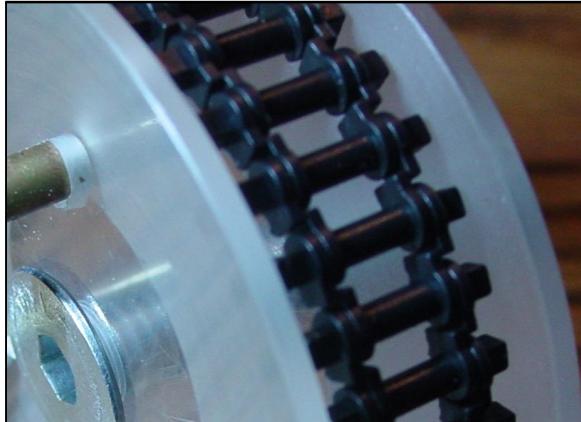
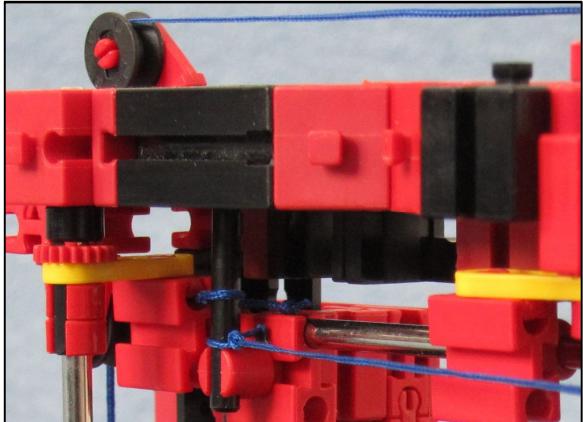


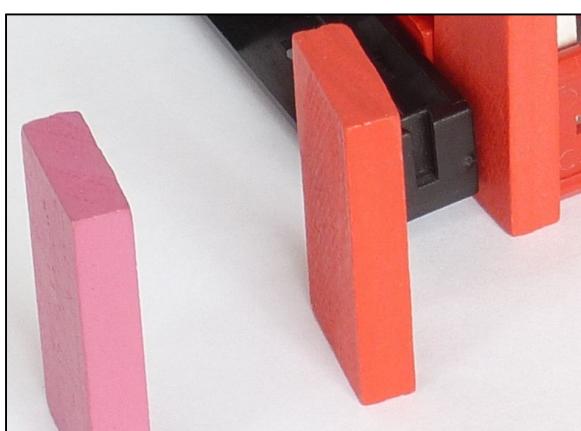
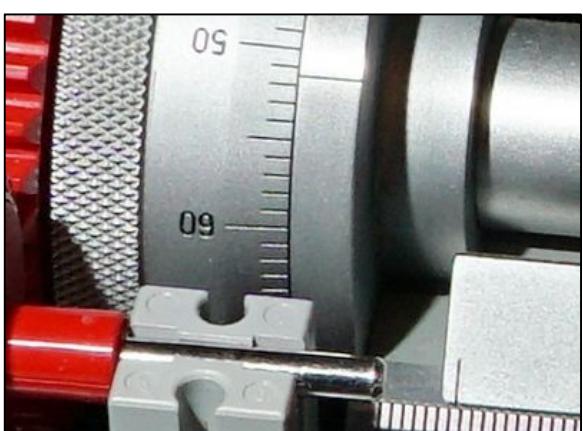
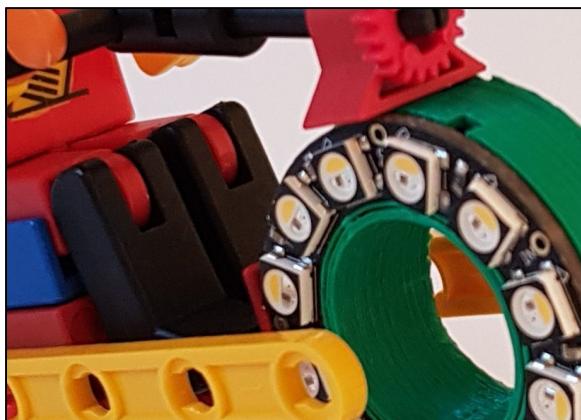
ft:pedia

Heft 1/2018



Time	Source	Destination
7 7.241041338	192.168.0.78	127.0.0.1
8 7.241221753	192.168.0.78	127.0.0.1
9 7.241242993	192.168.7.2	192.168.0.78
10 7.241259077	192.168.7.2	192.168.0.78
11 7.241265594	192.168.0.78	127.0.0.1

ame 9: 68 bytes on wire (544 bits), 68 bytes captured (544 bits)
nx cooked capture
ternet Protocol Version 4, Src: 192.168.7.2, Dst: 192.168.0.78 (Transmission Control Protocol, Src Port: 65000, Dst Port: 521)
Source Port: 65000
Destination Port: 52146
[Stream index: 1]
[TCP Segment Len: 0]
Sequence number: 1 (relative sequence number)
Acknowledgment number: 5 (relative ack number)
1000 = Header Length: 32 bytes (8)
Flags: 0x010 (ACK)
Window size value: 342
[Calculated window size: 43776]



Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

Vielfalt statt Einfalt

Calliope mini heißt die neue Geheimwaffe der Bundesregierung gegen den Fachkräftemangel – ein kleiner Controller, basierend auf dem BBC micro:bit, der 2016 in Großbritannien an Siebtklässler verteilt wurde. Entwickelt und eingeführt wurde er mit Unterstützung von Geesche Jost, Design-Forscherin und Internetbotschafterin der Bundesregierung, im Herbst 2016 und ist inzwischen beim Cornelsen-Verlag für rund 35 € für Grundschulen erhältlich. Kern ist ein 32-bit ARM Cortex M0-Prozessor mit 16 kB RAM und 256 kB Flash-Speicher – immerhin das Achtfache eines Arduino. Er verfügt über Motortreiber, eine SPI- und eine I²C-Schnittstelle und kann via USB über verschiedene Web-IDEs programmiert werden, unter anderem mit dem [Open Roberta Lab](#) in der an Scratch angelehnten Sprache NEPO.

Auf der Spielwarenmesse hat fischertechnik einen Calliope-Kasten für Schulen vorgestellt, der drei einfache Modelle (Fußgängerampel, Händetrockner und Schranke) und eine schöne Einsteigeranleitung enthält. Leider nutzen die Modelle im Kasten nur den Solarmotor – dabei sind für Einsteiger und Grundsünder starke Motoren wichtiger als der Gyro- und Beschleunigungssensor, über den der Calliope verfügt. Ein Calliope ist im Kasten nicht enthalten – sicherlich nicht allein wegen des Preises, sondern weil er ohne Gehäuse und CE-Zeichen daherkommt.

Dirk Fox, Stefan Falk

Die Steuerung von Modellen über eine integrierte Entwicklungsumgebung (IDE) mit Web-Schnittstelle ist seit Ende 2016 allerdings auch mit den fischertechnik-Controllern möglich: Für den LT und den TXT hat fischertechnik ScratchX um entsprechende Befehle erweitert; auch der BT Controller soll in Kürze unterstützt werden. Diese Controller haben einem Calliope mini voraus, dass sie kurzschlussfest in einem geschlossenen, fischertechnik-kompatiblen Gehäuse verbaut sind. Zudem ist Scratch in Schulen deutlich verbreiteter als NEPO.

Wer seinen TXT mit der Community Firmware ausstattet, kann ihn außerdem nicht nur mit dem Scratch-ähnlichen [Brickly](#) via WebIDE programmieren, sondern auch mit der StartIDE App ganz ohne Computer – direkt auf dem Touchscreen des TXT.

Und wem die fischertechnik-Controller nicht preiswert genug sind: Jetzt hat die Community auch noch einen fischertechnik-kompatiblen kurzschlussfesten Arduino im Gehäuse hervorgebracht – den ftDuino, mit 9V-Motortreibern und I²C-Schnittstelle.

Mehr zu diesen Möglichkeiten in dieser ft:pedia... Oder frei nach Goethe: „*Warum in die Ferne schweifen? Sieh, das Gute liegt so nah.*“

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

Vielfalt statt Einfalt	2
Mini-Modelle (Teil 19): fischertechnik tanzt in den Mai	5
Kaulquappen (Teil 9): Eine schräge Übertragung	7
Zahnräder mit variablem Z	9
Löten leichtgemacht	12
Urlaubskasten-Modell 5: Partner-Ventilator	17
Dominostein-Aufsteller	18
Schiebetüren	24
Vom elektromechanischen Betätiger zur elektrischen „Dampfmaschine“	34
fischertechnik-3D-Drucker 2.0	40
Automatische Differentialsperre	47
Neopixel für alle	53
Arduino-Sensoren am TX(T)	64
Scratch mit fischertechnik	69
fischertechnik meets BASCOM	79
ftDuino – Open-Source trifft Konstruktions-Baukasten....	85
Hacker im Kinderzimmer: Remote-Angriff auf den TXT .	92
startIDE für die Community Firmware – Programmieren direkt auf dem TXT oder TX-Pi	102
startIDE (1): Messen und Experimentieren	108
startIDE (2): Seilwinde	110
startIDE (3): TXT im freien Fall	112
startIDE (4): Focus Stacking	115

Termine

Was?	Wann?	Wo?
Nordconvention	28.04.2018	Schulzentrum Wedemark (Hannover)
Süd-Süd-Convention	31.05.2018	Gartenschule Karlsruhe
Schul-Robotik-Cup	16.06.2018	Bismarck-Gymnasium Karlsruhe
Fanclub-Tag	30.06.2018	Waldachtal

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, Gerhard Birkenstock,
Stefan Busch, Stefan Falk, Dirk Fox, Stefan Fuss, Ralf
Geerken, Peter Habermehl, Till Harbaum, Rolf Meingast,
Thomas Püttmann, Christian Riebeling, Rüdiger Riedel,
Martin Wanke, Dirk Wölffel.

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

Modell

Mini-Modelle (Teil 19): fischertechnik tanzt in den Mai

Rüdiger Riedel

Leider fehlt dem fischertechnik-Mann die Partnerin zum Tanzen – die fischertechnik-Frau. Wie wäre es mit Herrn und Frau Fischer [1]? Waldachtal, bitte helfen.

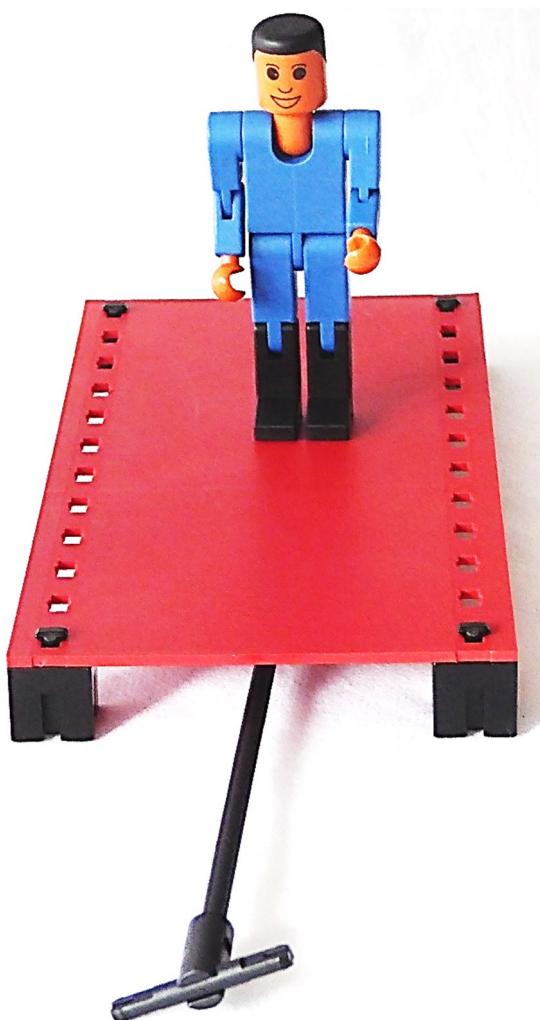


Abb. 1: fischertechnik-Mann auf der Platte

Bis dahin legt Herr Fischer eine Solo-Platte hin: Eine Statikplatte 180 · 90 wird mit vier Riegelsteinen bestückt. Die lange Rastachse erhält zwei Rastadapter. In einen kommt die kurze Rastachse, in den zweiten ein Neodym-Magnet mit 4 mm Durchmesser und 10 mm Länge (diesen Magnet gibt es im Internet z. B. bei fischerfriendsman.de; er ist kein fischertechnik-Produkt). In die Stiefel des fischertechnik-Mannes werden zwei dieser Magnete gedrückt. Eventuell müssen einige ausprobiert werden, um zwei gut passende zu finden. An einem Stiefel sollte der Nordpol herausschauen, am anderen der Südpol.

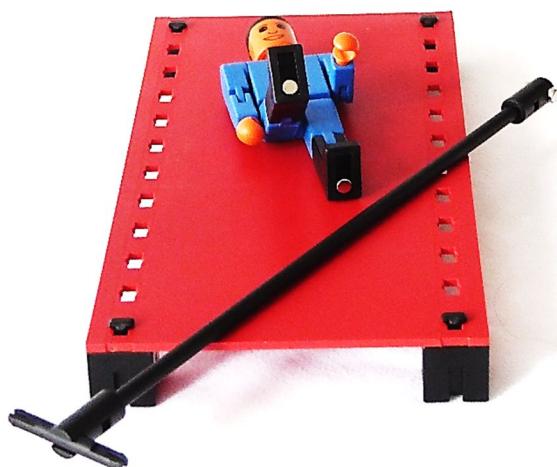
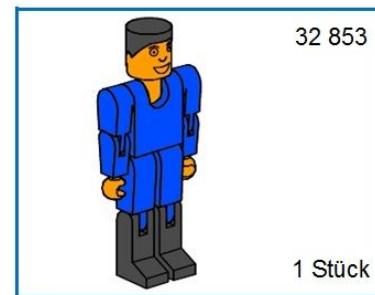
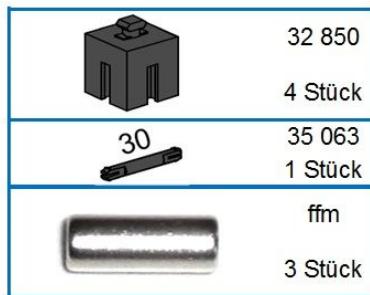
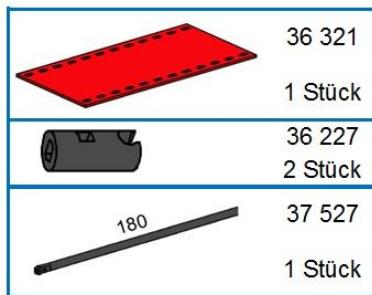


Abb. 2: Zeig' her die Stiefel

Das Spiel ist ganz einfach: Wird der Griff der Schiebestange waagerecht gehalten, kann Herr Fischer vor und zurück dirigiert werden. Bei senkrechtem Griff wird nur ein Fuß angepackt und Herr Fischer beginnt bei kreisenden Bewegungen der Schiebestange zu tanzen – rechts herum, links herum, im Dreiviertel-Takt.



Referenzen

- [1] Peter Habermehl: [Herr und Frau Fischer](#) im ft community-Bilderpool, 2015.

Abb. 3: Stückliste

Tipps & Tricks

Kaulquappen (Teil 9): Eine schräge Übertragung

Ralf Geerken

Mit Ketten und der Ur-Version der fischertechnik-Kegelzahnräder gelingt eine kraftschlüssige Übertragung auf schräg stehende Achsen, wie dieser Beitrag zeigt.

In manchen Bausituationen hat man ja das Problem, dass eine Drehbewegung auf eine andere Achse übertragen werden muss, die aber leider nicht parallel zur Antriebsachse liegt. In vielen Fällen wird das Problem mit Kardangelenken ([31044](#), [35971](#), [35115](#) + [38446](#)) gelöst.

Eine 90 Grad Übertragung ist auch mit

- einem Z40 + Z10 oder mit
- zwei Rastkegelzahnrädern möglich.

Es gibt noch einige Möglichkeiten mehr, eine Drehbewegung um 90 Grad zu übertragen. Nur zwei weitere davon seien hier noch genannt:

- Das Differential 85983 oder [75218](#) (mit Differentialkäfig [31411](#) und Differential-Kegelzahnrädern [31414](#), [35143](#) oder [137196](#)), welches dann die Drehzahl obendrein noch halbiert bzw. verdoppelt, wenn eine Seite befestigt wird sowie
- die Kegelzahnräder Z10 [35146](#) (neuere Bauform) und [31051](#) (alte Bauform).

Auf das Kegelzahnrad [31051](#) der alten Bauform möchte ich hier etwas näher eingehen, weil man damit nämlich noch andere Winkel als 90 Grad übertragen kann. Mithilfe einer Zahnradkette lässt sich nämlich ein Winkel von 7,5° problemlos und kraftschlüssig übertragen. Kleinere Winkel stellen kein Problem dar, größere Winkel lassen sich bis ca. 12° noch sehr gut übertragen.

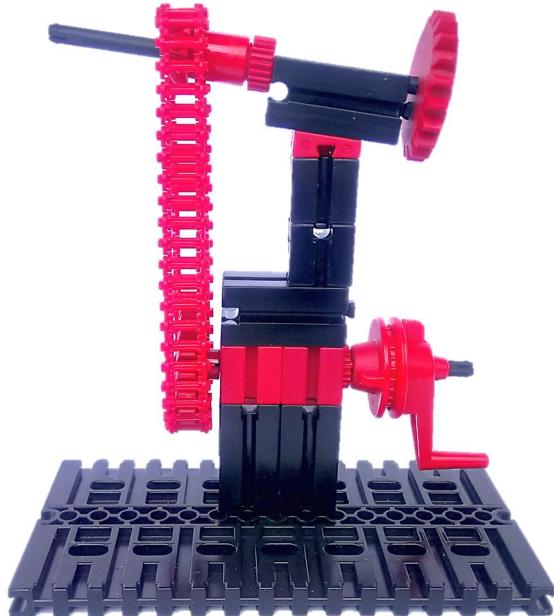


Abb. 1: Schräge Übertragung

Bei 15 Grad war dann Schluss, weil dann der breitere Teil des Kegelzahnrad in die Außenseite der Kette greift.

Bisher ist das noch nicht so weltbewegend, denn man kann diese Winkel, wie Martin Romann [1] und Thomas Püttmann [2] das in ihren Uhrenkonstruktionen bereits vor Jahren gezeigt haben, auch mit normalen Zahnrädern erreichen. Diese werden dazu einfach in einer Ebene gegeneinander gekippt. Diese Konstruktion hat aber den Nachteil, dass der Abstand der beiden Achsen festgelegt ist und dass sich die Drehrichtung umkehrt, was je nach Situation auch erwünscht ist.



Abb. 2: Schräg in zwei Ebenen

Eine weitere Möglichkeit bei der Kombination aus Kegelzahnrad und Zahnradkette ist es, die beiden Achsen nicht nur in einer Ebene winklig zu versetzen, sondern sie in der zweiten Ebene auch noch gegeneinander zu verdrehen. Das ist ebenfalls in einem Winkel bis ca. 12° problemlos möglich (Abb. 2).

Die neuere Variante des Kegelzahnrads [35146](#) ist übrigens für diese Konstruktion ziemlich ungeeignet, weil sich die Zähne schon bei viel kleineren Winkeln in der Kette verklemmen.

Es sind bestimmt noch viele weitere Kombinationen möglich. Diese zu (er)finden möchte ich aber ab dieser Stelle euch überlassen.

Referenzen

- [1] Martin Romann: [Pendeluhr](#) in der ft Community, 2010.
- [2] Thomas Püttmann: *Die Rast-O'Clock-Uhr.* [ft:pedia 1/2017](#), S. 42-47.

