe hinaus ergibt sich die Möglichkeit, um 45° gedrehte Bausteine zu verwenden.

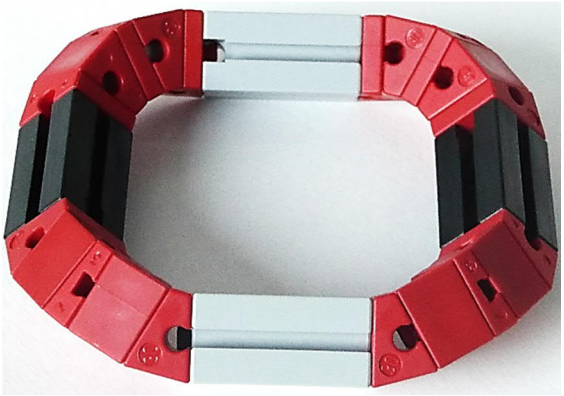


Abb. 35: Die Arena

Daraus lässt sich zum Beispiel eine Pyramide konstruieren; den Ansatz zeigt Abb. 36.

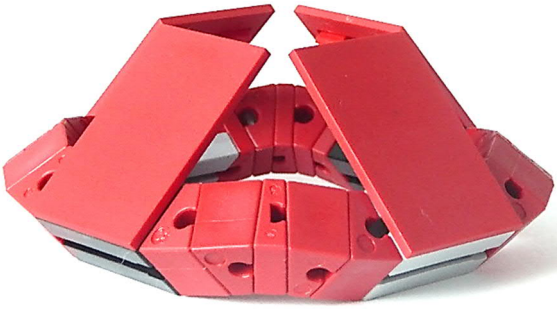


Abb. 36: Die Pyramide

Der kleine Dreh

All diese Figuren basieren auf diesem ganz einfachen Prinzip:

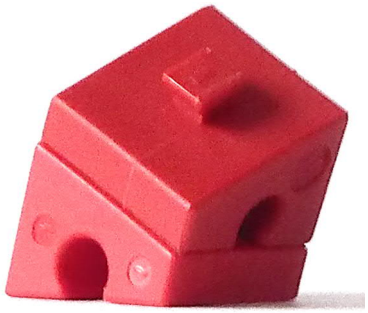


Abb. 37: Der kleine Dreh

Quellen

[1] Rüdiger Riedel: *Die Welt der ft-Winkelbausteine*. ft:pedia 1/2017, S. 17-23.

[2] Rüdiger Riedel: *Die Welt der ft-Winkelbausteine (Teil 2)*. ft:pedia 4/2018, S. 38-48.

Modell

Urlaubskasten-Modell 6: Berg- und Talbahn

Stefan Falk

Aus dem Wohnzimmer-Dienstreisen-Urlaubs-Notfallkasten aus ft:pedia 1/2016 [1] lässt sich eine einfache Berg- und Talbahn herstellen – bitteschön!

Das Vorbild

In der Anleitung der ersten fischertechnik-Statik-Kästen findet sich folgendes reizvolle Modell zum Ur-Kasten 300S [3]:

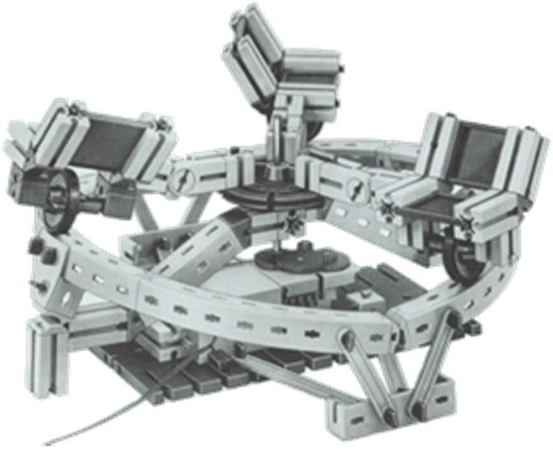


Abb. 1: Das Statik-Vorbild

Ein Motor treibt drei Wagen an, die auf einer aus Statik-Bogenstücken 60° gebauten Kreisbahn laufen. An zwei Stellen ist – hier mit Statik-Scharnieren – ein Knick eingebaut.

Das geht natürlich, wenngleich nicht so rund, auch mit Grundbausteinen, wie der folgende Modellvorschlag zeigt (Abb. 2). Der Modellvorschlag kommt mit den Teilen aus, die im „Urlaubskasten“ [1] und seiner Ergänzung laut [2] enthalten sind.

Der Aufbau

Die Mechanik ist einfach: Eine Kurbel geht über ein Kardangelenk auf ein Rast-Z20, welches das Z40 mit dem Karussell antreibt.

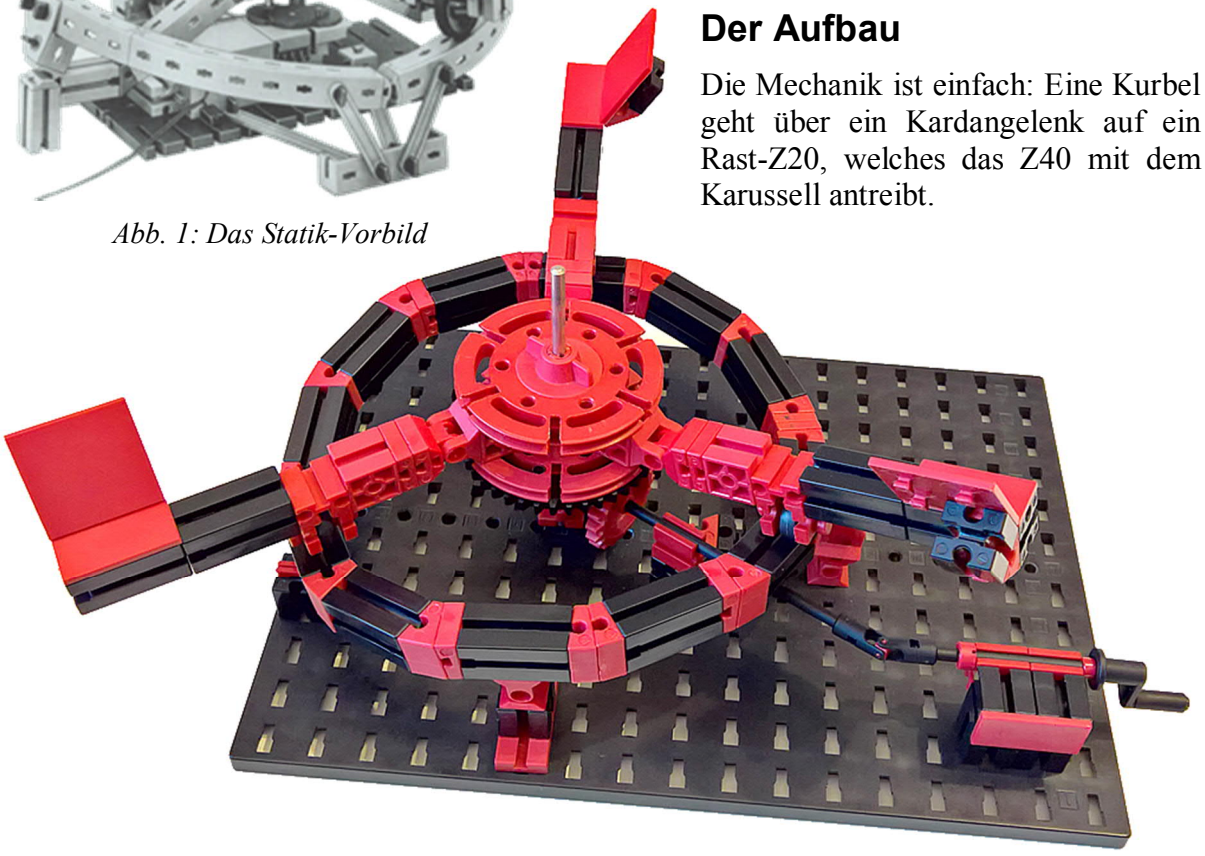


Abb. 2: Das Urlaubskasten-Karussell

Die drei Sitzträger sind in fischertechnik-Drehscheiben befestigt und können dank je eines Gelenkbausteins der Berg- und Talbahn folgen. Letztere ist einfach aus BS30 mit verschiedenen Winkelsteinen darin aufgebaut.

Der Antrieb

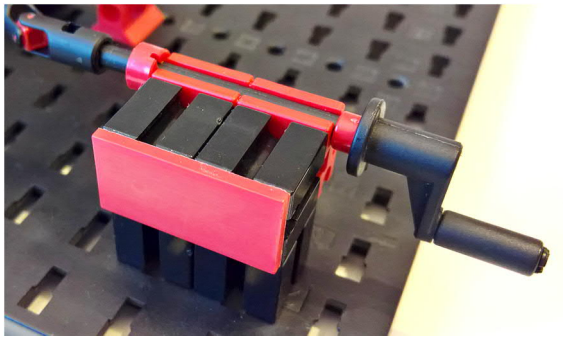


Abb. 3: Kurbel von vorne

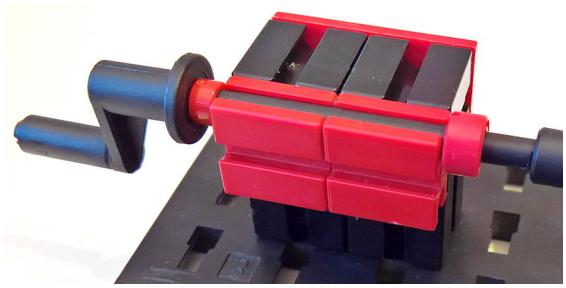


Abb. 4: Kurbel von hinten

Zweckmäßigerweise beginnen wir bei der Kurbel: Zwei BS15 mit zwei Zapfen, zwei BS15, eine Platte 15·30 zur Versteifung und zwei BS7,5 für die Achsführung genügen schon. Die Achse wird auf Kurbelseite mit einem Abstandsring 3 und auf der Abtriebsseite mit einem Klemmring fixiert. Das Ganze ergänzen wir wie in Abb. 5 zum Grundaufbau.

Der Ring

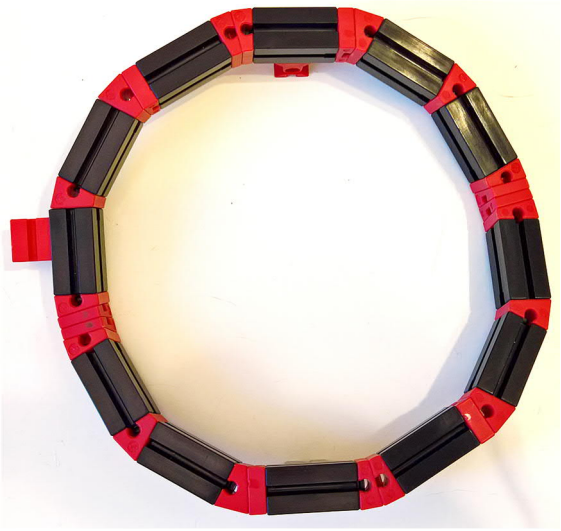


Abb. 6: Der Ring für die Berg- und Talbahn

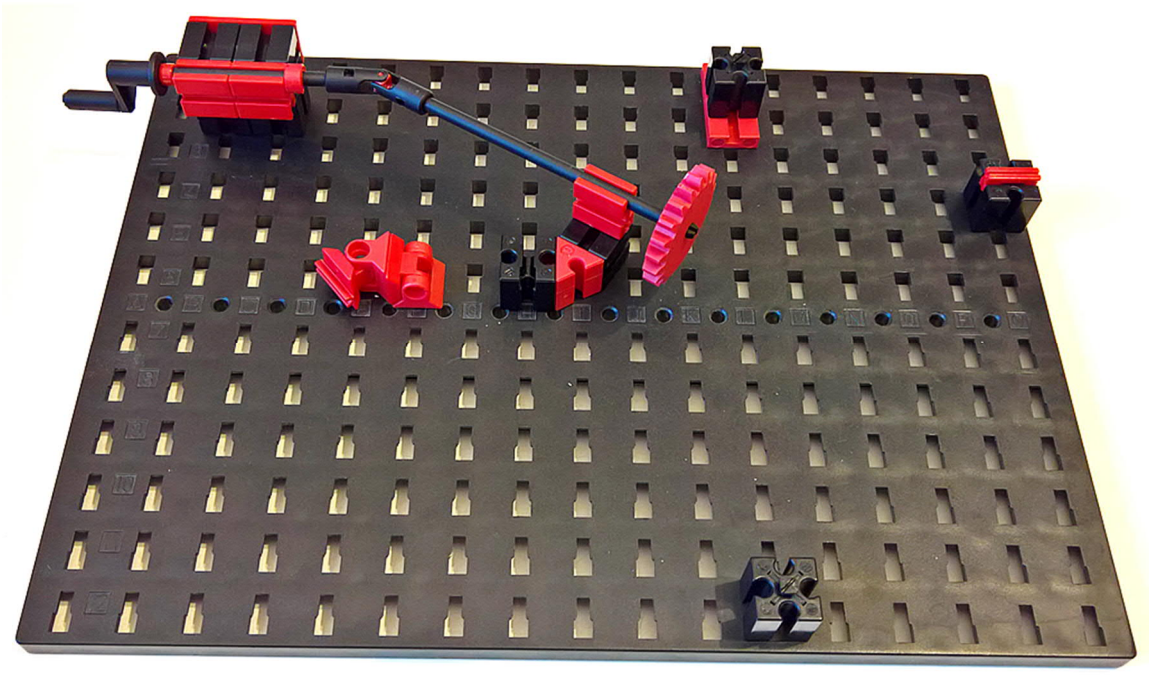


Abb. 5: Der Grundaufbau

Der Ring für die Berg- und Talbahn wird gebaut, wie es Abb. 6 zeigt.

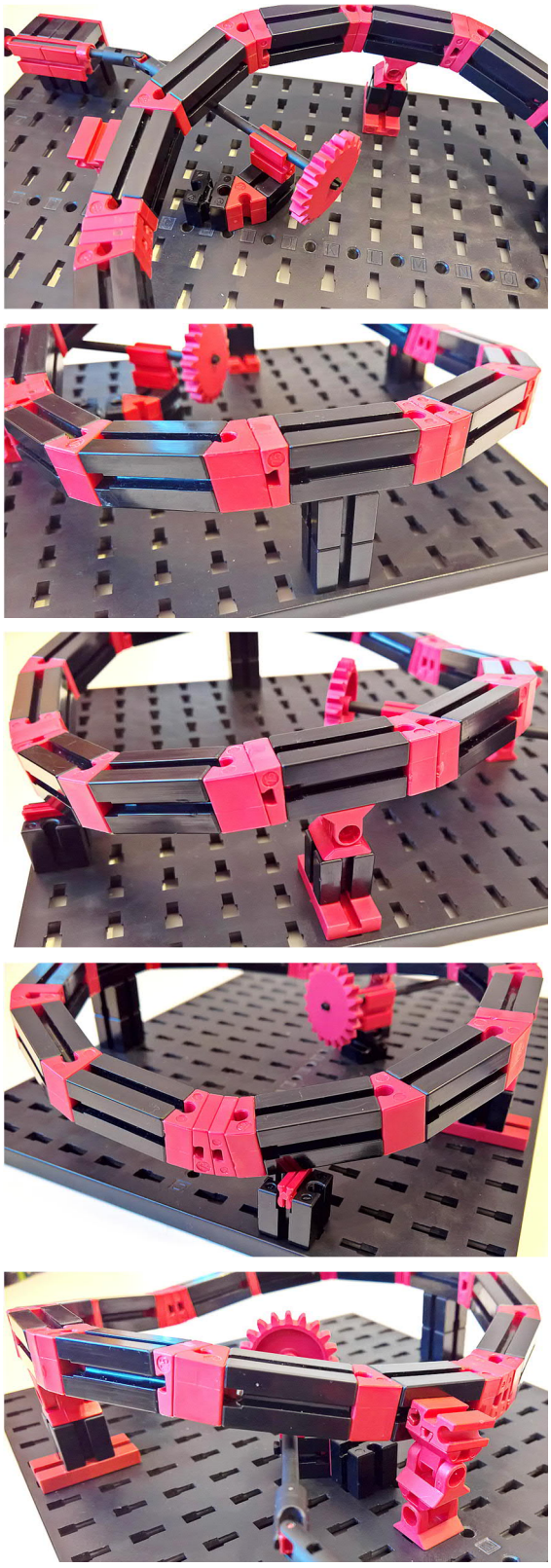


Abb. 7-11: Zur Befestigung des Rings

Die Abbildungen 7-11 zeigen, wie die Bahn an verschiedenen Stellen an der Grundplatte befestigt wird.

Die Sitzträger

Die Tragarme für die Sitze beginnen mit einem BS15 mit zwei Zapfen (hier einer mit Bohrung), der später in die Drehscheiben eingeschoben wird.

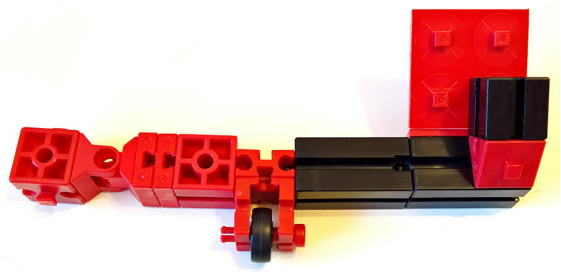


Abb. 12: Einer von drei Sitzträgern

Über ein Gelenk, zwei BS5 und einen weiteren BS15 mit zwei Zapfen geht es zu einem BS7,5, der nach unten die Rolle (die im Urlaubskasten nämlich laut [2] ergänzt wurde) und nach rechts zwei BS30 trägt. Auf dem sind eine Platte 15·30 als Sitz und über Winkelstein 60° und BS15 eine Platte 30·30 als Sitzlehne angebracht.

Der Mittelteil

Wie in Abb. 13 gezeigt, wird eine Metallachse 110 gelagert und mit einem Klemmring im unteren Rollenbock fixiert. Oben trägt er zwei Drehscheiben, die untere mitsamt einem Z40. Die obere Drehscheibe bekommt eine normale Nabe, die untere und das Z40 bekommen eine Flachnabe.

Abb. 13: Der angetriebene Mittelteil

Abb. 14 schließlich zeigt, wie die Tragarme in die Drehscheiben eingebracht und der gesamte drehbare Teil auf der Grundplatte so befestigt wird, dass das Z20 gerade satt ins Z40 eingreift.

Und dann schaut mal, ob es jemand schafft, am Karussell vorbeizugehen und *nicht* unbedingt auch mal kurbeln zu wollen – viel Spaß!

Quellen

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

[2] Falk, Stefan: *Ein kleines Update für den Urlaubskasten*. ft:pedia 3/2018, S. 6.

[3] fischertechnik: *Bauanleitung Statik 100S – 400S*. fischerwerke, Tumlingen, 1970/1971.

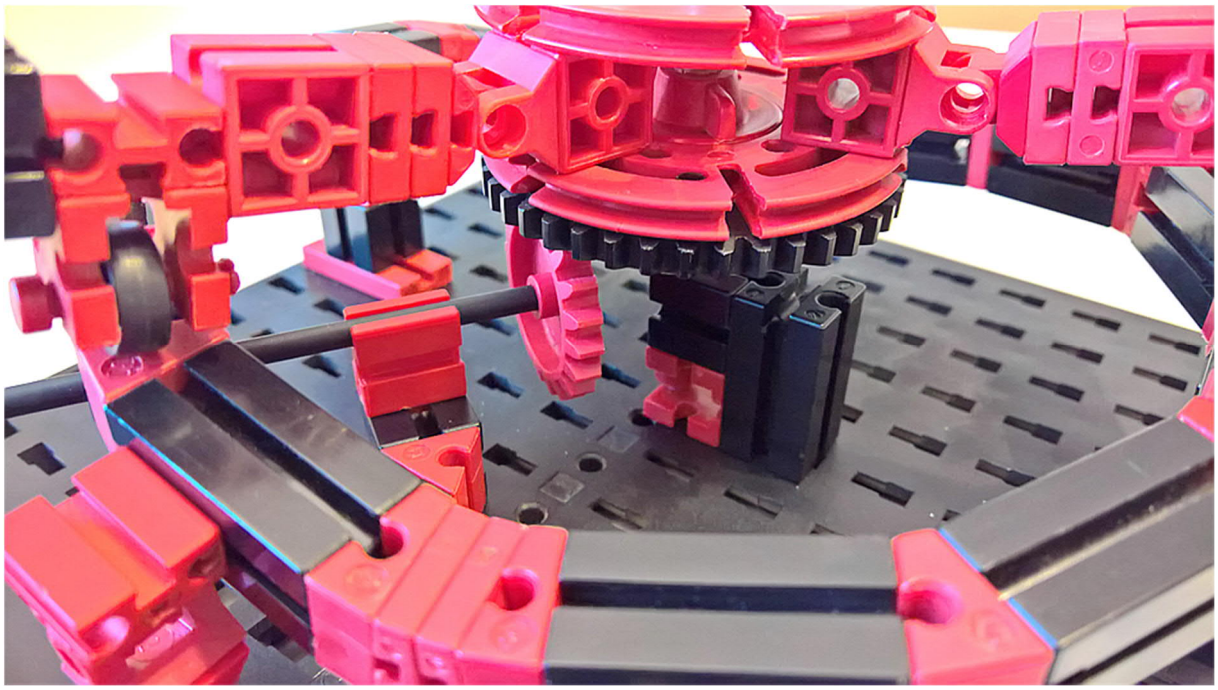


Abb. 14: Die Befestigung des drehbaren Teils mitsamt Tragarmen auf der Grundplatte

Modell

Kugel-Rotationsbeschleuniger

Rüdiger Riedel

Alle rollen abwärts, wir rollen aufwärts: Durch Rotation eilt die Kugel bergauf.

Man könnte auch sagen: Überall geht es bergab, aber bei uns geht es auch bergauf!

Die Anregung zu diesem Artikel erhielt ich durch „Jürgens Kugelbahn-Seiten“ im Internet, wo er seinen „Dreher“ vorstellte [1]. Die Idee ist, eine Kugel in schnelle Rotation zu versetzen und mit der gespeicherten kinetischen Energie vorwärts und aufwärts zu rollen. Die Motordrehzahl des S-Motors liegt bei ca. 9.500 U/min, und da der Durchmesser des antreibenden Rades fast dreimal so groß ist wie der der Kugel, wird diese sich deutlich schneller drehen.

Der Beschleuniger wird gebaut

Das V-Rad 23·10 (36581 rot, 154452 grün) mit dem Gummireifen 32,5·8 (34995) wird vom S-Motor in schnelle Drehung versetzt, auf Abb. 2 im Uhrzeigersinn, rechts herum. Die Kugel wird dadurch in Linksdrehung versetzt, sie bleibt an Ort und Stelle. Wird sie freigegeben, z. B. durch Absenken der Antriebsachse, setzt sie ihre Drehung in eine schnelle Fortbewegung nach links um.

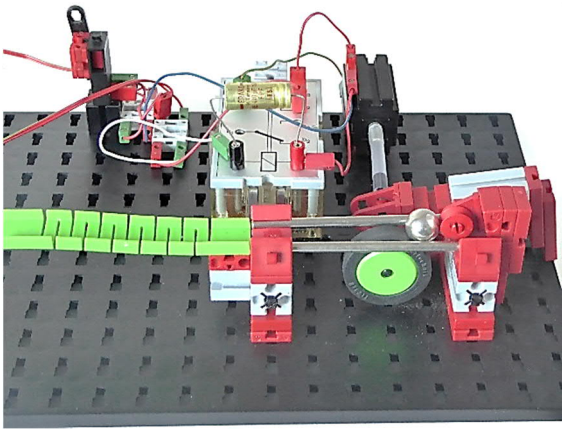


Abb. 2: Kugelbeschleuniger

Die Schienen am Start werden von zwei Achsen 80 gebildet. Die Auflager rechts und links sind unterschiedlich. Das linke besteht aus zwei Winkelsteinen 60° und

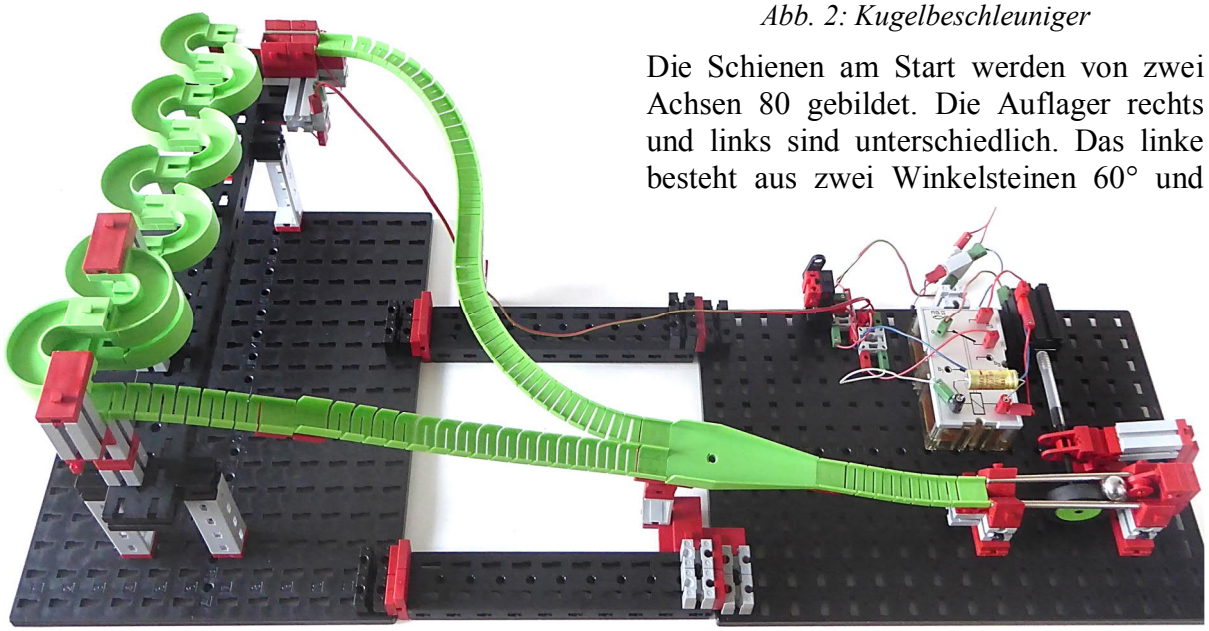


Abb. 1: Die Kugelbahn

zwei Winkelsteinen 7,5°, das rechte wieder aus zwei Winkelsteinen 60° und darunter zwei Winkelsteinen 15°! Die Anordnung ist nicht symmetrisch, die Kugel hängt links etwas tiefer und das Antriebsrad hat mehr Platz.

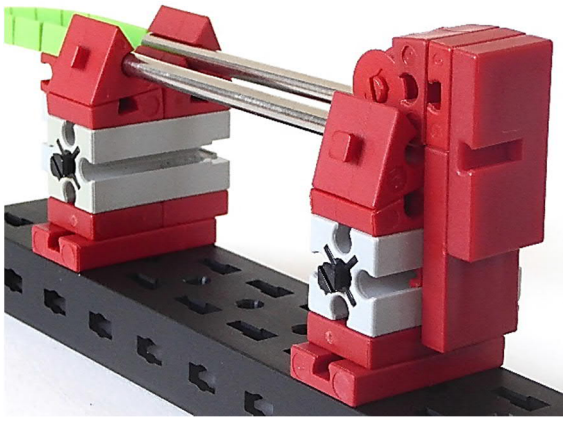


Abb. 3: Die beiden Auflager

Die Achsen werden zum Ausgang des Beschleunigers geschoben, zum Übergang auf die Flexschienen; sie klemmen ausreichend fest (Abb. 4).

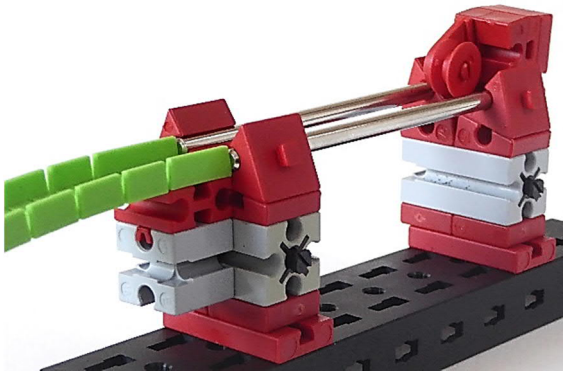


Abb. 4: Andere Perspektive der Auflager

Die Seilrolle 12·5,5 (38258) im V-Radhalter 10 (35668) hält die Kugel an ihrem Platz, die richtige Neigung erhält sie durch den Winkelstein 15°. Den Antrieb bewirkt ein S-Motor (32293).

Jetzt brauchen wir noch ein fischertechnik-Fremdteil: Ein etwa 45 mm langes Stück Silikonschlauch mit 4 mm Innendurchmesser und 6 mm Außendurchmesser. Das eine Ende wird über die Motorschnecke geschoben, das andere über eine Kunststoffachse 60.

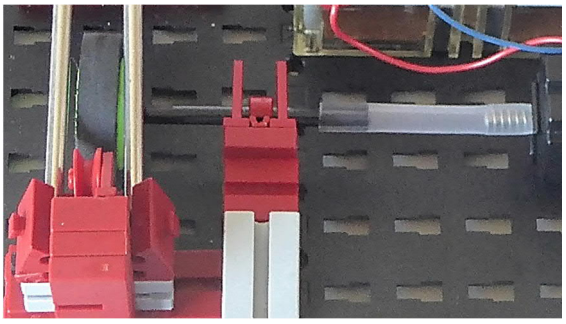


Abb. 5: Der Antriebsstrang

Die Antriebsachse wird beweglich gelagert, siehe Abb. 5 und 6.

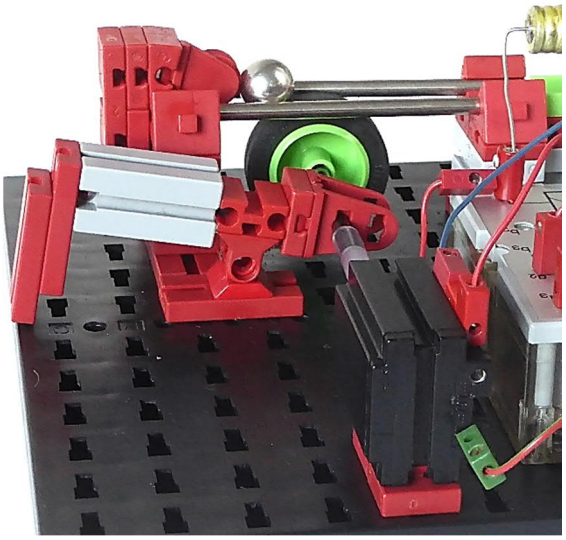


Abb. 6: Beschleuniger von hinten

Auf einem Baustein 5 15·30 (35049) sitzt mit einem Federnocken (31982) das Gelenk aus Gelenkwürfel-Zunge und Gelenkwürfel-Klaue 7,5 (31426 und 31436). Der Haltearm wird gebildet aus (von rechts nach links in Abb. 6):

1. zwei S-Kupplungen 15 2 (38253),
2. einem Baustein 5,
3. einem horizontal verbauten Baustein 7,5,
4. einem Baustein 30 und
5. zwei Bausteinen 5 15·30.