Tipps & Tricks

Zahnräder mit variablem Z

Gerhard Birkenstock

In fast jedem fischertechnik-Modell wird ein Getriebe gebraucht. Glücklich, wer keine genaue Übersetzung braucht. Aber wehe man benötigt genau die eine, und sie will einfach nicht mit den Zahnräden $Z=10, 15, 20, 30, 32, 40$ von fischertechnik zusammenpassen. Hier kommt eine Lösung.

Hintergrund

Zwar lassen sich auch ausschließlich mit fischertechnik-Zahnräden nahezu beliebige Über- oder Untersetzungen unter Verwendung von Differenzialen konstruieren, wie Thomas Püttmann gezeigt hat [1]. Kompakter (und mit weniger Reibungsverlusten) geht es, wenn man die erforderlichen Zahnräder mit passender Zähnezahl zur Verfügung hat [2, 3].

Mechanischer Aufbau

Auch mir stand mal wieder dieses Problem ins Haus. Aber dieses Mal wollte ich eine Lösung. Nichts Halbes. Etwas Richtiges, und variabel sollte es sein.

Um Zahnräder mit vorgegebener Zähnezahl Z zu erhalten, habe ich schon viele Lösungen gesehen: Von geklebten Kunstwerken bis zu mit CNC-Maschinen gefrästen Präzisionsteilen. Leider habe ich nichts mit reinen fischertechnik-Teilen entdeckt – von den oben erwähnten Konstruktionen mit Differenzial einmal abgesehen.

Meine Idee bedient sich des schon oft verwendeten Grundgedankens der fischertechnik-Kettenglieder – wie zum Beispiel von Harald Steinhaus (siehe Abb. 1 und [4]) und Triceratops (siehe seine „[Trickfibel](#)“). Dabei wurden die Ketten um runde fischertechnik-Komponenten (Räder, Rollen, Zahnräder, Naben, Differentialkäfig) gelegt.



Abb. 1: Ketten-Zahnräder (Harald Steinhaus)

Um aus Ketten Zahnräder mit beliebiger Zähnezahl zu machen, braucht man jedoch einen variablen Träger für die unterschiedlichen Durchmesser. Nach einem Hin und Her ist mir die in Abb. 2 skizzierte Lösung eingefallen.

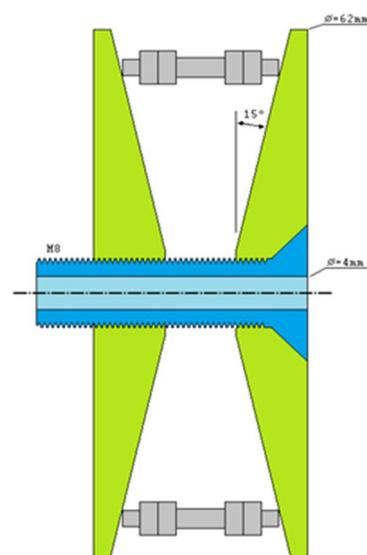


Abb. 2: Variabler Träger für Ketten-Zahnrad

Eine M8-Schraube wird mit 4,1 mm durchbohrt, passend für die fischertechnik-Achsen. Zwei Kunststoffscheiben bekommen zentrisch eine Bohrung von 6,4 mm. Diese Bohrungen werden mittels Gewindesteckern auf M8 erweitert. Einseitig werden beide Scheiben mit einem Konus von 15° versehen. Um die Vorrichtung mit der fischertechnik-Welt zu verbinden, werden noch drei Bohrungen mit 4 mm Durchmesser gesetzt. Hier passen die Seilscheibe und die beiden Zahnräder Z30 und Z40 dran.

Nun werden die beiden Scheiben so auf die M8 Schraube gedreht, dass die Schrägen sich nach innen gegenüber stehen. In diesen variablen Zwischenraum wird eine Kette vorgefertigter Länge platziert. Die beiden konischen Flächen zentrieren die Kette automatisch beim Zusammendrehen und klemmen das System zu einer kraftschlüssigen Einheit zusammen. Da die Kettensegmente gleichmäßig zentriert und belastet werden, kommt es zu keiner Stauchung einzelner Kettensegmente.

Ist die benötigte Zähnezahl gering, ist der Eingriff zwischen den Scheiben sehr tief. Dann muss mit einem weiteren großen Zahnrad die Eingriffsebene herauf geholt werden (siehe Abb. 3).

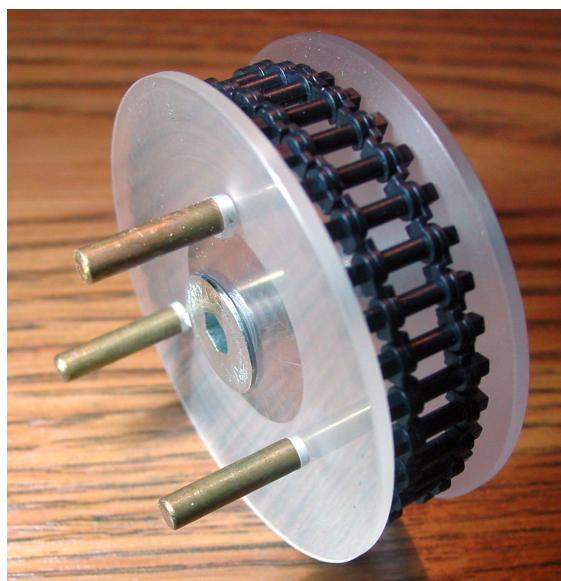


Abb. 3: Fertige Zahnräder-Konstruktion

Übersetzung

Nun zur eigentlichen Übersetzungs-Aufgabe. Rein mechanisch möchte ich von einer Minutenwelle zur Stundenwelle mit 24 h Umlaufdauer. Das Verhältnis ergibt sich zu:

$$i = 24h \cdot 60m = 1440m$$

Um dies mit möglichst wenigen Zahnrädern zu erreichen, habe ich folgende Aufteilung gewählt:

$$i = 1440 = \frac{40 \cdot 36 \cdot 10}{1 \cdot 1 \cdot 10}$$

Die jeweiligen x/1-Übersetzungen sind Schneckengetriebe. Hier wird ein Zahn pro Umdrehung transportiert.

In Abb. 4 ist die Anordnung gut zu erkennen. Das schwarze Z15 holt die Eingriffsebene hoch und macht eine Drehrichtungs-umkehr. Vom Sekundenrad werden über die Getriebe das Minutenrad und das Stundenrad angetrieben.

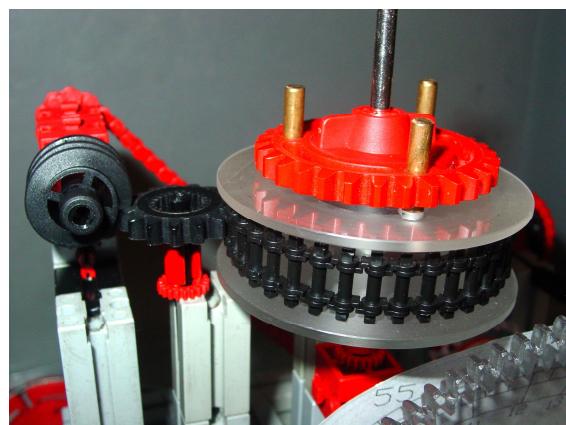


Abb. 4: Selbstbau-Z36 im Einsatz

Technische Verbesserungen

Eine Verbesserung wären Stufen auf den konischen Flächen (Abb. 5). Die Ketten würden sich ablegen und damit zentrieren. Die Abstände der Stufen sind im Durchmesser mit

$$D = \text{Modul} \cdot Z - 2$$

für die neuen Kettenglieder mit ihren seitlichen quadratischen Zapfen und mit

$$U = \text{Modul} \cdot Z - 4$$

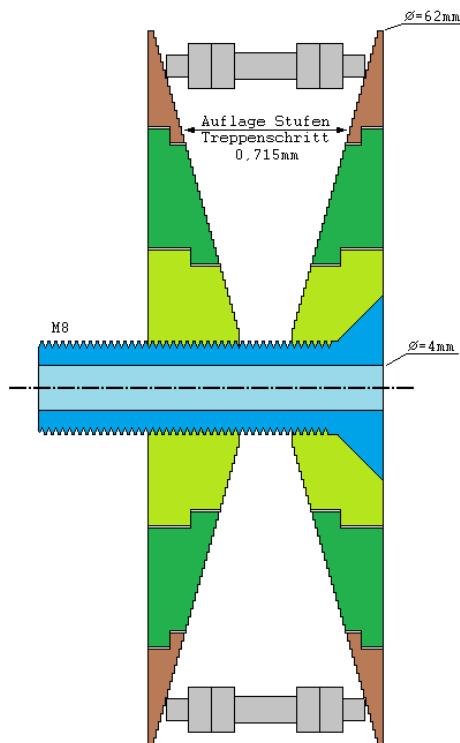


Abb. 5: Gestufte Schrägen und Ringsegmente

für die alten roten Kettenglieder mit 4 mm Kettenhöhe zu berechnen.

Dabei steht das Modul für die von fischer-technik vorgegebenen 1,5.

Eine weitere Verbesserung wären Ringseg-mente (Abb. 5). Diese lassen sich je nach Bedarf stapeln. Der Durchmesser der Klemmscheiben würde nach Bedarf ge-wählt und das abtriebende Zahnrad nicht mehr stören.

Referenzen

- [1] Thomas Püttmann: [Getriebe](#). ft:pedia 4/2014, S. 12-19.
- [2] Gerhard Birkenstock: [Uhrwerk mit Z80 und Z100](#). ft:pedia 4/2014, S. 20-24.
- [3] Gerhard Birkenstock: [Power-Synchronmotor](#). ft:pedia 3/2017, S. 42-45.
- [4] Harald Steinhaus: [Kaulquappen \(Teil 5\)](#). ft:pedia 3/2014, S. 14-16.

Tipps & Tricks

Löten leichtgemacht

Christian Bergschneider

Viele Modding-Artikel in der ft:pedia stellen mit einfachen Schaltungen praktische Erweiterungen für fischertechnik vor. Auch wenn für den Nachbau das nötige Elektronikwissen meist im Artikel beschrieben wird, bleibt das Handwerk des Löten unerklärt. Doch Löten ist nicht schwer und man kann es an einem Abend lernen.

Die Geschichte des Löten



Abb. 1: Die berühmte Totenmaske von Tutanchamun wurde im 14. Jahrhundert vor Christus gelötet

Archäologische Funde beweisen, dass die Menschheit schon 5000 v. Chr. das Löten als Technik zur Metallverarbeitung beherrschte. Löten wurde z. B. für die Herstellung von Kult- oder Schmuckgegenständen verwendet, um die damals bekannten Metalle Gold, Silber und Kupfer zu verbinden. Dabei wurden Kupfersalze in der CO-Atmosphäre eines Holzkohlefeuers reduziert, die Kupferanteile verbanden sich in einer chemischen Reaktion mit Gold oder Silber zu einer lötfähigen Legierung. Die Legierung ist die Mischung von mindestens zwei Metallen; sie hat andere Eigenschaften als jede ihrer Komponenten. Das prähistorische Lot hatte z. B. mit 889°C einen

deutlich niedrigeren Schmelzpunkt als die reinen Metalle wie Gold mit 1064°C. Somit konnten Verbindungen geschaffen werden, ohne dabei die Einzelteile erneut zu verformen.



Abb. 2: gelötete Bauteile auf dem fitPi

Heute wird das Löten hauptsächlich in der Elektrotechnik angewandt. Mit dem Löten werden leitende Verbindungen zwischen den Bauteilen hergestellt und die Bauteile selbst auf der Platine befestigt.

Werkzeug

Für den Anfang braucht man keine Spitzenausstattung. Für eine solide Grundausstattung muss man 50 bis 100 € investieren, je nachdem was sich bereits in der eigenen Werkzeugkiste findet.



Abb. 3: Handelsüblicher Lötkolben

Es gibt sehr viele verschiedene Arten von Lötkolben. Für Elektronikbauteile genügt

ein normaler, elektrischer Lötkolben (Abb. 3). Der Lötkolben sollte eine *Dauerlötspitze* haben, am besten in Bleistiftspitzenform. Die Dauerlötspitze ist mit einem edlen Metall überzogen, so dass diese nicht oxidieren kann. An oxidiertem Kupfer würde das Lot nicht halten und könnte nicht gelötet werden. Die Dauerlötspitze muss man u. U. extra kaufen. Die Leistung des Lötkolbens sollte zwischen 15 und 30 W liegen. Hat der Lötkolben eine zu geringe Leistung, so kann man die Lötstelle nicht ausreichend erhitzten, bei zu hoher Leistung wird es der Platine und den Bau teilen schnell einmal zu heiß.

Bei einer *Lötstation* kann man die Löttemperatur genau einstellen. Der Lötkolben wird je nach Nutzung nachgeregelt, so dass man sehr schnell und effektiv arbeiten kann. Aufgrund der hohen Anschaffungskosten, ist eine Lötstation für den Anfänger jedoch nicht sinnvoll.

Ein normaler Lötkolben ist so ausbalanciert, dass er gut auf dem Arbeitsplatz liegen bleibt. Wer etwas mehr lötet, wird recht bald in einen *Lötständer* investieren. Dieser hält den Lötkolben bei Nichtgebrauch griff bereit und hat meist noch für ein Lötschwämmchen Platz (Abb. 4).



Abb. 4: Lötständer mit Lötschwämmchen

Am Lötschwämmchen streift man überflüssiges Lot ab. Vor dem Arbeiten wird das Lötschwämmchen leicht angefeuchtet. Niemals eine Dauerlötspitze mit spitzem Werkzeug oder Schmirgelpapier bearbeiten, da sonst die Beschichtung abgeschabt würde!

Lötabsaugungen sind ebenfalls nur für den Profi. Sie saugen die Lötdämpfe auf und neutralisieren sie in einem Filter. Die hohen Kosten können durch gutes Lüften gespart werden.

Beim *Lötzinn* muss man sich die berühmte Gretchenfrage stellen: Bleihaltige Lote setzen beim Löten bleihaltige Dämpfe frei und sind später auf der Mülldeponie Sondermüll. Deshalb sind sie schon längst im kommerziellen Bereich verboten. Allerdings sind die Löteigenschaften von bleihaltigem Lot etwas besser, so dass es sich viel leichter verarbeiten lässt. In bleifreiem Lot wird das Blei durch Silber ersetzt. Der Schmelzpunkt liegt etwas höher und das Lot fließt schlechter, so dass bleifreies Lot etwas mehr Übung benötigt. Egal für welches Lot man sich entscheidet, es sollte Flussmittel im Lötzinn enthalten sein.

Am Anfang sollte man das Lot etwas dicker wählen (0,75 bis 1 mm Durchmesser). Für feine Bauteile wie SMDs benötigt man mit Lot 0,5 bis 0,75 mm Durchmesser. 100 Gramm reichen als Erstausstattung völlig aus; man kann damit sehr viele Schaltungen löten.

Das Lot und der Lötkolben werden beim Löten zwischen 350 und 450°C heiß. Eine feste Unterlage schützt den Arbeitsbereich vor Brandflecken und hilft später beim Aufräumen kleiner Kabelreste und Lotspritzern.



Abb. 5: Entlötpumpe

Wer lötet, macht auch Fehler. Dann hilft eine *Entlötpumpe*, das vorhandene Lot abzusaugen (Abb. 5). Hier gibt es kaum technische Unterschiede.

Die normale Spitzzange aus dem Werkzeugkasten hilft dabei, die Pins der Bauteile zurechtzubiegen und beim Löten ggf. auch festzuhalten.

Den besten Halt gibt jedoch eine „Dritte Hand“ (Abb. 6). Bauteile beim Löten einfach mit der eigenen Hand festzuhalten ist die schlechteste Alternative und führt zu verbrannten Fingern.



Abb. 6: Dritte Hand

Mit einem kleinen *Seitenschneider* werden Kabel und Pins der Bauteile auf die richtige Länge zurechtgeschnitten. Wie die Spitzzange findet sich dieser häufig im heimischen Werkzeugkasten.

Eine *Abisolierzange* hilft beim Entmanteln von Kabeln. Ein einfaches Modell mit Rändelschraube ist völlig ausreichend, Automatikzangen bieten für den Preis keinen angemessenen Vorteil. Verarbeitet man ausschließlich Drähte, so reicht häufig auch ein Seitenschneider aus.

Kleiner Lötkurs

Löten lernt man am besten an einer einfachen Schaltung. In unserem Lötkurs soll eine LED an einen fischertechnik-Batteriekasten angeschlossen werden. Als Bauteile werden ein Widerstand mit $470\ \Omega$, eine LED, etwas Kabel, ein Stück Prototypenplatine sowie zwei fischertechnik-Stecker benötigt (Abb. 7).

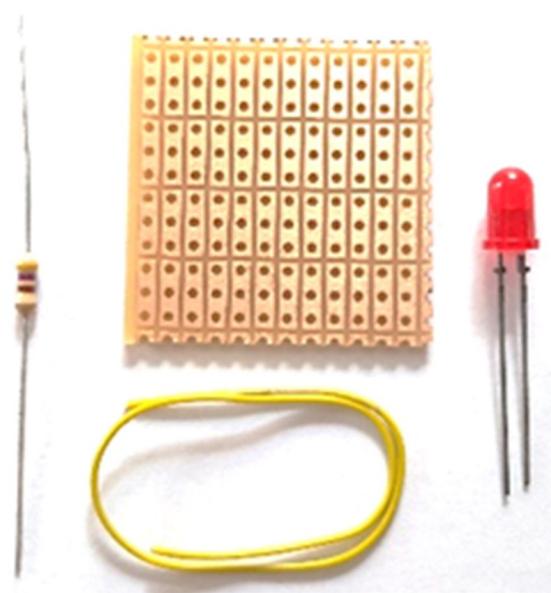


Abb. 7: Bauteile für die Lötbübung

Die Schaltung besteht nur aus einer LED, die mit einem Vorwiderstand in Reihe geschaltet wurde. Die LED-Farbe und Größe kann frei gewählt werden. Für den Betrieb an der fischertechnik-Bordspannung von 9 V muss der Vorwiderstand zwischen 360 und $680\ \Omega$ liegen.

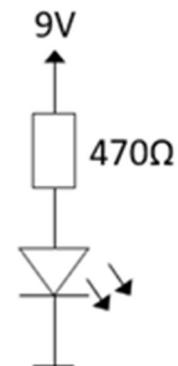


Abb. 8: Schaltung des Übungsbeispiels

Als Erstes werden die Drähte am Widerstand mit der Spitzzange in der Nähe des Bauteils um 90° umgebogen und in die Platine gesteckt. Der Körper des Widerstands soll direkt auf der Platine aufliegen.

Anschließend wird der Widerstand auf der Unterseite der Platine festgelötet: Die heiße Spitze des Lötkolbens an Platine und Bauteil ansetzen und Lot an der Lötspitze zuführen. Das Lot wird flüssig und legt sich

sowohl auf die Kupferseite als auch um das Bauteil.

Die Lötdauer sollte so kurz wie möglich sein, damit sich das elektronische Bauteil nicht unnötig erwärmt. Zu langes Erhitzen führt an der Platine dazu, dass sich die Kupferschicht vom Trägermaterial löst.