Der letzte Baustein links wird so eingestellt, dass die Antriebsachse waagerecht ausgerichtet ist und das Antriebsrad sich leicht nach unten drücken lässt.

Nun können wir einen Probestart durchführen. Den Motor schließen wir am besten an ein regelbares Netzgerät an und drehen

es auf etwa 6 V auf. Die vollen 9 V sind zu viel; das Rad dreht sich so schnell, dass der Gummireifen durch die Fliehkräfte aufgeweitet wird und herunterrutschen kann. An das Ende der Schienen sollten wir einen Kartondeckel legen, der die Kugel auffängt. Jetzt legen wir die Kugel auf wie in Abb. 6 lassen sie etwa drei Sekunden andrehen und senken dann das Antriebsrad ab.

Und wenn du nicht aufpasst ist – kladderadatsch – die Kugel futsch.

wusch! zoing! zong! wisch! klonk! weg!

Der elektrische Start

Als nächstes fragen wir uns, ob der Start der Kugel automatisiert werden kann.

Schalten wir beim Kugelandrehen den Motor aus, rollt die Kugel manchmal davon. Wenn wir den Motor ausschalten und kurzschließen, rollt die Kugel immer davon! Das lässt sich einfach mit einem Taster und der Schaltung nach Abb. 7 verwirklichen.

Die Kugel wird entsprechend Abb. 6 platziert. Dann wird der Taster kurz gedrückt und die Kugel angedreht.

Beim Loslassen wird der Motor kurzgeschlossen und stoppt sofort. Die Umfangsgeschwindigkeit des Antriebsrades wird kleiner als die der Kugel, somit rollt sie über das Rad hinweg und „düst ab“.

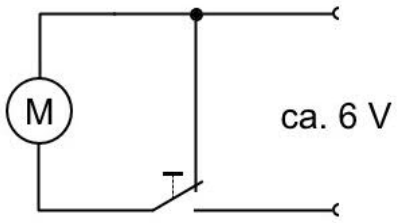


Abb. 7: Start mit dem Taster

Das Relais

Wir können einen weiteren Schritt zur Vollautomatisierung vornehmen und ersetzen den Taster durch zwei Stromschienen. Rollt eine Kugel darüber, schließt sie den Stromkreis eines Relais, welches dann den Motor kurzschließt und dadurch den Start einer zweiten Kugel auslöst.

Der Halter für die Stromschienen hat als Basis zwei Bausteine 2,5 15·45 2+2Z (38277) und zwei Bausteine 5 15·30 3N (38428) sowie zwei Bausteine 30 und vorne in Abb. 8 einen Baustein 15 sowie einen auf der Rückseite als Träger. Oben sitzen auf zwei Bausteinen 5 15·30 (35049) zwei Winkelsteine 7,5° und zwei Winkelsteine 60°, in welche die beiden Achsen 30 eingelegt werden.

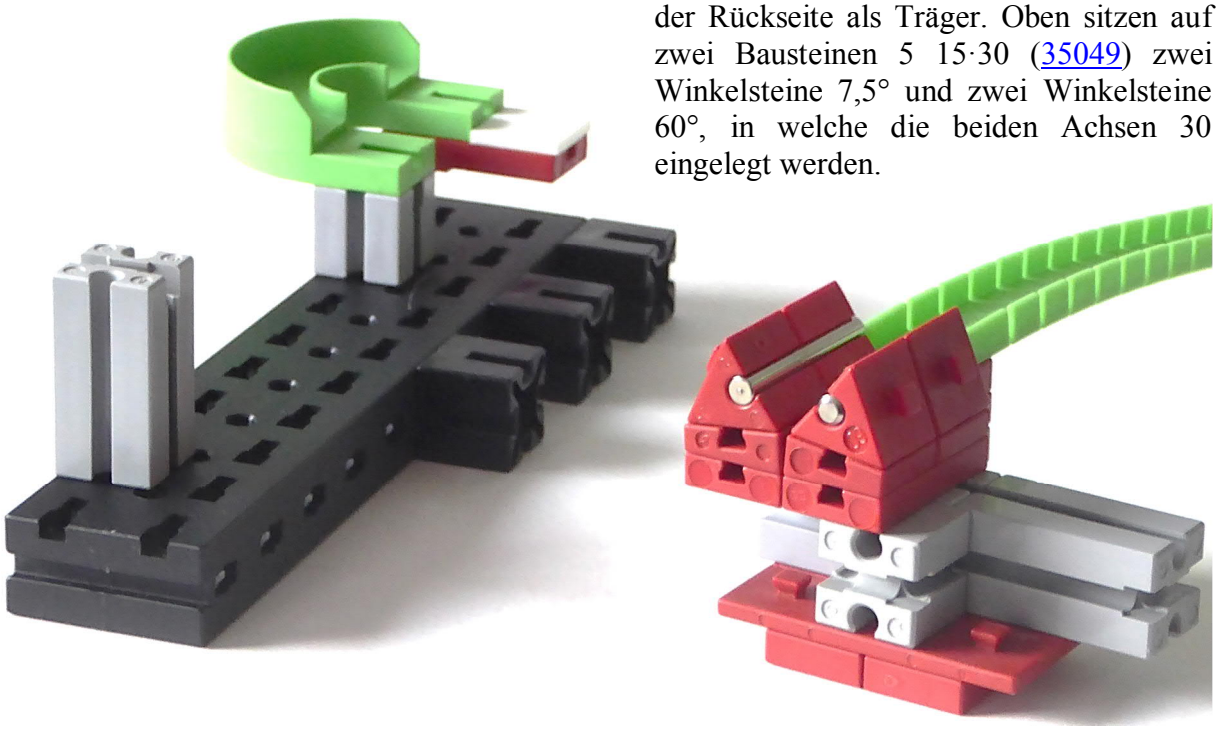


Abb. 8: Die Stromschienen

Der Übergang von der letzten Kugelbahn-Kurve wird durch einen Baustein 5 15·30 und eine Bauplatte 15·15 1Z hergestellt. Der gesamte Halter wird zwischen zwei S-Riegelsteine 15·15 (32850) geschoben.

Der elektrische Anschluss an die Achsen erfolgt durch Unterklemmen der blanken Drahtenden.

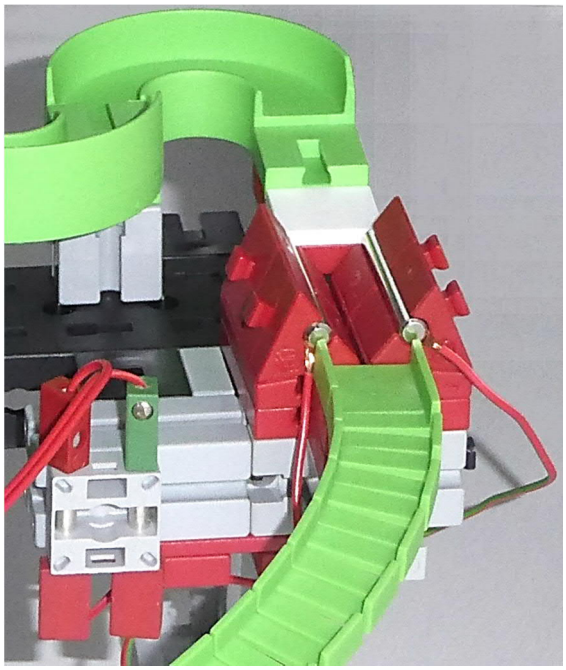


Abb. 9: Anschluss der Stromschienen

Die schaltende Kugel wird meistens nicht gleichmäßig über die Stromschienen hinwegrollen, sondern ein wenig hüpfen. Deswegen wird das Relais nicht sauber ansprechen. Das können wir verbessern: Wir überbrücken die Anschlüsse des Relais mit einem Elektrolytkondensator. Die elektrische Schaltung geht aus Abb. 10 hervor.

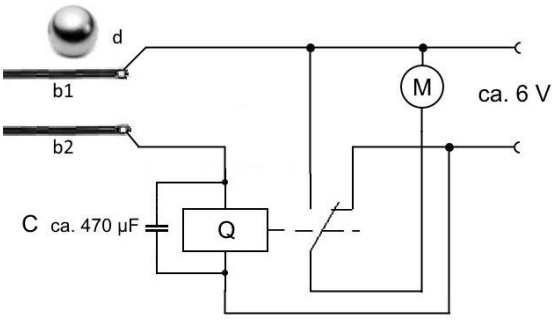


Abb. 10: Automatisierung

Q = das Relais, z. B. ft-Nr. 37683

b = die beiden Achsen 30 als Stromschienen

M = der Motor

d = die Kugel, die das Relais auslöst

C = Elektrolytkondensator (auf richtige Polung achten!)

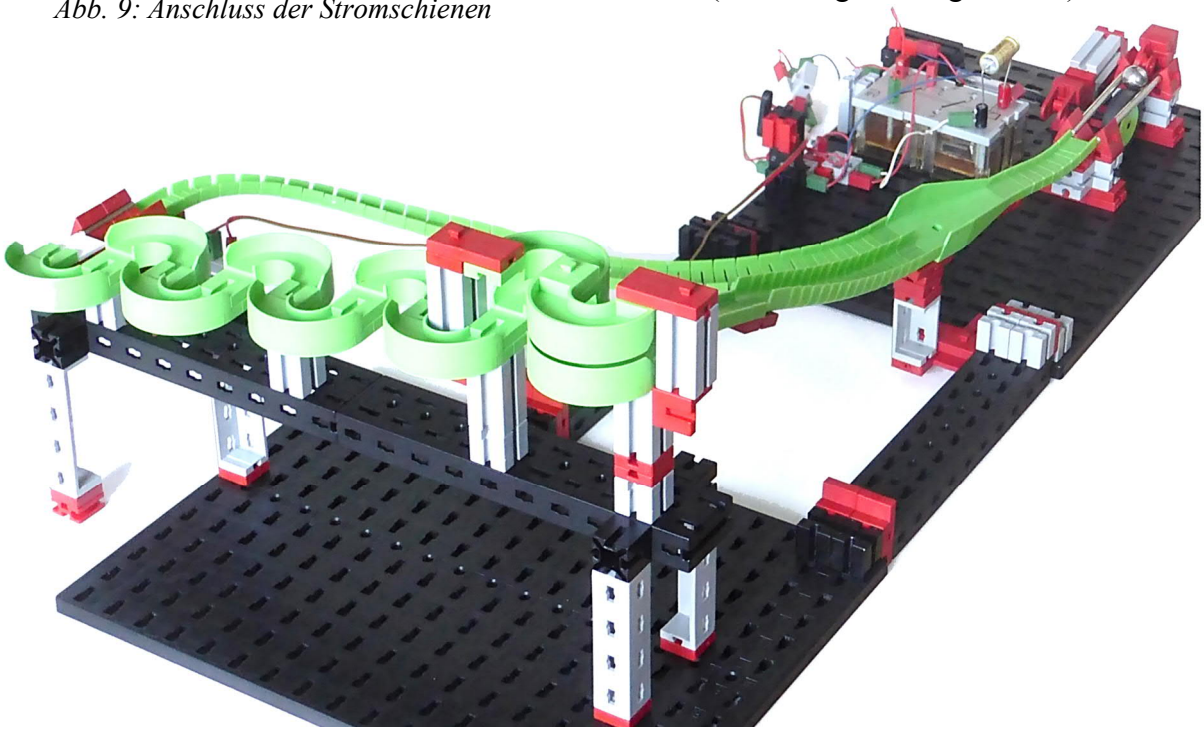


Abb. 11: Der Kugelfang

Der nächste Schritt ist eine kleine Kugelbahn, mit der die Möglichkeiten ausprobiert werden.

Die Kugelbahn

Die Kugelbahn ist im Prinzip eine Treppe mit 5 mm hohen Stufen, aufgebaut aus 9 Kugelbahn-Kurven 180° (158962) auf zwei waagerechten U-Trägern 150 (32854). Der Anstieg vom Start zur ersten Kurve besteht aus einem Kugelbahn-Flexprofil 90 (155901), der Kugelbahn-Wechselweiche (151716) ohne den Weichensteller, einem Kugelbahn-Flexprofil 180 hoch (159783) und einem weiteren Flexprofil 90. Die Weiche wird in zwei Richtungen mit je einem Winkelstein 7,5° schräg gestellt: Schräg nach oben und schräg nach vorne. Die losrollende Kugel wird somit sicher auf den linken Ausgang der Weiche geführt.

Der Aufstieg der Kugeln erfolgt nicht immer mit der gleichen Geschwindigkeit, sondern zufallsbedingt mal schneller, mal langsamer. Haben wir die Motordrehzahl so eingestellt, dass die Kugeln zuverlässig oben ankommen, dann neigen einige dazu, über die erste Kurve hinaus zu springen. Das soll der Kugelfang über den ersten beiden Kugelbahn-Kurven verhindern. Er besteht aus zwei verkehrt herum angebrachten Kugelbahn-Kurven entsprechend Abb. 11. Nachdem eine Kugel die Stromschienen passiert hat, erfolgt der Abstieg über zwei Flexprofile 180 (143234) in die breite Seite der Weiche zurück zum Start.

Wir starten

Der Automatikbetrieb kann beginnen. Eine Kugel legen wir in den Beschleuniger (Abb. 6) und schalten die Stromversorgung ein. Eine zweite Kugel lassen wir irgendwo auf der serpentinenförmigen Treppe hinunterrollen und hüpfen. Sobald sie die Stromschienen erreicht, zieht das Relais an und die erste Kugel startet. Anschließend fällt das Relais wieder ab, die zweite Kugel rollt in die Startposition und wartet auf die Auslösung, und so weiter…

Zur Vervollständigung der Anlage können wir noch einen Ein-Schalter hinzufügen. In Abb. 2, links oben, sehen wir eine aus dem Bilderpool bekannte Variante bestehend aus einem Minitaster (37780 oder 37783), einem Baustein 7,5, einem S-Strebenadapter (31848) und einer S-Strebe 15 I (36914).

Eine weitere Verbesserung bringt eine Diode am Eingang, so dass der Motor immer richtig herum dreht und vor allem der Elektrolytkondensator nicht falsch gepolt werden kann. Will man mit kleinerer Spannung als 6 V arbeiten, muss das Relais an einem eigenen Stromkreis mit 6 bis 9 V betrieben werden, weil es sonst nicht zuverlässig anspricht.

Nun ein fröhliches wrrrr! wusch! zoing! dongdongdong! klonk!
Und hoffentlich nicht „weg!“

Quellen

[1] Jürgens Kugelbahn-Seiten: *Dreher*. Jürgen Kintscher, 2000-2003.

[2] Rüdiger Riedel: *Video des Kugel-Rotationsbeschleunigers*. Yotube, 2019.

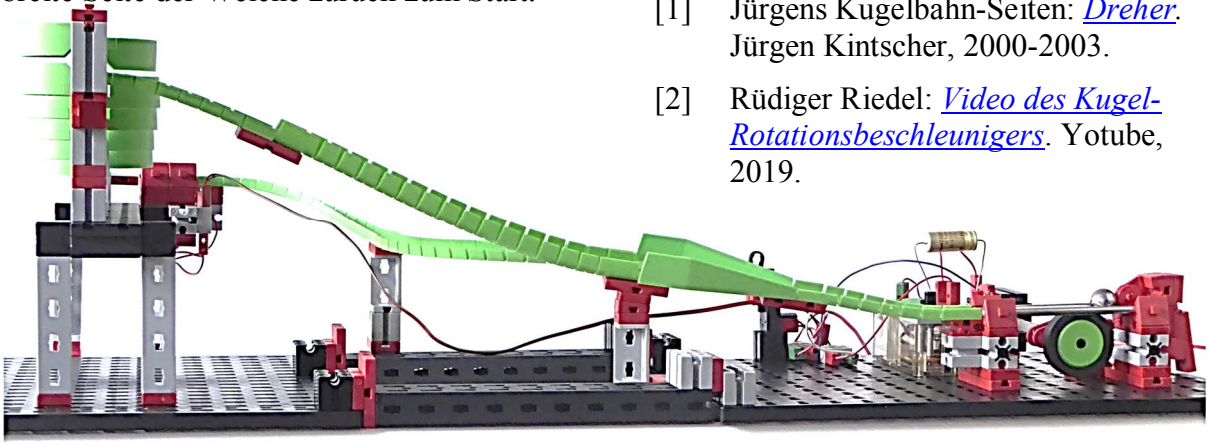


Abb. 12: Beschleunigt aufwärts

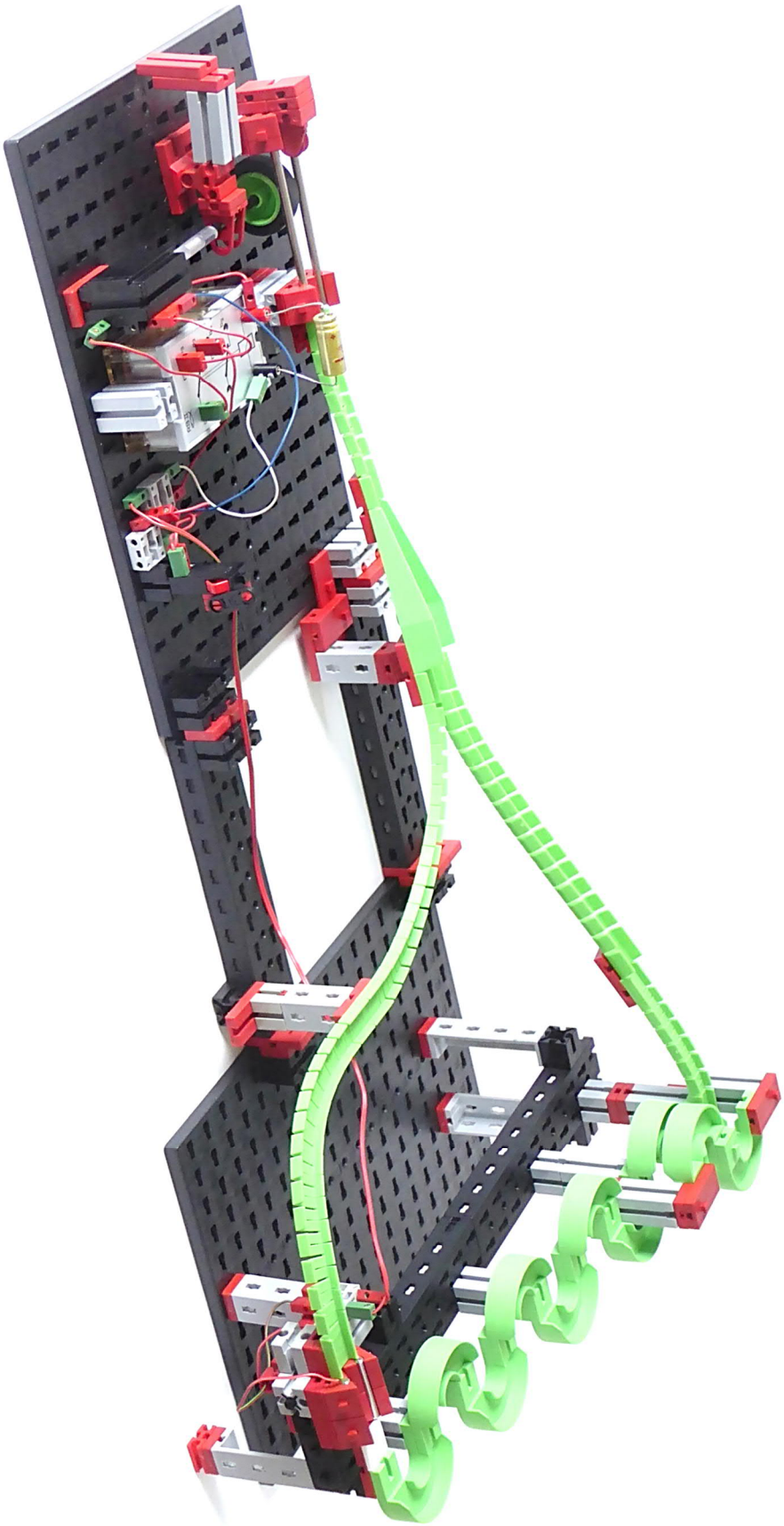


Abb. 13: Gesamtansicht der Kugelbahn

Modell

Elektrisch verstellbares Teleskopstativ aus fischertechnik

Leon Schnieber

In der Schule belegte ich das Wahlfach Astronomie. Da unser Lehrer neben dem gewöhnlichen Lehrstoff auch versucht, uns für die Sternenbeobachtung zu begeistern, bauten wir mit ihm außerhalb des Unterrichts ein „Baumarkt-Teleskop“ [1]. Schon beim Bau fasste ich den Entschluss, die Ausrichtung des Fernrohrs mit Hilfe von fischertechnik-Teilen zu motorisieren.

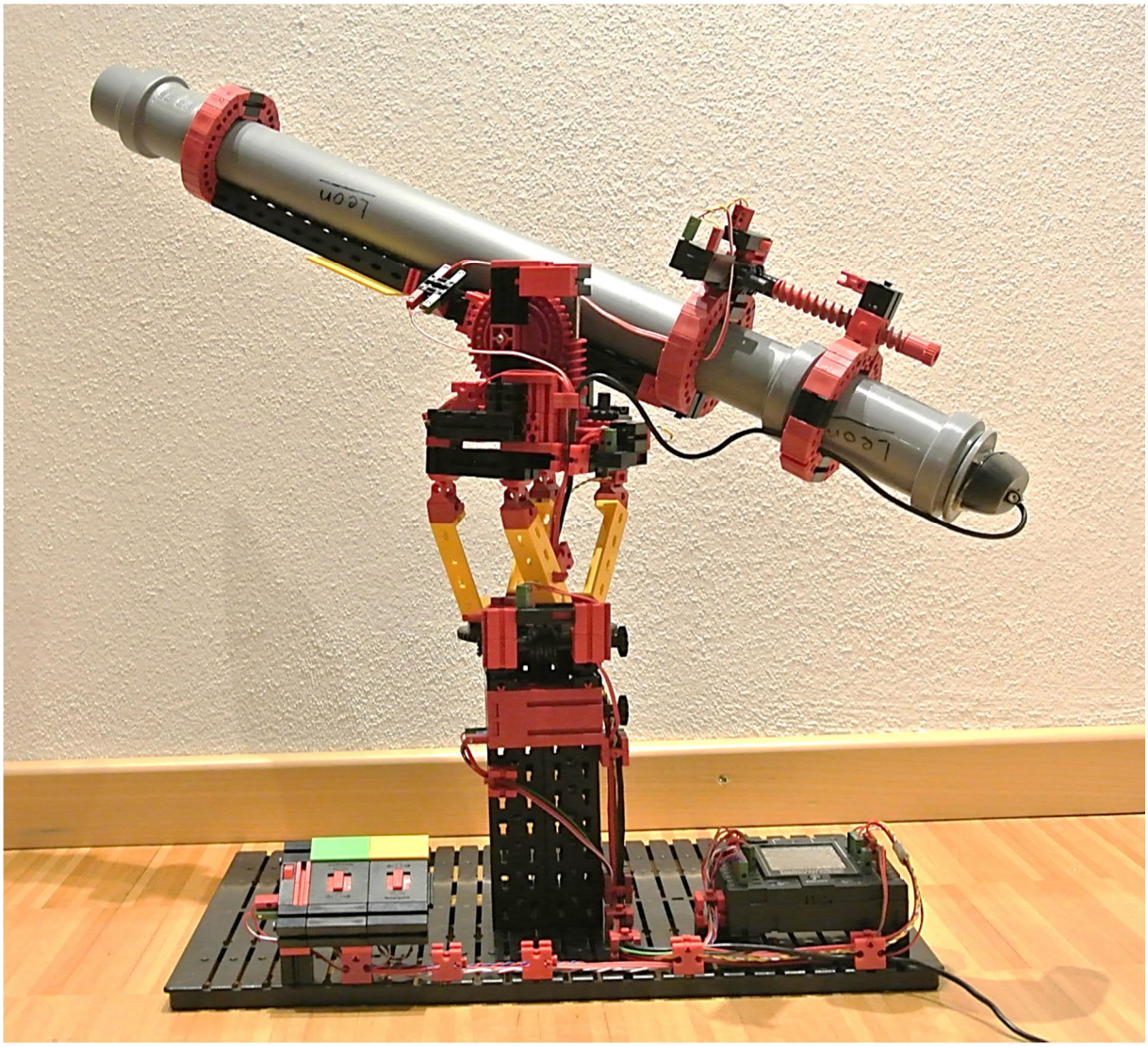


Abb. 1: Gesamtansicht des Teleskops

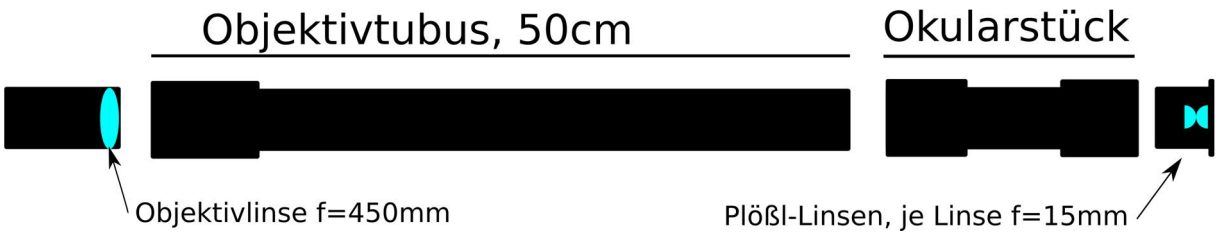


Abb. 2: „Explosionsansicht“ des Teleskops: Das Rohrstück links steckt im Tubus, die Muffe rechts im Okularstück. Dieses steckt locker auf dem Objektivtubus, so dass es sich leicht bewegen lässt.

Aufbau

Die Basis dieses Teleskops ist ein ca. 50 cm langes HT-Rohr aus dem Baumarkt. Der Anleitung des Bausatzes [1] folgend sägten wir am schmalen Ende des Rohres ca. 7 cm ab. Die Objektivlinse mit einer Brennweite f = 450 mm klebten wir auf die glatte Kante des beim Sägen neu entstandenen Reststücks.

Ein Muffenstopfen dient bei unserem Teleskop als Okular. Hinter das mittig gebohrte Loch klebten wir zwei Linsen mit je 15 mm Brennweite, so dass ein Plössl-Okular entstand, durch das man dann später den Sternenhimmel sieht. Um die Fokussierung zu erleichtern, steckten wir den Stopfen in eine Doppelmuffe und verklebten ihn dort. Am anderen Ende entfernten wir dagegen die Gummilippe, damit sich das Okular zum Scharfstellen vor- und zurückschieben lässt (Abb. 2). Das Innere des langen Rohrstückes kleideten wir außerdem mit schwarzem Tonpapier aus. Das minimiert später die Reflexionen von eindringendem Störlicht in den Objektivtubus.

Funktionsweise

Eigentlich ist das Baumarkt-Teleskop ein Kepler-Teleskop: Die Objektivlinse erzeugt vom beobachteten Objekt ein Zwischenbild am Brennpunkt des Objektivs. Für ein scharfes Bild muss der Betrachter das Okular nun so einstellen, dass das Zwischenbild exakt im Okular-Brennpunkt liegt. Das Bild, das der Betrachter durch das Okular sieht, steht auf dem Kopf.

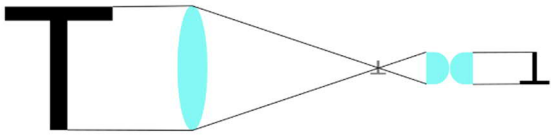


Abb. 3: Verlauf der Lichtbündel zwischen Objektiv und Okular. Links das reflektierte Licht vom Objekt, rechts das auf dem Kopf stehende „virtuelle“ Bild des Betrachters

Der theoretische Vergrößerungsfaktor V eines Teleskops errechnet sich aus dessen Brennweite f:

$$V = \frac{f_{Objektiv}}{F_{Okular}} = \frac{450\ \text{mm}}{15\ \text{mm}} = 30$$

Für die tatsächlich erreichbare Vergrößerung ist allerdings die Objektivöffnung entscheidend. Je größer sie ist, desto mehr Licht tritt in das Teleskop ein und es werden mehr Sterne sichtbar. Auch der Durchmesser des Okulars spielt eine Rolle, im Fall des Baumarkt-Teleskopes ist es der der Augenpupille. Für die praktisch nutzbare Normalvergrößerung V_{normal} gilt folgende Formel:

$$V_{normal} = \frac{d_{Objektiv}}{d_{Augenpupille}} = \frac{40\ \text{mm}}{5\ \text{mm}} = 8$$

Mit der bestehenden Linsenkonstellation ist also mindestens eine achtfache Vergrößerung möglich. Bei optimalen Luftbedingungen kann sich dieser Wert aber dem theoretischen Vergrößerungsfaktor von 30 annähern.

Aufbau des Stativs

Die Anforderungen an das Stativ waren schnell aufgestellt: Das Rohr muss sich in mindestens zwei Achsen bewegen, um den gesamten Abendhimmel abdecken zu können. Außerdem muss die Übersetzung der Motoren so klein wie möglich sein, um feine Justierungen des Teleskoprohres vornehmen zu können. Ist die Schrittweite zu groß, kann man z. B. eine Sternenkonstellation nicht genau im Bild positionieren.

Anfangs plante ich einen Automatik-Modus für das Teleskop, zum Beispiel in Kombination mit einer automatischen Kamera für große Panoramaaufnahmen des Nachthimmels. Diese Idee verwarf ich aber recht schnell; ein Zusammensetzen der Aufnahmen ist durch fehlende Kanten schwierig zu automatisieren und wäre mit unverhältnismäßig viel Nacharbeit am Computer verbunden gewesen.

Die Basis des Teleskopstativs ist eine Grundplatte 1000. Auf dieser habe ich mittig, aber etwas erhöht einen Drehkranz montiert. Dieser ist motorisiert und kann den Teleskopaufbau um knapp 360° drehen. Auf ihm sitzt, ebenfalls etwas erhöht, eine weitere Achse. Dieser Aufbau ist um 30° nach oben geneigt, ansonsten hätte der Kippmechanismus das Teleskoprohr nicht senkrecht in den Himmel richten können. Die zweite Achse ist etwas umständlich gelagert, da ich keinen weiteren Drehkranz besitze. Eine Metallachse führt direkt durch den Führungs-U-Träger auf der unteren Seite des Teleskoprohrs. Auf der einen Seite der Metallachse steckt ein Zahnrad Z40. Dieses ist über einen seiner drei Stifte mit dem U-Träger verbunden, sodass bei einer Bewegung des Rohraufbaus nichts durchrutschen kann. Seitlich am Zahnrad ist eine Schnecke montiert und über ein Getriebe mit einem S-Motor verbunden. (Abb. 4) Ein Taster mit Impulszahnrad Z4 gibt die nötige Rückmeldung an den Controller.

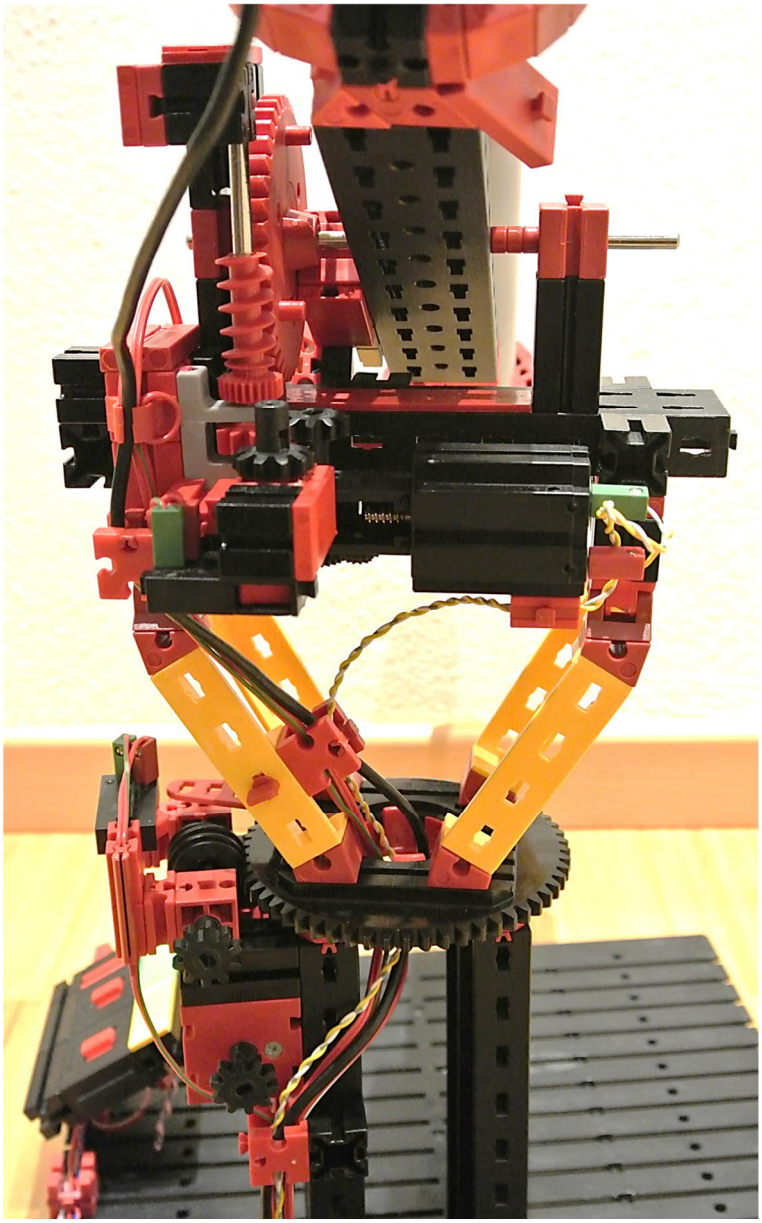


Abb. 4: Direkt hinter der Metallachse: Einer der drei Stifte im Z40 sichert durch einen am U-Träger montierten „Baustein 15 mit Loch“ das Teleskoprohr gegen Schlupf.

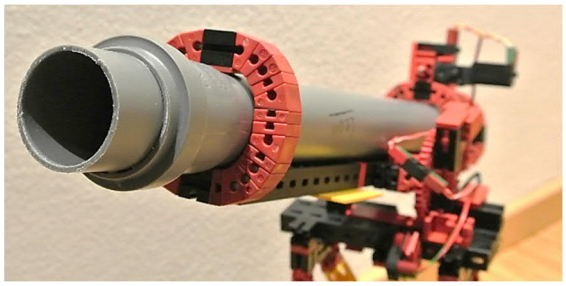


Abb. 5: Vorderer Haltering für das Teleskoprohr

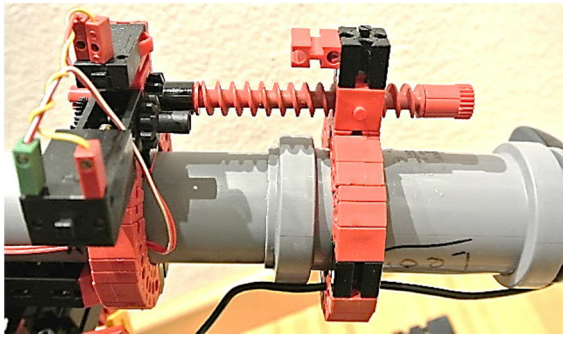


Abb. 6: Der rechte Teil der Fokussiereinheit kann frei verschoben werden

Auf dem U-Träger des Neigemechanismus ist über zwei Halteringe aus Winkelsteinen das Teleskop befestigt (Abb. 5). Der Schwerpunkt liegt ungefähr über der Metallachse. Die Kräfte, die auf die Achse wirken, sind dadurch kleiner. Am wichtigsten ist allerdings die Fokussiereinheit: Die Muffe am Teleskop kann, je nach Entfernung des zu betrachtenden Objektes, nach vorne oder hinten bewegt werden, um das Bild scharfzustellen. Eine Gewindestange und ein Haltering führen die Rohrmuffe linear entlang, ein XS-Motor bewegt diese Achse (Abb. 6). Eine Rückmeldung der Impulse ist hier nicht nötig, da die Fokussierung entweder manuell vom Bediener oder über eine Auswertung des Kamerabildes vorgenommen wird.

Steuerung

Bei der Sternenbeobachtung verbringt man die meiste Zeit mit dem Suchen nach Sternbildern oder anderen interessanten Konstellationen; Panoramen fertigt man dagegen eher selten an. Deshalb habe ich den Fokus der Steuerkonsole auf den manuellen Betrieb gelegt. Zwei Polwendeschalter stellen die beiden Freiheitsgrade des Stativs dar. Ein Taster ermöglicht eine feinere Dosierung der Impulse: Ein simples Programm wertet die Zustände der Polwendeschalter und des Tasters aus und setzt diese in entsprechende Motorenbewegungen um.

Drückt man den Taster kurz, fährt das Teleskop so wenig wie möglich in die gewählte Richtung. Hält man ihn für längere Zeit gedrückt, bewegt sich das Teleskop für eine längere Strecke und bremst es beim Loslassen sanft ab.

Nach den ersten Tests an einem wolkenfreien Abend stellte sich schnell heraus, dass das Okular vor allem in dem Zenit zugewandten Positionen zu tief liegt, um mit dem Auge heranreichen zu können: Zwischen dem Rohr-Ende und der Grundplatte sind nur wenige Zentimeter Platz. Deshalb klebte ich eine Webcam auf das Teleskopende, so dass deren Bild über einen Monitor schnell und bequem sichtbar wurde. Entfernt man außerdem den Infrarotfilter, kann man weitere interessante Bilder in Bereichen sehen, die das menschliche Auge sonst nicht wahrnehmen kann.

Fazit

Der Aufwand hat sich in meinen Augen auf jeden Fall gelohnt. Die Ergebnisse, die mit einem so einfachen Linsenaufbau möglich sind, finde ich sehr beeindruckend. Der Vergrößerungsfaktor dieses Selbstbau-Teleskops ist bei guter Witterung sogar vergleichbar mit den damaligen Instrumenten Galileo Galileis [2].

Quellen

[1] Hüning, Klaus: *Das Baumarkt-Teleskop*. ninmax GmbH, Astromedia, Landsberg am Lech, 2018.

[2] Spektrum der Wissenschaft Verlagsgesellschaft mbH: *Lexikon der Physik: Teleskop*. Heidelberg, 1998.

Schaltungstechnik

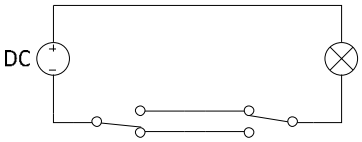
Motorsteuerungen (5): Schrittschaltwerke mit Wechselschaltung oder: Die Macht des XOR

Stefan Falk

In verschiedenen Beiträgen in der ft:pedia haben wir solche Schaltungen bereits verwendet, aber sie verdienen mal eine eigene, ausführliche Betrachtung: Die Verwendung von zwei Tastern für eine Schritt-Steuerung.

Wechselschaltung mit Lampe

Mit zwei Tastern oder Schaltern können wir eine Lampe bequem von zwei Stellen aus steuern [1]:



Schaltung 1: Wechselschaltung

Egal, ob die Lampe gerade leuchtet oder nicht, wir können sie von jedem der beiden Schalter aus umsteuern – probiert es!

Die Lampe leuchtet in Schaltung 1 genau dann, wenn beide Schalter „unten“ (betätigt) oder beide „oben“ (unbetätigt) stehen. Sie leuchtet nicht, wenn einer betätigt und der andere unbetätigt ist.

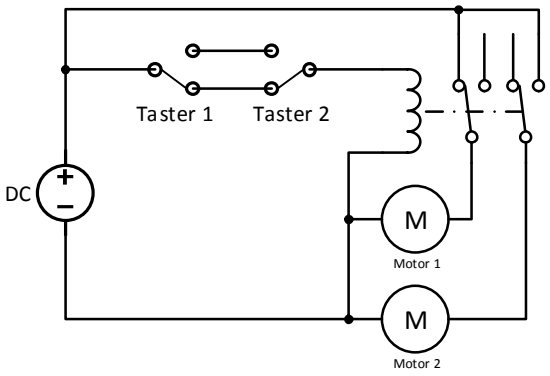
Wechselschaltung mit zwei Verbrauchern

Nun wollen wir das erweitern: Anstatt eine Verbraucher (die Lampe) wollen wir nun zwei (Lampen oder Motoren) so steuern, dass immer genau einer von beiden aktiv ist. Schaltung 1 muss also so ergänzt werden, dass es einen zweiten Verbraucher gibt, der genau dann Strom bekommt, wenn die Lampe *nicht* leuchtet. Wenn wir zwei Lampen verwenden würden, würde also eine immer dann leuchten, wenn die andere gerade nicht leuchtet, und umgekehrt.

Die zweite Lampe muss also dann leuchten, wenn die beiden Schalter in Schaltung 1 unterschiedlich stehen (einer betätigt, einer unbetätigt). Dazu gibt es mindestens zwei Möglichkeiten: Eine mit zwei Tastern und einem Relais und eine nur mit drei Tastern.

Wechselschaltung mit Relais

Das Problem dabei ist, dass wir wieder etwas einschalten müssen, wenn *kein* Strom anliegt (wenn unsere erste Lampe also *nicht* leuchtet). Das geht, wie wir in [1] gesehen haben, z. B. mit einem Relais. In Schaltung 1 wird der Verbraucher (die Lampe) durch ein Relais ersetzt, das Umschaltkontakte bietet und so in beiden Stellungen (angezogen oder abgefallen) je einen anderen Verbraucher einschalten kann:



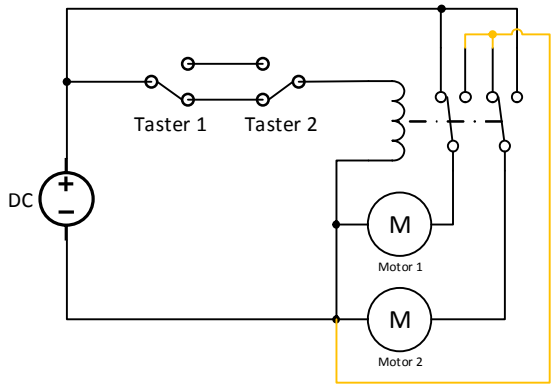
Schaltung 2: Wechselschaltung für zwei Verbraucher mit Relais

Das Relais haben wir absichtlich so verschaltet, wie Schaltung 2 zeigt, und nicht einfach die beiden Motoren an einem einzigen Wechselkontakt angeschlossen – wir werden gleich noch sehen, warum.

Die Schaltung funktioniert nun so: Wenn beide Taster denselben Zustand haben, zieht das Relais an, andernfalls fällt es ab (würde man die beiden Leitungen zwischen den Tastern überkreuzen, wäre es genau andersherum). Bei angezogenem Relais läuft einer der Motoren (hier *Motor 2*, denn das Schaltbild zeigt immer die Ruhestellung), bei abgefallenem Relais der andere (hier also *Motor 1*).

Kurzschlussbremse

Durch zwei weitere Leitungen können wir die in [2] beschriebene Kurzschlussbremse verwenden. Die bewirkt, dass ein ausgeschalteter Motor nicht lange nachläuft, sondern schlagartig und praktisch sofort anhält:



Schaltung 3: Wechselschaltung mit Kurzschlussbremse

Die orangefarben eingezeichneten zusätzlichen Leitungen schließen einen auszuschaltenden Motor kurz (beide Motoranschlüsse liegen dann am selben Pol der Stromversorgung). Diese Erweiterung ist der Grund für die Art, wie herum in Schaltung 2 das Relais eingezeichnet wurde.

Das erste mir bekannte Modell mit einer solchen Steuerung ist der *Bohrautomat* aus dem hobby-3-Begleitbuch 3-2:

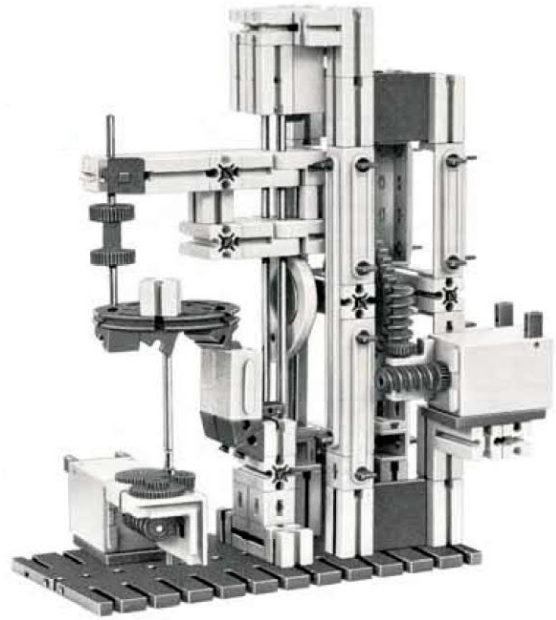


Abb. 1: Modell „Bohrautomat“ aus dem hobby-Begleitbuch 3-2 von 1973

Der linke Motor dreht eine fischertechnik-Drehscheibe mit Werkstücken (BS15 darauf, im Bild ist nur einer angebracht), der rechte Motor lässt den „Bohrer“ herunter ins Werkstück und wieder nach oben herausfahren. Die Schaltung bewirkt, dass immer genau einer der Motoren läuft und der andere stillsteht.

Funktionsmodell mit Relais

Unter Verwendung eines beliebigen fischertechnik-Relais oder des ft:pedia-Selbstbau-Relais aus [1] können wir das einfache Funktionsmodell aus Abb. 2 bauen und nach Schaltung 2 oder Schaltung 3 verdrahten.

Jeder Motor dreht sein Z40, und deren drei Zapfen betätigen je einen Taster. Das Relais wird damit angesteuert, sodass ein Motor dann läuft, wenn beide Taster *denselben* Zustand haben (beide gedrückt oder beide nicht gedrückt, das nennt man die *Identität*), während der andere läuft, wenn beide Taster einen *unterschiedlichen* Zustand haben (einer gedrückt und einer nicht, das ist das „exklusive ‚Oder‘“, „exclusive OR“ oder kurz „XOR“).

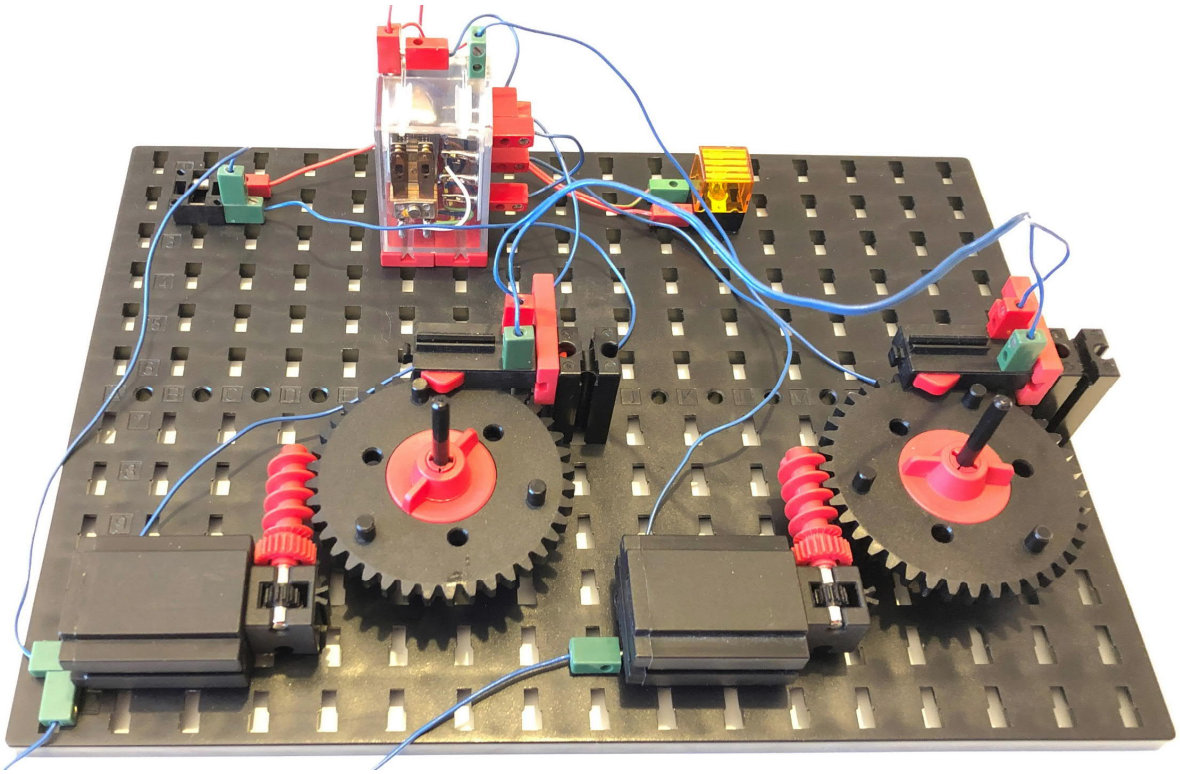


Abb. 2: Funktionsmodell mit Z40 und em5-Relais

Wenn man die Motoren hinreichend langsam laufen lässt, erkennt man, dass der sich endlos wiederholende Gesamtablauf aus vier Teilschritten besteht. Wir fangen mit einem willkürlich gewählten an:

1. Beide Taster nicht gedrückt. Der Motor *I*, der bei Identität der Taster läuft, dreht sein Z40, bis sein Taster betätigt wird.
2. Der erste Taster wird vom Zapfen seines Z40 gedrückt. Motor *I* stoppt und der Motor *X*, der beim XOR der beiden Taster läuft, läuft, bis auch sein Taster von einem Zapfen betätigt wird.
3. Da beide Taster wieder denselben Zustand haben (beide sind ja jetzt betätigt), stoppt Motor *X* wieder und Motor *I* läuft, bis der Zapfen von seinem Taster entfernt ist und den Taster loslässt. Das ist nur ein kurzer Moment, denn der Zapfen ist ja viel kleiner als der freie Raum zwischen zwei Zapfen.
4. Nun haben beide Taster wieder einen anderen Zustand – das XOR schlägt an, Motor *I* stoppt und Motor *X* läuft, bis auch sein Zapfen seinen Taster freigibt. Dann haben wir wieder den Zustand wie in 1. und das Spiel beginnt von neuem.

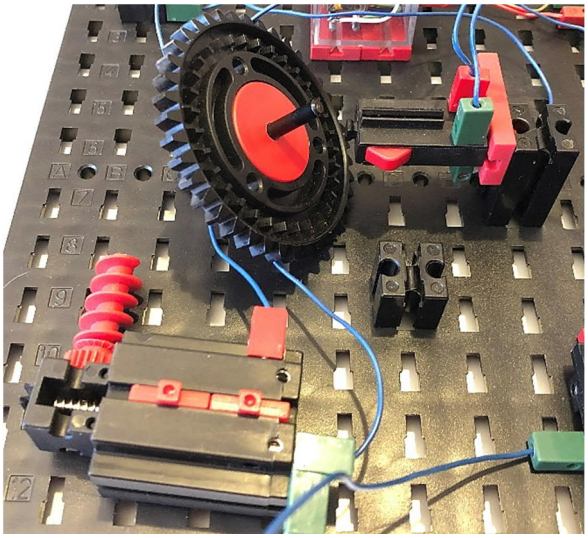


Abb. 3: Zum Aufbau des Z40-Modells

Wechselschaltung mit einem zusätzlichen Taster

Jetzt kommt Trick 17: Das geht auch ohne Relais! Wir brauchen lediglich einen dritten Taster anstelle eines Relais, allerdings verlieren wir bei dieser Spar-Schaltung die Kurzschlussbremse. Die folgende Schaltung erfüllt aber prinzipiell denselben Zweck.

Der wichtige Punkt ist, dass zwei der drei Taster direkt mechanisch gekoppelt werden. Das geht mit den fischertechnik-Minitastern ganz wunderbar, denn ihr roter Stößel wirkt durchgehend und kann einen weiteren, direkt dahinter befindlichen Taster mit betätigen.

Das erste Modell mit den Z40 ist extra so gebaut, dass ihr leicht einen zweiten Taster anbringen könnt. Damit beide zuverlässig hinreichend gleichzeitig betätigt werden, empfiehlt es sich, die Zapfen auf Stößelseite z. B. mit einer Platte 5·15·30 3N (38428) zu fixieren. Ihr könnt Schaltung 4 also leicht im Z40-Modell ohne Relais ausprobieren.

Funktionsmodell ohne Relais

Nur zur Abwechslung sei hier eine weitere Konstruktionsart dargestellt:

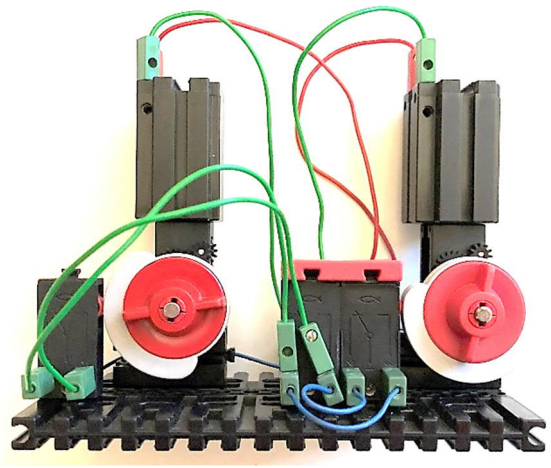


Abb: 4: Das Schaltscheiben-Modell von vorne

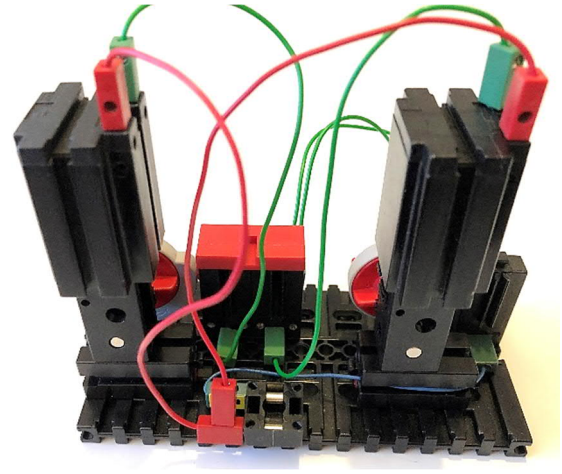
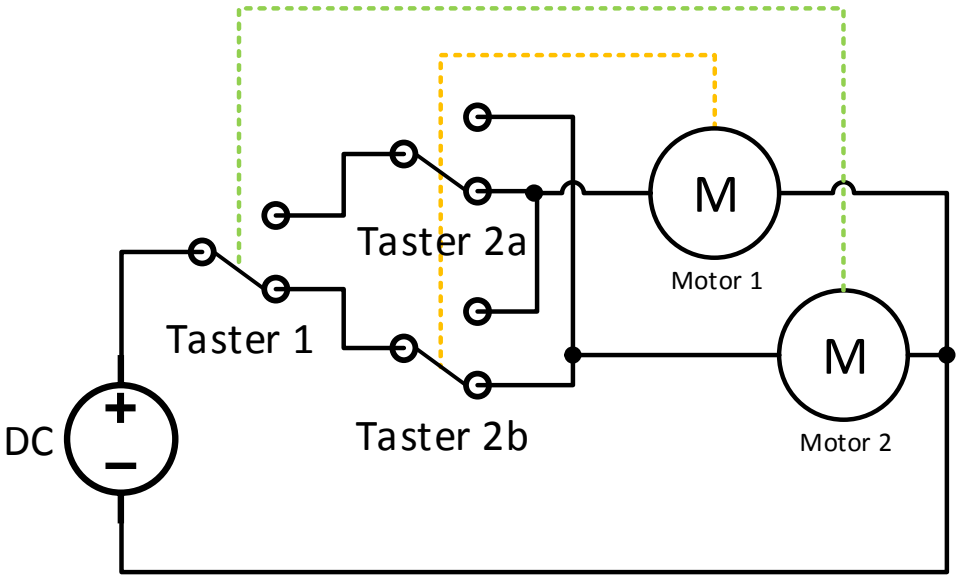


Abb. 5: Das Schaltscheiben-Modell von hinten



Schaltung 4: Wechselschaltung mit drei Tastern

Hier werden in jeder der beiden Naben je zwei Schaltscheiben (37728 grau oder 37727 schwarz) verwendet. Durch Verdrehen gegeneinander kann man den „offenen“ Winkel einstellen, bei dem der daran anliegende Taster unbetätigt bleibt. Die Schaltung ist einfach Schaltung 4 – mehr brauchen wir nicht!

Eine Stangenvorschub-Maschine

Was machen wir nun mit dieser Schaltung? Nun, wir können sie überall einsetzen, wo zwei Bewegungen immer abwechselnd vor sich gehen sollen. Anders als bei einer zeitlichen Programmsteuerung [4] können die vier Teilbewegungen aber beliebig und vor allem unterschiedlich lang dauern, denn es sind die Taster, die den jeweils nächsten Zustand einleiten.

Ein Beispiel ist der oben gezeigte Bohrautomat. Ein anderes ist die in Abb. 7 gezeigte Stangen-Vorschub-Maschine. Sie besteht aus den grün bezeichneten Teilen:

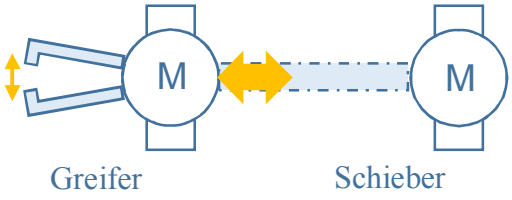


Abb. 6: Schema der Vorschub-Maschine

Es gibt einen motorbetriebenen Greifer, der die zu schiebende Stange festhalten oder loslassen kann. Dieser Greifer kann von einem zweiten Motor in Schubrichtung vor und zurück geschoben werden. Ein Schub nach links in Abb. 6/7 verläuft dann so:

1. Der Greifer liegt „hinten“ (in Abb. 6/7 wäre das also rechts, wenn die Stange nach links geschoben werden soll). Der Greifer packt zu und hält die zu schiebende Stange fest.

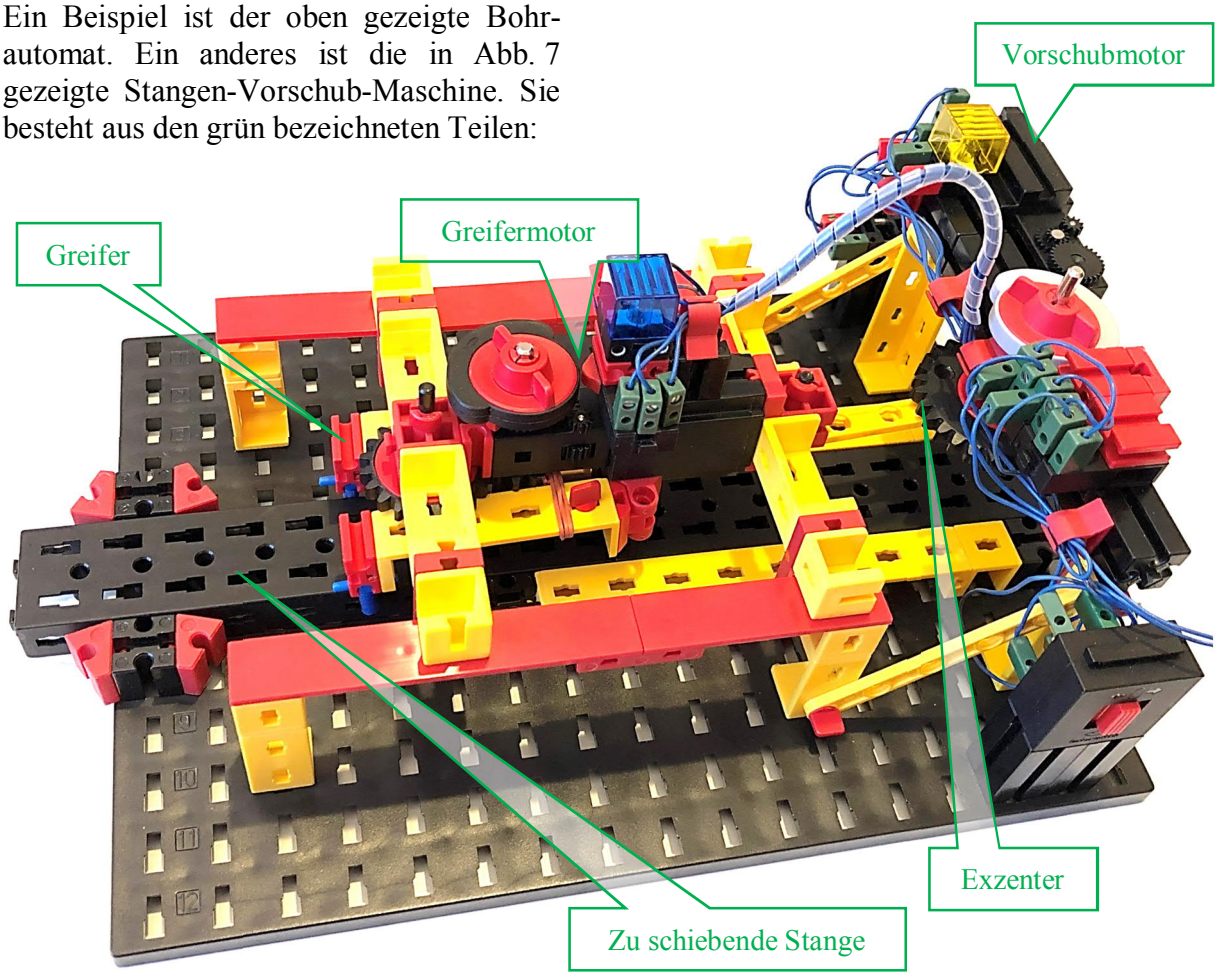


Abb. 7: Stangenvorschub

2. Der Schieber schiebt den Greifer nach „vorne“ (in Abb. 6/7 also nach links). Da der Greifer die Stange gerade festhält, wird die Stange also mit geschoben.
3. Der Greifer lässt los.
4. Der Schieber fährt zurück. Weiter bei 1, um die Stange immer weiter zu schieben.