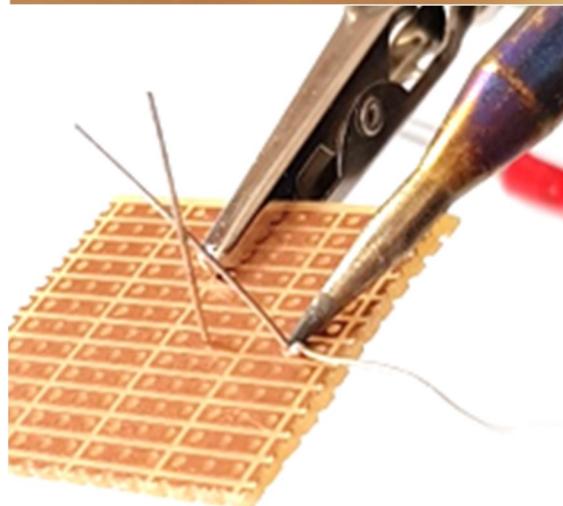


Abb. 9: Der Widerstand wird aufgesteckt (oben) und mit der Platine verlötet (unten)

Den Lötpunkt danach kurz abkühlen lassen und kontrollieren: Das Lot muss sich glatt um alle Metallteile gelegt haben.

Sind zwei Lötpunkte aus Versehen miteinander verbunden worden, muss das Lötzinn wieder entfernt werden. Dazu den Lötpunkt nochmals erhitzten und mit der Entlötspumpe überflüssiges Lot absaugen.

Bevor nun die LED aufgesteckt und verlötet wird, müssen noch die Drahtenden des Widerstands am Lötpunkt abgeschnitten werden.

In unserem Fall ist die Polung der Leuchtdiode egal, da am Batteriekasten ggf. die

Polung vertauscht werden kann. Als Halbleiter ist die LED beim Verlöten etwas temperaturempfindlicher als der Widerstand.

Nun müssen nur noch zwei Kabelenden angelötet werden: Ein kurzes Stück Isolierung mit der Abisolierzange oder dem Seitenschneider entfernen und die Litze oder den Draht von der Bestückungsseite aufstecken. Die Isolierung muss direkt an der Oberseite der Platine anliegen; sonst kann es später bei größeren Schaltungen zu unerwünschten Kurzschlüssen kommen.

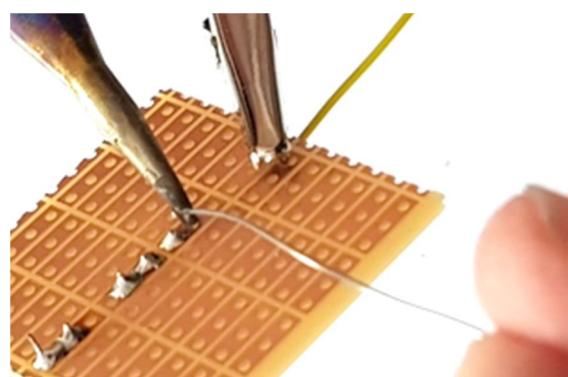


Abb. 10: Beim Löten des Kabelendes hält die „Dritte Hand“ Platine und Kabel fest

Das Kabel und die Platine werden am einfachsten in der dritten Hand eingeklemmt, so dass beide Hände für das Löten frei sind. Die Lötseite der Platine ist dann nach oben zu drehen und das Kabel zu verlöten.

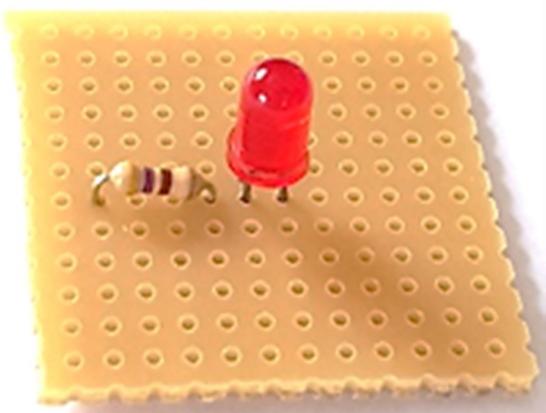


Abb. 11: Widerstand und LED auf der Platine

Bei Litzen muss ggf. das Ende des Kabels verzinkt werden, damit es besser in die Platine gesteckt werden kann. Dazu die

freien Kabelenden zwischen Daumen und Zeigefinger verdrillen, das Kabel in der dritten Hand befestigen und das Ende mit Lot überziehen.

Überstehende Kabelenden auf der Lötseite werden wieder mit dem Seitenschneider abgeschnitten. Dabei sollten die Platine oben und die Zange unten sein, so dass die Schnipsel nicht auf die Platine fallen und sich irgendwo verfangen können. Auf der Bestückungsseite sollte nochmals die Isolierung kontrolliert werden, um Kurzschlüsse vorzubeugen.

Sind beide Kabelenden eingelötet, so werden diese noch mit einem fischertechnik-Stecker konfektioniert. Die braucht ihr nur noch an der Batteriebox anzuschließen und die LED leuchtet.

Referenzen

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

Bildnachweis: Abb. 3, 4 und 5 entstammen mit freundlicher Genehmigung den Webseiten von [Ersa](#).

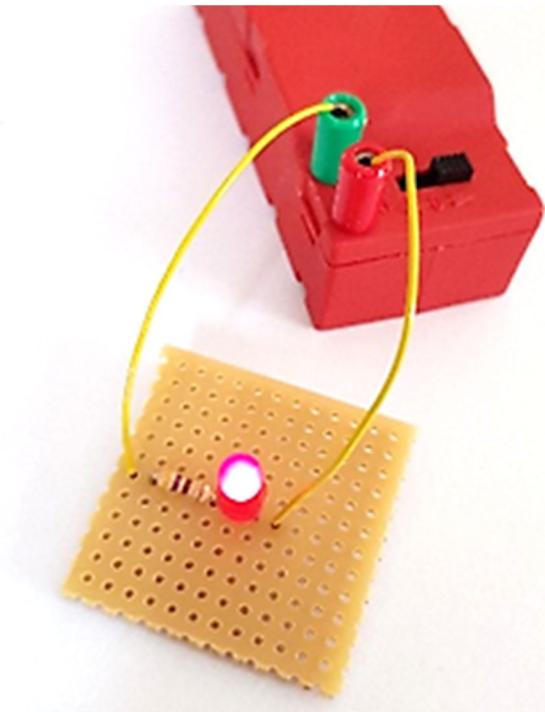


Abb. 12: Die fertige Schaltung

Modell

Urlaubskasten-Modell 5: Partner-Ventilator

Stefan Falk

Die Teile des Wohnzimmer-Dienstreisen-Urlaubs-Notfallkastens [1] aus ft:pedia 2016-1 genügen für diesen partnertauglichen Ventilator – der Sommer kommt ja bestimmt!



Federnocken und Baustein 15 auf einer Rast-aufnahmearchse 22,5 ([130593](#)).

Der Gag ist nun, dass der Ventilator um die Achse des Z40 selbst schwenkbar ist. So kommen beide fischertechnik-Fans abwechselnd in den Genuss des angenehmen Luftstroms. Fröhliches Kurbeln!

Referenzen

- [1] Stefan Falk: *Der Wohnzimmer-Dienstreisen-Urlaubs-Notfallkasten.* ft:pedia 2016-1, S. 31-36.
- [2] Stefan Falk: *Raffiniertes mit Achsen.* ft:pedia 2013-3, S. 38-41.

Zwei kühlungsbedürftige Personen können sich gegenübersetzen. Jeder steht eine Kurbel zur Verfügung. Deren lange Achsen treffen sich in der Mitte bei drei Rast-Kegelzahnrädern [35061](#). Die zentrale Achse geht zu einer Übersetzung vom Rast-Z20 aufs Rast-Z10 und von dort über Z40/Z10 zu den Ventilatorflügeln. Die Flügel sitzen mit je zwei Winkelsteinen 60° über Baustein 5 15·30 3N ([38428](#)),

Modell

Dominostein-Aufsteller

Thomas Püttmann

Millionen von Fernsehzuschauern konnten in den 80er und 90er Jahren lange Reihen aus Dominosteinen umkippen sehen. Auch heute noch faszinieren solche Kettenreaktionen große und kleine Kinder überall auf der Welt. Das Aufstellen der Reihen ist allerdings zeitaufwendig und nichts für Nervenschwache. Was liegt da näher, als ein ferngesteuertes fischertechnik-Fahrzeug zu bauen, das die Aufgabe zu einem reinen Vergnügen macht?

Dominosteineinreihen umkippen zu sehen fasziniert Kinder und Erwachsene gleichermaßen. Und die Faszination ist umso größer, je länger die Reihen und je aufwendiger die Effekte sind. Wer aber schon einmal versucht hat, interessante Reihen mit mehr als 200 Steinen aufzustellen, weiß, wie anstrengend es ist, stets einen funktionierenden Abstand zu finden, und wie frustrierend es sein kann, wenn man versehentlich einen Stein umwirft und damit das Ergebnis der letzten halben Stunde in wenigen Sekunden zunichte macht.

Natürlich gab es schon immer Aufstellhilfen in Form von Kämmen. Wer sich aber auch nur etwas für Technik begeistert, wird das im Jahr 2010 von Matthias Wandel auf [YouTube](#) vorgestellte LEGO-Fahrzeug viel spannender finden. Aus einem horizontalen Magazin schiebt es die Steine während der Fahrt mit einem Exzentermechanismus aus. Allerdings kann es nur geradeaus fahren. Hier wird nun eine ferngesteuerte fischertechnik-Variante präsentiert, die die Steine auch in Kurven oder Schlangenlinien aufstellen kann.

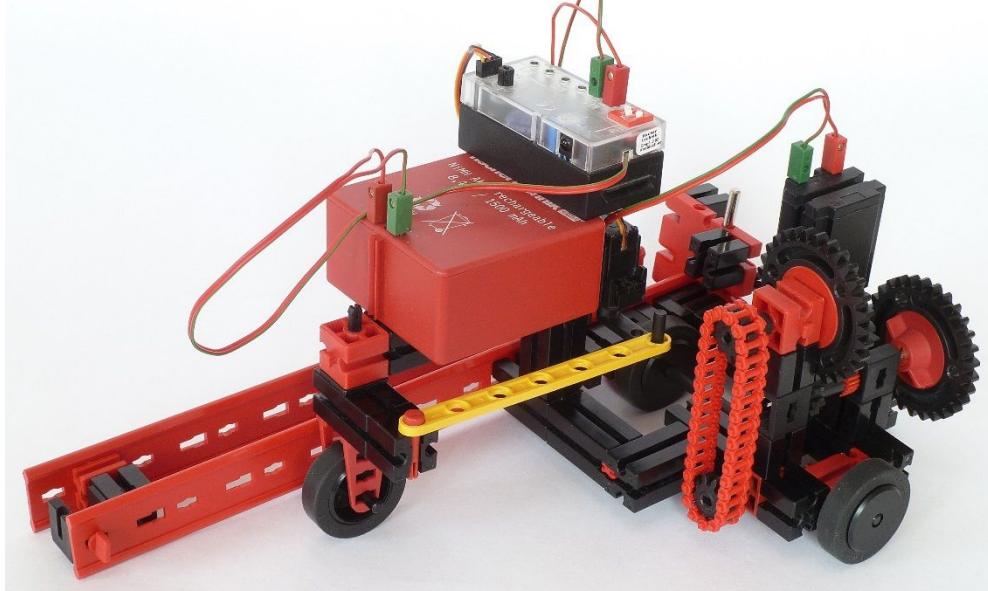


Abb. 1: Der ferngesteuerte Dominostein-Aufsteller

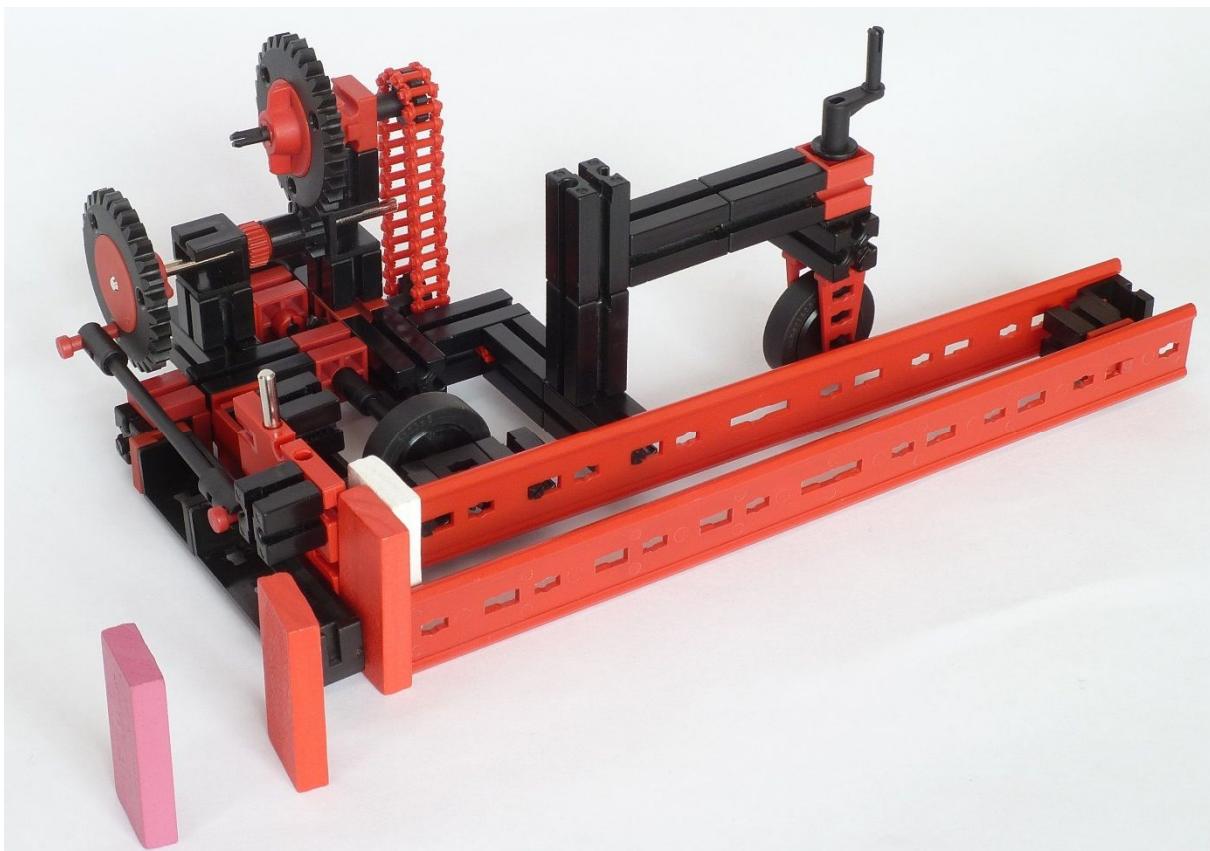


Abb. 2: Der unmotorisierte Dominostein-Aufsteller in Funktion

Funktion

Wie das Fahrzeug funktioniert erkennt man am besten anhand von Abb. 2. Die Dominosteine befinden sich dicht gestapelt zwischen den beiden roten Laufschienen. Sie werden mit dem Fahrzeug mitbewegt und gleiten dabei über den Boden. Auf der Abbildung ist das Magazin gerade fast leer, um einen besseren Blick auf das Innere des Fahrzeugs zu ermöglichen.

Die Steine werden von der roten Grundplatte 45 x 45 ausgeschoben, die an der hinteren Kante des Fahrzeugs in einem Schlitz geführt wird. Die Schiebebewegung ergibt sich durch eine Schubstange aus der Drehbewegung des Zahnrad Z30 ganz links in Abb. 2.

Am anderen Ende der Welle, auf der dieses Zahnrad Z30 sitzt, befindet sich ein Z10, das ein weiteres Zahnrad Z30 kämmt. Dieses zweite Z30 dreht sich daher nur ein Drittel so schnell wie das erste. Über zwei

Rastritzel Z10, die mit einer Kette verbunden sind, und zwei Kegelzahnräder wird die Drehbewegung des zweiten Z30 weitergegeben an die Vortriebsachse des Fahrzeugs. Auf dieser Achse sitzt, quasi in der Mitte des Fahrzeugs, das Vortriebsrad. Fahren und Schieben sind also insgesamt so miteinander verkoppelt, dass während einer vollen Umdrehung des Vortriebsrads drei Dominosteine in gleichmäßigem Abstand ausgeschoben werden.

Man kann das Fahrzeug per Hand schieben und lenken. Vor allem Kinder haben aber noch mehr Freude an der motorisierten und ferngesteuerten Version aus Abb. 1.

Zusammenbau

Wir beginnen den Zusammenbau mit der inneren der beiden roten Laufschienen. An dieser Laufschiene bauen wir zwei schwarze Bausteine 15 an und schieben einen schwarzen Winkelträger 15 mit zwei Zapfen dazwischen, siehe Abb. 3.

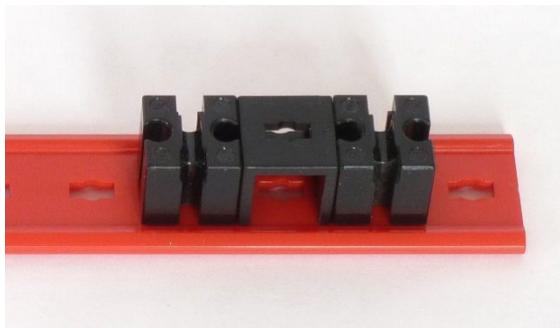


Abb. 3: Laufschiene mit Anbau

Eine Grundplatte 45 x 45 schiebt den jeweils letzten Dominostein aus der Reihe nach außen. Diese Platte muss an der Rückseite des Fahrzeugs an einer glatten Fläche entlang gleiten. Abb. 4 zeigt die glatte Hinterkante des Fahrzeugs mit einer Bauplatte 4 x 1.

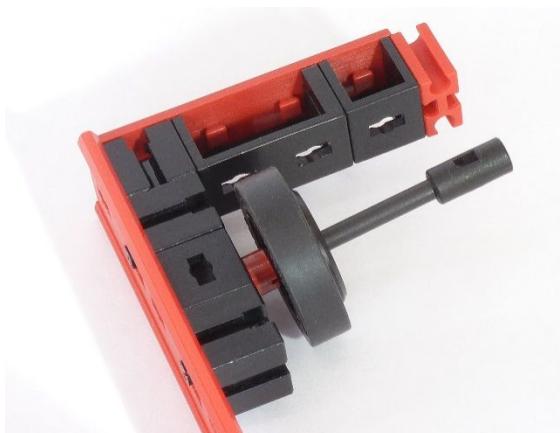


Abb. 4: Angebaute Hinterkante und eingesteckte Vortriebsachse

Das Fahrzeug wird durch ein Rad 23 mit einem Reifen 32 vorwärtsbewegt. In diesem Rad steckt eine Rastachse 45.

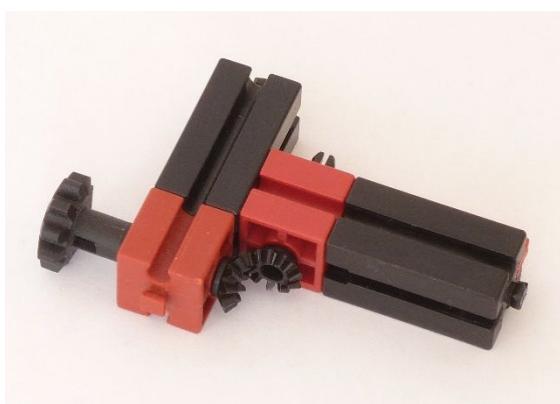


Abb. 5: 90°-Umsetzung der Drehbewegung

Um das Herausschieben der Bausteine mit der Vorwärtsbewegung des Fahrzeugs zu verkoppeln, muss die Drehbewegung der Antriebsachse um 90° gedreht werden. Dazu dient das Kegelzahnradgetriebe aus Abb. 5.

Ein zweites Rad in der Flucht der Antriebsachse stützt das Fahrzeug auf der anderen Seite, siehe Abb. 6.

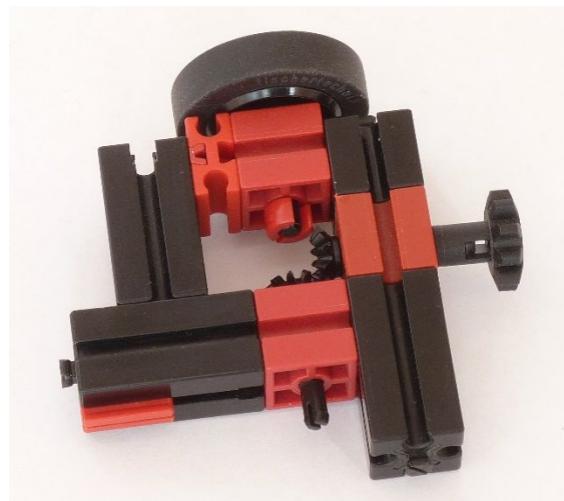


Abb. 6: Unterbau auf der linken Seite des Fahrzeugs

Mit Hilfe dreier zusätzlicher Bausteine 30 und eines Verbindungsstücks 15 wird der Unterbau des Fahrzeugs fertiggestellt, siehe Abb. 7.

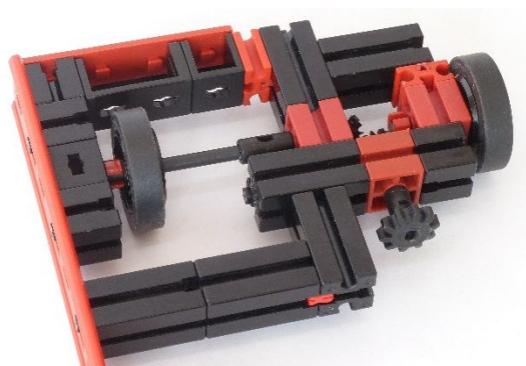


Abb. 7: Vollständiger Unterbau

Damit die Dominosteine einander sicher umwerfen, dürfen die Abstände zwischen den Steinen nicht zu klein und nicht zu groß sein. Unsere Steine haben eine Abmessung von 45 mm x 20 mm x 6 mm. Experimente ergaben, dass drei bis vier Dominosteine

aufgestellt werden müssen, wenn das Vortriebsrad unseres Fahrzeugs eine ganze Umdrehung macht. Das Rad, das über die Pleuelstange die Steine ausschiebt, dreht sich daher dreimal so schnell wie die Vortriebsachse.

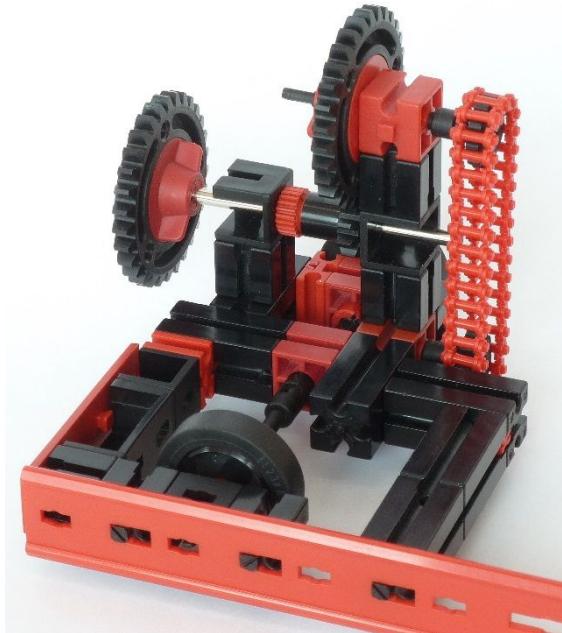


Abb. 8: Die Verkopplung von Vorwärts- und Ausschubbewegung

Genau dies leistet der Aufbau in Abb. 8. Über eine Kette wird die Drehbewegung auf eine obere Welle übertragen, auf der ein Z30 sitzt, das ein Z10 auf einer mittleren Welle kämmt. Diese mittlere Welle dreht sich daher dreimal so schnell wie die Vortriebswelle. Auf ihr sitzt ein weiteres Z30, das für das Ausschieben der Dominosteine verantwortlich ist.

In Abb. 9 sieht man die Führung der nicht abgebildeten schiebenden Grundplatte 45 x 45 hinten am Fahrzeug. Der Verbund aus einem Winkelträger 30 und einem Winkelträger 60 sollte sich möglichst dicht über dem Boden befinden, ohne zu schleifen. Der Schlitz für den Schieber sollte eine konstante Breite besitzen. Das klappt bei mir mit gelben Winkelträgern deutlich besser als mit schwarzen – leider, denn die schwarzen sehen an dieser Stelle meines Erachtens deutlich besser aus.

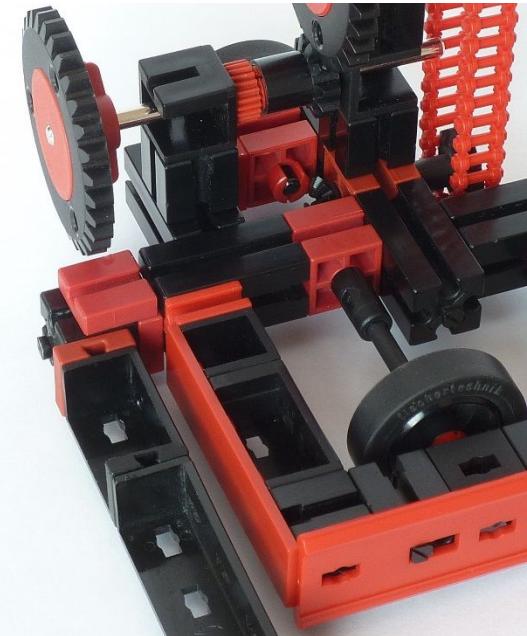


Abb. 9: Führung des Schiebers

Die Schubstange besteht aus einer Rastachse 75, zwei Rastadapters und zwei roten Rastachsen 20. Die schiebende Platte ist eine Grundplatte 45 x 45, an der als Gewichte zwei Bausteine 15 angesteckt und eine Achse 30 eingesteckt ist. Dadurch wird ein Verkanten während des Schiebens verhindert. Wichtig ist, dass die Rastachse 20 nur so weit in die Bohrung der Grundplatte gesteckt wird, dass sie nicht auf der anderen Seite hinausragt. Ansonsten wird nicht nur ein Dominostein ausgeschoben, sondern ein zweiter mitgenommen.

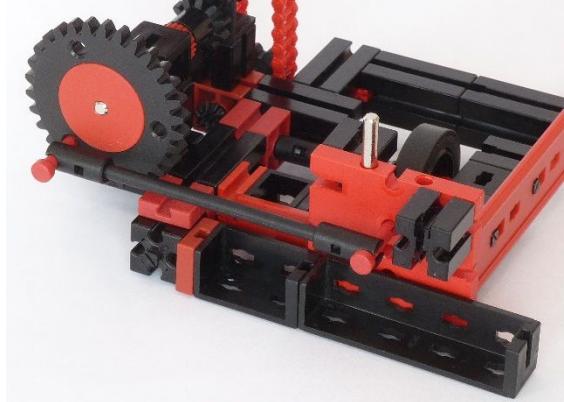


Abb. 10: Schubstange und schiebende Grundplatte 45 x 45 mit Gewichten

Die Dominosteine werden außen durch eine zweite Laufschiene geführt. In Abb. 11 ist zu sehen, mit welchen Zwischenteilen diese an der inneren Laufschiene befestigt wird.



Abb. 11: Die äußere Führungsschiene der Dominosteine

Das Vorderrad besteht aus einem Rad 23 mit einem Reifen 32, einer K-Achse 30, zwei Kupplungsstücken 3, einem Baustein 30 und einer Rastachse mit Platte.

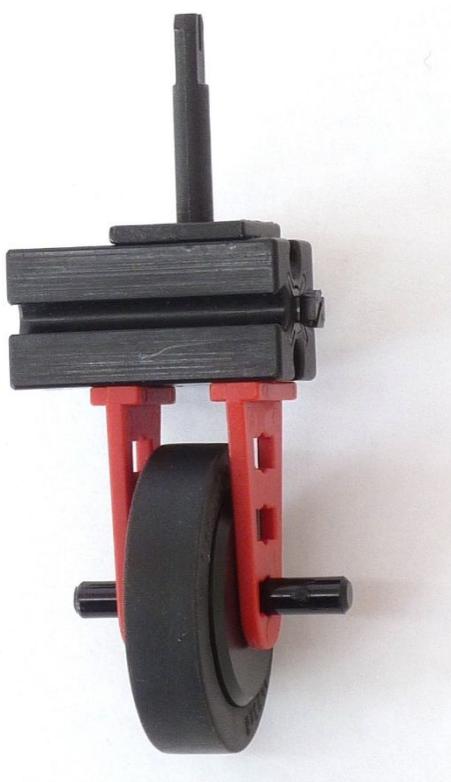


Abb. 12: Der Aufbau des Vorderrads

Das Fahrzeug wird mit dem IR-Controller ferngesteuert (siehe Abb. 1).

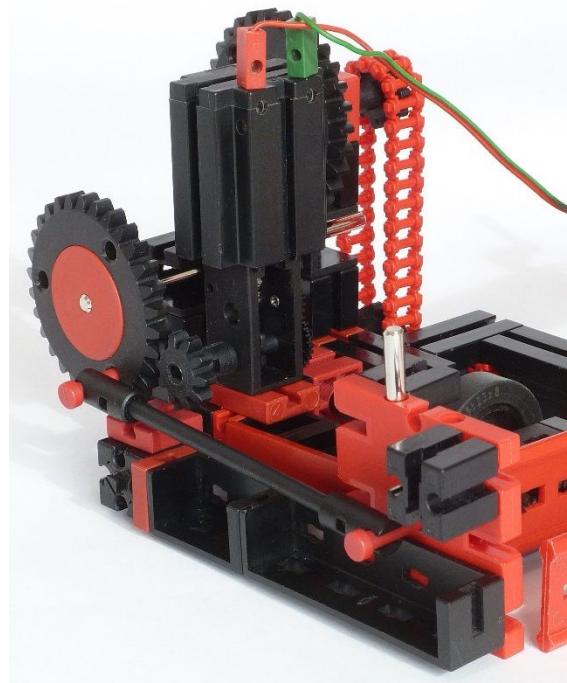


Abb. 13: Der Anbau des Motors

Dazu werden zwei weitere Bausteine 15 ergänzt: einer zur Verlängerung des Lenkarms über dem Vorderrad und einer hinter dem Akku. An dem zweiten Baustein 15 wird dann auch der Empfänger angesteckt. Der Servo wird an der Rückseite der vertikalen Bausteinsäule angebaut und lenkt das Fahrzeug über eine I-Strebe 90 als Schubstange.

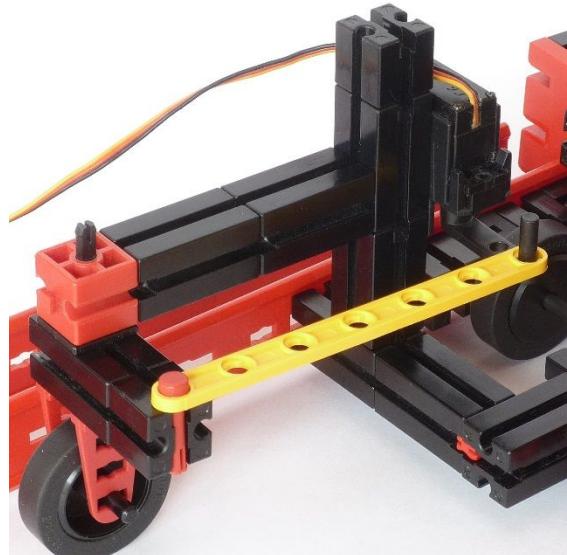


Abb. 14: Die ferngesteuerte Lenkung mit Servo und I-Strebe 90 als Schubstange

Bemerkungen

Der Aufsteller kommt ohne Änderungen mit Dominosteinen verschiedener Größen zurecht. Selbst Jengasteine lassen sich damit aufstellen. Falls die Steine kürzer sind als 45 mm, sind die Abstände zwischen den Dominosteinen allerdings zu groß. Dann sollte das obere Z30 gegen ein Z40 ausgetauscht werden. Natürlich muss dabei auch ein Baustein 7,5 eingefügt und die Kette verlängert werden.

Mit fischertechnik lassen sich auch viele tolle Effekte in die Dominostrecken einbauen. Für anregende Ideen verweise ich auf das Video [2]. Kombinationen aus Dominostrecken und Kugelbahnstrecken sind besonders attraktiv.

Zum Austesten habe ich das Fahrzeug zweimal in unseren Kindergarten mitgenommen, da dort viel freie Fläche und genügend Dominosteine vorhanden sind. Die Kinder haben begeistert damit gespielt und gemeinschaftlich lange Reihen aufgestellt. Durch den offenen Aufbau lassen sich die Funktion des Fahrzeugs und das Zusammenspiel der einzelnen technischen Komponenten (Räder, Getriebe, Schubmechanismus, Lenkung, Fernsteuerung) auch kleineren Kindern schon sehr gut erklären.

Vielen Dank an dieser Stelle an das freundliche Team des Kindergartens Liebfrauen in Altenbochum für die Unterstützung!

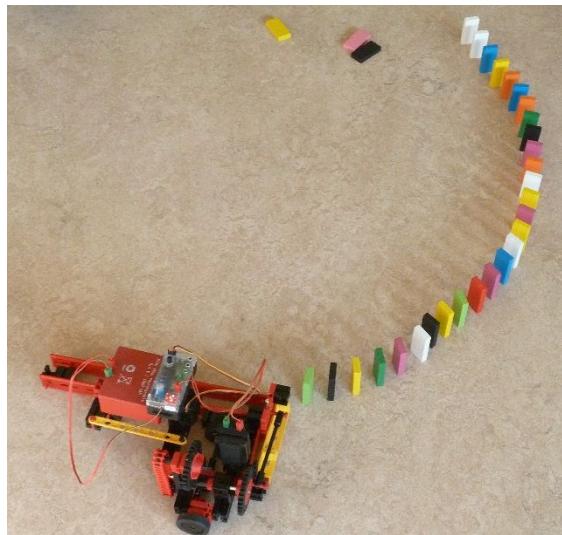


Abb. 15: Das Fahrzeug im Einsatz – das Magazin ist leer und muss neu aufgefüllt werden

Links

- [1] Matthias Wandel: [Lego domino row building machine](#), Youtube.
- [2] hevesh5, kaplamino: [Unconventional Domino Tricks](#), Youtube.
- [3] Thomas Püttmann: [Dominosteine aufstellen mit fischertechnik](#), Youtube.
- [4] Thomas Püttmann: [Liste der Einzelteile zum Bau des Dominosteine-Aufstellers](#), Downloadbereich der ft-Community.

Modell

Schiebetüren

Stefan Busch

Wie spannend das Thema „Türen“ ist, hat Stefan Busch in ft:pedia 3/2017 mit der Vorstellung von Schwenktüren gezeigt. Damit ist das Thema natürlich noch lange nicht erschöpfend behandelt – daher folgt nun eine Fortsetzung über Schiebetüren ...

Einleitung

Das typbestimmende Merkmal von Schiebetüren besteht im horizontalen Verschub des Türblatts.¹ Der damit einhergehende minimale Raumbedarf in Durchlassrichtung stellt einen großen Vorteil gegenüber Schwenktüren dar und macht diesen Türtyp besonders für Verkehrsmittel interessant.¹

Einfache Schiebetüren



Abb. 1a: Außenschiebetüre eines Lieferwagens



Abb. 1b: Trag- und Führungsschiene

Die einfachste Ausführung einer Schiebetür besteht aus einer Tragschiene, dem daran verschieblich gelagerten Türblatt sowie meistens einer oder mehrerer Führungsschienen. Trag- und Führungsschiene sowie das Türblatt liegen hier vor der Fahrzeugwand (Abb. 1a/b).

Bei dem in Abbildung 2 und 3 gezeigten Modell erfolgt die Verschiebung des Türblatts anders als im Original mittels Spindelantrieb, der das Blatt gleichzeitig stabilisiert, sodass auf eine Führungsschiene verzichtet werden kann.

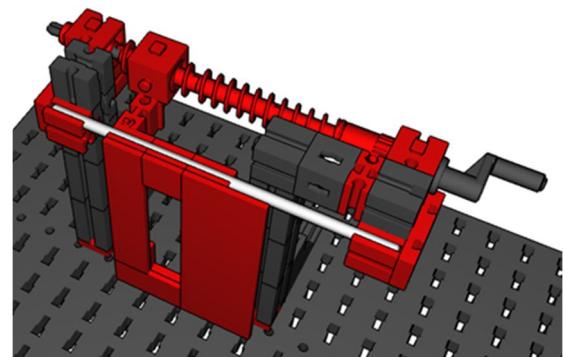


Abb. 2: Einfache Schiebetür in fast geschlossenem Zustand. Der Winkelträger 60, der die Pfosten des Portals verbindet (siehe Abb. 3), ist hier weggelassen, um die Verbindung von Türblatt und Spindelmutter zeigen zu können.

¹ Die Definitionen der zur Beschreibung von Türen wichtigen Begriffe finden sich im vorangegangenen Artikel über Schwenktüren [1].

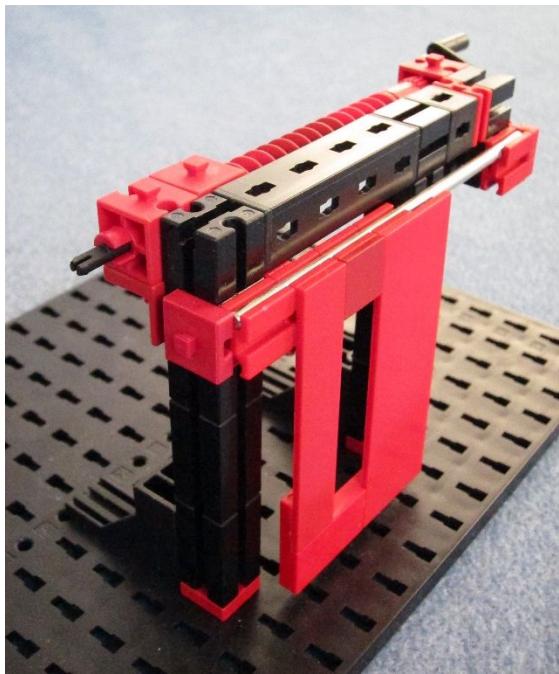


Abb. 3: Außenschiebetür² im Modell

Dem Vorteil einer unkomplizierten Konstruktion steht gegenüber, dass das Türblatt in Schließendlage nicht in der Wandebene liegt (gut zu sehen in Abb. 1b). Dies ist unter anderem bei Fahrzeugen wegen des Windwiderstands und aus ästhetischen Gründen häufig unerwünscht.

Schwenkschiebetüren

Ein in Schließendlage bündig mit der Außenwand abschließendes Türblatt muss beim Öffnungsvorgang zunächst nach außen bewegt werden, bevor es verschoben werden kann. Hierzu existiert eine Vielzahl technischer Lösungen, von denen hier nur einige beschrieben werden können.

Die gebogene Tragschiene

Die einfachste Möglichkeit besteht darin, Trag- und Führungsschienen nicht linear auszuführen, sondern so zu formen, dass das Türblatt beim Verschub entlang des durch die Schienen vorgegebenen Wegs

automatisch aus der Wandebene herausgeleitet wird. Charakteristisch für diese Konstruktion ist eine etwa mittig in der Außenwand liegende Schiene zur Führung der Nebenschließkante des Türblatts. Anwendung finden solche Türen besonders bei Lieferwagen (Abb. 4).