Die zu schiebende Stange ist bei uns eine Reihe von U-Trägern 150. Der Greifer wirkt wie eine Zange, die durch Gummikraft zusammen und durch einen Motor auseinander gedrückt wird (der wird später noch genauer beschrieben). Der Vorschub geschieht über einen Exzenter.

Im Greifer ist ein Taster verbaut, beim Exzenter-Motor sind es zwei – zunächst mal ganz nach Schaltung 4. Der Ablauf kann dann z. B. so sein:

1. Der Greifer liegt gerade rechts im Modell und ist geöffnet. Er packt nun die Stange durch seinen Motor.
2. Sobald er fest zugegriffen hat, ändert sich der Zustand seines Tasters. Das setzt den Greifermotor still und startet den für den Vorschub, der die vom Greifer festgehaltene Stange nach links aus dem Modell schiebt.
3. Sobald der Greifer ganz nach vorne geschoben wurde, ändert sich der Zustand des Vorschubmotors. Also wird der nun ausgeschaltet und der Greifermotor startet. Der bewirkt damit das vollständige Loslassen der Stange.
4. Erst und sobald das erreicht ist, wird der Greifermotor wieder aus- und stattdessen der Vorschubmotor eingeschaltet. Der zieht über den Exzenter den Greifer in die rechte Position – weiter geht’s endlos ab 1.

Das Grundgerüst

Es beginnt alles auf einer Bauplatte 500. Mit wenigen BS15, Winkelbausteinen 30° und Statikträgern (120 und 15 mit zwei Zapfen) ist die Führung für die zu transportierende U-Träger-Schiene hergestellt. Achtet darauf, diese Führung nicht zu stramm zu bauen, sondern lasst den U-Trägern genügend Spiel, sodass sie leichtgängig geschoben werden können.

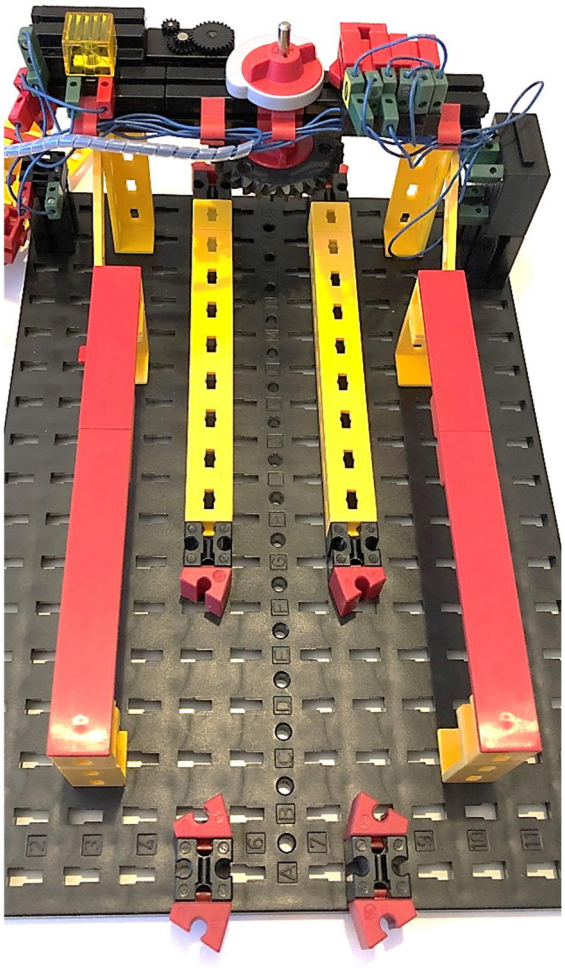


Abb. 8: Grundgerüst und Vorschubantrieb

Links und rechts nebendran entsteht aus Statikträgern 30 und 15 mit zwei Zapfen, Platten 15·90 und 15·60 sowie zur Verbindung derselben Bausteine 5·15·30 3N die Trägerschienen für den Schlitten mit dem Greifer.

Der Vorschub-Antrieb

Ganz am Ende sitzen Statikträger 60, auf denen einfach per Federnocken mittig eine Querstange aus BS15 mit Bohrung und links und rechts je zwei BS30 befestigt werden. Diese Stange steht links und rechts je 7,5 mm über die Statikträger hinaus.

Der S-Motor ist mit einem Verbinder 30 (mit einer abgeflachten Seite) an den Grundbausteinen angebracht. Er treibt über ein Rast-Z10 das Z30 für den Exzenter an. Der sitzt mit einer Rastachse und einer Riegelscheibe zwischen Z30 und Bausteinen als Abstandshalter auf derselben Metallachse, auf der obendrauf in einer Flachnabe die beiden Schaltscheiben sitzen. Die Riegelscheibe zwischen Z30 und Bausteinen sorgt für den richtigen Abstand, sodass das Z10 satt ins Z30 eingreift, ohne dass etwas hakt.

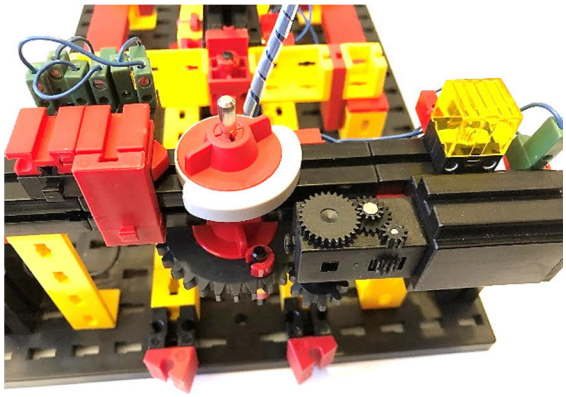


Abb. 9: Detailblick auf den Vorschub

Diese betätigen das Taster-Paar. Die beiden Taster sind einfach mit zwei BS7,5 mit Verbinder 15 miteinander gekoppelt, und der erste Taster ist mit einem Baustein 5·15·30 mit einer langen Nut und einem Federnocken an der Querstange befestigt.

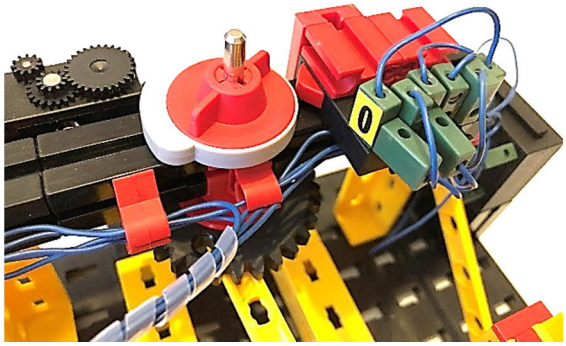


Abb. 10: Anordnung der beiden Taster beim Vorschub

Im Z30 steckt locker eine Achse 30 mit einer I-Strebe 75, oben und unten mit einem Klemmring gesichert. Die soll unseren Greifer vor und zurück schieben.

Der Greifer

Der Greifer verwendet ebenfalls einen S-Motor mit angebrachten Schaltscheiben, die hier aber nur einen einzigen Taster zu betätigen brauchen. In seiner oberen langen Nut steckt ein BS15, an dem per Federnocken ein BS7,5 sitzt, an dem der Taster befestigt ist.

An der Motor-Rückseite geht es weiter mit zwei Federnocken für einen BS 5·15·30 3N, in dessen mittlerer Nut ein BS7,5 mit Bohrung steckt. In die Bohrung kommt wieder eine Achse 30, die das andere Ende der Exzenter-Schubstange (der I-Strebe 75) aufnimmt.

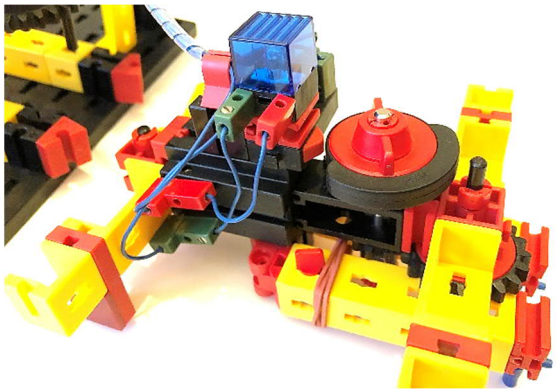


Abb. 11: Der Greifer-Schlitten

In den Seiten des BS15 mit Bohrung stecken zwei Tragarme, jeweils bestehend aus Statikträger 30, Baustein 5·15·30 und Statikträger 15, und zusätzlich sitzt auf dem nach unten ragenden Ende des roten Flachbausteins eine Platte 15·15. Die ist wichtig, damit der Schlitten zwar leichtgängig auf den Schienen geschoben werden kann, aber nicht zu viel Spiel in Querrichtung hat.

Vorne sitzen ein BS7,5 und wieder ein BS15 mit Bohrung. Der hält zwei gleich gebaute seitliche Träger. Unten wird vom Motor ein Rast-Z10 gedreht, und in der Achse des BS15 mit Bohrung sitzt eine Flachnabe mit einem Z20.

Ein normales Haushaltsgummi, einige Male verschlungen, zieht die zwei Arme zusammen, die über Gelenkbausteine an einem unten am Motor angebrachten BS15 zwei

Statikträger 60 beweglich halten. Die zwei S-Riegel mit Riegelscheibe sichern das Gummi dagegen, zu weit zu den Gelenken zu rutschen und also zu wenig zu ziehen.

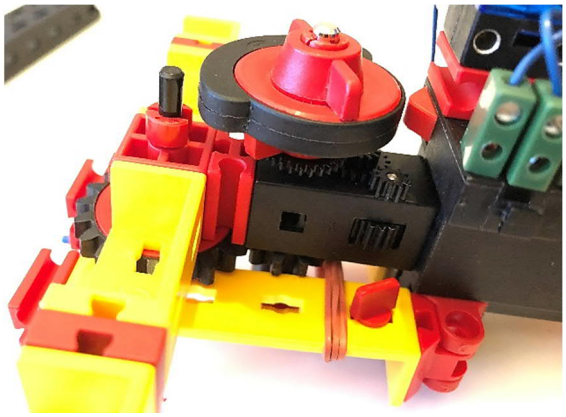


Abb. 12: Detailblick auf die Greifmechanik

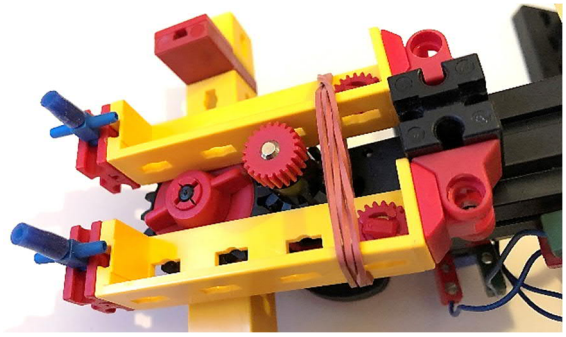


Abb. 13: Die Greifmechanik von unten

Ganz vorne an den Statikträgern 60 sitzen per Federnocke BS7,5, in denen je ein Pneumatik-T-Stück steckt. Darauf sitzen kurze (ca. 10 mm lange) Stücke Pneumatik-Schlauch, und so wird der Greifer wunderbar griffig – die zu bewegenden U-Träger rutschen nicht einfach durch.

Durch die Untersetzung von 2:1 zwischen Z10 und Z20 erreichen wir, dass pro halber Umdrehung der Schaltscheiben (also immer, wenn sich der Tasterzustand *ändert*) eine Vierteldrehung des Z20 mit der Flachnabe erfolgt.

Damit – so müsst ihr die Schaltscheiben justieren – steht mit jeder Halbdrehung der Schaltscheiben die Nabenmutter zwischen den Greifarmen mal längs (und lässt das Gummi die Greifarme zupacken) und mal quer (und drückt die Greifarme gegen die Gummikraft auseinander).

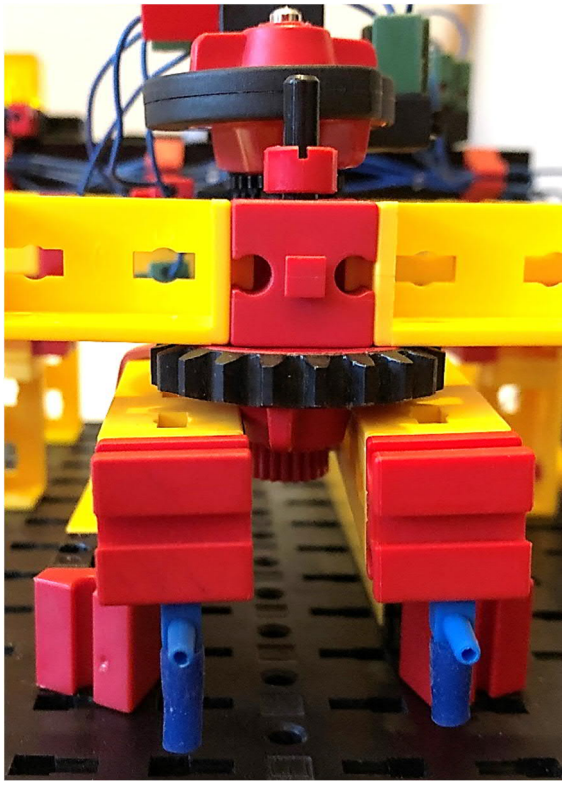


Abb. 14: Der Greifer ist geschlossen

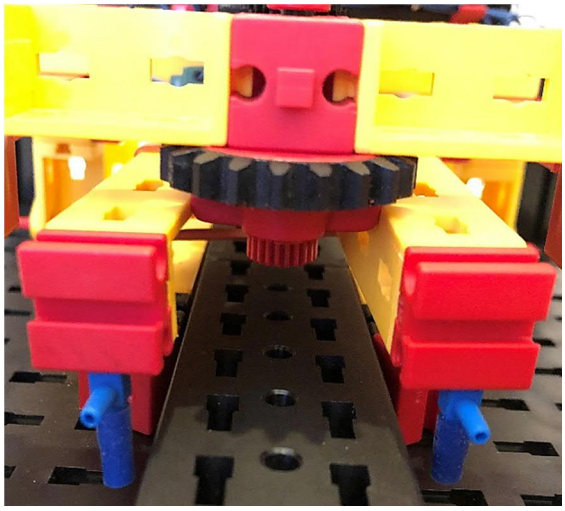
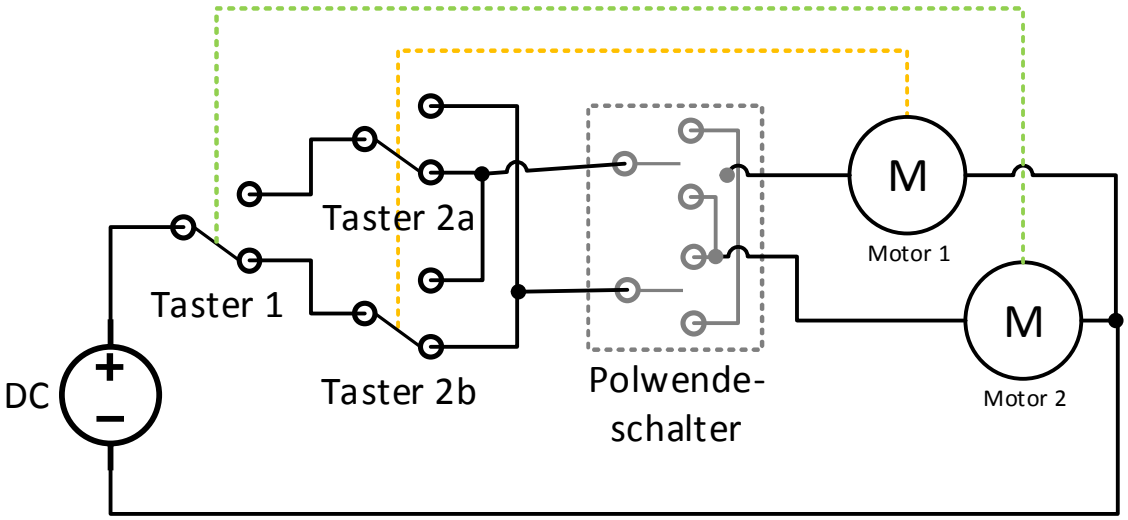


Abb. 15: Der Greifer wird durch die Nabenmutter geöffnet



Schaltung 5: Um einen Umschalter der Bewegungsrichtung ergänzte Schaltung 4

Funktionsweise

Wenn ihr alles einfach nach Schaltung 4 verkabelt, funktioniert das Modell wie folgt. Wir fangen in einem willkürlich gewählten Zustand an:

1. Der Greifer ist gerade geschlossen (greift zu) und der Schlitten steht ganz hinten. Einer der Motoren läuft – nehmen wir an, das sei der Vorschubmotor. Der dreht sein Z30 eine halbe Umdrehung und schiebt den Greiferschlitten also nach vorne. Eine festgehaltene U-Träger-Stange würde also nach vorne aus dem Modell herausgeschoben werden.
2. Nach der halben Umdrehung ändern sich seine Taster, der Vorschubmotor wird aus- und der Greifermotor eingeschaltet. Der dreht seine Schaltscheiben eine halbe Umdrehung und seine Z20-Nabenmutter also eine viertel Umdrehung – der Greifer öffnet sich. Dann ändert sich der Zustand des Tasters am Greifer.
3. Der Greifermotor geht also aus und der Vorschubmotor dreht wieder eine halbe Umdrehung. Also wird der geöffnete Greifer zurückgezogen, und zwar ohne die U-Träger zu bewegen. Bis die Taster des Vorschubs ihren Zustand wieder ändern…
4. Dann wird der Vorschubmotor aus- und der Greifermotor wieder eingeschaltet. Seine halbe Umdrehung bewirkt eine Vierteldrehung der Nabenmutter, die steht also wieder längs zum Schlitten und lässt die Greifarme zupacken. Das Spiel geht von vorne los, indem der Greifer in geschlossenem Zustand wieder nach vorne geschoben wird.

Umschaltbare Schubrichtung

Es könnte aber auch andersherum laufen: Je nachdem, ob der Greifer immer „hinten“ zupackt und vorne loslässt oder umgekehrt, wird die Stange in die eine oder die andere Richtung transportiert. Wenn wir die Schubrichtung gerne steuern möchten, so können wir das diesmal nicht dadurch tun, dass wir die Stromversorgung umpolen. Eine halbe Umdrehung des Exzenters im Vorschub bewirkt genau dasselbe, egal in welcher Richtung diese Drehung erfolgt, und analog gilt dasselbe auch für den Greifer.

Wir können aber die *Reihenfolge* ändern, wenn wir einen Polwendeschalter in Schaltung 4 einbauen und so zu Schaltung 5

kommen. Hier empfiehlt sich die aktuelle Version des Schalters (36708), weil der in der Mittelstellung das Modell auch gleich ganz ausschalten kann.

Wir vertauschen also nicht die Polung der Motore, sondern lediglich die Tatsache, welcher Motor bei Gleichstand aller Taster laufen soll und welcher bei Ungleichstand. Das allein ergibt den richtigen Ablauf für eine beliebig wählbare Transportrichtung der U-Träger-Stange.

Unter [5] findet sich auch ein Video der Maschine. Im nächsten Beitrag dieser Reihe werden wir eine ähnlich wirkende Schaltung elektronisch (mit einem E-Tec-Modul) realisieren.

Quellen

[1] Falk, Stefan: *Motorsteuerungen (Teil 4)*. ft:pedia 4/2011, S. 6-20.

[2] Falk, Stefan: *Motorsteuerungen (Teil 1)*. ft:pedia 1/2011, S. 4-8.

[3] fischertechnik: *Bohrautomat* in *Experimente + Modelle hobby 3 Band 2*. Fischer-Werke, Tumlingen, 1971, S. 56-62.

[4] Falk, Stefan: *Programmsteuerungen*. ft:pedia 1/2013, S. 4-19.

[5] Falk, Stefan: *Motorsteuerungen 5 – Schrittschaltwerk mit Wechselschaltung Mittel*. Youtube, 2019.

Computing

Neues von startIDE: Feldvariable, Servos, I2C

Peter Habermehl

Seit der in ft:pedia 4/2018 vorgestellten Version 1.5 von startIDE [1, 2] gab es zahlreiche Erweiterungen der Programmier-App. Aktuell ist sie in Version 1.7 im App-Store der community firmware verfügbar. Mit Version 1.6 wurden Feldvariable eingeführt, mit Version 1.6.6 die Ansteuerung von Servos, die über ein I2C-Shield an diverse Controller angeschlossen werden können, und mit Version 1.7 Funktion zum direkten Schreiben und Lesen auf dem I2C-Bus sowie einige mathematische Bitoperatoren. Die neuen Funktionen werden im Folgenden kurz vorgestellt.

Feldvariable

Die Einführung von Index- bzw. Feldvariablen gehörten zu den häufigsten Anwenderwünschen. Da die Erweiterung der vorhandenen Variablenverwaltung um indizierte Variable insbesondere wegen der problematischen Einbindung in das grafische Programmiersystem nicht praktikabel war, wurde ein dynamisches Datenregister geschaffen, das mit verschiedenen startIDE-Funktionen manipuliert werden kann.

Eine Feldvariable, in startIDE als Array bezeichnet, ist letztlich einfach eine Liste von Integer-Werten, auf die über ihren Positionsindex zugegriffen wird.

Mit dem Befehl `ArrayInit <Arrayname> <Initialbedatung>` wird ein Array in startIDE initialisiert. Die Initialbedatung ist optional; wird sie weggelassen, so ist das Array leer, enthält also keine Elemente.

Mit dem Befehl `Array <Variable> <Operator> <Array> <Index>` erfolgt der Zugriff auf die im Array abgelegten Daten. So kann mit `Array <Variable> readFrom <Array> <Index>` der Inhalt des Arrays an Position `<Index>` in die `<Variable>` übertragen werden.

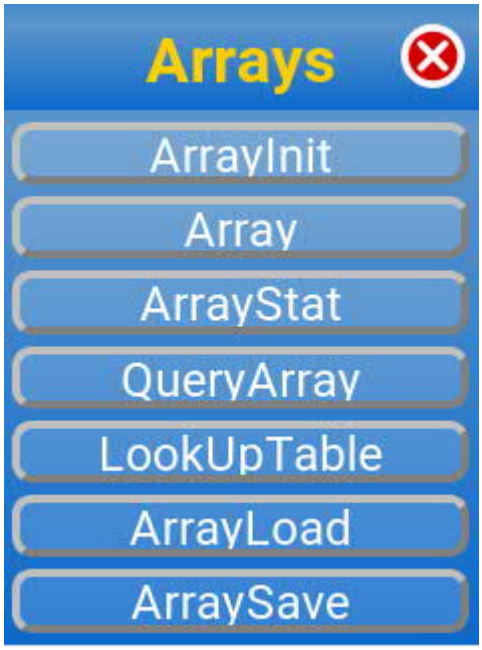


Abb. 1: die neuen Array-Befehle

Umgekehrt schreibt `Array <Variable> writeTo <Array> <Index>` den Inhalt der `<Variable>` an Position `<Index>` in das Array.

`Array <Variable> appendTo <Array> 0` hängt den Inhalt der Variable als neuen Eintrag an das Ende des Array an, unabhängig von einem eventuell angegebenen Index. Dabei wächst das Array um einen Eintrag.

`Array <Varfiable> insertTo <Array> <Index>` fügt den Inhalt der Variable an der Stelle `<Index>` in das Array ein, wobei der vorherige Inhalt und alle darauffolgenden Indizes um eine Stelle verschoben werden. Der Inhalt wird also nicht überschrieben und das Array wächst um einen Eintrag.

Umgekehrt entfernt `Array <Variable> removeFrom <Array> <Index>` den Eintrag an Position `<Index>` aus dem Array, nachdem der Wert in die `<Variable>` übernommen wurde. Die Größe des Arrays verringert sich dabei um einen Eintrag, alle Folgeeinträge werden entsprechend verschoben.

Array-Befehle:

Befehl: **ArrayInit myData**
Array: leer
Index: -

Befehl: **ArrayInit myData 17;10;72**
Array: 17 | 10 | 72
Index: 0 1 2

Befehl: **Array <Variable=42> appendTo myData 0**
Array: 17 | 10 | 72 | 42
Index: 0 1 2 3

Befehl: **Array <Variable=-5> insertTo myData 2**
Array: 17 | 10 | -5 | 72 | 42
Index: 0 1 2 3 4

Befehl: **Array <Variable> removeFrom myData 3**
Array: 17 | 10 | -5 | 42
Index: 0 1 2 3

Befehl: **Array <Variable=20> writeTo myData 1**
Array: 17 | 20 | -5 | 42
Index: 0 1 2 3

ArrayStat Ergebnisse:

sizeOf	min	max	minIdx	maxIdx	mean
0	-	-	-	-	-
3	10	72	1	2	33
4	10	72	1	2	35
5	-5	72	2	3	27
4	-5	42	2	3	16
4	-5	42	2	3	19

Abb. 2: Grundlegende Array-Funktionen

Mit der Funktion `ArrayStat <Variable> <Operator> <Array>` lassen sich statistische Informationen über ein Array gewinnen.

Zunächst liefert `ArrayStat <Variable> sizeOf <Array>` die Anzahl der Elemente des Arrays. `ArrayStat <Variable> min <Array>` ermittelt den kleinsten im Array enthaltenen Wert. Der Operator `minIdx` würde dazu den Index zurückliefern; analog funktionieren `max` und `maxIdx`.

Abb. 2 zeigt eine Abfolge von Array-Befehlen, ihre Auswirkung auf das Array und die mit der `ArrayStat`-Funktion abrufbaren Informationen.

Dabei steht `<Variable>` für eine startIDE-Variable, `<Variable=x>` für eine Variable mit dem aktuellen Wert `x`.

Um den Inhalt eines Arrays auf dem Display – oder, wenn aktiviert, in die Log-Datei – auszugeben, ist der Befehl `QueryArray <Array>` zu benutzen.

Über `LookUpTable` (Abb. 3) können Kennlinien durch Arrays definiert werden. Als Interpolationsmethoden stehen „nächster Nachbar“ und „linear“ zur Verfügung. Damit ist z. B. die Umrechnung des Widerstands eines NTCs in eine Temperatur möglich. Auch kann z. B. die Positionsvorgabe für einen Servo von Winkelgrad in das Ansteuer-Tastverhältnis umgerechnet werden. Dies ist in Listing 1 dargestellt.

Abb. 3: LookUpTable

Interessant bei dem Beispiel in Listing 1 ist die Begrenzung des Eingangswerts durch die Kennlinie. Der Servo-Ansteuerbereich ist von 145 Steps bei -90° bis 575 Steps bei +90° Drehwinkel definiert (Details dazu siehe weiter unten in Abschnitt 2, Servo-Ansteuerung).

Die Eingangskennlinie konvertiert alle Soll-Winkel von -10.000° bis -90° auf 145 Steps und alle Soll-Winkel oberhalb von 90° bis 10.000° auf 575 Steps. Dazwischen wird linear interpoliert. Im Beispiel ist der gewünschte Stellwinkel +35°, das Ansteuer-Tastverhältnis wird über das LookUpTable berechnet und mit dem Servo-Befehl an den Servo ausgegeben.

```
# Servo
Init stellwinkel 35
Init pwm_out 0
ArrayInit zielwinkel -10000;-90;90;10000
ArrayInit pwm_tv_out 145;145;575;575
#
LookUpTable pwm_out zielwinkel linear pwm_tv_out stellwinkel
Servo FTD S00 pwm_out
Delay 1000
```

Listing 1: Servo-Ansteuerung über Kennlinie

Ebenfalls auf Anwenderwunsch wurden die Befehle `ArrayLoad` und `ArraySave` ergänzt. Mit ihnen können Arrays auf der SD-Karte abgespeichert und wieder geladen werden.

Ein Zugriff auf die gespeicherten Daten ist auch über das Webinterface von startIDE möglich. So können z. B. Kalibrierwerte auf dem PC in eine Textdatei geschrieben und diese dann als Array auf den TXT/TX-Pi übertragen werden. Umgekehrt können z. B. Messwerte von startIDE in einem Array erfasst und dieses dann zur Offline-Auswertung abgespeichert und auf einen PC übertragen werden.

Eine Besonderheit dieser Funktionen ist, dass mit der Option `userSelect` während der Programmausführung der Benutzer die zu ladende bzw. speichernde Datei auswählen kann. Alternativ kann ein Array `byName` geladen werden, d. h. die abgespeicherte Datei muss den gleichen Namen haben wie das im startIDE-Code definierte Array.

Beim Abspeichern wird das Array, wenn nicht `userSelect` angegeben wurde, unter seinem Namen abgespeichert. `replace` bewirkt, dass eine evtl. vorhandene Datei gleichen Namens überschrieben wird, `rename` führt zu einer automatischen Umbenennung in der Form

`<Arrayname>_YYYYMMDD-HHMMSS.arr`

mit Datums- und Uhrzeitangabe. Dies ist insbesondere für Messdatenserien interessant.

Servo-Ansteuerung

Mit dem unter [5] vorgestellten und als 3D-Druck-Daten verfügbaren Mini-Servo-System (siehe auch [6]) entstand der Wunsch, diese Servos auch von startIDE aus ansteuern zu können. Dazu wurde der Funktionsgruppe `Ausgänge` der `Servo`-Befehl hinzugefügt.

Allerdings war es dazu auch erforderlich, die Servos hardwaremäßig anzubinden. Die Wahl fiel auf das Adafruit PMW Servo Shield bzw. einen günstig verfügbaren China-Clon dieses Shields [6, 7]. Diese Shields werden über das I2C-Protokoll angesprochen.

Auf dem TXT-Controller ist die direkte Ansteuerung des I2C-Busses ab der aktuellen Version 1.88 der ftrobopy-Library vom 4.1.2019 möglich [8]. Da diese noch nicht in die community firmware integriert wurde, wird der I2C-Bus des TXT z. Zt. noch nicht unterstützt.

Bereits implementiert und ansteuerbar sind der I2C-Bus des ftDuino sowie der servoDuino [6, 9]; der Anschluss des Servo Shields an den I2C-Bus des ftDuino ist gemäß des ftDuino-Handbuchs vorzunehmen, als Gehäuse mit integrierter Spannungsversorgung kann das servoShield-Design verwendet werden [10].

Der `Servo`-Befehl (Abb. 4) selbst hat drei Parameter, nämlich das Controller-Device, an dessen I2C-Bus das Servo Shield angeschlossen ist, z. Zt. also ftDuino FTD oder servoDuino SRD, den Port bzw. Kanal (0-15), an dem das Servo angeschlossen ist, und den PWM-Ansteuerwert in 12-bit-Auflösung von 0-4095.

Standard-Servos arbeiten üblicherweise in einem Bereich von 5% bis 12,5% Einschaltzeit, das bedeutet, dass der untere Anschlag des Servos bei

$$0{,}05 \cdot (2^{12} - 1) = 204{,}75$$

aufgerundet also bei 205 Schritten, der obere bei

$$0{,}125 \cdot (2^{12} - 1) = 511{,}85$$

aufgerundet also bei 512 Schritten liegt. Die Mittelstellung liegt dementsprechend zwischen den beiden Werten bei ungefähr 358 Schritten.