Abb. 4: Gebogene Führungsschiene einer Schiebetür

Schiebetüren mit gebogenen Trag- und Führungsschienen stellen gewissermaßen den Übergang zwischen den einfachen Schiebetüren und den im Folgenden beschriebenen Schwenkschiebetüren dar, bei denen das Türblatt durch eine separate Schwenkbewegung ausgestellt wird. Verschub und Schwenken des Türblatts werden dabei durch einen einzigen Antrieb bewirkt, um die richtige Abfolge der Bewegungen zu gewährleisten. Die Realisierung eines mög-

² Bei Außenschiebetüren liegt das Türblatt vor der Wandebene, bei sogenannten Taschenschiebetüren dahinter. Hier gleitet das Türblatt bei der Öffnung in ein Tasche.

lichst einfachen Synchronisierungsgtriebes im gewählten Maßstab stellt die Haupt schwierigkeit beim Bau der Schwenkschiebetüren dar.

Ausschwenken des Türblatts

Schwenkschiebetüren finden beispielsweise als Straßenbahntüren Verwendung (Abb. 5). Beim hier gezeigten Typ wird das Türblatt von einem Schwenkarm getragen, der sich entlang einer mit dem Portal verbundenen Tragschiene verschieben lässt. So kann das Türblatt zu Beginn des Öffnungsvorgangs ausgelenkt und anschließend parallel zur Außenwand in die Offenendlage verschoben werden. Angetrieben wird dabei lediglich der Verschub des Gleitlagers, die Steuerung des damit verbundenen Schwenkarms erfolgt über eine sogenannte Kulissenführung: Ein am Schwenkarm befindlicher Kulissenstein (im gezeigten Beispiel eine Rolle) wird beim Verschub in einer Bahn, der Kulisse (hier ein U-Profil), entlang eines durch die Form dieser Bahn vorgegebenen Weges geführt und steuert dadurch die Bewegung des Schwenkarms.³



Abb. 5: Straßenbahntür

In Abb. 5 ist das Türblatt mit seinem Trag arm (ganz links) zu erkennen, daran anschließend der Schwenkarm mit einer durch die Kulisse geführten Rolle und schließlich

das Gleitlager mit der Tragschiene. Der Antrieb ist nicht zu sehen.

Zur Führung des Türblatts befinden sich an dessen Innenseite zwei Schienen, in die Rollenlager eingreifen, welche wiederum über Schwenkkarme an einer vertikalen Drehsäule im Portal gelagert sind (Abb. 6). Diese Anordnung erlaubt eine glatte, nicht durchbrochene Außenwand am Fahrzeug.

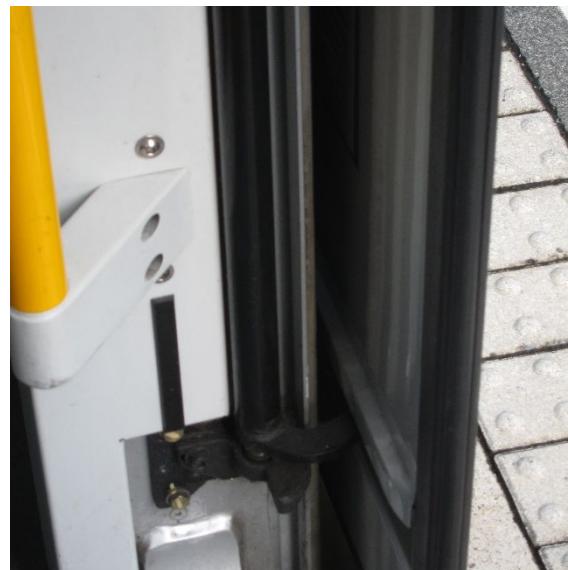


Abb. 6: Straßenbahntür: Unterer Teil der vertikalen Drehsäule mit Rollenlager sowie die zugehörige Führungsschiene im Türblatt; der obere Teil der Säule mit dem Betätigungshebel ist in Abb. 5 hinter dem Schwenkarm zu sehen.

Abb. 7 zeigt das Türblatt und dessen Führung im Modell. Vorbildgerecht befinden sich die Führungsschienen am Türblatt (Metallachsen 60), werden hier aber von Gleitlagern (Bausteinen 7,5) gehalten. Um deren Bewegung nicht zu behindern, dürfen die Bauplatten, die das Türblatt bilden, im Bereich der Führungsschienen keine Zapfen aufweisen. Leider fiel dieser Bedingung der „Einbau“ eines Fensters im Türblatt zum Opfer.

³ Kulissenführungen fanden sich vor allem bei Steuerungen von Dampfmaschinen, speziell bei Dampfloks [3].

Die Schwenkarme sind durch Mitnehmer und Klemmbuchsen kraftschlüssig mit einer Achse 150 verbunden.

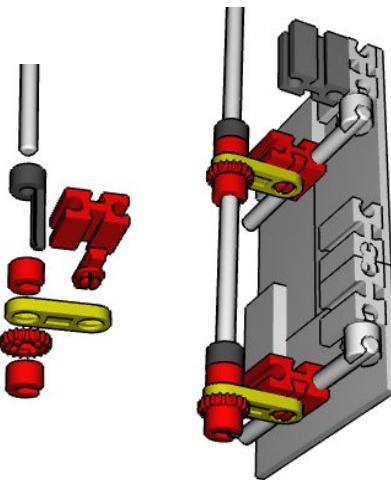


Abb. 7: Türblatt mit Führung (rechts): Der Tragarm des Türblatts ist dunkelgrau dargestellt. Aufbau der Schwenkarme (links).

Bei der Konstruktion des Schwenkarms, der den Tragarm des Türblatts mit der Tragschiene verbindet, ist sicherzustellen, dass das Türblatt ausschließlich um die Hochachse geschwenkt werden kann, da es andernfalls verkantet und die Führungsschienen nicht mehr leicht durch die Lager gleiten.

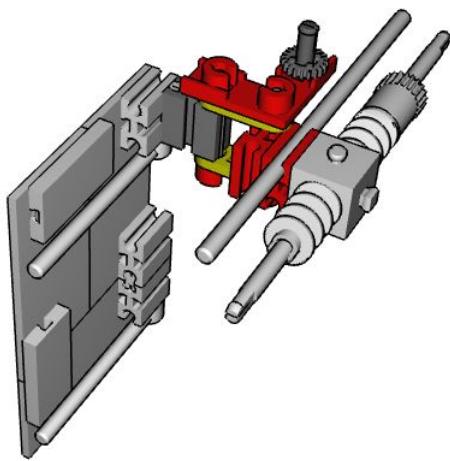


Abb. 8a: Tragschiene mit Gleitlager, Schwenkarm und Tragarm des Türblatts

Um die nötige Bauhöhe für ein verwindungssteifes Gelenk zu erhalten, wird der Schwenkarm daher aus zwei ober- und unterhalb des Tragarms angeschlagenen

Streben 15 gebildet; eine L-Lasche dient als Antriebshebel (Abb. 8).

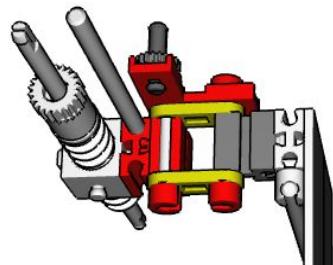


Abb. 8b: Besonders zu beachten ist der Einbau der L-Lasche und der V-Achse (dunkelgrau) am freien Arm der Lasche

Die Steuerung des Schwenkarms erfolgt durch eine am freien Arm des Antriebshebels mittels Riegelscheibe Z20 fixierten V-Achse 17, dem Kulissenstein. Zu beachten ist, dass die Stege der L-Lasche nach oben weisen und die V-Achse nur minimal unterhalb der Lasche hervorsteht, damit in eingeschwenkter Position ein ausreichender Abstand zwischen Hebel und Antriebsspindel bleibt.

Tragschiene und Spindelgetriebe sind durch zwei Träger am Portal befestigt, die vertikale Drehsäule reicht durch den Querträger des Portals hindurch und wird durch zwei weitere Achslager an der Portalstütze gehalten (Abb. 9).

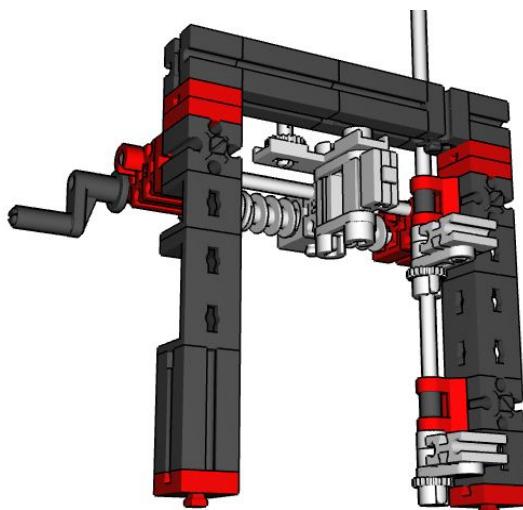


Abb. 9a: Portal, Vorderansicht

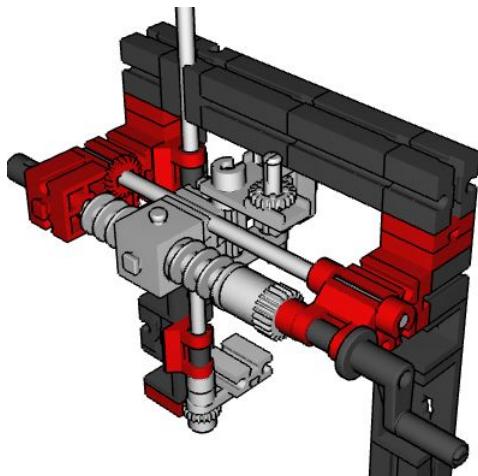


Abbildung 9b: Ansicht der rückwärtigen Träger von Tragschiene und Antriebsspindel

Am Modell wird deutlich, dass die Tragarm und Türblatt verbindende Achse nicht mit der Drehachse der Gleitlager zusammenfallen darf, da dann die Lage des Türblatts unbestimmt ist, was dessen freie Drehung um diese gemeinsame Achse erlaubt. Im Original wird diese Instabilität durch eine spezielle Ausführung der Rollenlager vermieden.

Zur Führung des Kulissensteins (V-Achse, Abb. 8) und damit zur Steuerung des Aus- oder Einschwenkens des Türblatts dient die am Querträger des Portals befestigte Kulissenführung (Abb. 10). In der Führung ist die als Kulissenstein fungierende V-Achse 17 zu erkennen.

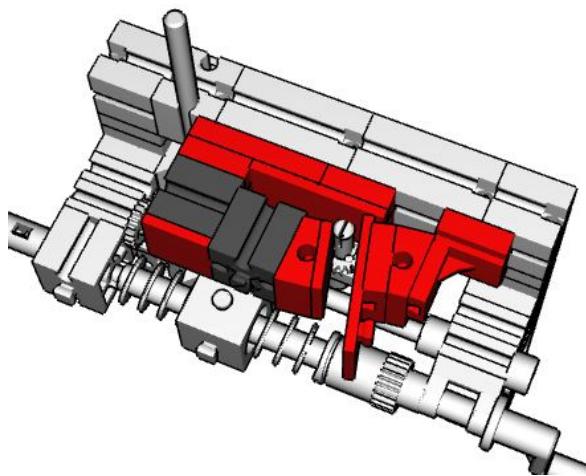


Abbildung 10: Kulissensteuerung

Das Ausschwenken der Gleitlager der Führungsschienen im Türblatt erfolgt durch ein federbelastetes Hebelgetriebe (Abb. 11): Beim Öffnen der Tür gibt ein Mitnehmer am Spindelgetriebe einen Hebel frei, der durch eine Feder (Gummiband) in seine Offenendlage bewegt wird. Das angeschlossene Hebelwerk stellt über die vertikale Schwenksäule die Schwenkkarre der Führungslager aus. Beim Zusammenbau des Hebelwerks ist zu beachten, dass dessen Konstruktion nicht sauber gelungen ist: Das Gelenk an der Drehsäule darf nicht zu stramm sitzen, da es auch eine geringe vertikale Drehung ausgleichen muss, die durch die Bewegung des vertikalen Hebels zustande kommt. Hier wäre ein Kugelgelenk im fischertechnik-Sortiment hilfreich.

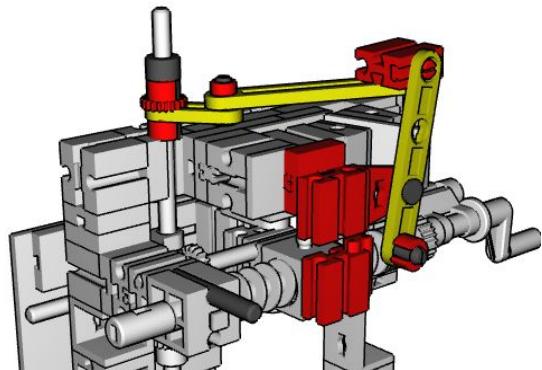


Abbildung 11a

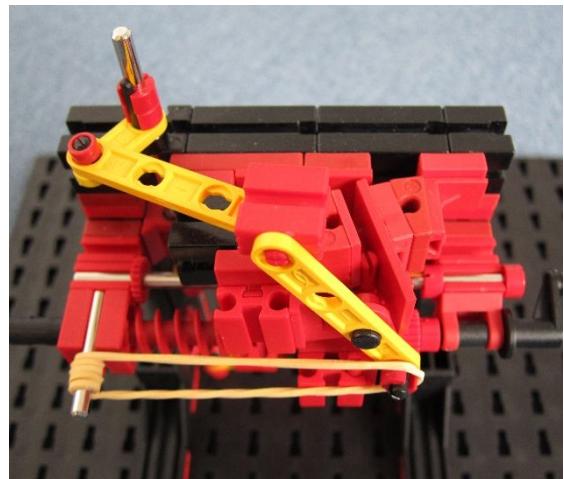


Abbildung 11b: Hebelwerk zur Betätigung der vertikalen Schwenkachse

Am Modell ist das Gummiband zwischen Hebel und Widerlager sowie die Wicklungen am Widerlager zu erkennen. Ein zufriedenstellendes Funktionieren dieses Modells erfordert eine optimale Anordnung verschiedener Komponenten:

- Die Positionierung der Kulissenführung am Querträger des Portals entscheidet darüber, wann der Schwenkarm des Türblatts einschwenkt und somit auch über die Schließendlage der Hauptschließkante des Türblatts. Entsprechend bestimmt die Position des antriebsseitigen Hebels des Hebelgetriebes und die Stellung des Hebels der vertikalen Drehsäule die Schließendlage der Nebenschließkante.

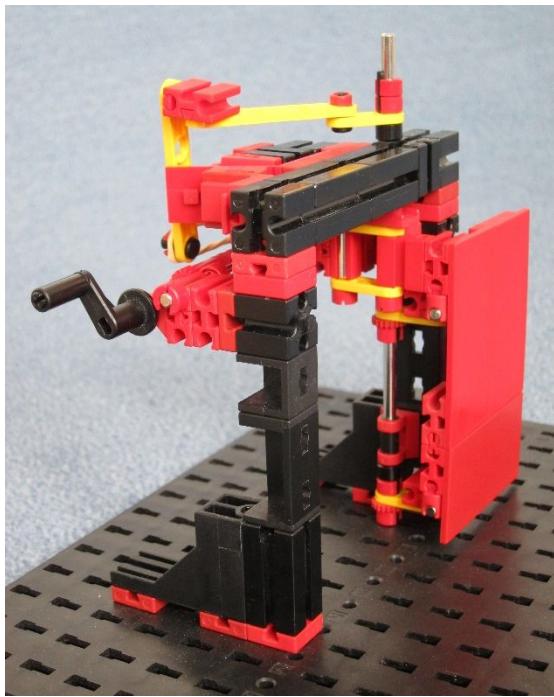


Abbildung 12: Vorderansicht des Modells

- Die Zugspannung der Feder muss einerseits ausreichen, um die Führung zu Beginn des Öffnungsvorgangs vollständig auszuschwenken, darf andererseits aber nicht zu hoch sein, um den Schließvorgang nicht zu behindern. Am besten lässt sich dies mit einem Gummiband realisieren, da sich hier die Spannung durch die Anzahl der Wicklungen am

Widerlager (Achse 30 am Portal) gut einstellen lässt (Abb. 11b).

Achtung: Das Auffinden der optimalen Abstimmung erfordert hier mehr Geduld als bei den anderen vorgestellten Türmodellen. Die in den Abbildungen gezeigten Anordnungen sollten aber einen guten Ausgangspunkt für die Feinjustierung zeigen.

Eine interessante Variation dieses Türtyps zeigt Abb. 13. Hier wird der Tragarm der Tür um eine horizontale Achse geschwenkt. Deutlich erkennbar ist der rote, kulissengeführte Tragarm des Türblatts sowie die um eine horizontale Achse drehende Schwinge, die Tragarm und Tragschiene verbindet. Unter der Tragschiene ist die Antriebsspindel sichtbar.



Abbildung 13: Oberer Teil einer Schwenkschiebetür eines Doppelstockwagens von Bombardier

Ausschwenken der Tragschiene

Der beim Modell des vorstehend beschriebenen Schwenkschiebetürtyps erforderliche aufwendige Antrieb der vertikalen Drehachse und die Schwierigkeiten mit der stabilen Positionierung des Türblatts in Offenendlage lassen sich vermeiden, indem das Türblatt gemeinsam mit der gesamten Tragschiene ausgeschwenkt wird.

Das Patent EP0920591A3 [4] beschreibt hierzu die Verwendung eines ebenen viergliedrigen Koppelgetriebes, gebildet aus Portal (Gestell), zwei gegenläufigen Schwenkarmen (Schwingen) und der Tragschiene mit ihren Halterungen (Koppel).

Wichtig für das hier beschriebene Prinzip ist die Anordnung der Schwingen zueinander. Das Gestell des Koppelgetriebes ist in Abb. 14 nicht dargestellt.

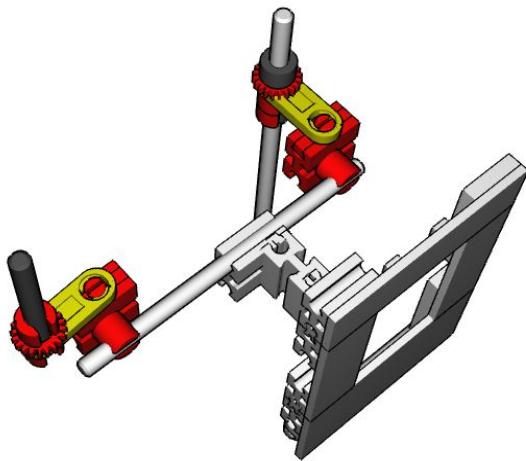


Abb. 14: Modell einer Viergelenkkette zur Auslenkung einer Tragschiene

Durch die zueinander rechtwinklige Anordnung der Schwingen werden Tragschiene und Türblatt zu Beginn des Öffnungsvorgangs schräg ausgestellt, befinden sich nach Abschluss des Schwenkvorgangs aber wieder parallel zur Außenwand. Dabei kann sich die Tragschiene je nach Ausführung noch innerhalb des Fahrzeugs befinden oder über die Seitenwand hinausgeschwenkt sein. Wie schon im vorgenannten Beispiel wird die Schwenkbewegung durch eine Kulissenführung bewirkt, direkt angetrieben wird wieder nur der Verschub des Gleitlagers auf der Tragschiene. Auch hier werden zusätzlich Rollenlager ausgeschwenkt, die in Führungsschienen im Türblatt eingreifen.

Zwar war beim Vorbild kein Beispiel dieses Türtyps zu finden, das eine ungehinderte Ansicht aller Komponenten erlaubte, die Anordnung der einsehbaren Teile der Tür eines ICEs (Abb. 15) und die Bewegung ihres Türflügels weisen aber darauf hin, dass diese Tür nach dem hier beschriebenen Prinzip arbeitet.

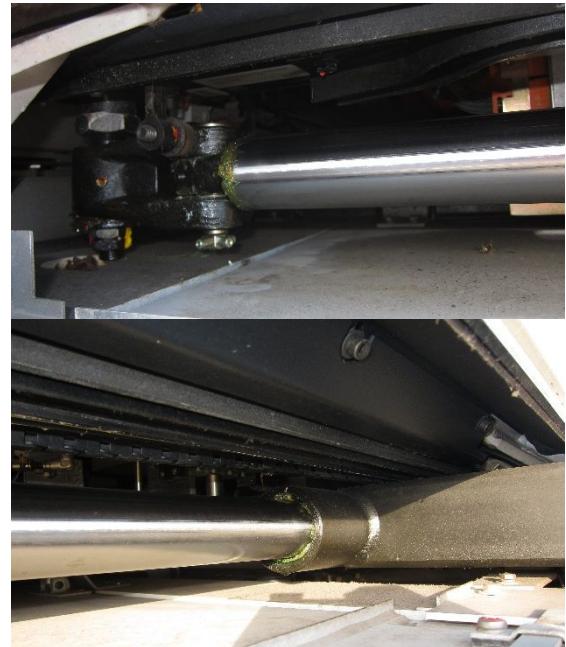


Abb. 15: Komponenten der Tür eines ICE-Wagens der Baureihe 403 (ICE 3): Tragschiene mit dem der Hauptschließkante zugeordneten Schwenklager und dem gebogenen Teil der Kulisse (oben), Gleitlager und Tragarm des Türblatts sowie gerader Teil der Kulisse (unten)

Die Beschreibung des Modells beginnt wieder beim Türblatt (Abb. 16). Abweichend zum Original kann hier auf die obere Führungsschiene verzichtet werden, was wieder den ‚Einbau‘ eines Fensters ins Türblatt ermöglicht.

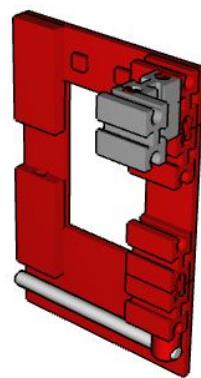


Abb. 16: Türblatt mit integrierter Führungsschiene; Tragarm und Gleitlager sind grau dargestellt

Abb. 17 zeigt die Trag- und Führungselemente der Tür. Das Gestell des Koppelgetriebes ist in dieser Darstellung der besseren Übersichtlichkeit wegen nicht dargestellt. Zur richtigen Ausrichtung des Türblatts müssen die der Nebenschließkante zugeordnete Schwinge und der Schwenkarm des Führungslagers jederzeit in die gleiche Richtung weisen. Zur Synchronisation der Schwenkbewegungen dient im Modell ein Getriebe aus zwei S-Riegelscheiben und einem Klemmring Z36. Trotz seines vergleichsweise geringen Raumbedarfs ließ sich das Getriebe nicht wie beim Vorbild oberhalb der Türöffnung unterbringen. Das Getriebe kann bei diesen Türtyp deutlich einfacher ausgeführt werden als beim vorherigen Modell, da alle Schwenkachsen ortsfest sind, also nicht verschoben werden müssen.

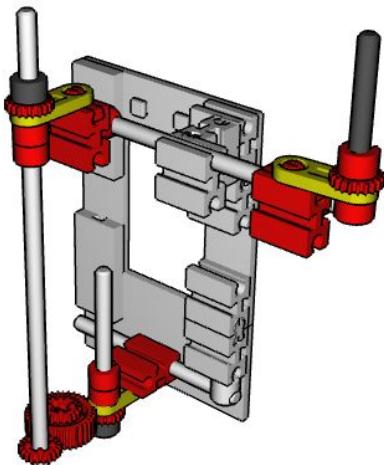


Abb. 17: Trag- und Führungselemente bestehend aus Koppelgetriebe (oben) und schwenkbarem Gleitlager (unten) sowie dem Getriebe zur Synchronisierung der Schwenkbewegungen

Die beweglichen Komponenten werden über drei Achsen im Portal gelagert (Abb. 18). Im Schwenkbereich des Koppelgetriebes bestehen die Portalstützen aus je einem Statikwinkelstein 30, um Platz für die Schwingen zu schaffen. Die der Nebenschließkante zugeordnete Portalstütze weist unten zusätzlich eine – nicht Vorbild

gerechte – Aussparung für den Schwenkarm des Führungslagers auf. Am Querträger des Portals dient ein Baustein 5 als Anschlag zur Begrenzung des Schwenkbereichs der der Hauptschließkante zugeordneten Schwinge des Koppelgetriebes (Abb. 18 oben und 19). Bei genauer Be trachtung findet sich ein solcher Anschlag auch im Original (Abb. 15).

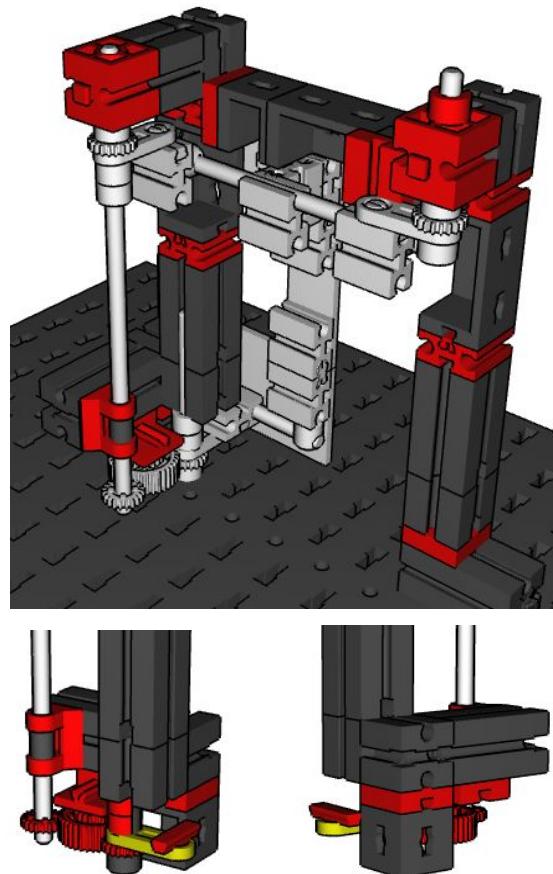


Abb. 18: Portal. Gesamtansicht (oben) und Detail der Aussparung für den Schwenkarm von vorn (links) und von hinten (rechts); das Gleitlager ist nicht abgebildet

Die Steuerung der Schwenkbewegung erfolgt vorgbildgerecht durch eine Kulissenführung (Abb. 19). Bei richtiger Einstellung wird die Koppel beim Öffnen der Tür maximal ausgelenkt (Positionierung des rückwärtigen Teils der Kulisse) und die als Kulissenstein fungierenden Stange leichtgängig, aber spielfrei durch den parallel zum Portal verlaufenden Anschlag sowie die um 30° angewinkelte Nut geführt.

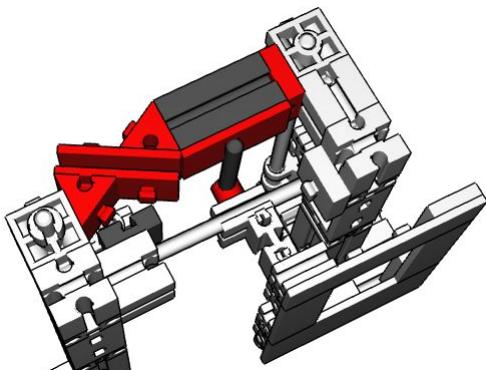


Abb. 19: Kulissenführung; der Querträger des Portals ist nur teilweise abgebildet; sichtbar ist der als Anschlag der Schwinge dienende Baustein 5 (dunkelgrau)

Vorder- und rückwärtiger Teil bilden eine Bahn mit einer um 30° angewinkelten Nut. Eine am Gleitlager des Türblatts befestigte Achse 40 bildet den Kulissenstein.

Die Bewegung des Türblatts wird durch einen umlaufenden Seilzug bewirkt, der beidseitig auf das Gleitlager des Türblatts wirkt. Der Antrieb erfolgt über eine Seiltrommel. Mehrere Wicklungen des Seils stellen eine kraftschlüssige Verbindung sicher, die Seilspannung lässt sich durch vertikales Verschieben der Seilwindengests justieren. Das Seil wird über Rollen am Portal entlang geführt, sein der Nebenschließkante zugeordnetes Ende läuft durch das entsprechende Schwenklager der Tragschiene, also im freien Raum zwischen Baustein 7,5 und Strebenadapter (Abb. 20).

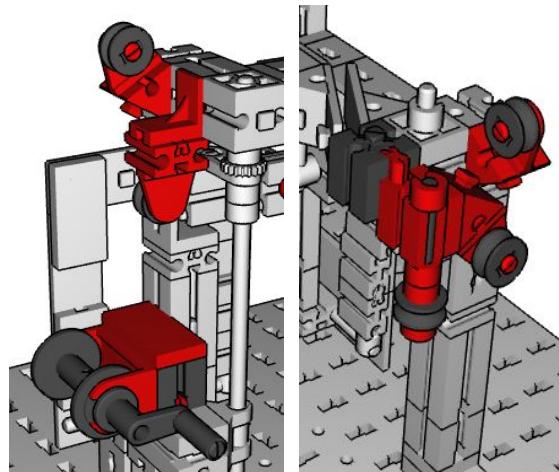


Abb. 20: Anordnung der Seilrollen zur Seilführung auf der Seite der Haupt- (rechts) und der Nebenschließkante (links)

Bei optimaler Einstellung aller Komponenten (Länge der Koppel, Winkel der Schwingen, Position der Kulisse und Spannung des Seilzugs) lässt sich die Tür durch Drehen der Kurbel vorgbildgerecht öffnen und schließen, ohne gegenüber der Türöffnung zu verkanten (Abb. 21 und 22).

Referenzen

- [1] Stefan Busch: [Schwenktüren](#). ft:pedia 3/2017, S. 23-33.
- [2] Wikipedia: [Schiebetür](#).
- [3] Wikipedia: [Kulissensteuerung](#).
- [4] Dietmar Dilcher: [Schwenkschiebetür für Fahrzeuge](#). EP0920591A3.

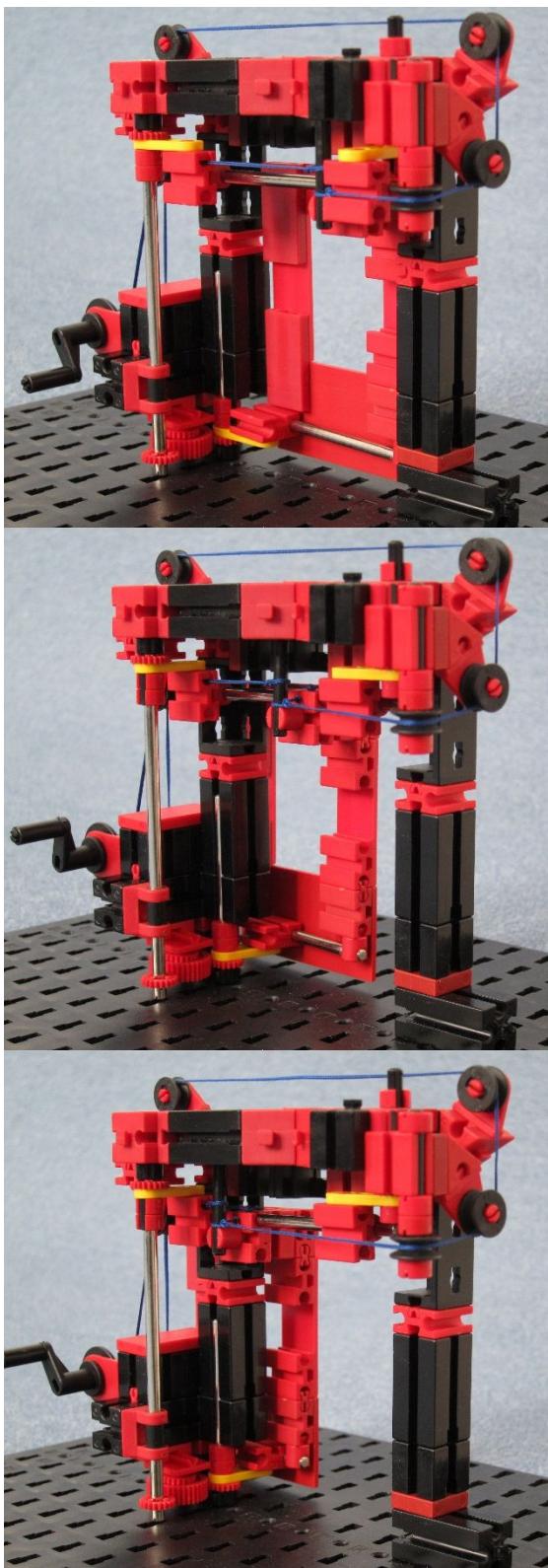


Abb. 21: Vollständiges Modell; besonders beachtenswert ist das durch das Schwenkgelenk geführte Seil

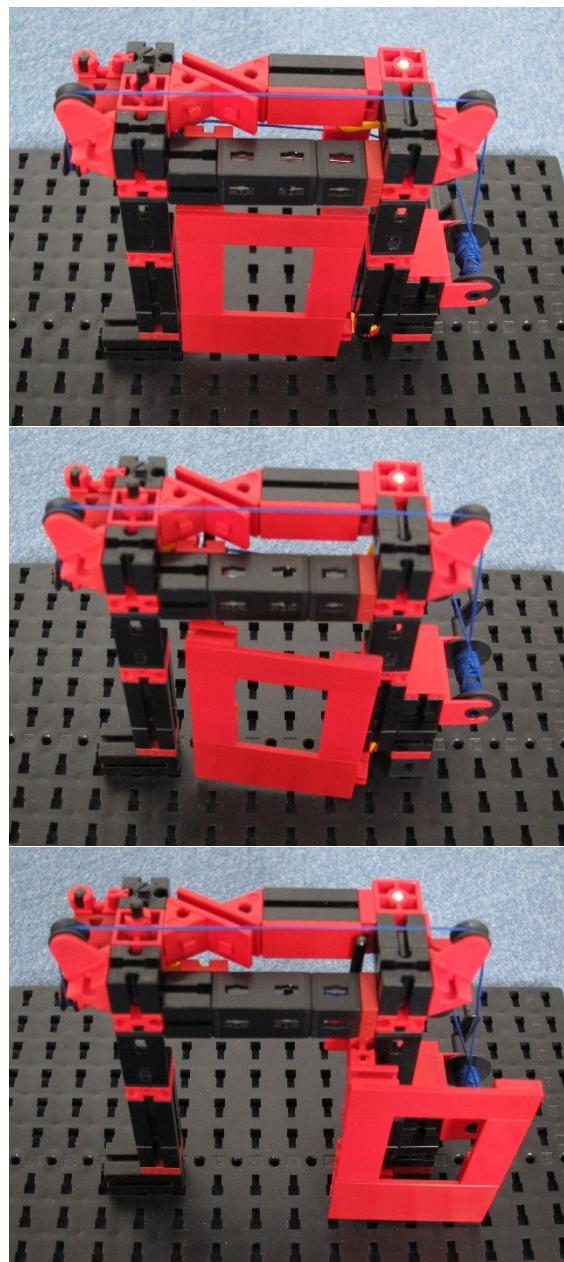


Abb. 22: Vollständiges Modell; deutlich sichtbar ist die Kulissenführung

Modell

Vom elektromechanischen Betätiger zur elektrischen „Dampfmaschine“

Rüdiger Riedel

Ein Jahr habe ich gebraucht, um aus einem einfachen Funktionsmodell „Dampfmaschine“ [1] eine überzeugende elektromechanische Konstruktion zu entwickeln. Entscheidend waren die verbesserte Anordnung der Dauermagnete und der Einsatz des BT Smart Controllers für die doppeltwirkende Funktion. Die „Kolben“ arbeiten jetzt sowohl ziehend als auch drückend.

Entwicklungsgang

In Abb. 1 ist mein erster Versuch zu sehen, die Funktion einer Dampfmaschine mit den fischertechnik-Elektromagneten nachzubilden. Die Maschine lief sauber mit etwa 12 V Spannung nach dem Prinzip der Gleichstrommaschine in [3].

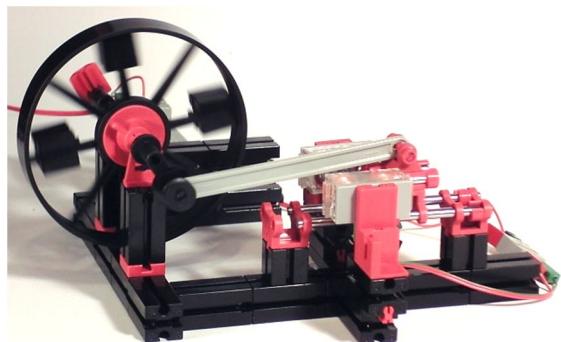


Abb. 1: Meine erste elektrische Dampfmaschine

Sie war einfach wirkend, also nur ziehend an der Kurbel oder nur drückend, je nach Polung. Zwei Stabmagnete 4 x 25 mm wurden für den „Kolben“ verwendet. Die Steuerung ist, kaum zu erkennen, auf der Rückseite: Eine Gelenkwürfel-Zunge trägt einen kleinen Magneten (4 x 10 mm) und der regt einen Reedkontakt an, einmal je Umdrehung, der wiederum die beiden Elektromagnete ein- und ausschaltet.

Nicht vergessen darf man die Schutzbeschaltung! Sonst ist die Freude am Reedkontakt schnell dahin. Ich habe dazu auf der Rückseite der Maschine die parallel geschalteten E-Magnete mit einem 220-Ohm-Widerstand überbrückt [3].

Eine erste Verbesserung der Maschine bringt der Tipp von geometter, statt der Rollenböcke BS7,5 zu verwenden, die wesentlich leichter gleiten.

Die Neodym-Magnete mit 4 mm Durchmesser und 25 mm Länge passen nicht recht zur Geometrie der Elektromagnete. Der Polabstand beträgt etwa 16 mm, 15 mm lange Magnete sollten besser funktionieren. Mein Ziel war aber, die schon eingeführten Neodym-Magnete mit 4 mm Durchmesser und 10 mm Länge auch hier zu verwenden.

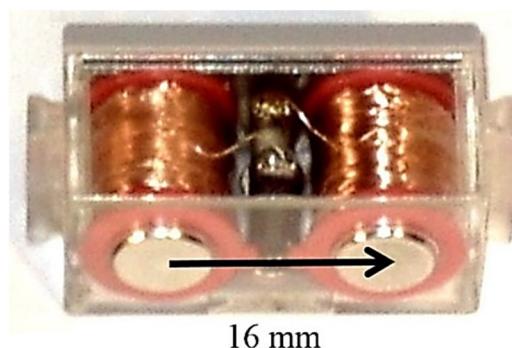
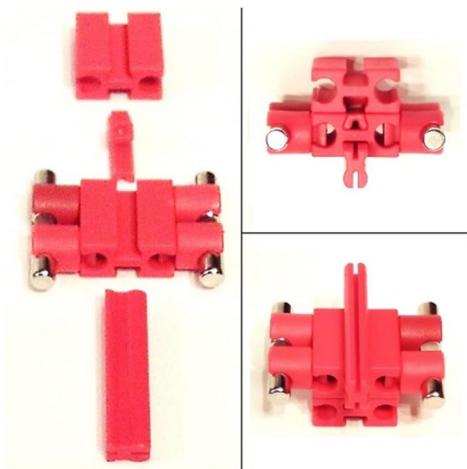


Abb. 2: fischertechnik-Elektromagnet 32363, gleiche Abmessungen wie der ältere Typ 31324

Die Lösung zeigt Abb. 3: Die beiden Stabmagnete auf jeder Seite sind mit gleichnamigen Polen gegeneinander gerichtet. Den Abstand stellt man mit der dünnen Seite eines V-Bausteins 15 Eck (38240) oder Winkelsteins 10 x 15 x 15 (38423) ein.