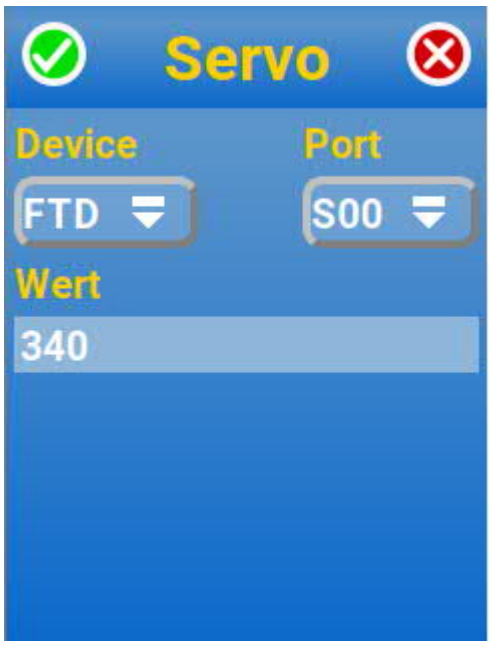


Abb. 4: Servo-Befehl

Weil startIDE allerdings den vollen PWM-Arbeitsbereich von 0-100% Tastverhältnis zulässt (entsprechend 0-4095 Schritte), ist darauf zu achten, den Servo nicht über seine mechanischen Grenzen hinaus anzusteuern, um eine Beschädigung zu vermeiden.

Mit dem dem startIDE-Release v1.7 beigefügten Projekt `servoTest` (Abb. 5, siehe auch das Video [11]) können Servos eingemessen werden, indem man manuell ihre Anschläge ansteuert. Der jeweilige Anschlag ist erreicht, wenn ein weiteres Senken bzw. Erhöhen des Ansteuerwertes keine weitere Positionsänderung des Servos mehr bewirkt.

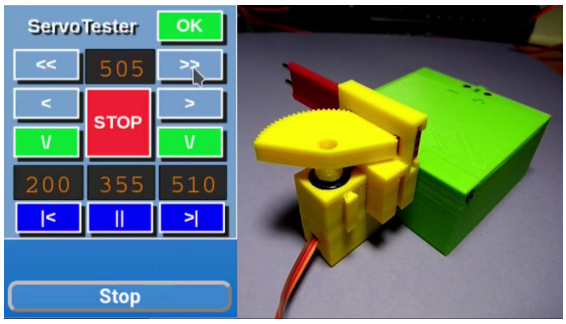


Abb. 5: servoTest [11]

Üblicherweise ist das Erreichen der mechanischen Endlage von einem Brummen bzw. Knurren des Servos begleitet. Der Ansteuerwert der unteren Endlage sollte um 5-10 Schritte vergrößert, der der oberen Endlage entsprechend verringert werden, damit der Arbeitsbereich des Servos sicher zwischen den mechanischen Anschlägen liegt. Alternativ kann man auch +/- 90° Servo-Position anfahren und die Ansteuer-Werte dazu ermitteln.

Die so ermittelten Ansteuer-Grenzen können in das Array `pwm_tv_out` aus Listing 1 übernommen werden.

I2C-Kommunikation

Nachdem startIDE intern zur Ansteuerung der Servos ohnehin um I2C-Funktionalitäten erweitert wurde, lag es nahe, auch dem Anwender Zugriff auf den I2C-Bus zu ermöglichen.

Daher gibt es nun in der Funktionsgruppe `Interaktion` den Unterpunkt `Communication`, der die beiden Befehle `I2CWrite` und `I2CRead` umfasst. `I2CWrite <Device> <Array>` schreibt den Inhalt des angegebenen `<Array>` als Nachricht auf den I2C-Bus des angegebenen `<Device>`, also z. Zt. ftDuino oder servoDuino. Es findet dabei keine Fehlerbehandlung statt.

Die Nachricht setzt sich dabei aus positiven 8-bit-Werten zusammen. Üblicherweise beginnt eine I2C-Botschaft mit der I2C-Adresse des Devices, gefolgt von der Adresse des angesprochenen Registers sowie evtl. Daten, die in das angegebene und die folgenden Register des I2C-Devices geschrieben werden sollen.

`I2CRead <Device> <Array>` liest Daten aus den Registern eines I2C-Bus-Geräts. Dabei wird im `<Array>` als erster Eintrag die Adresse des auszulesenden Devices angegeben, als zweiter Eintrag die Anzahl der zu lesenden Bytes. Nach erfolgter Ausführung des Befehls enthält das `<Array>` die gelesenen Daten, ebenfalls byteweise. Über `ArrayStat <Variable> sizeOf <Array>` lässt sich ermitteln, ob die gewünschte Anzahl Bytes gelesen wurde.

```
01 # Bosch BME280
02 Init byte 0
03 ArrayInit message 118;208
04 I2CWrite FTD message
05 #
06 ArrayInit message 118;1
07 I2CRead FTD message
08 Array byte readFrom message 0
09 QueryVar byte
```

Listing 2: I2C-Kommunikation

Ein Beispiel zur I2C-Kommunikation zeigt Listing 2. Dem am ftDuino angeschlossenen BME280-Umweltsensor von Bosch [12] mit der I2C-Adresse 76h (= 118dez) wird mit `I2CWrite` der Wert D0h (= 208dez) übermittelt, was in diesem Fall bedeutet, dass der interne Adresszeiger des Sensors auf das Register D0 gesetzt wird, in dem die Kennung dieses Sensors steht.

Weitere Daten werden nicht geschrieben, sondern es werden jetzt mit `I2CRead` ein Byte ab Adresse D0h gelesen und die gelesenen Daten in das Array `message` übertragen.

Der erste (und in diesem Beispiel auch einzige) Wert des Array wird in Zeile 8 in die Variable byte geschrieben und deren Inhalt auf dem Display ausgegeben. In diesem Beispiel würde die Zahl 96 (= 60h) zurückgegeben. Dies ist die interne Kennung des Sensors.

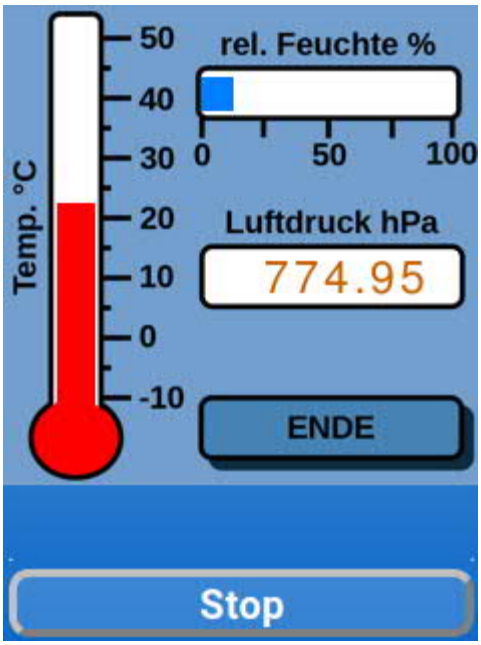


Abb. 6: Auswertung des I2C-Sensors BME280

Eine grafische Ausgabe der Sensorwerte des BME280 zeigt Abb. 6. Der gesamte Rahmen der Balken- und Numerik-Anzeigen wurde als PNG-Grafik geladen (siehe `Canvas load` bei den weiteren Neuerungen unten). Lediglich die farbigen Balken und die Ziffern werden live mit den startIDE-Grafikfunktionen gezeichnet.[1]

Weitere Neuerungen

Um die Kommunikation auf dem I2C-Bus etwas zu erleichtern, wurde der `Calc`-Befehl um einige Operatoren erweitert:

- Der `sign`-Operator wandelt die als `Operand1` übergebene Zahl anhand ihrer Binärdarstellung in eine `Operand2`-bit große, vorzeichenbehaftete Zahl um.
- Der `unsign`-Operator arbeitet umgekehrt zum `sign`-Operator und wandelt die als `Operand1` übergebene Zahl in eine `Operand2`-bit große positive Zahl um.
- Die Operatoren `bitAnd`, `bitOr` und `bitXOr` liefern entsprechend eine bitweise Und-, Oder- bzw. Exclusive-Oder-Verknüpfung der beiden Operanden zurück.
- Der `bitShift`-Operator verschiebt die Bits von `Operand1` um `Operand2` Stellen nach links. Ein negativer `Operand2` führt zu einem Bitshift nach rechts.

Beispielsweise liefert

```
Calc x 255 sign 8
```

in der Variablen `x` den Wert -1 zurück.

Der `bitShift`-Operator

```
Calc x 255 bitShift 8
```

liefert 65280 zurück:

```
255dez = 1111 1111b
1111 1111b << 8dez
  = 1111 1111 0000 0000b
  = 65280dez
```

Eine kleine, aber recht hilfreiche Neuerung betrifft die Grafikfunktionen. Die Leinwand-Befehle wurden um `Canvas load` erweitert. Damit ist es möglich, PNG-Dateien, die zuvor per Webinterface auf den TXT/TX-Pi übertragen wurden, in die Grafik-Leinwand zu laden und dann mit den startIDE-Grafikfunktionen weiter zu bearbeiten. So können z. B. grafische Benutzerschnittstellen einfach mit einem Grafikprogramm gezeichnet werden (als Beispiel dafür siehe das `servoTest`-Projekt oben).

Schließlich wurde die Funktion `FromSys` noch um einige Datenquellen erweitert: `FromSys <Variable> aktXPos` liefert die aktuelle `x`-Position des Mauszeigers bzw. die Berührungskoordinate auf der Grafikleinwand (Canvas) zurück, als Entsprechung zum bereits vorhandenen Befehl `touchXPos`, der die Koordinate bei Auftreten des Berührungs-/Loslassereignisses liefert. `FromSys <Variable> aktYPos` liefert entsprechend die `y`-Position. Und schließlich kann mit `FromSys <Variable> touch` der aktuelle Berührungsstatus (0:

[1] Der Luftdruckwert ist korrekt, dieser Artikel ist auf einer Höhe von ca. 2000 m.ü.M. entstanden.

keine Berührung; 1: Berührung) der Canvas abgefragt werden.

Ein Beispiel zur Verwendung dieser neuen Datenquellen bringt startIDE mit dem Projekt „microPaint“ mit (Abb. 9).

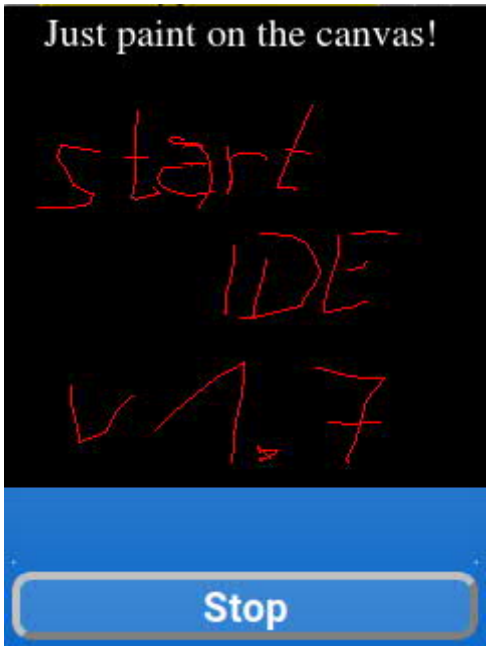


Abb. 9: microPaint – Malen auf Touchscreen

Quellen

[1] Peter Habermehl: *startIDE für die Community Firmware – Programmieren direkt auf dem TXT oder TX-Pi*. ft:pedia 1/2018, S. 102-107.

[2] Peter Habermehl: *Grafik auf dem TXT mit startIDE*. ft:pedia 3/2018, S. 37-40.

[3] Peter Habermehl: *Einbindung des TXT- bzw. TX-Pi-Touchscreens in startIDE*. ft:pedia 4/2018, S. 49-52.

[4] Peter Habermehl: *startIDE Handbuch v1.7*. GitHub.

[5] Jan: *fischertechnik SG90 mini servo system*. Thingiverse, 15.09.2018, und zugehöriger Thread im ftc-Forum.

[6] Peter Habermehl: *Servo-Ansteuerung mit servoShield und servoDuino*. ft:pedia 1/2019, in dieser Ausgabe.

[7] Dirk Fox: *I²C mit dem TX(T) – Teil 16: Servo-Driver*. ft:pedia 2/2017, S. 41-47.

[8] Torsten Stuehn: *fischertechnik TXT Python*. GitHub. Siehe auch: Torsten Stuehn: *Programmierung des TXT unter Python*. ft:pedia 2/2017, S. 58-62.

[9] Peter Habermehl: *servoDuino: An Arduino sketch to serve as an USB-I2C-bridge for PCA9685 servo shields*. GitHub.

[10] Peter Habermehl: *fischertechnik I2C servo shield case*. Thingiverse, 27.12.2018.

[11] LeadCalibrator: *startIDE mit Servos*. YouTube, 21.12.2018.

[12] Bosch BME280 Manual: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280-DS002.pdf. Version 1.6, 20.09.2018.

[13] startIDE-Thread im ftc-Forum.

Computing

startIDE (6): Sonar

Rolf Meingast

Ein drehbarer Ultraschallsensor (Sonar) soll seine Umgebung erkunden und die Ergebnisse auf dem Bildschirm anzeigen [1]. Dieser Beitrag beschäftigt sich mit dem Aufbau und dem Programm zur Steuerung, das auf dem TXT mit der App startIDE läuft [2].

Aufbau des Sonars

Der Ultraschallsensor ist mit dem TXT verbunden und direkt auf der Achse eines Mini-Servos befestigt [3].

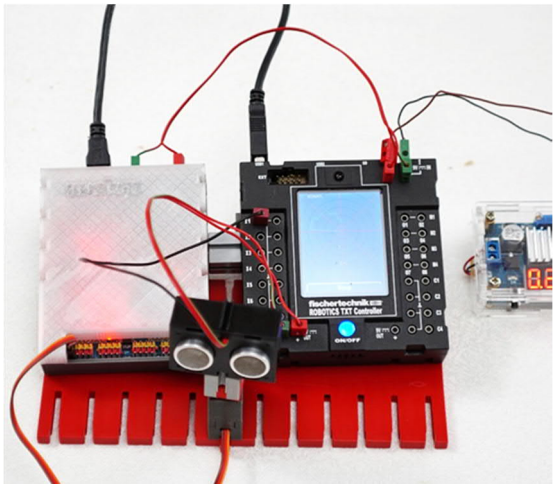


Abb. 1: Aufbau des Sonars

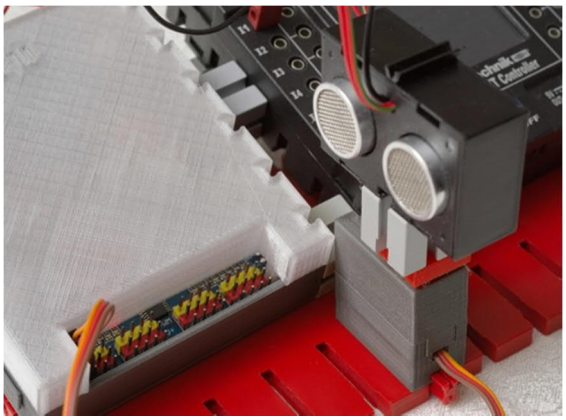


Abb. 2: Anschluss des Mini-Servo am servoDuino [4]

Der Servo ist am servoDuino [4] angeschlossen, der über USB mit dem TXT verbunden ist (Abb. 1, 2).

Anmerkungen zum Programm

Der maximale Drehwinkel des Servos beträgt 180°. In den Arrays `d_alpha.arr` und `Servo-n.arr` sind Winkel (in 1/10°) und Steuerwerte für den Servo gespeichert:

```
0;70;130;195;245;290;…
130;140;150;160;170;180;…
```

Diese Werte wurden mit der App zum Einmessen ermittelt [5].

Zu Beginn des Programms werden diese Arrays initialisiert und geladen.

Nach dem Start muss man entscheiden, ob ein Testlauf oder ein Dauerlauf erfolgen soll. Dann erfolgt die Eingabe des gewünschten Drehwinkels, der Schrittweite und der Pausendauer zum Betrachten des Ergebnisses auf dem Display des TXT.

Abb. 3 zeigt das Ergebnis eines Durchlaufs für einen Drehwinkel von 160°. Die freien Bereiche sind grün gekennzeichnet.

Die Achsen, Kreise und Beschriftungen werden im Modul Sonarschirm erzeugt, das die Sonargrafik zeichnet.

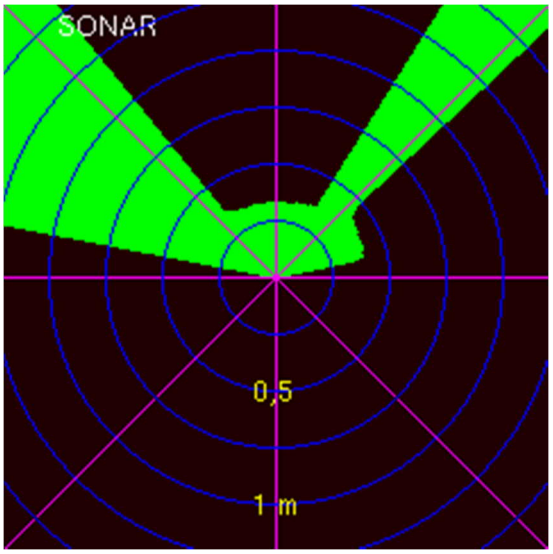


Abb. 3: Darstellung als Sonarschirm

Referenzen

[1] Dirk Fox: *Radar und Sonar*. ft:pedia 2/2011, S. 4-8.

[2] Peter Habermehl: *startIDE für die Community Firmware – Programmieren direkt auf dem TXT oder TX-Pi*. ft:pedia 1/2018, S. 102-107.

[3] Jan: *fischertechnik SG90 mini servo system*. Thingiverse, 15.09.2018, und zugehöriger Thread im ftc-Forum.

[4] Peter Habermehl: *servoDuino: An Arduino sketch to serve as an USB-I2C-bridge for PCA9685 servo shields*. GitHub.

[5] LeadCalibrator: *startIDE mit Servos*. YouTube, 21.12.2018.

Programm

```
# SonarTXTS31
# Januar 2019
Log Clear
ArrayInit d_alpha
ArrayLoad d_alpha byName
ArrayInit Servo-n
ArrayLoad Servo-n byName
Init z 300
Init Vorzeichen -1
Init msecPause 8000
Init Winkel 180
Init Startwinkel 0
Init Endwinkel 180
Init Schrittweite 1
Init dSchrittweite 10
Init Schritte 180
Init menu 0
Init dauerlauf 0
Init alpha 0
Init x 0
Init y 0
Init j 0
Init k 0
Init r 100
#
# Hauptmenue
#
Tag mainloop
FromButtons menu Testlauf
Dauerlauf STOP
IfVar menu == 1 test
IfVar menu == 2 Dauerlauf
Stop
#
Tag Dauerlauf
Init dauerlauf 1
#
Tag test
#
FromDial Winkel 1 180  Drehwinkel
FromDial Schrittweite 1 10
Schrittweite
FromDial msecPause 0 20000  msec
Calc Startwinkel 180 - Winkel
Calc Startwinkel Startwinkel / 2
Calc Endwinkel 180 - Startwinkel
Calc Schritte Winkel /
Schrittweite
Calc dSchrittweite Schrittweite *
10
#
Tag dauernd
#
Log Clear
Log 1
#
# Start Position
```

```
#
Tag start
Init dalpha Startwinkel
Calc dalpha dalpha * 10
LookUpTable z d_alpha linear
Servo-n dalpha
Servo SRD S00 z
#
Canvas show
Canvas clear
Call Sonarschirm 1
# Sonarmitte
Pen plot 120 120
Pen areaAdd 120 120
Call zeichnen
#
#
Tag neuePosition
Calc dalpha dalpha + dSchrittweite
LookUpTable z d_alpha linear
Servo-n dalpha
Servo SRD S00 z
Call zeichnen
LoopTo neuePosition Schritte
#
#
Pen areaDraw k j
Call Sonarschirm 1
Canvas log
Call Pause
IfVar dauerlauf == 1 dauernd
Jump mainloop
#
Module zeichnen
Calc alpha dalpha / 10
# Entfernung messen
FromIn TXT 1 D r
Delay 10
Calc r r * 10
Calc x r cos alpha
Calc x x / 10
Calc k x + 120
Calc y r sin alpha
Calc y y / 10
Calc j 120 - y
Color pen 0 255 0
Pen lineTo k j
Pen areaAdd k j
Canvas update
MEnd
#
Module Pause
TimerClear
Tag Zeit
IfTimer < msecPause Zeit
MEnd
#
# Sonarschirm
#
Module Sonarschirm
Color pen 255 255 255
Color paper 33 0 0
Text Helvetica 12 SONAR
Pen text 24 14
Canvas update
# Koordinatensystem
Color pen 255 0 255
Pen move 0 0
Pen lineTo 240 240
Pen move 120 0
Pen lineTo 120 240
Pen move 240 0
Pen lineTo 0 240
Pen move 0 120
Pen lineTo 240 120
Color pen 255 255 255
Color paper 33 0 0
Color pen 0 0 255
Pen move 95 95
Pen circleTo 145 145
Pen move 70 70
Pen circleTo 170 170
Pen move 45 45
Pen circleTo 195 195
Pen move 20 20
Pen circleTo 220 220
Pen move -5 -5
Pen circleTo 245 245
Pen move -30 -30
Pen circleTo 270 270
Color pen 255 255 0
Text Helvetica 12 0,5
Pen text 110 175
Text Helvetica 12 1 m
Pen text 110 225
Canvas update
MEnd
```

Computing

startIDE (7): Psychrometer

Rolf Meingast

Dieser Beitrag beschäftigt sich mit dem Aufbau eines fischertechnik-Psychrometers und dem Programm zur Steuerung, das auf dem TXT oder dem TX-Pi mit der App startIDE läuft [1].

Hintergrund

Unter einem *Psychrometer*, vorgestellt von *Ernst Ferdinand August* (1795-1870) im Jahr 1825, verbessert von *Richard Aßmann* (1845-1918) 1892, versteht man ein Gerät zur Bestimmung der relativen Luftfeuchtigkeit.

Zwei Thermometer, Th1 und Th2, von denen das zweite um sein Quecksilbergefäß eine Hülle aus leichtem Gewebe trägt, befinden sich in einem Luftstrom. Dieser Luftstrom streicht an beiden Thermometergefäßen vorbei. Benetzt man die Hülle am zweiten Thermometer mit Wasser von Raumtemperatur, so verdunstet das Wasser umso schneller in die vorbeiströmende Luft, je trockener diese ist.

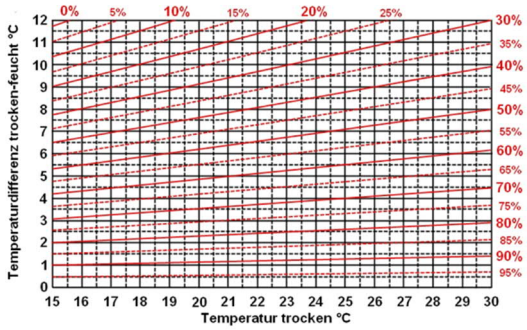


Abb. 1: Psychrometerdiagramm

Infolge der dabei auftretenden Verdunstungskälte kühlt sich Th2 ab. Man kann aus dem Temperaturunterschied zwischen dem trockenen und dem feuchten auf die Feuchtigkeit der Luft schließen. Meist entnimmt man zur abgelesenen Temperaturdifferenz in Abhängigkeit von der Raumtemperatur den Wert der relativen Luftfeuchte aus einer Tabelle oder einem Diagramm, z. B. dem Psychrometerdiagramm in Abb. 1.

Aufbau des Psychrometers

Als Thermometer verwende ich zwei NTC10K Temperaturfühler.[2] Der mit einem feuchten Papiertuch umwickelte befindet sich in einem kleinen Metallzylinder und ist am Eingang I1 des ftDuino [2] (oder eines TXT) angeschlossen, der zweite an I2.

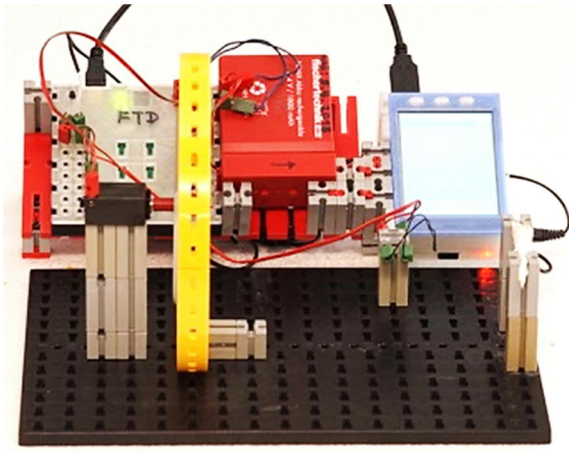


Abb. 2: fischertechnik-Psychrometer

[2] Diese beiden Fühler hatte ich zufällig; natürlich kann man auch zwei beliebige andere Temperaturfühler verwenden.

Der Motor mit Luftschraube wird an den Motorausgang M1 angeschlossen.

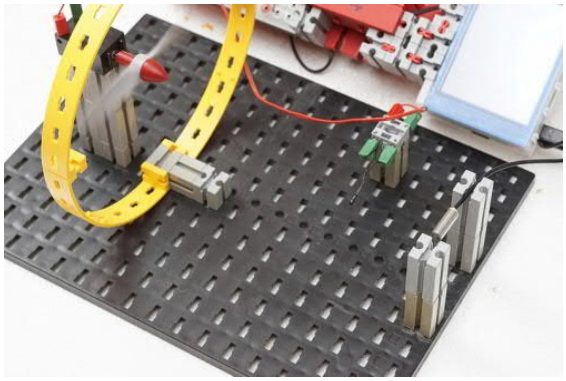


Abb. 3: Ventilator und Temperaturfühler

Durchführung der Messung

Das Papiertuch um Th2 wird befeuchtet, der Ventilator eingeschaltet und die beiden Temperaturen T1 und T2 abgelesen. Die Temperatur T2 sinkt nach einigen Minuten auf einen kleinsten Wert T2min.

Wenn nicht mehr genügend Wasser verdunstet, steigt die Temperatur T2 wieder an. Anhand der Werte von T1 und der Temperaturdifferenz T1-T2min kann aus dem Psychrometerdiagramm die relative Luftfeuchte entnommen werden. Für Umgebungstemperaturen in der Nähe von 22°C wird die Luftfeuchte direkt auf dem Display angezeigt.

Temperaturmessung

Die Temperaturmessung mit einem NTC10k erfolgt durch Messung des elektrischen Widerstands R. Anhand einer T(R)-Formel kann daraus die Temperatur ermittelt werden.

Ich habe mich für einen anderen Weg entschieden. Beide Heißleiter wurden zusammen mit einem Thermometer in eiskaltes Wasser gehalten, die Widerstände R1 und R2 und die Temperatur T bestimmt. Diese Werte R1, R2 und T wurden in den Arrays `data_R1.arr`, `data_R2.arr` und `data_T12.arr` gespeichert. Durch Zugabe von heißem Wasser wurde die Temperatur dann schrittweise erhöht. Zur Erhöhung der Genauigkeit wurde die Temperatur in 1/10°C eingegeben.

Hängt man die nächsten Daten an die ersten Daten (`append`) nehmen mit steigender Temperatur die Widerstände ab. Um aus gemessenen Widerstandswerten im Psychrometerprogramm die Temperaturen zu ermitteln, müssen die Widerstandswerte im R-Array aber stetig steigen. Deshalb werden zunächst Startwerte in die Arrays eingetragen und die folgenden Werte davor (Position 0) eingefügt. Zum Schluss werden die Startwerte, die sich ja dann am Array-Ende befinden, wieder entfernt.

Programm zur Array-Erzeugung

```
# new
# ARR_dgrad_gen23
# Kalibrieren von
# zwei NTC-Widerstaenden
# an I1 und I2 des FTD
Message ARR_dgrad_gen12'Okay
# Eingabe der Widerstands-
# werte in dezi °C
# dies muss bei der
# Auswertung beachtet werden!
Print Temperatur in 1/10 °C
Delay 2000
# Wenn sich die Werte R1 und R2
# nicht mehr aendern:
Print R_ENDE:Taste TX-Pi
Delay 2000
# Ende der Eingabe
Print EingabeENDE:T>1000
Delay 2000
Init integer 0
Init T_i 0
Init R1_i 0
Init R2_i 0
ArrayInit data_T12
ArrayInit data_R1
ArrayInit data_R2
# in die Arrays werden
# Nullen als erste
# Elemente geschrieben
Init i 0
Array T_i appendTo data_T12 0
Array R1_i appendTo data_R1 0
Array R2_i appendTo data_R2 0
# Eingabewiederholung
Tag A
Clear
FromIn FTD 1 R R1_i
Delay 100
FromIn FTD 2 R R2_i
```

```
# Arrayindex i
QueryVar i
# Anzeige beider
# Widerstandswerte
QueryVar R1_i
QueryVar R2_i
Delay 100
FromSys integer dispBtn
Delay 100
# Widerstandmessung Wdh.,
# wenn keine Taste des
# TX-Pi gedrueckt wird.
IfVar integer == 0 A
# Temperatureingabe
# in dezigrad
FromKeypad T_i -250 32768
IfVar T_i > 1000 E
# T und R-Werte werden
# in die Arrays an den
# Anfang eingefuegt
Array T_i insertTo data_T12 0
Array R1_i insertTo data_R1 0
Array R2_i insertTo data_R2 0
Calc i i + 1
Message neue Temperatur
einstellen!'Okay
Jump A
# Ende der Eingabe und
# Speichern der Arrays
Tag E
ArrayStat i sizeOf data_T12
Calc i i - 1
Array integer removeFrom data_T12
i
Array Integer removeFrom data_R1 i
Array Integer removeFrom data_T12
i
ArraySave data_T12 replace
ArraySave data_R1 replace
ArraySave data_R2 replace
```

Listing 1: Erzeugung der Arrays

Zusätzlich wurden „per Hand“ aus dem Psychrometerdiagramm Arrays mit Werten der relativen Feuchtigkeit (data_rLF.arr) und Temperaturdifferenz in dGrad (data_dT .arr) bei Raumtemperatur (22°C) erzeugt:

```
100;96;92;87;83;80;76;72;68;65;61;
57;54;50;47;44;40;38;35;31;28;25;1
5;

0;5;10;15;20;25;30;35;40;45;50;55;
60;65;70;75;80;85;90;95;100;105;12
5;
```

Diese beiden Arrays wurden über das Webinterface von startIDE mit einem Internetbrowser auf den TX-Pi geladen.