*Abb. 3: Der Läufer oder Kolben.
Links der Aufbau, rechts oben von vorne,
rechts unten die Unterseite*

Damit lässt sich schon etwas anfangen:

Der elektromagnetische Betätiger

Der Aufbau ergibt sich aus Abb. 4 bis 9. Für einen optimalen Abstand zwischen Läufermagneten und den Elektromagneten sorgen die beiden Baustein-Kombinationen

WS7,5 – WS15 – WS7,5 – BS5 – BS5 2Z

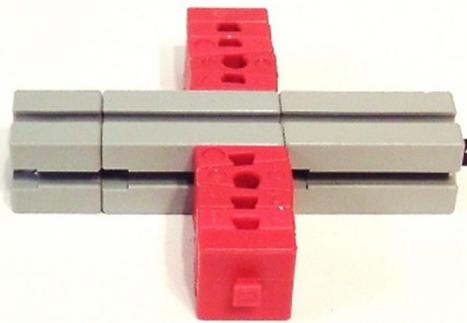


Abb. 4: Der Betätiger, Basis

Im Betrieb stößt der Verbinder 30 des Läufers rechts und links an die Minitaster, damit der Läufer nicht über die beiden Ruhepositionen hinausschießt. Eine elektrische Funktion haben die Minitaster nicht.

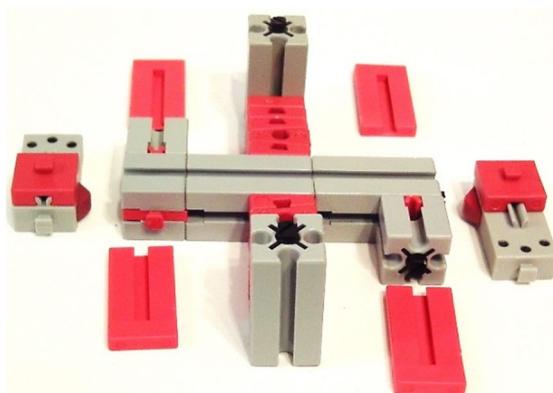


Abb. 5: Zusammenbau

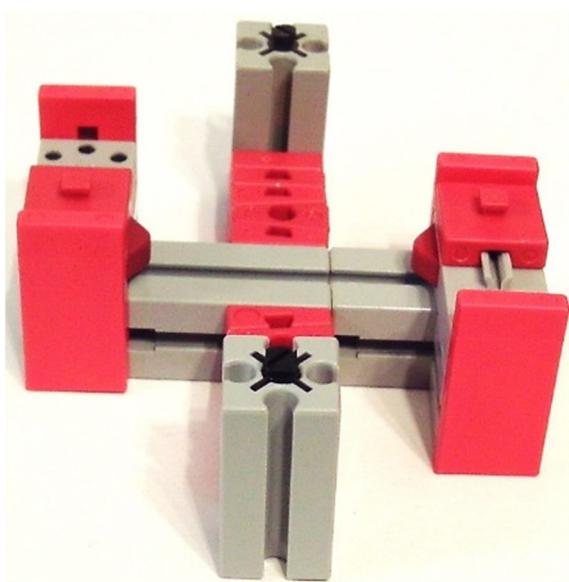


Abb. 6: Das Gerüst steht

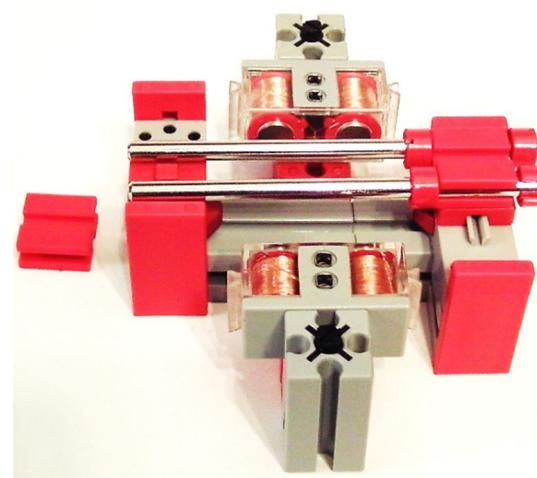


Abb. 7: Die Achsen im rechten BS7,5 sollten etwas verschiebbar bleiben

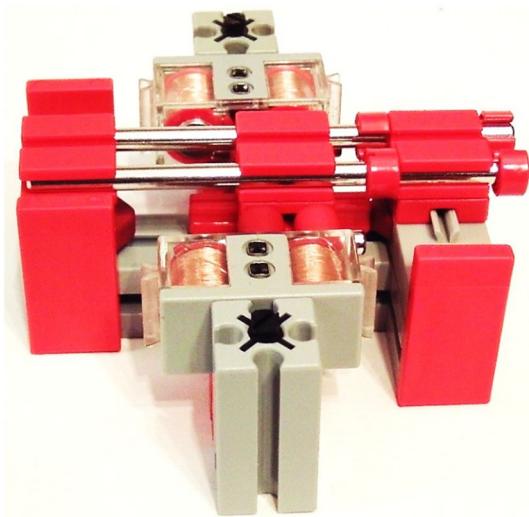


Abb. 8: Der Läufer ist eingefügt

Dieser Läufer hat zwei stabile Positionen bei einem Hub von etwa 13 mm.

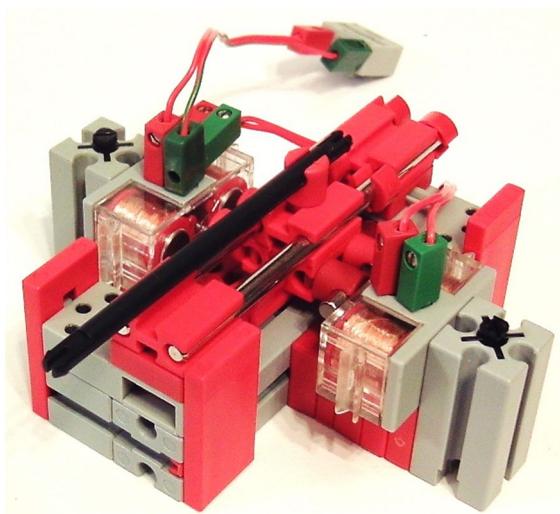


Abb. 9: Der anschlussbereite Betätiger

Angesteuert wird der Betätiger über einen Umschalter, also von Hand, oder über einen Controller, der die wechselnde Polarität der Elektromagnete erzeugt. Die Kraftentwicklung folgt einem Bogen: Am Anfang und Ende klein, in der Mitte der Bewegung am größten.

Von Vorteil ist die Schnelligkeit: 10 Hz (d. h. zehn volle Bewegungszyklen pro Sekunde vor und zurück) sind problemlos möglich. Man kann auf die Begrenzungen verzichten und so einen größeren Weg überstreichen, wenn das z. B. von Kurbeln und einem Schwungrad unterstützt wird.

Die doppeltwirkende Zweizylinder-Dampfmaschine

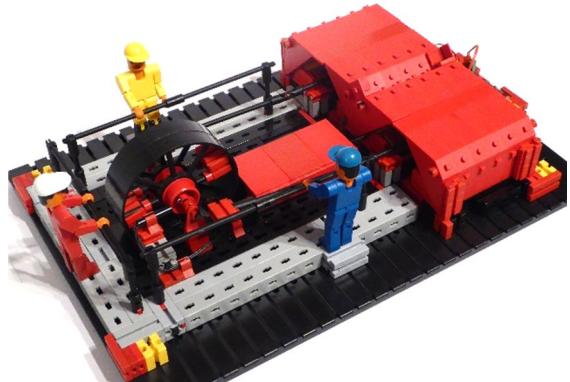


Abb. 10: Zweizylinder-Dampfmaschine



Abb. 11: Die Zylindergehäuse sind teilweise entfernt

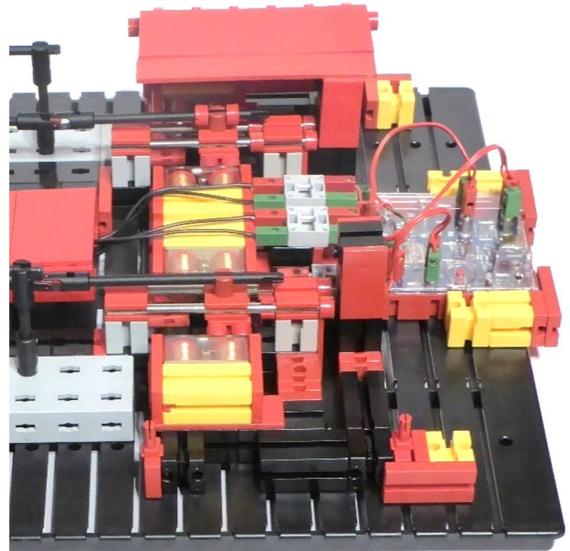


Abb. 12: Einblick in die Zylinder, rechts der BT Smart Controller

Für mein Modell einer Dampfmaschine setze ich zwei etwas modifizierte Betätigter ein. Der Abstand der Elektromagnete wird auf jeder Seite von zwei Baustein-Kombinationen eingestellt, s. o. Dadurch können die Anschlüsse und die Verbindungsleitung

der beiden E-Magnete verdeckt geführt werden (Abb. 14).



Abb. 13: Der modifizierte Betätiger

Bei diesem Betätiger werden die älteren Elektromagnete 31324 verwendet. Die sind zwar schwächer als der neuere Typ 32363, aber gebraucht noch leichter erhältlich.

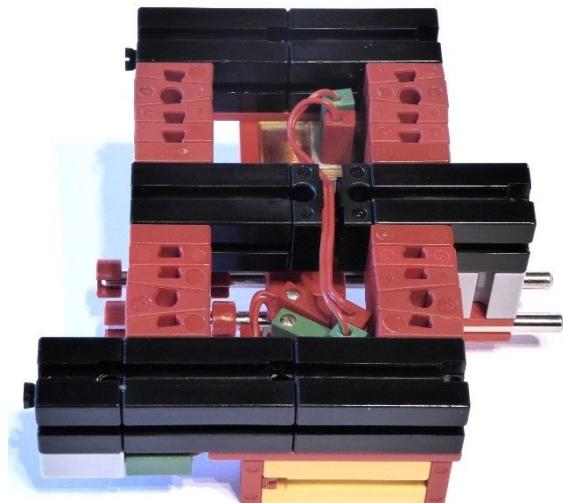


Abb. 14: Modifizierter Betätiger von unten

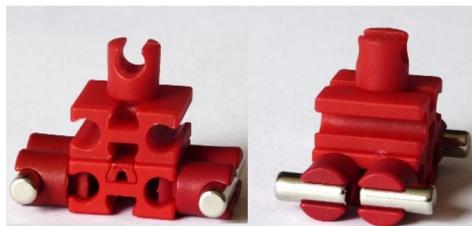


Abb. 15: Der modifizierte Läufer von vorn und von der Seite

Die Höhe der E-Magnete zu den Magneten des Läufers wird durch Bausteine 2,5 15 x 45 2+2Z (38277) eingestellt.

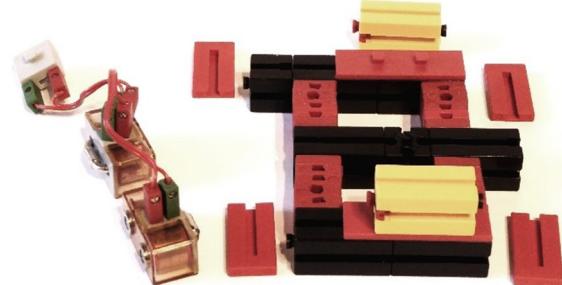


Abb. 16: Der modifizierte Betätiger vor dem Einbau der E-Magnete

Jeweils zwei Rastachsen mit Kardangelenk simulieren das Schubkurbelgetriebe.

Für das Schwungrad habe ich wegen des besseren Aussehens zwei Speichenräder (36916) genommen, die mit zweimal drei BS7,5 zusammengehalten werden.

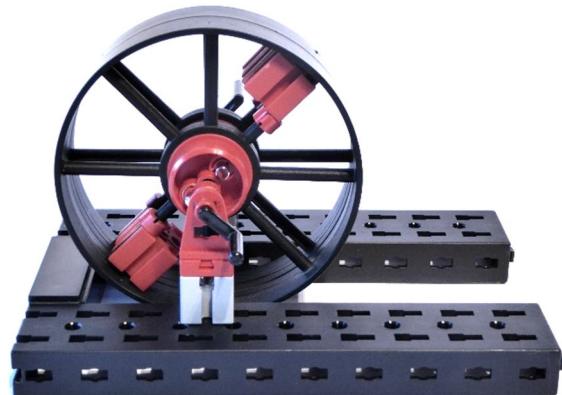


Abb. 17: Schwungrad

Nun hat der Betätiger einen kleinen aktiven Hub von ungefähr 13 mm. Der passt nicht gut zur Rastkurbel (35088), deren Hub bei 23 mm liegt. Deshalb setze ich zwei Kurbeln 40 KR11 (38447) ein, um 90° gegeneinander versetzt. Die haben einen Hub von annähernd 18 mm. Die Kurbeln erhalten eine Scheibe (105195), das Kurbelgestänge wird mit je einem BS7,5 ohne weitere Befestigung angelenkt.

Die Positionserkennung übernehmen zwei um 90° versetzt angeordnete Reedkontakte (36120), siehe Abb. 18.



Abb. 18: Die Reedkontakte

Angeregt werden sie von zwei Stabmagneten 4 x 10 mm, die in zwei Klemmbuchsen 5 stecken. Zusammen passen sie genau in den Zwischenraum des V-Rades 23 x 10 (das habe ich mir abgeschaut von der „fischertechnik Classic Linie Mobile Dampfmaschine“).

Die Zylindergehäuse bestehen aus den Seitenteilpaaren der Baggerschaufel (37353) und je zweimal vier Baggerschaufel-Mittelteilen (37355). Außerdem werden noch zwei Bauplatten 15 x 90 6Z (38245) benötigt, um die Mitte abzudecken.

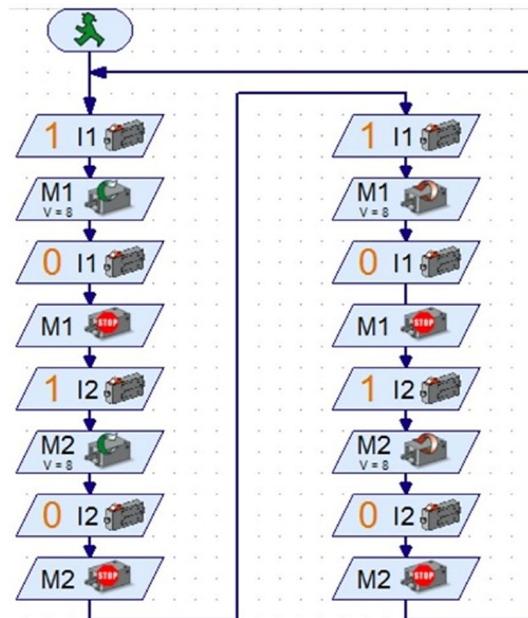


Abb. 19: Ablaufsteuerung

Aufgebaut habe ich das Ganze auf Basis des Schwungradgerüstes in Abb. 17. Über ein

paar Zwischenstücke werden die Betätiger an den U-Trägern befestigt. Zum besseren Transport benutze ich eine Bauplatte 1000 und sichere die Maschine mit den gelben und roten Bausteinen an den Ecken.

Das Steuerprogramm des BT Smart Controllers zeigt Abb. 19.

Anmerkung

Wenn alles sauber aufgebaut ist, fest sitzt und die Steuermagnete und Reedkontakte ausgerichtet sind, läuft die Maschine ruhig und elegant. Um etwas anzutreiben fehlt ihr allerdings die Leistung. Das liegt an der zu geringen Spannung, die der BT Smart Controller abgibt: Es sind 7,8 V in beiden Richtungen bei 9 V Versorgung durch ein stabilisiertes Netzgerät. Auch sind die älteren E-Magnete (32363) schwächer als die neueren (31324), dafür aber immer noch gebraucht erhältlich.

Die E-Magnete sind im Dauerbetrieb mit 12 V belastbar [2]. Bei dieser Maschine beträgt die Einschaltzeit etwa 70 %, somit könnte die Versorgungsspannung bis auf 17 V angehoben werden.

Ein [Video](#) der elektrischen Dampfmaschine habe ich in Youtube hochgeladen.

Referenzen

- [1] Rüdiger Riedel: [Elektrische Dampfmaschine](#). Bilderpool der ftcommunity.
- [2] Rüdiger Riedel: [Der Elektromagnet: Was kann er \(vertragen\)?](#) ft:pedia 4/2016, S. 46-51.
- [3] Rüdiger Riedel: [Funktionsmodelle von Gleich- und Wechselstrommotoren](#). ft:pedia 4/2016, S. 52-58.
- [4] Dirk Fox, Thomas Püttmann: [Technikgeschichte mit fischertechnik](#). dpunkt-Verlag, 2015, S. 159-182.
- [5] Dirk Fox: [Die Dampfmaschine](#). ft:pedia 4/2012, S. 38-45.



Abb. 20: Die elektromechanische Dampfmaschine mit Bedienungspersonal

Modell

fischertechnik-3D-Drucker 2.0

Dirk Wölffel

Im Frühjahr 2016 erschien der fischertechnik-Bausatz eines 3D-Druckers (536624). Mit diesem Selbstbausatz erhält der Käufer einen einfachen Zugang zu der zukunftsweisenden und faszinierenden Technologie des 3D-Drucks. Nachdem der Preis sich etwas gesenkt hatte, habe ich mir auch einen fischertechnik-3D-Drucker zugelegt und damit meine ersten Erfahrungen mit dem Aufbau des Druckers und dem 3D-Druck gesammelt. Nach zwei Jahren intensiven Einsatzes wurden verschiedene Schwächen im Aufbau und der Druckqualität sichtbar. Deshalb beschloss ich, den 3D-Drucker mit verschiedenen Modifikationen neu zu konstruieren.

Das Original

Der fischertechnik 3D Printer (Abb. 1) hat einen einfachen Aufbau aus Alu-Profilen als Grundgestell. Die Schrittmotoren sind ausreichend dimensioniert.

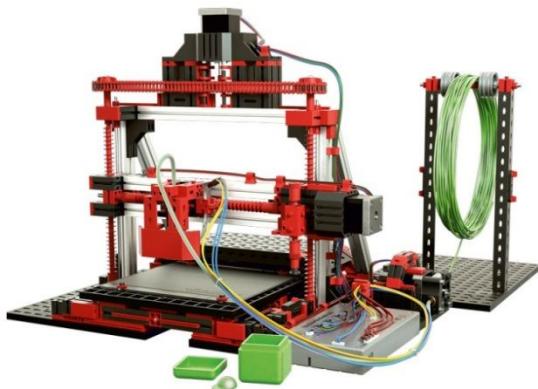


Abb. 1: 3D Printer 536624

Die Neukonstruktion

Grundlage für die Neukonstruktion (Abb. 2) waren, die von mir festgestellten baulichen Schwächen des 3D Printers [1]:

- keine kompakte Bauform (mehrteiliger Aufbau)
- die Höhe des Druckbetts lässt sich nicht einstellen
- die Halterung für die Filamentrolle ist nicht geeignet

- der 3D Printer lässt sich schwer reinigen
- das Einstellen der Z-Achse über die Schnecke ist problematisch
- die [Abtriebshülse für den Schrittmotor](#) (160549) der X-Achse ist nicht langlebig
- Die Achsen haben keine Kugellager
- zum Kühlen beim 3D-Druck fehlt ein Lüfter
- das Druckbett ist nicht ausreichend eben
- eine Beleuchtung fehlt



Abb. 2: 3D-Drucker 2.0

Die Abmaße des neuen 3D-Druckers 2.0 (Abb. 2) umfassen eine Länge von 30 cm und 32 cm Breite bei einer Höhe von 34 cm. Er wiegt stattliche 8,2 kg.