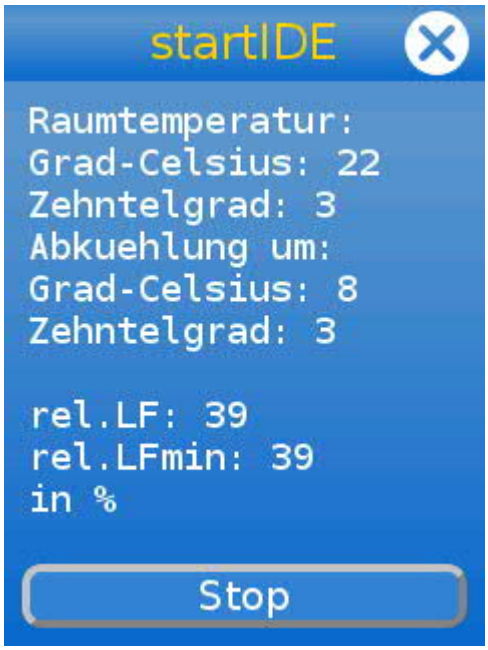


Abb. 4: startIDE-Ausgabebildschirm

Messprogramm (mit Kommentaren)

```
# new
# Hygrometer-36
# 20181224
Print Psychrometer
Print mit TX-Pi
Print und FTD
Delay 2000
Clear
ArrayInit data_T12
ArrayInit data_R1
ArrayInit data_R2
ArrayInit data_dT
ArrayInit data_rLF
#
# Laden der Arrays
# NTC10k Daten
# erzeugt mit
# ARR_dgrad_gen23
# Temperatur in
# deziCelsius Grad!
ArrayLoad data_T12 byName
ArrayLoad data_R1 byName
ArrayLoad data_R2 byName
# PsychrometerDaten
# bei 22°C
ArrayLoad data_dT byName
ArrayLoad data_rLF byName
#
# Ventilator ein
#
Motor FTD 1 r 255
#
# Beginn der Messungen
#
```

```
Init Grad-Celsius 0
Init Zehntelgrad 0
Init i 0
Init korr 0
Init R_mess 0
Init Temp 0
Init Temp1 0
Init Temp2 0
Init rel.LF 100
Init deltaTemp 0
Init rel.LFmin 100
#
Tag Messungen
#
# NTC #1 an I1 des FTD
#
Init HL 1
# Widerstandsmessung an I1
FromIn FTD 1 R R_mess
# aus R_mess wird die
# Temperatur Temp
LookUpTable Temp data_R1 linear
data_T12 R_mess
# Temp1:= Dezigrad an I1:
Init Temp1 Temp
#
# NTC #2 an I2 des FTD
#
Init HL 2
# Raumtemperatur in
# dezigrad
# Widerstandsmessung an I2
FromIn FTD 2 R R_mess
LookUpTable Temp data_R2 linear
data_T12 R_mess
# Temp2:= Temp
Init Temp2 Temp
Print Raumtemperatur:
Call TEMP-Ausgabe 1
#
# Berechnung der
Temperaturdifferenz
#
Calc deltaTemp Temp2 - Temp1
IfVar deltaTemp <= 0 O
Init Temp deltaTemp
Print Abkuehlung um:
Call TEMP-Ausgabe 1
#
# Berechnung der
# rel. Luftfeuchte
# Berücksichtigung der
# Abweichung von 22°C
Calc korr Temp2 - 220
Calc korr korr * 10
Calc korr korr + 25
Calc korr korr / 50
Calc deltaTemp deltaTemp - korr
IfVar deltaTemp < 0 O
LookUpTable rel.LF data_dT linear
data_rLF deltaTemp
IfVar rel.LFmin < rel.LF M
Init rel.LFmin rel.LF
Tag M
Print
QueryVar rel.LF
Tag O
QueryVar rel.LFmin
Print in %
#
Tag weiter
#
Delay 2000
Clear
Jump Messungen
#
#
Module TEMP-Ausgabe
#
# Wert der Temperatur in
# dezigrad in Temp wird
# in grad umgerechnet:
Calc Grad-Celsius Temp div 10
# Rest dezigrad
Calc Zehntelgrad Temp mod 10
#Ausgabe
QueryVar Grad-Celsius
QueryVar Zehntelgrad
MEnd
```

Listing 2: Messprogramm

Referenzen

[1] Peter Habermehl: *startIDE für die Community Firmware – Programmieren direkt auf dem TXT oder TX-Pi*. ft:pedia 1/2018, S. 102-107.

[2] Till Harbaum: *ftDuino – Open Source trifft Konstruktionsbaukasten*. ft:pedia 1/2018, S. 85-91. Siehe auch: ftDuino.de und Till Harbaum: *fischertechnik compatible arduino*. GitHub.

Computing

startIDE (8): Messung von Temperatur und relativer Luftfeuchtigkeit mit dem Si7021

Rolf Meingast

Der Sensor Si7021 von Silicon Labs ist für die Messung von Temperatur und Luftfeuchtigkeit konzipiert. Nach der Erweiterung von startIDE [1] um I2C-Funktionalitäten [2] lag es nahe, die mit einem Si7021 ermittelten Messwerte mit denen des fischertechnik-Psychrometers [3] zu vergleichen.

Aufbau

Der Si7021 [4] ist ein I²C-Sensor mit 1,9-3,6V Betriebsspannung, maximal 0,18 mA Strombedarf und bis zu 400 kHz I²C-Busfrequenz. Adafruit bietet ein Breakout-Board, das mit Betriebsspannungen von sowohl 3,3V als auch 5V arbeitet (Abb. 1).

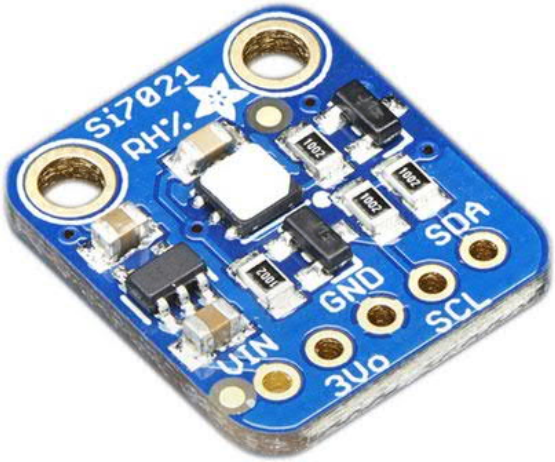


Abb. 1: Adafruit-Breakout-Board mit I²C-Sensor Si7021

Der Versuchsaufbau ist einfach: Das Board wird über ein 4-adriges Kabel an die Pins 5V (VIN), GND, SCL und SDA des ftDuino [5] angeschlossen, der über USB-Kabel mit einem TX-Pi [6] verbunden ist. Die I2C-Adresse des Sensors ist 40h = 64dez.

Programm

Als Programmierumgebung habe ich startIDE gewählt. In einer Endlosschleife (Zeilen 2 bis 42) werden die Sensordaten ausgelesen, in Temperatur (in °C) bzw. relative Luftfeuchtigkeit (in %) umgerechnet und auf dem Display des TX-Pi angezeigt. Mit dem Befehl F3h = 243dez wird die Temperaturmessung initialisiert.

Zuerst werden beide Werte im Array `data` gespeichert und anschließend über den ftDuino an den Sensor übertragen. Nach einer kurzen Pause (40 ms) wird das Ergebnis (zwei Byte) in das Array `data` übertragen. Die beiden Werte aus dem Array werden in den Variablen `n256` und `n1` gespeichert und in die Dezimalzahl `p` umgewandelt (Zeile 18 f.).

Die Umwandlung des Messwerts in die Temperatur in °C erfolgt in den Zeilen 20 bis 22 anhand der Formel aus dem Datenblatt [4]:

$$Temp = \frac{175{,}72 \cdot Temp\_Code}{65536} - 46{,}85$$

Anschließend wird das Ergebnis angezeigt (Zeile 23).

```
# new
Tag Wdh
Clear
Print Si7021
Print
ArrayInit data 64;243
I2CWrite FTD data
Delay 40
ArrayInit data 64;2
I2CRead FTD data
Delay 10
Init n256 0
Init n1 0
Array n1 readFrom data 1
Init p 0
Init Temp_in_°C 0
Calc p n256 * 256
Calc p p + n1
Calc p p * 176
Calc p p / 65536
Calc Temp_in_°C p - 47
QueryVar Temp_in_°C
ArrayInit data 64;245
I2CWrite FTD data
Delay 20
ArrayInit data 64;2
I2CRead FTD data
Init f256 0
Init f1 0
Init Feuchte_in_% 0
Array f256 readFrom data 0
Array f1 readFrom data 1
Calc p f256 bitShift 8
Calc p p + f1
Calc p p * 125
Calc p p / 65536
Calc Feuchte_in_% p - 6
QueryVar Feuchte_in_%
Print
Delay 1000
Jump Wdh
```

Listing 1: Temperatur-Messprogramm

Analog wird bei der Bestimmung der relativen Luftfeuchtigkeit verfahren.

Die Umrechnungsformel ist

$$\%RH = \frac{125 \cdot RH\_Code}{65536} - 6$$

Die folgenden Screenshots der Sensor- (Abb. 2) und der Psychrometermessung (Abb. 3, [3]) sowie der Blick auf mein Haarhygrometer (Abb. 4) zeigen, dass die Messungen des Sensors und des Psychrometers übereinstimmen und – mit geringer Abweichung – durch das Haarhygrometer bestätigt werden.

Abb. 2: Messergebnisse des Sensors

Abb. 3: Messergebnisse des fischertechnik-Psychrometers

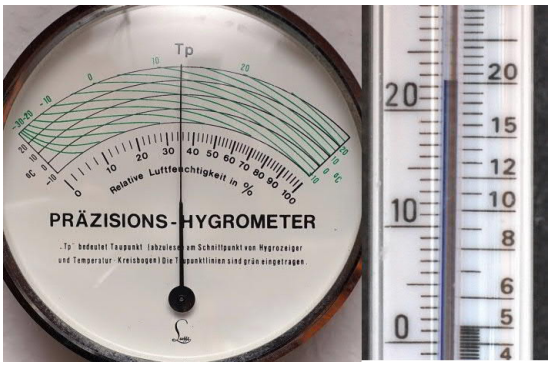


Abb. 4: Haargyrometer und Thermometer

Quellen und Referenzen

[1] Peter Habermehl: *startIDE für die Community Firmware – Programmieren direkt auf dem TXT oder TX-Pi*. ft:pedia 1/2018, S. 102-107.

[2] Peter Habermehl: *Neues von startIDE: Feldvariable, Servos, I2C*. ft:pedia 1/2019, in dieser Ausgabe.

[3] Rolf Meingast: *startIDE (7): Psychrometer*. ft:pedia 1/2019, in dieser Ausgabe.

[4] Silicon Labs: *Si7021-A20: I²C Humidity and Temperature Sensor*. Datasheet, Rev. 1.2, 8/2016.

[5] Till Harbaum: *ftDuino – Open Source trifft Konstruktionsbaukasten*. ft:pedia 1/2018, S. 85-91. Siehe auch: ftDuino.de und Till Harbaum: *fischertechnik compatible arduino*. GitHub.

[6] Peter Habermehl: *Der (schnelle Weg zum) TX-Pi*. ft:pedia 1/2019, in dieser Ausgabe. Siehe auch: Till Harbaum: *TX-PI – A Raspberry PI setup for fischertechnik*. GitHub.

Computing

I²C mit dem TX(T) – Teil 17: Luftdruck- und Temperatursensor (2)

Dirk Fox

Für den Bosch-Drucksensor BMP085 hat Georg Stiegler in Ausgabe 1/2013 [1] einen Treiber für ROBO Pro vorgestellt. Inzwischen hat der Sensor mehrere Nachfolger bekommen – den BMP180 und, Ende 2013, den BMP280. Letzterer erreicht eine sehr hohe Genauigkeit und liefert außerdem einen äußerst präzisen Temperaturwert.

Der Luftdruck

Auf der Erdoberfläche lastet das Gewicht der Erdatmosphäre. Das sind auf Meereshöhe etwa 10^5 N/m², pro Quadratmeter also beachtliche 10 Tonnen – etwa der Druck einer 10 m hohen Wassersäule. Gemessen wird der Druck in Hektopascal (hPa); auf Meereshöhe liegt er im Mittel (bei angenommenen 15°C) bei 1013,25 hPa. Bewegt man sich in höheren Gefilden, sinkt der Luftdruck alle acht Höhenmeter um etwa 1‰ (ca. 1 hPa). Damit lässt sich aus einer Messung der Druckdifferenz die Höhe über dem Meeresspiegel berechnen. Der Druckabfall erfolgt mit zunehmender Höhe jedoch nicht linear sondern hat einen exponentiellen Verlauf, da die erdnahen Luftschichten eine größere Dichte haben und daher schwerer sind (Abb. 1).

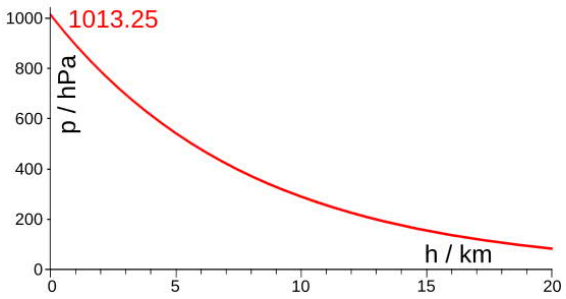


Abb. 1: Verlauf des Luftdrucks im Referenzmodell (Klaus-Dieter Keller, Wikipedia, CC BY-SA 3.0) [2]

Er hängt auch von der Temperatur ab: je geringer die Temperatur, desto größer die Dichte und desto höher der Luftdruck. Will man die Höhe also einigermaßen präzise bestimmen, benötigt man neben einem Drucksensor eine möglichst genaue Temperaturmessung.

Die Höhe in m lässt sich dann nach der *internationalen Höhenformel* (Gl. 1) wie folgt bestimmen [2]:

$$h = \frac{288{,}15\ \mathrm{K}}{0{,}0065\,\frac{\mathrm{K}}{\mathrm{m}}} \cdot \left(1 - \left(\frac{p(h)}{p_0}\right)^{\frac{1}{5{,}255}}\right)$$

$$\approx 44330{,}77 \cdot \left(1 - \left(\frac{p(h)}{p_0}\right)^{0{,}190295}\right)\mathrm{m}$$

Gl. 1: Internationale Höhenformel

Dabei gibt $p(h)$ den gerade gemessenen und, als Bezugswert, p_0 den auf Meereshöhe „reduzierten" Luftdruck an: den Druck, der am Messpunkt theoretisch herrschte, wenn der Punkt auf Meereshöhe läge. Dieser Wert hängt jedoch von der Wetterlage ab und kann zwischen ca. 950 hPa und deutlich über 1040 hPa schwanken. Für eine exakte Höhenberechnung muss der aktuelle Wert von p_0 daher bekannt sein.

Einige Städte veröffentlichen diesen wetterabhängigen Bezugswert p_0 mit ihren meteorologischen Daten im Internet. Alterna-

tiv kann man mit der Höhenformel, aufgelöst nach p_0 (siehe Gl. 2), diese „Reduktion“ für einen Punkt bekannter Höhe berechnen und anschließend zur Bestimmung von weiteren Höhenangaben in gleicher Wetterlage verwenden.

$$p_0 = \frac{p}{\left(1 - \frac{h}{44330{,}77}\right)^{5{,}255}} \text{ hPa}$$

Gl. 2: Höhenformel, aufgelöst nach p_0

Aus der Differenz dieses reduzierten Referenzwertes zum mittleren Luftdruck auf Meereshöhe von 1013,25 hPa lassen sich Aussagen über die Wetterlage ableiten: höherer Luftdruck = schönes, niedrigerer = schlechtes Wetter.

Der BMP280

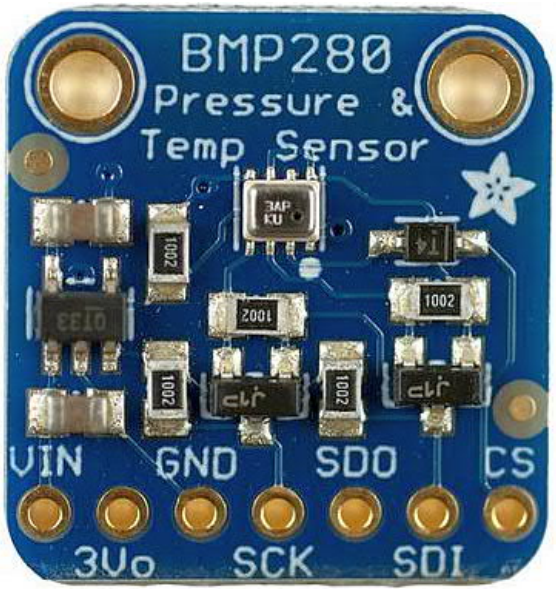


Abb. 2: BMP280 (Bosch/Adafruit)

Mit dem BMP280 hat Bosch Ende 2013 einen Luftdrucksensor mit einer sehr hohen Messgenauigkeit von etwa 0,12 hPa (oder Bar; entsprechend etwa einem Höhenmeter) herausgebracht. Eine wichtige Rolle spielt dabei der integrierte Temperatursensor, der einen Temperaturausgleich vornimmt. Er hat mit 0,0003°C eine deutlich höhere Auflösung und ist mit +/-0,01°C viel genauer als der in ft:pedia 4/2015 vorgestellte Temperatursensor MCP9808 von Microchip [3].

Neben dem I²C- unterstützt der Sensor auch das SPI-Protokoll (nutzbar am ftDuino oder Arduino). Adafruit bietet seit 2015 ein BMP280-Breakout-Board, erhältlich z. B. bei Exp-Tech für rund 11 €.

Technische Daten

- Spannung: 3-5 V
- Strom: 2,7 µA bis max. 1,12 mA (Peak)
- Messbereich: 300-1100 hPa (entspricht ca. -500 bis 9.000 m Höhe)
- Messrate: 23 bis 181 Hz
- Genauigkeit Druck: ± 1 m (0,12 hPa)
- Genauigkeit Temperatur: 0,01°C
- I²C-Bus: bis 3,4 MHz (*high speed mode*)
- I²C-Adresse: 0x77 (fest)

Anschluss an TXT und TX

Wie das Adafruit-Board mit dem Vorgänger-Sensor BMP180 unterstützt das I²C-Board des BMP280 sowohl 3,3 V- als auch 5 V-Logik, sodass der Sensor sowohl am TX als auch am TXT ohne Level-Shifter genutzt werden kann.

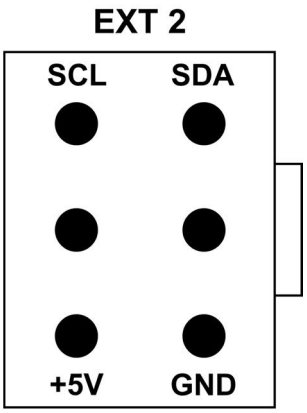


Abb. 3: Belegung der I²C-PINs am EXT 2-Anschluss des TX Controllers

Beim TX wird der I²C-Sensor mit dem Erweiterungsport EXT 2 verbunden (Abb. 3). Die Konfektionierung eines Anschlusskabels mit Pfostenbuchse ist in [4] beschrieben; man kann aber auch einfach vier

Female-Jumper für die Verbindung benutzen. Verbunden werden die Anschlüsse VIN-+5V (Spannungsversorgung), GND-GND, SCL-SCK (Takt) und SDA-SDI (Daten).

Beim TXT erfolgt der Anschluss an der EXT-Buchse, die mit 3,3 V-Logik arbeitet. Der Stromverbrauch des Sensors ist mit maximal 1,12 mA so niedrig, dass für die Stromversorgung des Sensors (VIN) der 3V3-PIN des EXT-Ausgangs genutzt werden kann (Abb. 4).

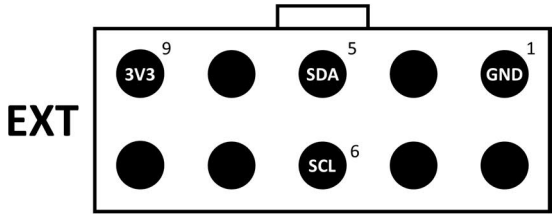


Abb. 4: Belegung der I²C-PINs am EXT-Anschluss des TXT Controllers

Der integrierte Spannungswandler des Adafruit-Boards liefert am Pin ‚3Vo' (Abb. 2) eine 3,3 V-Ausgangsspannung, über die weitere Sensoren mit bis zu 100 mA versorgt werden können. Am TXT muss man dafür allerdings eine leistungsstärkere 3,3 V-Stromversorgung als den 3V3-Pin des EXT-Ausgangs nutzen, z. B. über einen Step-Down-Wandler [5, 6].

Mit dem ftDuino lässt sich der Sensor wie beim TX ebenfalls über eine 6-polige I²C-Pfostenbuchse verbinden: Die Pin-Belegung des 5 V-Logik-Anschlusses entspricht der des TX.

Arbeitsweise

Nach dem Einschalten befindet sich der Sensor im *Sleep Mode*. Wählt man den *Normal Mode*, nimmt der Sensor in regelmäßigen Zeitabständen Messungen vor. Im *Forced Mode* wird eine sofortige Messung initiiert; anschließend kehrt der Sensor in den *Sleep Mode* zurück (Abb. 5).

Die Messauflösung kann – für den Temperatur- und den Luftdrucksensor separat – zwischen 16 und 20 bit gewählt werden. Durch die dadurch mögliche Rauschfilterung steigt die Auflösung des Drucksensors dabei von 2,62 Pa (mBar) auf 0,16 Pa, die des Temperatursensors von 0,005°C auf 0,0003°C.[3]

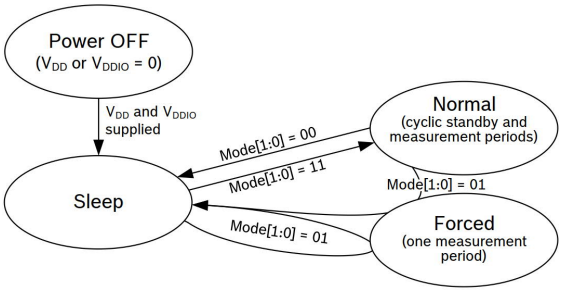


Abb. 5: Zustandsdiagramm [7]

Mit der Erhöhung der Messauflösung steigt allerdings der Aufwand für die A/D-Wandlung und damit auch der Stromverbrauch des Sensors. Der erreicht jedoch selbst bei der höchsten Auflösung maximal 1,12 mA – für TX und TXT eine vernachlässigbare Größe.

Die Dauer einer Luftdruckmessung (inklusive A/D-Wandlung) liegt bei maximaler Auflösung bei etwa 43 ms. Bei kontinuierlicher Messung (*Normal Mode*) sind – abhängig von der Auflösung – 26 bis 166 Messungen pro Sekunde möglich.

Kurzfristige Druckänderungen durch Ereignisse (Öffnen einer Tür o. ä.) können mit einem IIR-Filter (*infinite impulse response filter*) durch Mittelwertbildung ausgeblendet werden.

Register

Der BMP280 verfügt über 11 Byte-Register, über die der Sensor konfiguriert und gesteuert wird (Tabelle 1).

[3] Für die reine Messung des Luftdrucks genügt eine Auflösung des Temperatursensors von maximal 17 bit (= zweifache Überabtastung), da diese nur indirekt zu einer Verbesserung der Druckmessung beiträgt.

Register	Bedeutung
0xD0	*ID Register*
0xE0	*Reset Register*
0xF3	*Status Register*
0xF4	*Control Register*
0xF5	*Config Register*
0xF7-F9	*Pressure Measurement*
0xFA-FC	*Temperature Measurement*

Tab. 1: Register des BMP280

ID Register

Das *ID Register* (0xD0) ist mit dem Wert 0x58 vorbelegt. Darüber lässt sich erkennen, ob der (richtige) Sensor am Bus angeschlossen und unter der (festen) I²C-Adresse 0x77 erreichbar ist.

Reset Register

Über das *Reset Register* (0xE0) kann der Sensor durch Schreiben des Reset-Befehls 0xB6 zurückgesetzt werden. Dabei wird ein *Power On Reset* ausgelöst und der Sensor begibt sich in den *sleep*-Mode.

Status Register

Aus dem *Status Register* (0xF3) kann der aktuelle Zustand des Sensors ausgelesen werden. Dabei werden zwei Stati angezeigt:

- **Bit 0**: Update – Daten aus dem nichtflüchtigen Speicher werden kopiert (beim *Power On Reset* und vor jeder A/D-Wandlung).
- **Bit 3**: Messung – Daten werden gerade umgewandelt.

Mit dem Auslesen der Messwerte sollte gewartet werden, bis das Status-Register den Wert 0 angenommen hat.

Control Register

Das *Control Register* (0xF4) enthält Einstellungen der Sensoren zur Messung:

- **Bit 0, 1**: Arbeitsmodus *Sleep Mode* (00), *Forced Mode* (01 oder 10), *Normal Mode* (11).
- **Bit 2-4**: *Oversampling Pressure*, 0: keine Druckmessung, 1-5: 2^{i-1}-fache Überabtastung.
- **Bit 5-7**: *Oversampling Temperature*, 0: keine Temperaturmessung, 1-5: 2^{i-1}-fache Überabtastung.

Die geringste Auflösung der Temperatur- und der Druckmessung liegt bei 16 bit. Jede weitere Überabtastungsstufe erhöht die Auflösung um ein weiteres Bit (bis 20 bit).

Config Register

Im *Config Register* (0xF5) werden der IIR-Filter und die Standby-Zeit im Normal Mode, also die Wartezeit zwischen zwei Messungen konfiguriert:

- **Bit 0**: 0: 4-fach SPI-Interface, 1: 3-fach SPI-Interface.
- **Bit 2-4**: IIR-Filter (0: aus, 1: zwei, 2: fünf, 3: 11 und 4: 22 Messungen).
- **Bit 5-7**: Standby-Zeit (000: 0,5 ms, 001: 62,5 ms, 010: 125 ms, …, 111: 4 sec).

Pressure Measurement

Die Rohdaten der Luftdruckmessung stehen in den Registern 0xF7-0xF9. Die Bits 4-7 von Register 0xF9 werden abhängig von der gewählten Messauflösung ausgewertet.

Temperature Measurement

Die Rohdaten der Temperaturmessung stehen in den Registern 0xFA-0xFC. Die Auswertung des letzten Bytes hängt von der im *Control Register* gewählten Messauflösung ab.

Die Rohdaten von Druck- und Temperaturmessung müssen anschließend noch in die tatsächlichen Werte in den Einheiten hPa und °C umgerechnet werden. Das erfolgt nach zwei im Datenblatt vorgegebenen Algorithmen [7], die zur Beschleunigung

der je Messung erforderlichen Umrechnungen mit Zwischenwerten rechnen.

Daneben gibt es 12 Kalibrierungs-Register, die werkseitig eingestellte Kalibrierungswerte für die Temperatur- (3) und die Luftdruckbestimmung (9) enthalten. Diese 12 Register müssen ausgelesen werden, um aus den Rohwerten des Sensors die Temperatur (in °C) und den Luftdruck (in hPa) zu berechnen. Damit das Auslesen nicht von Messungen unterbrochen wird, empfiehlt es sich, diese Werte gleich zu Beginn des Programms im Sleep-Mode auszulesen und in globalen Variablen zu speichern.

ROBO Pro-Treiber

Der ROBO Pro-Treiber für den BMP280 findet sich in meiner I²C-Treibersammlung (fischertechnik-AG.de) [8]. Er umfasst die folgenden Befehle:

- `BMP280_GetDeviceID`: Liest über die I²C-Adresse 0x77 die *Device ID* des Sensors aus Register 0xD0 aus.
- `BMP280_Reset`: Löst einen *Power On Reset* des Sensors aus. Danach befindet sich der Sensor im Sleep-Mode.
- `BMP280_GetStatus`: Liefert den Status des Sensors zurück (> 0: laufende Messung/Konvertierung oder Initialisierung)
- `BMP280_Init`: Bei der Initialisierung des Sensors wird zunächst ein Reset durchgeführt und anschließend über das Auslesen der *Device ID* geprüft, ob der Sensor korrekt angeschlossen ist. Zur späteren Umrechnung der Roh-Werte werden die werkseitig voreingestellten Kalibrierungswerte des Sensors ausgelesen und in Variablen gespeichert. Schließlich werden der *Normal Mode* mit einer Messfrequenz von 125 ms aktiviert, für Temperatur und Druck die maximale 20-bit-Auflösung (0,0003°C und 0,16 mBar) gewählt und eine Mittelwertbildung über die jeweils fünf letzten Messwerte eingestellt.
- `BMP280_SetTempPrec`: Einstellung der Auflösung der Temperaturmessung (0: deaktiviert, 1-5: 16-20 bit).
- `BMP280_SetPressPrec`: Einstellung der Auflösung der Luftdruckmessung (0: deaktiviert, 1-5: 16-20 bit).
- `BMP280_SetMode`: Einstellung der Betriebsart (0: *Sleep Mode*, 1/2: *Forced Mode*, 3: *Normal Mode*)
- `BMP280_SetDuration`: Einstellung der Messfrequenz im *Normal Mode* (Pause zwischen zwei Messungen) von 0 (0,5 ms) bis 7 (4 s).
- `BMP280_SetFilter`: Einstellung des IIR-Filters (Zahl der Messungen für die Mittelwertbildung) von 0 (eine) bis 4 (22 Messungen).
- `BMP280_GetTemp`: Messung der Temperatur (Auslesen des Rohwerts und Umrechnung in °C)
- `BMP280_GetPressure`: Messung des Luftdrucks (Auslesen des Rohwerts und Umrechnung in hPa)
- `BMP280_GetAltitude`: Berechnung der Höhe in m aus dem Luftdruck-Rohwert und dem Referenz-Luftdruck auf Meereshöhe.
- `BMP280_GetSeaPressure`: Bestimmung des Luftdruck-Referenzwerts auf Meereshöhe aus der bekannten Höhe des Messpunkts.

Die Umrechnung der Temperatur- und Druckluft-Rohdaten in °C und hPa erfordert zahlreiche Fließkommaoperationen und ist relativ aufwändig. Daher **funktionieren die Berechnungen beim TX auch nur im Online-Mode** (offline werden selbst bei kleinster Messfrequenz keine Ergebnisse angezeigt); der TXT hingegen ist leistungsfähig genug, um die Berechnungen auch offline durchzuführen.

Beispielanwendungen

Wozu aber benötigt man die Höhe? Hilfreich ist sie auf jeden Fall, wenn man überall auf der Welt das perfekte Frühstücksei zubereiten möchte: So ändert sich mit der Höhe auch die Siedetemperatur des Wassers. Sie sinkt etwa alle 300 Höhenmeter um 1°C und liegt damit auf der Zugspitze nur noch bei ca. 90°C. Das Ei muss also entsprechend länger kochen.[4]

Außerdem kann der Sensor in einem Aufzug oder Parkhaus das aktuelle Stockwerk feststellen. Und aus der Höhenänderung kann die Geschwindigkeit einer Seilbahn oder eines Fallschirmsprungs berechnet werden – bei letzterem lässt sich sogar die Auslösung des Notschirms von dem Luftdrucksensor steuern (zugegeben, vielleicht nicht die wichtigste Anwendung für einen fischertechniker).

Es gibt aber noch weitere sinnvolle Anwendungen. So lässt sich mit dem Sensor eine Wetterstation programmieren: Die Kenntnis des Luftdrucks (bzw. dessen Veränderung) erlaubt Wettervorhersagen. Und der Vergleichswert p_0 liefert auch Informationen über Windrichtung und Windgeschwindigkeit, da die Luft aus Bereichen höheren Drucks zu solchen mit niedrigerem Druck strömt.

Kann man die Höhe seines Aufenthaltsorts mit einem Luftdrucksensor zutreffend feststellen, lässt sich auch die Genauigkeit der GPS-Positionsbestimmung verbessern: eine der drei Koordinaten ist damit gegeben [9].

Eine weitere mögliche Anwendung für unseren hoch präzisen Sensor ist die Verbesserung der Entfernungsmessung. Wie wir wissen, hängt die Geschwindigkeit des Schalls von der Temperatur der Luft ab ([10], Abb. 6).

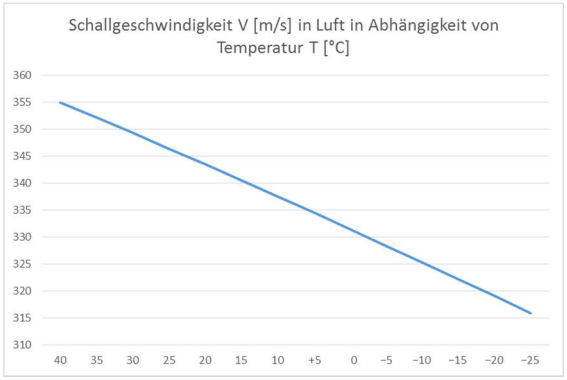


Abb. 6: Veränderung der Schallgeschwindigkeit in Abhängigkeit von der Lufttemperatur

Sie kann nach der folgenden Formel aus T (in °C) berechnet werden [11]:

$$V = 331{,}5 \cdot \sqrt{1 + (T/\,273{,}15)}\ \frac{\mathrm{m}}{\mathrm{s}}$$

Gl. 3: Berechnung der Schallgeschwindigkeit in Luft (in Abhängigkeit von Temperatur T)

So können wir bei der Abstandsberechnung nach einer Ultraschall-Messung die der Berechnung zu Grunde liegende Schallgeschwindigkeit an die aktuelle Lufttemperatur anpassen.

Referenzen

[1] Georg Stiegler: *I²C mit dem TX – Teil 3: Luftdruckmessung*. ft:pedia 1/2013, S. 32-38.

[2] Wikipedia: *Barometrische Höhenformel*.

[3] Dirk Fox: *I²C mit dem TX(T) – Teil 12: Temperatursensor*. ft:pedia 4/2015, S. 44-48.

[4] Dirk Fox: *I²C mit dem TX – Teil 7: Real Time Clock (RTC)*. ft:pedia 4/2013, S. 28-34.

[5] Dirk Fox: *I²C mit dem TX(T) – Teil 13: Farbsensor*. ft:pedia 1/2016, S. 79-89.

[4] Auf dem Mount Everest würde bei einer Siedetemperatur von etwa 70°C das Eiweiß übrigens nicht einmal stocken – da müsste man es schon in einem Dampfdrucktopf garen.

[6] Dirk Fox: *I²C mit dem TX(T) – Teil 16: Servo-Driver*. ft:pedia 2/2017, S. 41-47.

[7] Bosch Sensortec: *BMP280 Digital Pressure Sensor*, Data Sheet, v1.19, 08.01.2018.

[8] Dirk Fox: *I²C-Treiber für RoboPro*. fischertechnik-AG.de.

[9] Dirk Fox: *I²C mit dem TX – Teil 6: GPS-Sensor*. ft:pedia 3/2013, S. 54-62.

[10] Dirk Fox: *I²C mit dem TX – Teil 8: Ultraschall-Sensor*. ft:pedia 4/2013, S. 35-40.

[11] Eberhard Sengpiel: *Berechnen der Schallgeschwindigkeit in Luft und die wirksame Temperatur*.

Computing

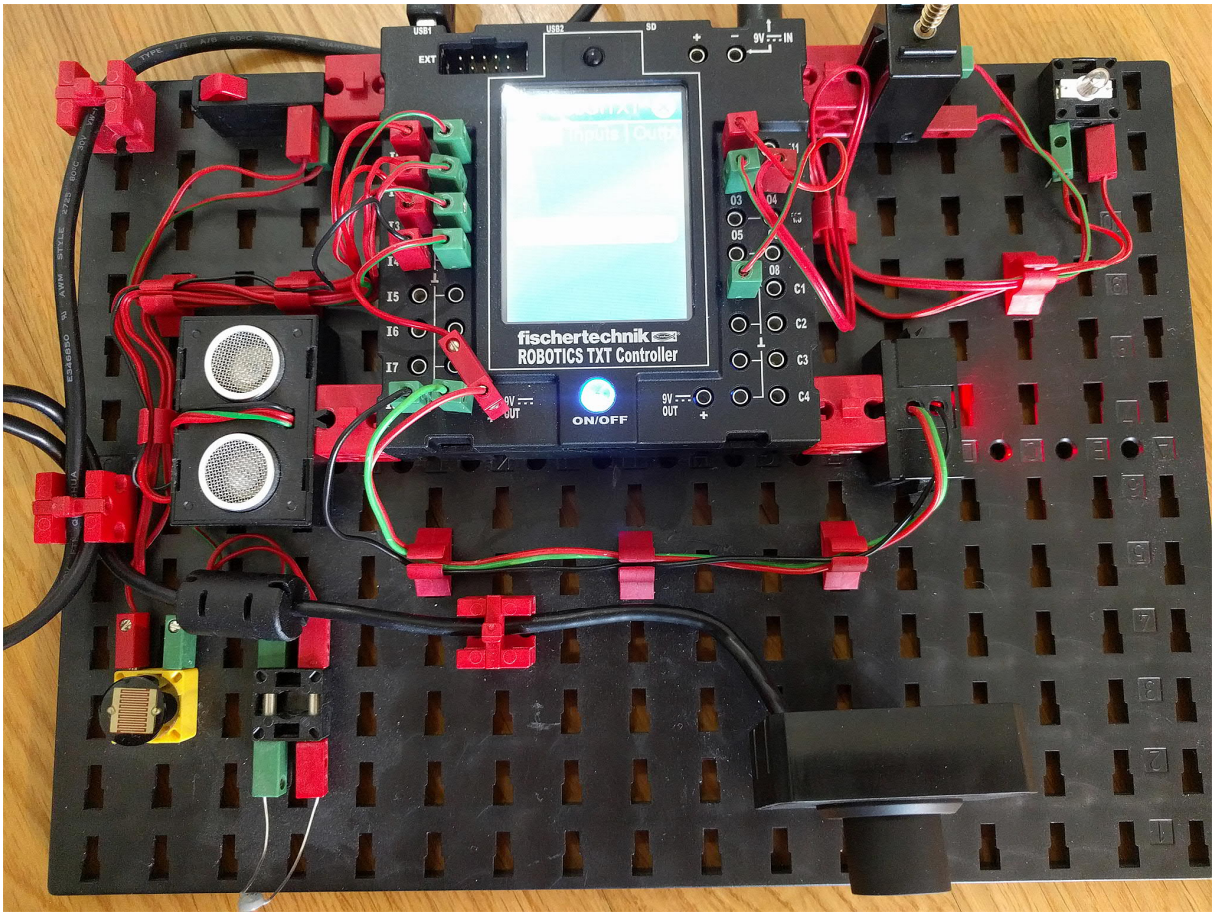
Sustainable smart home with the TXT

Martin Giger

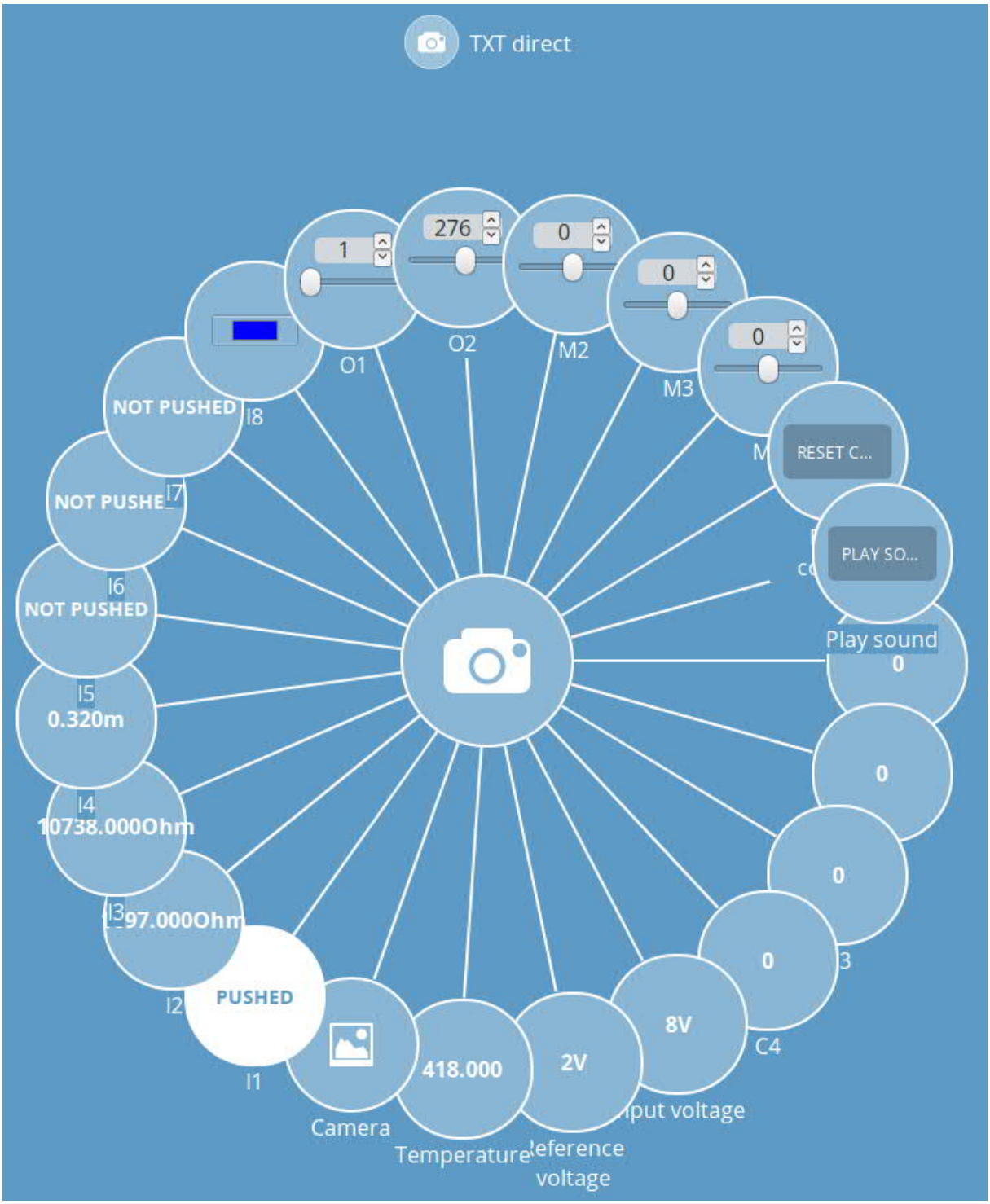
fischertechnik launched the smart home kit last year. A very good move on a conceptual level. Smart home and IoT (internet of things) are rapidly growing technology sectors. The unique placement of the TXT allows it to be a perfect introductory platform to this world.

However, the smart home platform from fischertechnik relies on a central cloud server. This leads to vendor lock-in and challenges to integrate it with other products. Making the cloud your primary interface also means you're vulnerable to internet outages. Of course, the risk really depends on the device type, but you wouldn't want your thermostat to stop working, just because your internet was down.

Mozilla started venturing into IoT recently [1]. They are trying to advocate for better privacy and user freedom by promoting interoperability. These goals are unified in



Img. 1: The TXT IoT demo setup



Img. 2: The web user interface

a proposal for a device API that is based on web protocols called “Web of Things” [2]. It is developed in collaboration with smart home and “industry 4.0” manufacturers. This API isn't necessarily implemented by each device directly, some devices are too low powered to provide a web server or are using other successful local mesh networks like Z-Wave or Zigbee. In these cases, a gateway that is connected to the user's local network would provide the web thing API.

The web thing API breaks physical devices down into three attributes: properties, actions and events. Properties as stateful values that can both be changed by the device and the user (I'll often refer to the user as the client). A device can however declare a property as read-only or add input value restrictions. Actions let the user execute an action on the device that either isn't stateful or affects multiple properties. Lastly, events are fired by the device to indicate a momentary effect that is not reflected in the state. Above that is a capabilities [3] system, with which devices can indicate the semantics of their features. For example, a light bulb can advertise itself as being a “Lamp”, it's brightness property will be a “BrightnessProperty” and the power toggle will be an “OnOffProperty”. These allow clients to expose appropriate UIs and behaviors for devices. The protocol is available over HTTP(S) and optionally WebSockets for real-time communication of changes.

Mozilla is developing both a reference gateway to control devices using this protocol and reference implementations in multiple languages to build web things with. The gateway also has an adapter system to bridge other smart home protocols to the web of things data model.

A TXT is easily powerful enough to run the web server required for the web thing API on its own and be a proper web thing. I've taken the python APIs from the community firmware [4] and Mozilla's python web thing library to create an app that can turn the TXT into a web thing. It currently exposes all inputs (I1-I8), outputs (M1-M4/O1-O8), telemetry of the TXT, the camera as well as the counter inputs (C1-C4) as properties. Only the actor properties are writable. Further it provides actions to play a built-in sound and reset a counter input.

The fischertechnik solution for smart home functionality uses MQTT. Why not just adapt that to the web of things? MQTT is only a protocol and the actual packets aren't standardized. Further, you would have to figure out how to make the TXT connect with a custom MQTT consumer. All in all, I decided it was much simpler and resilient to write an implementation of the web thing protocol that runs on the TXT.

The “WebOfTXT” app, the application connecting the TXT to the web of things I wrote, offers configuration for the sensor and actor types connected to the TXT. After the configuration is set, the user has to start the web thing server. Before starting the server, the schema for the thing is built based on the selections made by the user. A *QTimer* is started to get updated property values. As an exception, the actor properties are only changed by clients of the web thing API, setting the speeds or levels for the outputs. The two actions trigger the respective functions in the TXT API based on the provided inputs. All inputs from API clients are verified by the web thing python package to be valid inputs. For example, the actor levels are checked to be integers in the allowed range for the actor type. These limits are also published as a schema in the web thing API, so clients can also expose these limits to the user.

A special case is the camera. It is exposed as an image snapshot. However, the image is not served via the web thing API. HTTP can already handle images just fine and there's no need for scaffolding information. Furthermore, an image can't be written, only read. Thus, the image property just points to an image file hosted by the TXT. The TXT updates this image about ten times a second. However, refreshing it is entirely up to the client and no notification is sent when a new image is available. For all other properties, the web thing sends an event whenever a property value is changed, so the client can immediately display the updated value.

I found out, that starting the web thing server in the main app thread blocks the

TXT UI. Thus, the web thing server is started in a separate python thread with its own event loop. Otherwise, the application could not be stopped from the TXT, since it would no longer be registering touch screen inputs.

fischertechnik's smart home solution is focused on getting the user up and running with little effort and little first party infrastructure. It is relatively simple to build solutions for the TXT that can be used in conjunction with other smart home or IoT devices, though. I believe building your own solution is even more valuable for education on the topic, for example because you can easily consume the data yourself afterwards. Bringing up privacy, security and interoperability in IoT education is important, especially considering that major manufacturers are avoiding these topics.

Source code of my python bridge can be found in the *freaktechnik/WebOfTXT* GitHub repository [5]. The python script is only about 750 lines of code; however, it requires a couple of python packages to do the heavy lifting. Those are installed by the *build.sh* script, which produces a ZIP file that can be uploaded to the TXT via the web UI of the community firmware. This should be a good starting point to create a more specific web thing with the TXT and annotate it with the proper capabilities.

I want to thank all Mozilla employees and friends from the fischertechnik community that have helped me during the creating of this app, as it was my first time seriously writing python.

I've uploaded a demo video of the app in action (without TXT camera) to youtube [6].

References

[1] Mozilla Corporation: *Project Things*.

[2] Mozilla Corporation: *Web Thing API*.

[3] Mozilla Corporation: *WoT Capability Schemas*.

[4] fischertechnik Community: *TXT Community Firmware*.

[5] Giger, Martin: *WebOfTXT*. GitHub.

[6] Giger, Martin: *Web of TXT demo*. youtube.

Computing

ftDuino spielt Minecraft

Till Harbaum

fischertechnik trifft virtuelle Bausteinwelt!

Über Programmierumgebungen für fischertechnik-Controller und den ftDuino wurde in der ft:pedia schon Einiges geschrieben [1]. Häufig dreht es sich dabei um die Frage, wie so eine Umgebung aussehen muss, damit sie auch von Einsteigern und speziell Kindern leicht zu erlernen und mit Begeisterung zu verwenden ist. Dass die Antwort auf diese Frage unter Umständen an unerwarteter Stelle liegt, soll dieser Beitrag etwas erleuchten.

Dass man Kindern den Einstieg in die Programmierung so einfach wie möglich machen muss, damit sie schnelle Erfolgserlebnisse haben und motiviert bleiben, scheint eine Binsenweisheit zu sein. Wer aber jemals seinen Kindern beim Computerspielen zugeschaut hat, wurde dabei schnell eines Besseren belehrt. Ob es unzählige Pokemon-Monster sind, deren individuelle Eigenschaften der Nachwuchs im Schlaf runterrasselt oder ob es die speziellen Tricks beim Aufbau einer Zombiefalle in Minecraft [2] sind, die die Eltern ahnungslos stehen lassen, ist dabei egal. Es überrascht die Tatsache, dass Kinder in Computerspielen auch komplexeste Zusammenhänge spielend verstehen und umsetzen und Einfachheit wohl nicht unbedingt der Schlüssel zur Begeisterung zu sein scheint.

Es ist wohl etwas anderes, was dafür sorgt, dass Kinder begeistert bei der Sache bleiben. Warum also nicht statt in langweiligen immer gleichen Programmierumgebungen à la Scratch und RoboPro mal in einem Computerspiel programmieren? Was wäre, wenn man virtuelle und physikalische Welt verbinden könnte?

Ein Spiel, das geradezu prädestiniert dafür ist, ist *Minecraft*. Minecraft ist ein sogenanntes Aufbauspiel. Die gesamte dreidimensionale Minecraft-Spielwelt besteht aus Würfeln unterschiedlichen Materials, die der Spieler nahezu frei anordnen kann. Auch wenn es inzwischen alle möglichen Spielmodi gibt, so war das beherrschende Spielelement schon 2009 die Möglichkeit, die Welt frei durch das Versetzen von Blöcken zu verändern. Dabei gleicht Minecraft speziell im sogenannten „Kreativ-Spielmodus“ einem unendlich großen virtuellen fischertechnik-Kasten. Es gibt kaum Spielregeln, es gibt kein Ziel, es geht einzig und allein um das Bauen an sich.

Interessanterweise stellt Minecraft im Spiel eine elektrische Schaltungssimulation namens *Redstone* zur Verfügung sowie Sensoren und Aktoren in Form von Schaltern, Tastern, Lampen, Druckzylindern etc., die mit Redstone interagieren können. Auch einige Logik-Elemente sind im Spiel vorhanden. Findige (und sehr geduldige) Minecraft-Spieler bauen aus unzähligen dieser einfachen Elemente ganze Rechenwerke [3]. Eine Integration echter Elektronik drängt sich hier förmlich auf.

Spielmodifikationen – Mods

Die Kopplung zwischen realer Welt und einem Spiel wie Minecraft steht und fällt mit der Möglichkeit, eigene Objekte in die Spielwelt integrieren zu können. Minecraft

ist in der Programmiersprache Java geschrieben und bot schon immer die Möglichkeit, entsprechende Erweiterungen, sogenannte *Mods*, zu schreiben. Der Hersteller Mojang hat die dafür nötige API ins Spiel integriert. Die Benutzung dieser API ist jedoch nicht einfach und so haben findige Programmierer eine Art Zwischenschicht entwickelt, die auf der einen Seite auf der Original-API aufbaut, auf der anderen Seite aber mächtige und leichter zu benutzende Möglichkeiten zur Mod-Entwicklung bereitstellt.

Eine populäre Zwischenschicht dieser Art ist *Forge* [4]. Wer selbst einmal in die Mod-Entwicklung für Minecraft reinschnuppern möchte, findet unter *MinecraftByExample* [5] ein paar komplette Beispiele für den Einsatz von Forge. Auch die ftDuino-Mod ist auf Basis von Forge und den Beispielen von MinecraftByExample entstanden.

Die Installation von Minecraft und den Mods kann man üblicherweise getrost beliebigen Jungs ab acht Jahren überlassen, entsprechendes Know-How ist hier zum Erstaunen der meisten Eltern in der Regel ausreichend vorhanden. Fehlt der jugendliche Fachmann, dann zeigt ein Video-Tutorial [6] die nötigen Schritte ebenfalls im Detail.

Die ftDuino-Mod

Die ftDuino-Mod unterscheidet sich technisch kaum von beliebigen anderen Mods, die neue Spielelemente einfügen. Der einzig fundamentale Unterschied besteht darin, dass das Spiel Zugriff auf den USB-Anschluss des PCs zur Ansteuerung des ftDuino benötigt. Da die Mod wie das Spiel selbst in Java geschrieben ist, muss der Zugriff auf die Hardware aus Java erfolgen. Eine passende sogenannte Bibliothek dafür ist *JSSC* [7], die z. B. auch in der ebenfalls in Java geschriebenen Arduino-IDE zum Ansprechen der Arduinos verwendet wird.

Um den ftDuino ansprechen zu können, ist auf dem ftDuino ein entsprechender Sketch nötig. Hier kommt der auch an anderer Stelle verwendete *IoServer*-Sketch [8] zum Einsatz. Er erlaubt es, über USB die Ausgänge des ftDuino zu steuern sowie den Zustand der Eingänge vom USB-Anschluss aus abzufragen.

Für den Anwender sind also vier Schritte zur ftDuino-Nutzung in Minecraft nötig:

1. IoServer-Sketch auf dem ftDuino installieren
2. Minecraft installieren
3. Minecraft-Forge installieren
4. ftDuino-Mod installieren

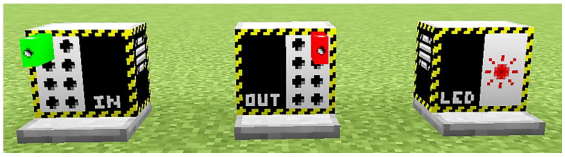


Abb. 1: Die drei ftDuino-spezifischen Minecraft-Blöcke

Die ftDuino-Mod stellt im Spiel ein paar zusätzliche Spielelemente zur Verfügung (Abb. 1). Die Verwendung ist einfach: Es gibt Ein- und Ausgangsblöcke. Durch virtuelle fischertechnik-Steckerchen wählt man in Minecraft den jeweiligen Anschluss am ftDuino aus. Ein elektrisches Signal am ftDuino-Eingang wird durch den Input-Block in Minecraft in ein virtuelles Redstone-Signal innerhalb der Minecraft-Welt umgesetzt. Es kann dort wie jedes andere Redstone-Signal verarbeitet und kann z. B. genutzt werden, um Aktionen im Spiel auszulösen. Umgekehrt können Redstone-Signale per Output-Block auf einen ftDuino-Ausgang geleitet, dort wieder in elektrische Signale umgewandelt werden – und so fischertechnik-Lampen und -Motoren betätigen!

Anwendungen

Die Minecraft-Welt mit fischertechnik zu verbinden erlaubt ein paar interessante Spielvariationen. So kann nicht nur der Spieler selbst Signale auslösen, es können auch *NPCs* (nicht vom Spieler gesteuerte

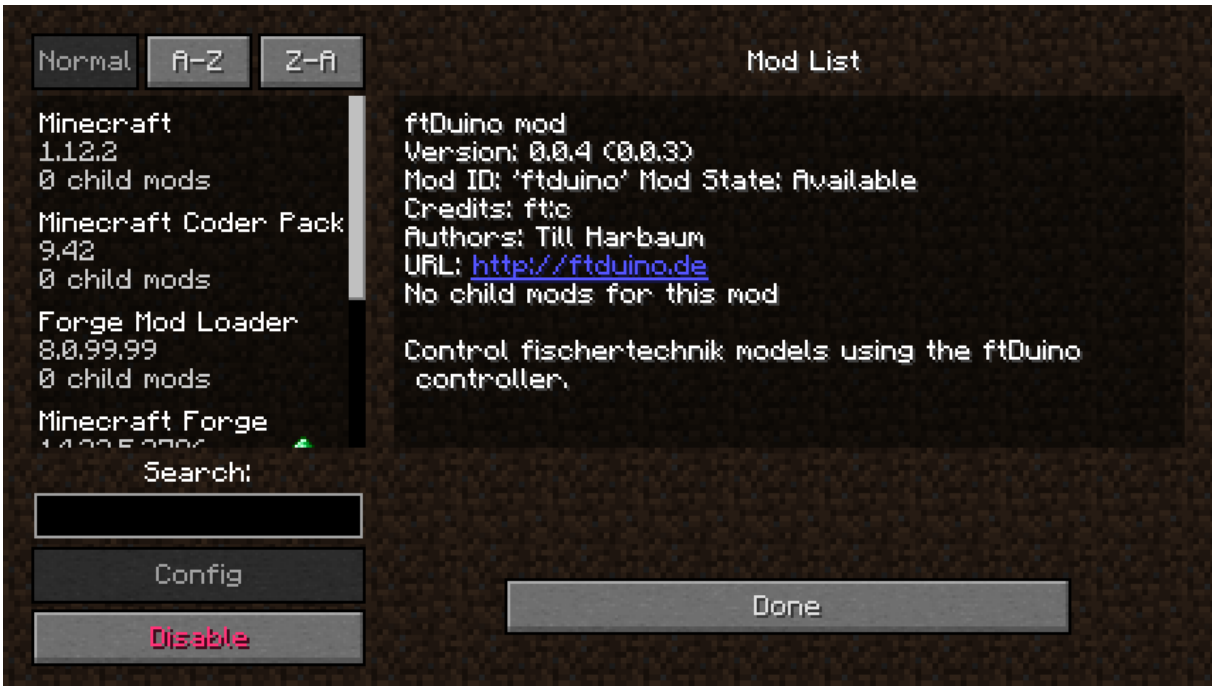


Abb. 2: Das ftDuino-Plugin wird von Minecraft erkannt

Figuren) wie z. B. feindliche Monster ein Ereignis auslösen. So kann eine Gefahr im Spiel z. B. eine Lampe in der realen Welt aktivieren.

Viele Minecraft-Objekte reagieren auf Redstone-Signale. Nicht zuletzt lässt sich Spiel-interner TNT-Sprengstoff auf diese Weise zünden. Die unbegrenzten (Sprengstoff-)Ressourcen im Spiel erlauben es, mit einem Druck auf einen fischertechnik-Taster ganze Berge in Minecraft zu sprengen.

Fazit

Grafische Programmierumgebungen müssen einfach sein? Fragt man Minecraft-Spieler, ist das nicht so. Programmierung mit 3D-Objekten in einer virtuellen Spielwelt kann erstaunlich flexibel sein. Statt didaktisch wertvoller Schaltelemente in kindgerechten Farben warten hier Monster und Berge aus Sprengstoff darauf, verkabelt zu werden und mit der fischertechnik-Welt zu interagieren.

Ist das pädagogisch sinnvoll? Das mag man in Frage stellen. Motiviert es Kinder, sich mit dem Thema auseinander zu setzen? Definitiv! Die offene Spielwelt fördert dabei das um-die-Ecke-Denken, und warum soll man eine fischertechnik-Ampelanlage nicht einmal durch Eisenbahnwaggons steuern, die in einer virtuellen Welt über virtuelle Schalter rollen? Es mag sicher einfachere Lösungen geben, aber cooler ist die Eisenbahn-Lösung allemal.

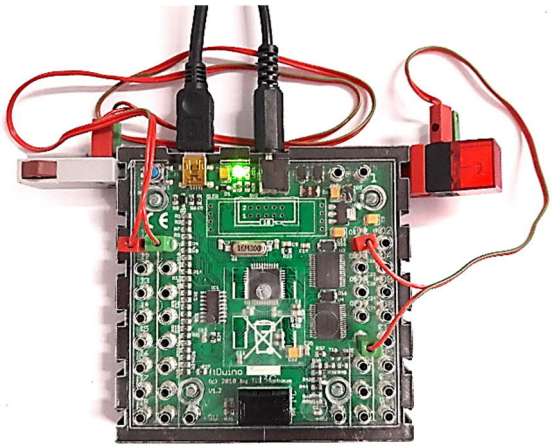


Abb. 3: fischertechnik-Schalter und Lampe am ftDuino reagieren auf Minecraft

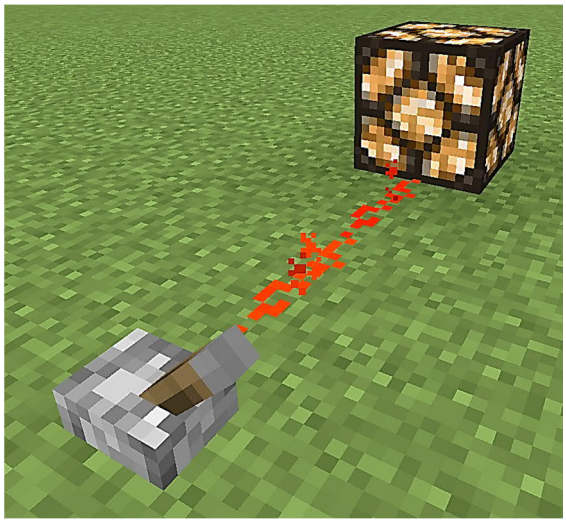


Abb. 4: Schalter und angeschlossene Lampe in Minecraft

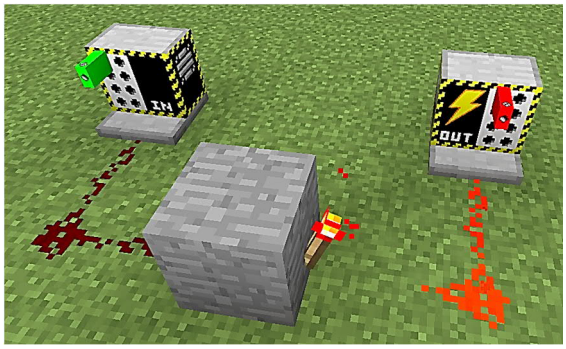


Abb. 5: Eine einfache Inverter-Schaltung in Minecraft zwischen ftDuino-Eingang I1 und -Ausgang O1

Quellen

[1] Till Harbaum: *ftDuino - Arduino für fischertechnik*.

[2] Minecraft.

[3] Minecraft: *Advanced redstone circuits*.

[4] Minecraft: *Minecraft Forge*.

[5] TheGreyGhost: *MinecraftByExample* auf Github.

[6] Kalimero2: *fischertechnik steuert Minecraft ftDuino* auf youtube.

[7] scream3r: *Java Simple Serial Connector* auf Github.

[8] Till Harbaum: *IoServer-Sketch* auf Github.

Abb. 6: fischertechnik-Werbeblöcke in Minecraft

Computing

Der (schnelle Weg zum) TX-Pi

Peter Habermehl

Der TX-Pi [1] als Teil des fischertechnik-Community-Controller-Universums bietet die Benutzeroberfläche der TXT Community Firmware [2] auf einem 3,2 bzw. 3,5 Zoll Touchscreen.

Einleitung

Der Raspberry Pi ist ein 2006 entwickelter Einplatinencomputer. Die aktuelle Version 3 verwendet einen ARM Cortex-A-Prozessor (64 bit) mit bis zu vier Kernen und einem Prozessortakt von bis zu 1,4 GHz; der Arbeitsspeicher kann auf 1 GB ausgebaut werden. Die Betriebsspannung liegt bei 5V – nicht sehr weit von den 9V eines TXT oder Arduino entfernt – und es gibt 26 GPIO-Pins, einen I²C-, CSI- und HDMI-Ausgang. Da kommen die Leistungsdaten des TXT nicht mit.

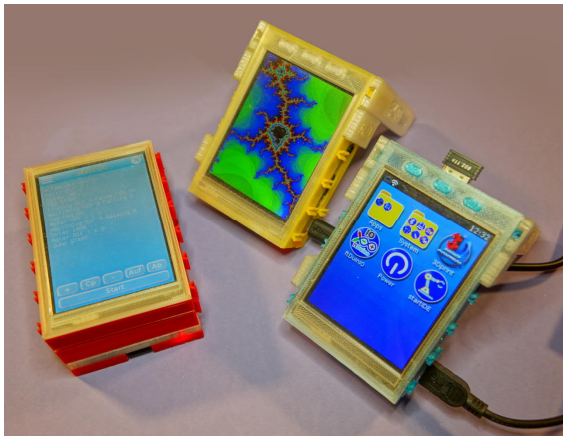


Abb. 1: Ein paar TX-Pis

Das TX-Pi-Projekt der fischertechnik Community ergänzt den Raspberry Pi um einen 3,2 bzw. 3,5 Zoll großen Touchscreen und – optional – ein 3D-gedrucktes, fischertechnik-kompatibles Gehäuse. Als I/O-Hardware können alle fischertechnik-Interfaces mit USB-Anschluss oder auch der ftDuino [3] verwendet werden. Auf dem TX-Pi lässt sich die Community Firmware installieren und nutzen. Damit eignet sich das Gerät auch hervorragend als Controller für den fischertechnik 3D-Drucker, wie die 3D-App der Community Firmware zeigt. Die bislang recht umständliche Installationsprozedur wurde nun kürzlich durch die Bereitstellung von SD-Karten-Images durch die Community deutlich erleichtert.

Abb. 2: TX-Pi im Acrylgehäuse

Hardware-Installation

Durch die jüngsten Updates und die Umstellung auf das aktuelle Raspbian Stretch ist das TX-Pi Setup kompatibel mit allen Raspberry-B-Varianten: Pi 1B, Pi 2B, Pi 3B und Pi 3B+. Man benötigt also lediglich einen Raspberry Pi, ein passendes Netzteil, eine mindestens 8 GB große microSD-Karte und einen Touchscreen der Typen WaveShare 3.2“ V3, 3.2“ V4 oder 3.5“ Typ A bzw. Typ B.

Wer keine Möglichkeit hat, sich das Gehäuse per 3D-Druck zu erstellen, kann auf diverse im Handel erhältliche Gehäuse zurückgreifen, siehe z. B. Abb. 2. So müssen lediglich die RPi-Platine und das Display in das bzw. die Gehäuse verbaut werden, und die Hardware ist bereit.

Software-Setup

Eigentlich neu ist nun die Installation der Software. Während bislang zunächst ein Standard-Raspbian als Betriebssystem installiert werden musste, um dann darauf aufsetzend per Setup-Script die TX-Pi-spezifischen Installationsanpassungen vorzunehmen, liegen unter [4] nun Image-Dateien bereit, die z. B. mit dem sehr einfach zu bedienenden Flash-Tool „Etcher“ [5], das für Windows, Linux und Mac erhältlich ist, auf eine SD-Karte geschrieben werden können.

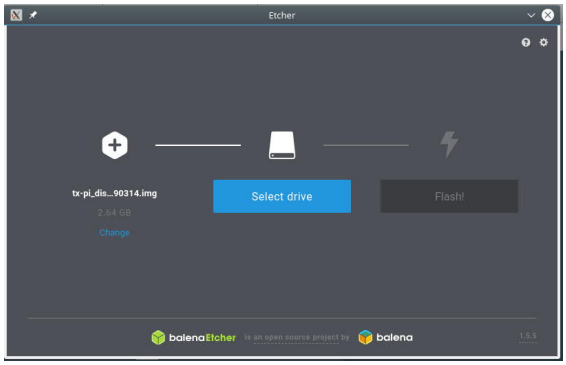


Abb. 3: Das Etcher-Hauptfenster

In Abb. 3 ist das GUI von Etcher dargestellt. Mit nur drei Schritten wird die SD-Karte beschrieben: Zunächst ist die Image-Datei, die von GitHub heruntergeladen wurde, auszuwählen. Etcher verarbeitet die Datei sowohl gepackt (im .zip Format) als auch entpackt (als .img).

Im zweiten Schritt ist das Laufwerk mit der SD-Karte, die beschrieben werden soll, auszuwählen.

Dies ist der einzig kritische Punkt, denn bei Auswahl des falschen Laufwerks kann man damit selbiges irreparabel überschreiben – also Obacht!

Mit einem Klick auf den „Flash“-Button wird das Beschreiben der SD-Karte gestartet. Es erscheint ein Fenster mit Fortschrittsanzeige – siehe Abb. 4. Nachdem die Karte beschrieben wurde, kann man sie direkt in den TX-Pi einsetzen und das Gerät starten.

Abb. 4: Der Schreibvorgang in Etcher

Beim Erststart des TX-Pi wird noch die Konfiguration vorgenommen, so dass z. B. das Dateisystem auf die Größe der Karte angepasst wird und SSH-Schlüssel für das Gerät erzeugt werden.

Der TX-Pi kann nun z. B. über Ethernet oder WLAN mit dem Internet verbunden werden, so dass der App-Store der Community Firmware genutzt werden kann.

Apps, die die I/O-Hardware des TXT ansprechen, können verständlicherweise nicht genutzt werden. Allerdings kann z. B. mit Brickly [6] ein über USB angeschlossener ftDuino gesteuert werden. Damit ist der TX-Pi kinderzimmertauglich.

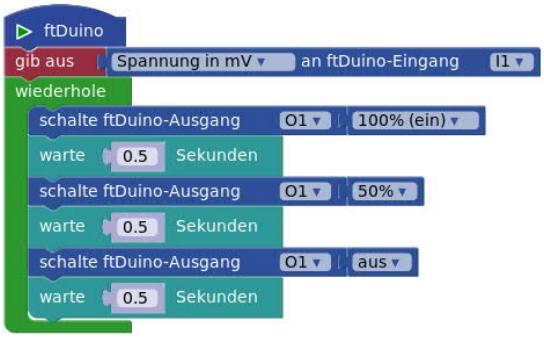


Abb. 5: Ein Brickly-Programm für den ftDuino auf dem TX-Pi

Mit der App „startIDE“ [7], mit der direkt auf dem Gerät programmiert werden kann

(Abb. 6), lassen sich fischertechnik Interfaces der Robo-Interface-Generation (das wären Robo IF, Robo I/O Extension, Robo LT Controller und mit Einschränkungen auch das Intelligent Interface) oder ftDuino und servoDuino [8] ansprechen.

Bild 6: Ampelsteuerung über ein Robo Interface mit startIDE

Fortgeschrittene Benutzer können das Gerät analog zum TXT unter der Community Firmware in Python [9] oder anderen auf dem RPi verfügbaren Sprachen programmieren. Die Bibliotheken „touchUI“ für die GUI-Gestaltung [10], „ftrobopy“ [11] für den Zugriff auf den TXT und „libroboint“ für den Zugriff auf die Robo Interfaces [12] sind bereits installiert.

Fernzugriff auf das Gerät erhält man per ssh mit der Benutzernamen-/Passwortkombination „ftc:ftc“ mit den Rechten als User bzw. mit „pi:raspberry“ mit der Möglichkeit, auch administrative Vorgänge (als Linux-Root-User) vorzunehmen.

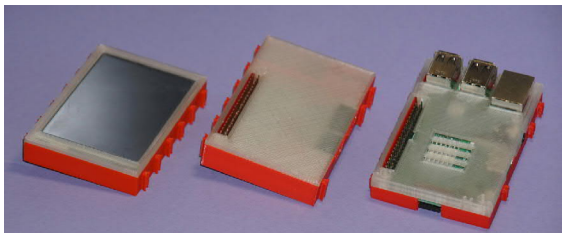


Abb. 7: Das TX-Pi I2C-breakout-Modul (in der Mitte)

Für Selbstprogrammierer ist abschließend noch das TX-Pi I2C Breakout Modul interessant (siehe Abb. 7, [13]). Es ermöglicht auf den I2C-Bus des Raspberry zuzugreifen; gleichzeitig sind Buchsen zur Spannungsversorgung des TX-Pi über das 9V-fischertechnik-System vorgesehen. Ambitionierte Bastler können auch weitere GPIO-Pins des Raspberry über die integrierten Buchsenleisten nach außen führen. Damit stehen der Community Firmware auf dem Raspberry Pi ganz neue Welten offen.

Referenzen

[1] Till Harbaum: *TX-Pi*. GitHub.

[2] Till Harbaum: *Auf zu neuen Ufern: Die Geschichte der „Community-Firmware“ für den TXT*. ft:pedia 4/2016, S. 59-67. Siehe auch: *Die TXT community firmware*.

[3] Till Harbaum: *ftDuino – Open Source trifft Konstruktionsbaukasten*. ft:pedia 1/2018, S. 85-91. Siehe auch: *ftDuino-Webseite*.

[4] fischertechnik-Community Firmware: *SD-Karten-Images*.

[5] Etcher.

[6] Till Harbaum: *Brickly auf dem TXT: Grafische Programmierung à la Google-Blockly*. ft:pedia 1/2017, S. 92-98. Siehe auch: *Brickly-Tutorial*.

[7] Peter Habermehl: *startIDE für die Community Firmware – Programmieren direkt auf dem TXT oder TX-Pi*. ft:pedia 1/2018, S. 102-107. Siehe auch: *startIDE*.

[8] Peter Habermehl: *Servo-Ansteuerung mit servoShield und servoDuino*. In dieser Ausgabe. Siehe auch: Peter Habermehl: *servoDuino: An Arduino sketch to serve as an USB-I2C-bridge for PCA9685 servo shields*. GitHub.

[9] *Community Firmware: Programmieren unter Python*.

[10] Till Harbaum: *touchUI*. Github.

[11] Torsten Stuehn: *Programmierung des TXT unter Python*. ft:pedia 2/2017, S. 58-62. Siehe auch: Torsten Stuehn: *fischertechnik TXT Python*. Github.

[12] Humpelstilzchen: *libroboint: fischertechnik ROBO Interface Library for Unix like systems*. Github.

[13] Peter Habermehl: *TX-Pi I2C breakout*. Thingiverse.com.

Computing

Servo-Ansteuerung mit servoShield und servoDuino

Peter Habermehl

Mit Servomotoren ist weit mehr möglich als der Einsatz als Lenkmotor, wie in den Fahrzeugmodellen von fischertechnik. Dass Servomotoren nicht einmal mit den fischertechnik-Controllern angesteuert werden können, ist daher schon lange ein ärgerlicher begrenzender Faktor. Zum Glück gibt es Abhilfe – Dank der Kreativität der Community…

Servomotoren waren im fischertechnik-System immer eine Randerscheinung. Zwar gehörte schon zur Funkfernsteuerung der 1980er Jahre ein Lenkservo, und auch das Infrarot- und Bluetooth-Fernlenkset brachten jeweils einen (Mini-)Lenkservo mit, aber die Ansteuerung erfolgte ausschließlich über die jeweiligen Fernbedienungsempfänger, sodass ein universeller Einsatz der Servos als Stellmotor in beliebigen Modellen nicht ohne weiteres möglich war.

Mit den auf Thingiverse verfügbaren Konstruktions- bzw. 3D-Druck-Daten für das fischertechnik-kompatible Mini-Servo-System [1] von Jan (Username juh), Abb. 1, kam automatisch die Frage nach einer dazu passenden Ansteuerelektronik auf, da nun mit dem Servo-System unterschiedlichste Antriebsaufgaben elegant gelöst werden können.

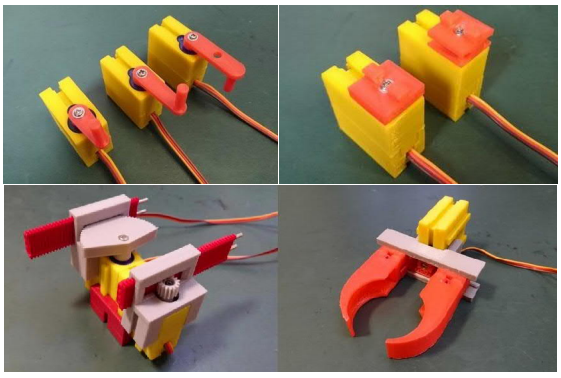


Abb. 1: Mini-Servos von juh [1]

Dabei waren zwei Hauptaufgaben zu lösen. Erstens arbeiten Modellbau-Servos üblicherweise mit einer Versorgungsspannung von 5 bis 6 Volt, während im fischertechnik-System 8 bis 9 Volt verwendet werden. Zweitens erfolgt die Ansteuerung des Servos über ein pulsweitenmoduliertes Rechtecksignal [2], das von den gängigen fischertechnik-Controllern nicht zur Verfügung gestellt wird.

Eine mögliche Lösung wäre die Verwendung eines PWM-Signalgenerators. Solche Schaltungen gibt es in vielfältigen Varianten, auch eine externe Steuerung ist möglich, so dass z. B. über eine Spannnungsmodulation ein Stellsignal für einen Servo erzeugt werden kann.

Eine relativ einfache Servo-Ansteuerung lässt sich allerdings auch realisieren, wenn man über einen Mikrocontroller ein I2C-PWM-Shield ansteuert. Da sowohl der TXT-Controller als auch der ftDuino über einen I2C-Anschluss verfügen und I2C-Servo-Shields im Maker-Umfeld recht gut verfügbar sind, wurde diese Möglichkeit hier weiter verfolgt. Derselbe Ansatz steckt auch hinter dem ftPi von Christian Bergschneider und Stefan Fuß [3] und dem ftPwrDrive-Controller [4].

Das in Abb. 2 gezeigte I2C-Servo-Shield aus dem Arduino-Umfeld (siehe auch [5]),

ein Clon[5] eines Adafruit-Designs, ist je nach Bezugsquelle ab ca. 3 € erhältlich. Es basiert auf dem PCA9685 [6], einem Mikrocontroller, der eigentlich zur Ansteuerung von LEDs entwickelt wurde und über 16 PMW-Kanäle mit 12bit Auflösung verfügt.

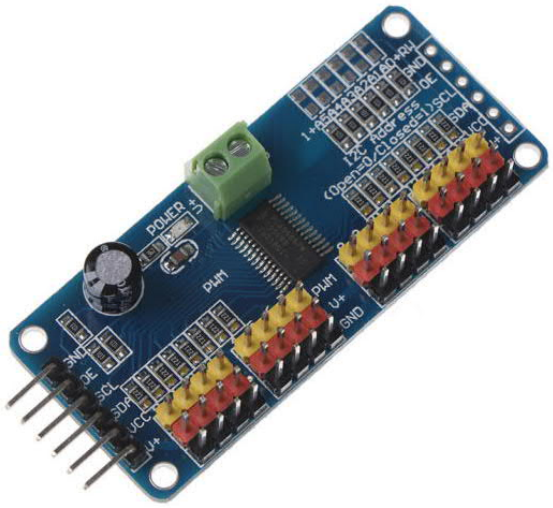


Abb. 2: PCA9685 I2C-Servo-Shield (nach Adafruit)

Das Shield bietet neben dem I2C-Bus-Anschluss die Möglichkeit zur Einspeisung einer separaten Servo-Versorgungsspannung. Weiterhin ist der I2C-Bus durchgeschleift, so dass der Anschluss weiterer I2C-Komponenten möglich ist.

Somit fehlte hardwareseitig nun nur noch eine leistungsfähige 5V-Spannungsquelle. Die Wahl fiel hierbei auf das in Abb. 3 gezeigte XL4015-Modul [7], einen Step-Down-Spannungswandler, der bei einem Eingangsspannungsbereich von 4-38V auf eine Ausgangsspannung von 1,8-36V einstellbar ist und der maximal 5A leistet, womit auch die parallele Versorgung mehrerer Servos kein Problem darstellt.[6]

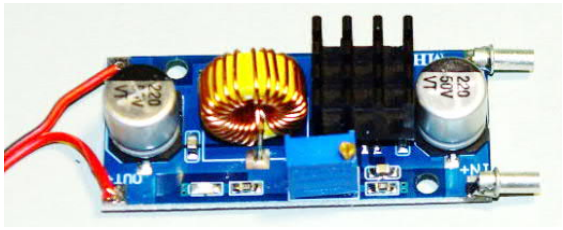


Abb. 3: XL4015-basierter Spannungswandler

Um diese beiden Bauteile fischertechnik-kompatibel zu machen, wurde ein Gehäuse zum 3D-Druck entworfen, das die Module aufnimmt und mit fischertechnik-Nuten zur Befestigung versehen ist. Der elektrische Aufbau ist relativ einfach: An die I2C-Lötanschlüsse an der rechten Seite des Servo-Shields wurden Vcc, Gnd, SCL und SDA nach der Belegung des 6-Pin-Anschlusses des ftDuino auf einen entsprechenden 6-poligen Wannenstecker gelegt. An die Eingänge des Step-Down-Wandlers wurden fischertechnik-kompatible 8,4 mm-Bundhülsen gelötet, die durch die Gehäuserückwand nach außen geführt wurden.

Nachdem der Step-Down-Wandler auf eine Ausgangsspannung von 5,2 Volt justiert wurde, wurden seine Ausgänge mit dem V+-Schraubterminal des I2C-Shields verbunden. Das komplette servoShield ist in Abb. 4 abgebildet, angeschlossen am ftDuino. Um das servoShield am TXT Controller zu betreiben, empfiehlt sich der Einsatz des I2C Extender von Björn Gundermann [8].

---

5 Anmerkung zur Verwendung „günstiger“ China-Hardware: Ich habe aufgrund eines im Internet gefundenen Tutorials dieses Shield bei eBay gekauft, ohne zu wissen, dass es eine Kopie des Adafruit-Originals ist. Da auch dieses und andere China-Module auf der von Adafruit kostenlos bereitgestellte Arduino-Library aufbauen, sei an dieser Stelle darauf hingewiesen, dass die chinesischen Module unabhängig von anderslautenden Beschreibungen der Anbieter Defizite gegenüber dem Original aufweisen können. Darüber hinaus ist es auch aus den anderen genannten Gründen eine gute Idee, Adafruit direkt durch den Kauf originaler Produkte zu unterstützen. Gleiches gilt auch für die Arduino-Produkte.

6 Eigene Messungen zur Stromaufnahme des SG90-Servo liegen nicht vor, das offizielle Datenblatt nennt keine Werte und im Internet gibt es verschiedene Aussagen. Der Spitzenstrom dürfte im Bereich von 250-500 mA liegen.

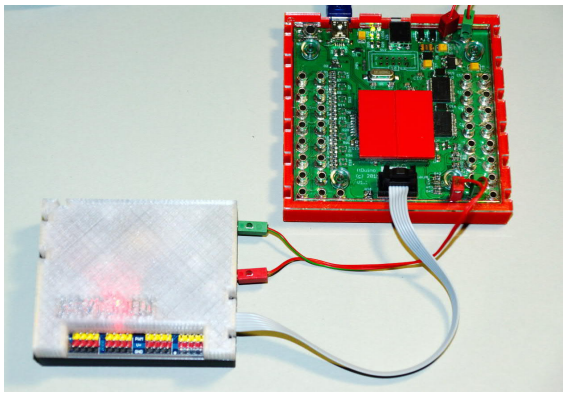


Abb. 4: servoShield am ftDuino

Softwareseitig gibt es verschiedene Lösungen zur Ansteuerung des servoShield. Am ftDuino liegt die Verwendung der Adafruit PCA9685 Servo Driver Library nahe, die sehr gut dokumentiert ist [9].

Um das servoShield unter Robo PRO mit dem TXT zu verwenden, ist auf die Robo PRO-I2C-Funktionen zurückzugreifen. Da der PCA9685 aber diesbezüglich ebenfalls sehr gut dokumentiert ist, ist auch die Ansteuerung über I2C vergleichsweise einfach zu bewerkstelligen. Ein fertiger Robo PRO-Treiber ist in [5] beschrieben und steht in der Robo PRO-I2C-Treibersammlung zum Download zur Verfügung.

Eine weitere, sehr einfache Möglichkeit zur Ansteuerung des servoShield (z. Zt. nur am ftDuino) bietet die App „startIDE“ für die community firmware des TXT und TX-Pi [TX-Pi], wie im Artikel zur aktuellen startIDE-Version in dieser Ausgabe der ft:pedia nachzulesen ist.

Nachdem mit dem servoShield erste Erfahrungen mit dem Einsatz von Servos in fischertechnik-Modellen gesammelt werden konnten, entstand die Idee, Servos für Controller ohne I2C-Bus auch direkt verfügbar zu machen. Das Ergebnis dieser Bemühungen ist der servoDuino, der die Komponenten des servoShield um einen Arduino Nano ergänzt (siehe Abb. 5).

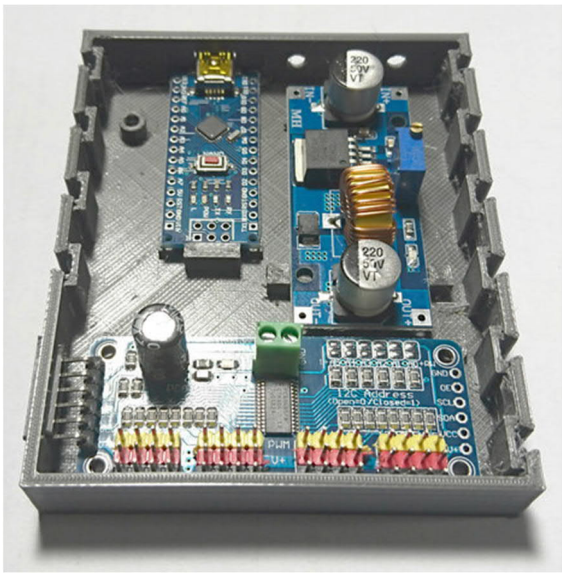


Abb. 5: servoDuino

Obwohl aus elektronischer Sicht nicht optimal, weil der Arduino empfindlich auf unsaubere Spannungsversorgung reagieren kann, wurde der Arduino parallel zum V+Terminal des Servo-Shields an den Ausgang des Step-Down-Wandlers angeschlossen. Details finden sich unter dem servoDuino-Link [10]. Im bisherigen Betrieb traten keine Stabilitätsprobleme auf.

Damit ergeben sich zwei Einsatzmöglichkeiten:

Mit entsprechender Programmierung des Arduino mit der Arduino-IDE unter Verwendung der vorgenannten Library kann das Gerät eigenständig laufen. Ohne weitere Eingänge wären damit zwar zunächst nur taktgesteuerte Antriebe möglich, aber das kann zum Betrieb z. B. eines Kugelvereinzelers oder eines Hubmechanismus' für eine Kugelbahn völlig ausreichen.

Darüber hinaus können GPIO-Anschlüsse des Arduino nach außen gelegt und als Eingänge zur Programmsteuerung verwendet oder am gehäuseseitig links durchgeschleiften I2C-Bus des Servo-Shields (hier passt eine 6-Pin-DuPont-Steckerleiste im 2,54 mm-Rastermaß) weitere I2C-I/O-Komponenten betrieben werden.

Mit dem im servoDuino-Repository [10] verfügbaren Arduino-Sketch lässt sich der servoDuino zusätzlich auch noch über USB ansteuern.

Der Arduino dient dann lediglich als USB-I2C-Bridge. Somit kann über den seriellen Monitor der Aruino-IDE oder jeden beliebigen USB-Host, der serielle Kommunikation über USB zulässt, mit simplen Klartext-Befehlen auf das Servo-Shield und den I2C-Bus zugegriffen werden.

Diese Möglichkeit wurde wiederum in startIDE genutzt, sodass der Betrieb des servoDuino als Servo-Controller und I2C-Bridge am TX-Pi (und natürlich TXT) unter startIDE möglich ist. Weitere Details hierzu finden sich in der startIDE- und servoDuino-Dokumentation.

Abschließend lässt sich sagen, dass mit Jans Mini-Servo-System fantastische neue Möglichkeiten im fischertechnik-System geschaffen wurden, und dass sich, 3D-Druck-Möglichkeiten vorausgesetzt, mit sehr wenig Elektronik-Bastelaufwand und geringen Materialkosten von 7-10 € für servoShield bzw. servoDuino auch eine zur Mehrheit der üblichen Robotik-Controller kompatible Ansteuerlösung schaffen lässt.

Die für alle Nicht-Bastler vielleicht beste Nachricht zum Schluss: Björn Gundermann erwägt, auch die Servos in seinem fischertechnik-Shop anzubieten, sobald der z. Zt. in Entwicklung befindliche ftPwdDrive-Controller [4], der ebenfalls Servos ansteuern kann, bei ihm erhältlich ist.

Quellen und Referenzen

[1] Jan: *fischertechnik SG90 mini servo system*. Thingiverse, 15.09.2018, und zugehöriger Thread im ftc-Forum.

[2] Wikipedia: *Servo*.

[3] Christian Bergschneider, Stefan Fuss: *Alternative Controller (3): Der ftPi – ein Motor Shield für den TX(T)*. ft:pedia 2/2016, S. 68-72.

[4] Dirk Wölffel, Christian Bergschneider, Stefan Fuss, Björn Gundermann, Christian Lauff: *Der ftPwrDrive-Controller für Schrittmotoren und Servos – Teil 1*. ft:pedia 4/2018, S. 67-70. Sieh auch den zugehörigen Thread im ftc-Forum.

[5] Dirk Fox: *I²C mit dem TX(T) – Teil 16: Servo-Driver*. ft:pedia 2/2017, S. 41-47.

[6] NXP: *PCA9685: 16-channel, 12-bit PWM Fm+ I²C-bus LED controller*.

[7] XLSEMI: *XL4015: 5A 180KHz 36V Buck DC to DC Converter*. Datasheet, Rev. 1.5.

[8] Björn Gundermann: *ft-Extender zum einfachen Anschluss von I2C-Geräten*.

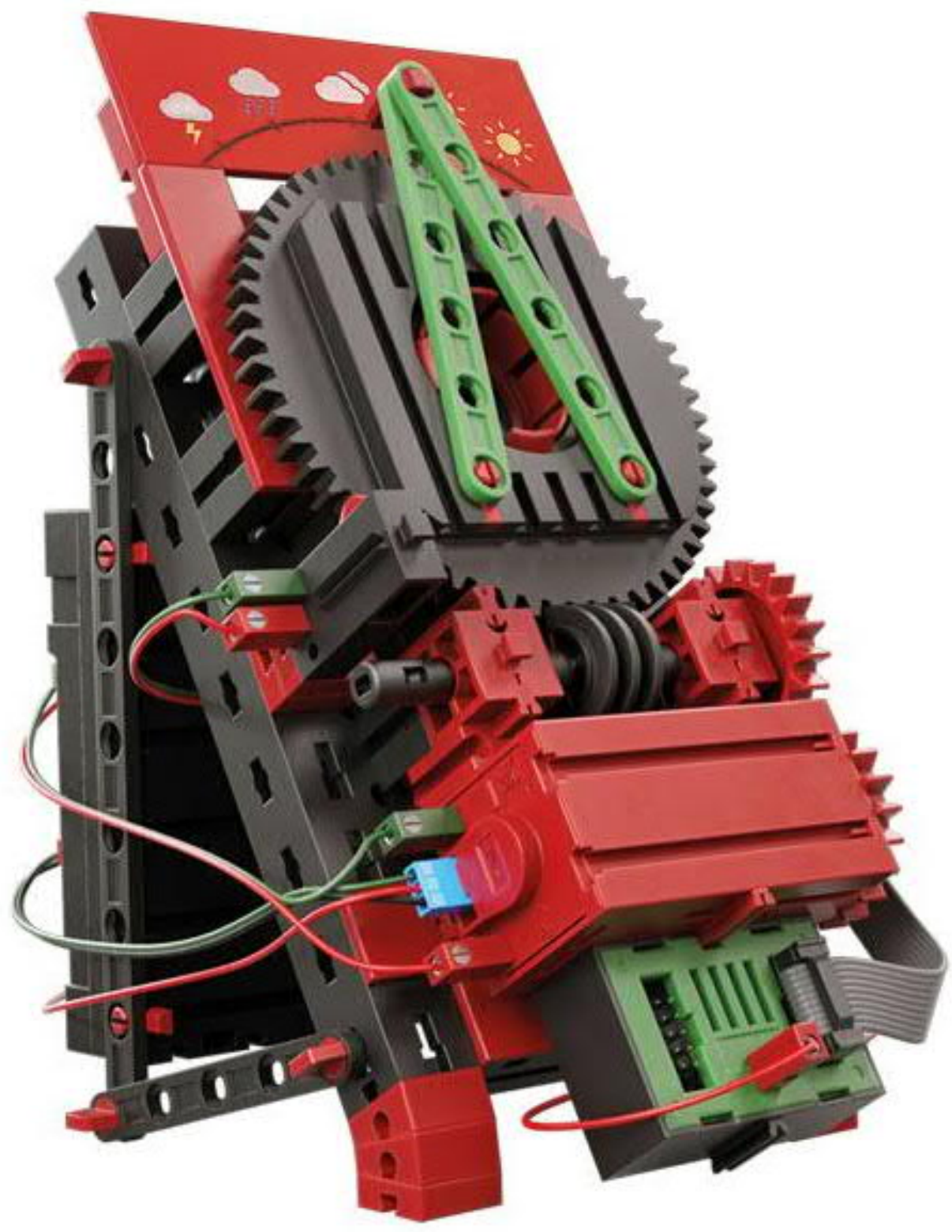
[9] Adafruit: *Adafruit-PWM-Servo-Driver-Library*. Arduino-Library für den PCA9685 Servo Driver, GitHub.

[10] Peter Habermehl: *servoDuino: An Arduino sketch to serve as an USB-I2C-bridge for PCA9685 servo shields*. GitHub.

[11] Peter Habermehl: *fischertechnik I2C servo shield case*. Thingiverse, 27.12.2018.

[12] Peter Habermehl: *Der (schnelle Weg zum) TX-Pi.* ft:pedia 1/2019, in dieser Ausgabe. Siehe auch: Till Harbaum: *TX-Pi*. GitHub.

[13] Till Harbaum: *ftDuino*.



Robotics TXT Smart Home Barometer