Als Grundlage für den Neubau des 3D Printers habe ich die Grundform des [Ultimaker²Go](#) (Abb. 3) gewählt. Der Aufbau schien mir kompakt genug, um meine Pläne umzusetzen.



Abb. 3: Ultimaker²Go

Die Bauform sollte nicht geschlossen sein, damit man die Funktionsweise auch noch erkennt. Hinzu kam der Wunsch, ein beheiztes Druckbett zu verbauen, damit sich das Warping (verziehen der Bauteile) beim Druck minimiert.

Die Elektronik

Um die Abmaße des Modelles zu finden habe ich nach einem beheizten Druckbett gesucht. Fündig geworden bin ich beim [Anycubic-Ultrabase-220x220mm](#) (Abb. 4) auf Ebay. Das Druckbett ist beschichtet und zeigt eine starke Haftung beim Erwärmen. Es lässt sich mit 12 Volt oder 24 Volt betreiben.



Abb. 4: Anycubic-Ultrabase-220x220mm

Als nächstes brauchte ich eine passende Stromversorgung und eine Temperaturregelung für das beheizte Druckbett. Der fischertechnik-3D-Controller besitzt leider keine Möglichkeit, ein beheiztes Druckbett anzuschließen. Bei der Wahl des Netzteils habe ich mich für das geregelte Schaltnetzteil [Surom 12V 30A DC 360W](#) (Abb. 5) entschieden.



Abb. 5: geregeltes Schaltnetzteil

Für die Temperaturregelung fiel die Wahl auf den [Inkbird-Temperaturregler mit 40 A SSR Relais und Temperatursensor](#) (Abb. 6). Der obere Wert (rot) zeigt die Ist-Temperatur an, der untere Wert (grün) die Soll-Temperatur. Somit hatte ich alle relevanten Bauteile zusammen, um mit dem Aufbau zu beginnen. Ich nahm das Druckbett als Vorlage und baute mit Alu-Profilen den Grundaufbau darum herum. Damit standen die äußeren Abmaße des 3D-Druckers fest.



Abb. 6: Inkbird-Temperaturregler mit SSR Relais und Temperatursensor

Den unteren Bereich des Grundkörpers habe ich dann so konstruiert, dass die elektronischen Komponenten dort untergebracht werden konnten (Abb. 7). Hier sieht man die Anordnung der verschiedenen elektronischen Komponenten. Das SSR (*Solid State Relais*) vorne rechts steuert das Netzteil über den Inkbird-Temperaturregler.

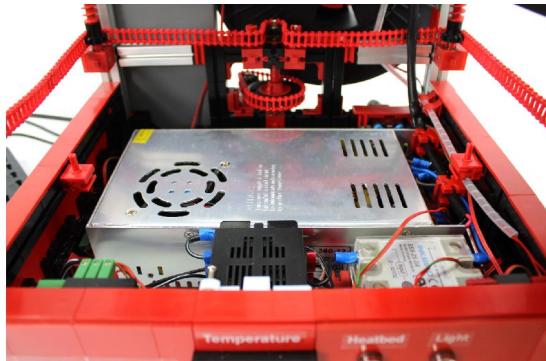


Abb. 7: Anordnung der Elektronik

In der Front des 3D-Druckers habe ich alle relevanten Bauteile übersichtlich verbaut. Zu sehen ist die PID-Temperaturregung (mittig), der Ein-/Aus-Schalter (rechts) für das Heizbett und das Licht (Abb. 8).



Abb. 8: Frontseite des 3D-Druckers

Für das Einschalten des Netzteils habe ich seitlich einen 220 Volt Wippschalter und eine Kaltgerätebuchse für das Netzkabel verbaut (Abb. 9).



Abb. 9: Wippschalter und Kaltgerätebuchse

Als Beleuchtung sind zwei 12 Volt LED-Streifen links und rechts verbaut (Abb. 10).

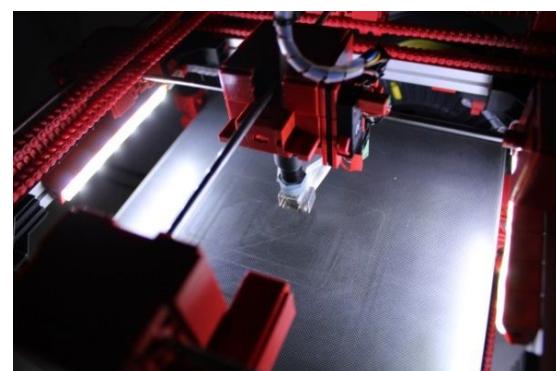


Abb. 10: Die Beleuchtung

Das Druckbett

Der Vorteil des beheizten Druckbetrts ist, dass das Verziehen der 3D-Druck-Bauteile minimiert werden kann. Außerdem lässt sich die Höhe des Druckbetts besser verstehen (Abb. 11).

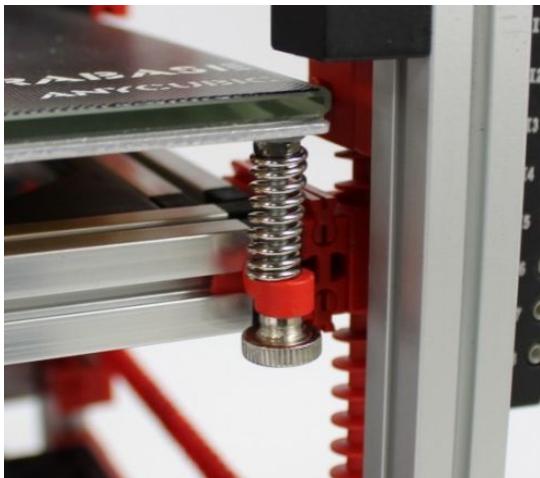


Abb. 11: Stellschraube Druckbett

Die Höhenverstellung

Auf der nächsten Abbildung ist die Höhenverstellung der Z-Achse zu erkennen (Abb. 12). Diese lässt sich sehr leicht und genau über ein Zahnrad Z15 verstellen.

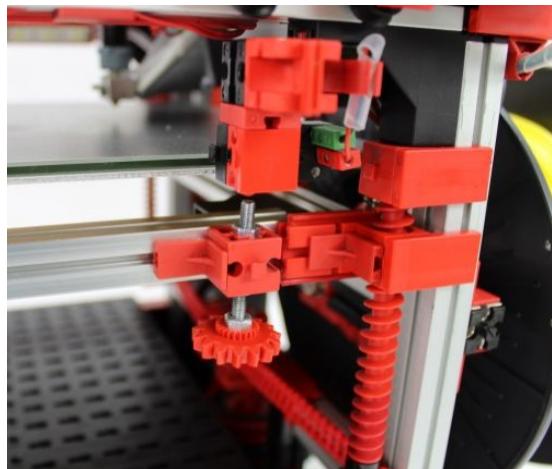


Abb. 12: Höhenverstellung der Z-Achse

Die Achsen

Für die Ansteuerung der Z-Achse habe ich mich für eine Aufhängung über Schnecken an vier Seiten entschieden, weil es sehr stabil ist und die Höhe sich besser verstehen lässt (Abb. 13).

Die X-Achse und die Y-Achse werden, wie beim Original, über Schnecken angetrieben; das ist in der seitlichen Vorder- (Abb. 14) und Rückansicht sehr gut zu erkennen (Abb. 15).



Abb. 13: Kettenantrieb der Z-Achse

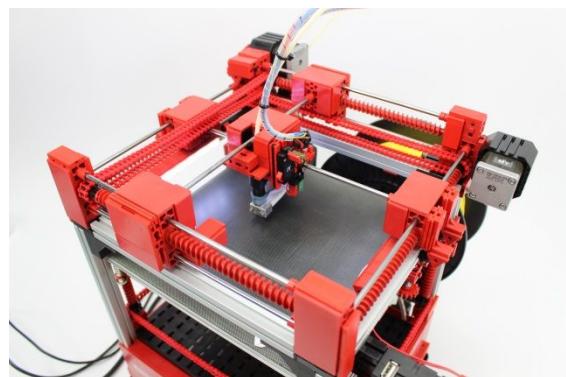


Abb. 14: seitliche Vorderansicht

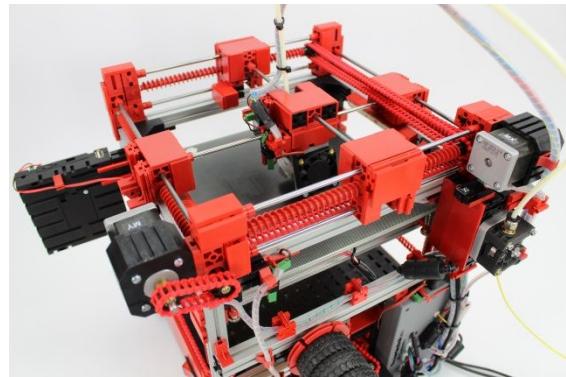


Abb. 15: seitliche Rückansicht

Die Filament Zuführung

Der Extruder ist auf der Rückseite des 3D - Druckers hochkant verbaut (Abb. 16). Der Druckkopf wird über die beiden X- und Y- Achsen geführt. Vorne am Druckkopf ist der Endschalter der Y-Achse zu erkennen (Abb. 17).

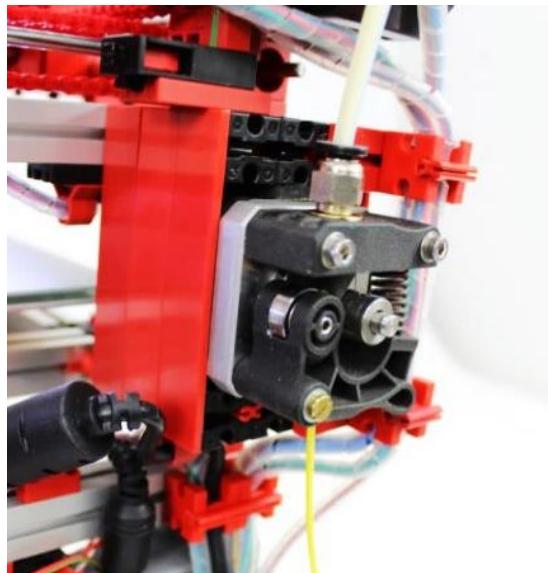


Abb. 16: Der Extruder

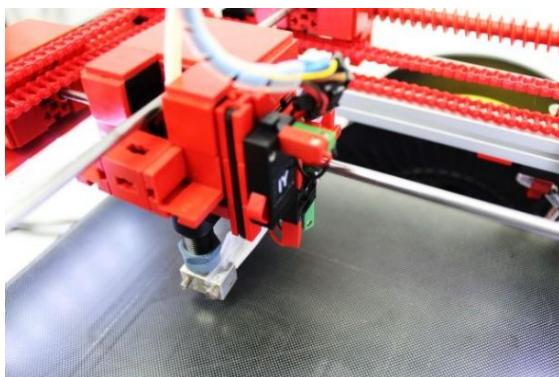


Abb. 17: Der Druckkopf

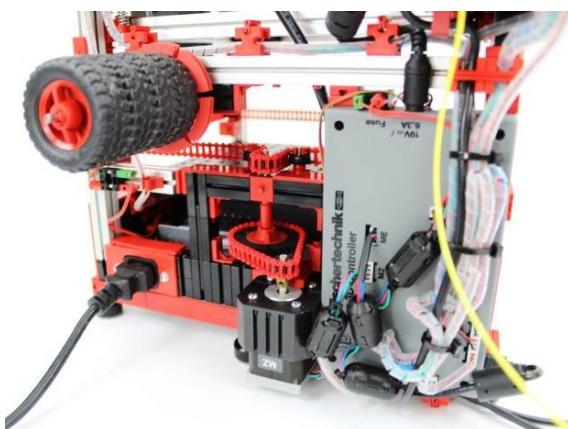


Abb. 18: fischertechnik-3D-Controller

Der Controller

Die Hardwareansteuerung erfolgt über den fischertechnik-3D-Controller. Der 3D-Controller wurde beim 3D-Printer auf der Rückseite verbaut. Gut zu erkennen ist auch die

Aufnahme für die Filamentrolle und der Antrieb der Z-Achse (Abb. 18).

Die Software

Der 3D-Drucker kann über das Programm 3D Print Control von fischertechnik (Abb. 19) angesteuert werden. Alternativ gelingt das auch über die Community Firmware mit der 3D Print App von Till Harbaum (Abb. 20) [2].

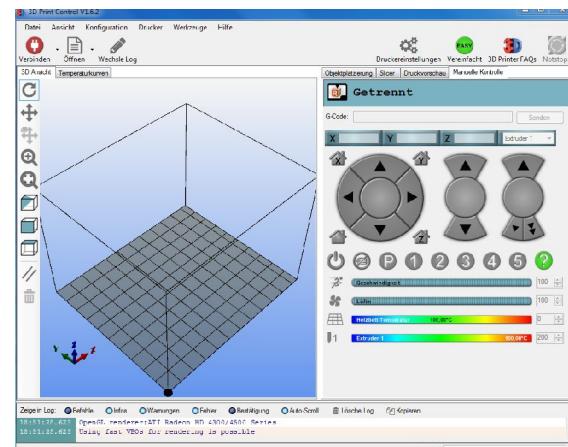


Abb. 19: fischertechnik 3D Print Control

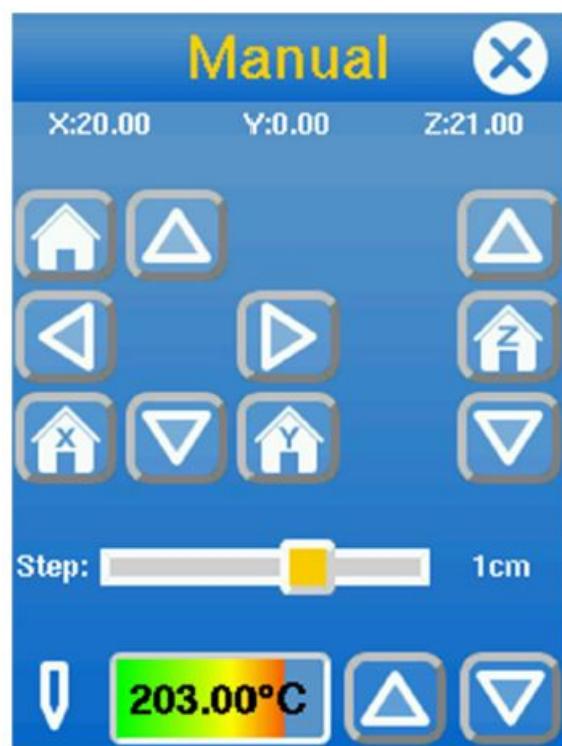


Abb. 20: ftcommunity 3D Print App

Fazit

Durch die Summe der verschiedenen baulichen Veränderungen konnte der 3D-Druck erheblich verbessert werden (Abb. 21, siehe auch das [Video](#)).

Dazu gehört auch der Komfort in der Bedienbarkeit des 3D-Druckers. Es zeigt sich, dass man mehr aus den fischertechnik Originalmodellen herausholen kann, wenn man Schwachstellen analysiert und nach alternativen Lösungen und Ansätzen sucht.

Referenzen

- [1] Dirk Wölffel: [*Neue ft-Teile selbst gemacht – 3D-Druck \(5\): Qualitätsverbesserung des ft-Druckers.*](#)
ft:pedia 1/2017, S. 72-76.

- [2] Till Harbaum: [*Frische Apps für den TXT Controller.*](#) ft:pedia 4/2016, S. 68-71.



Abb. 21: 3D-Druck-Teile

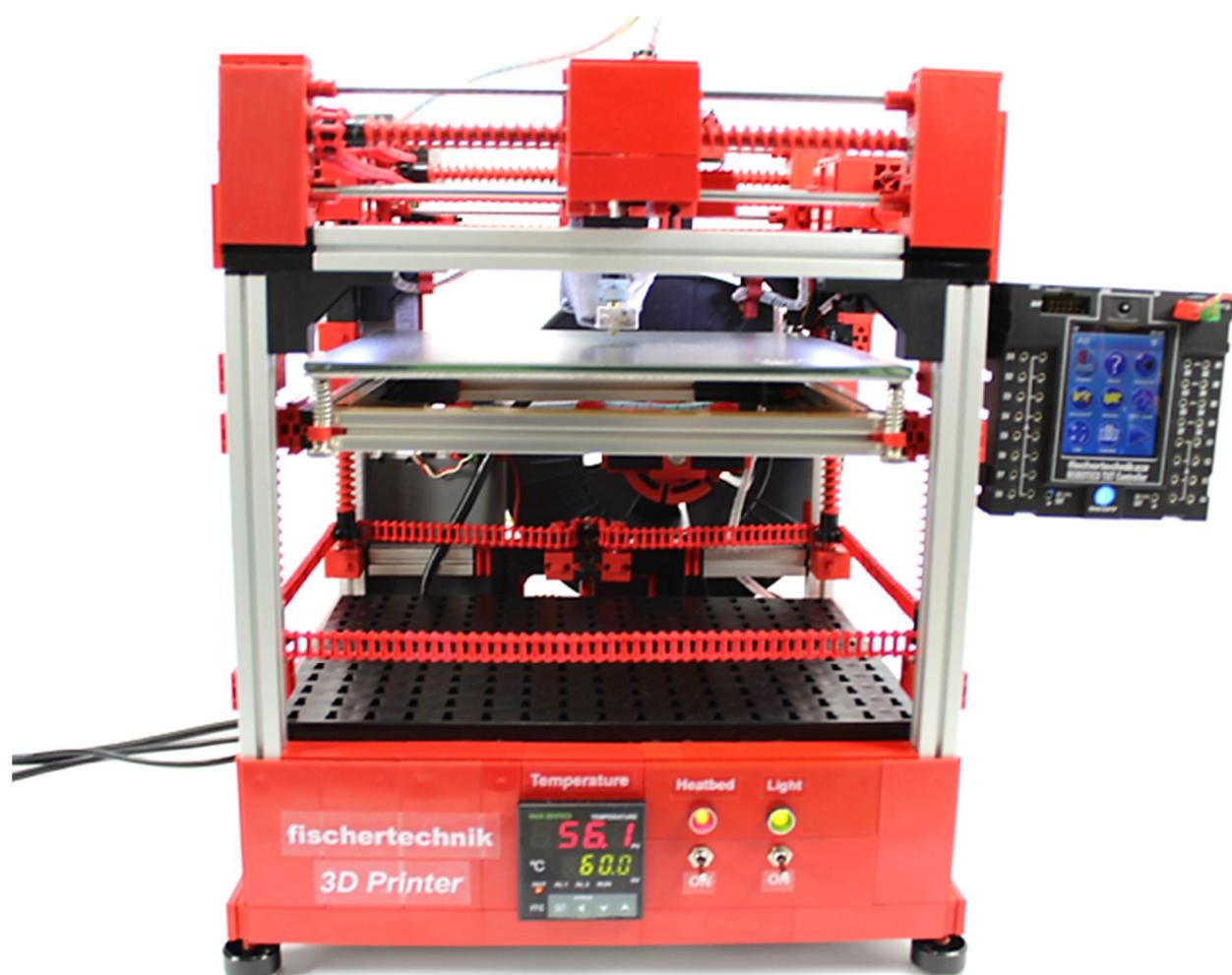


Abb. 22: fischertechnik-3D-Drucker 2.0

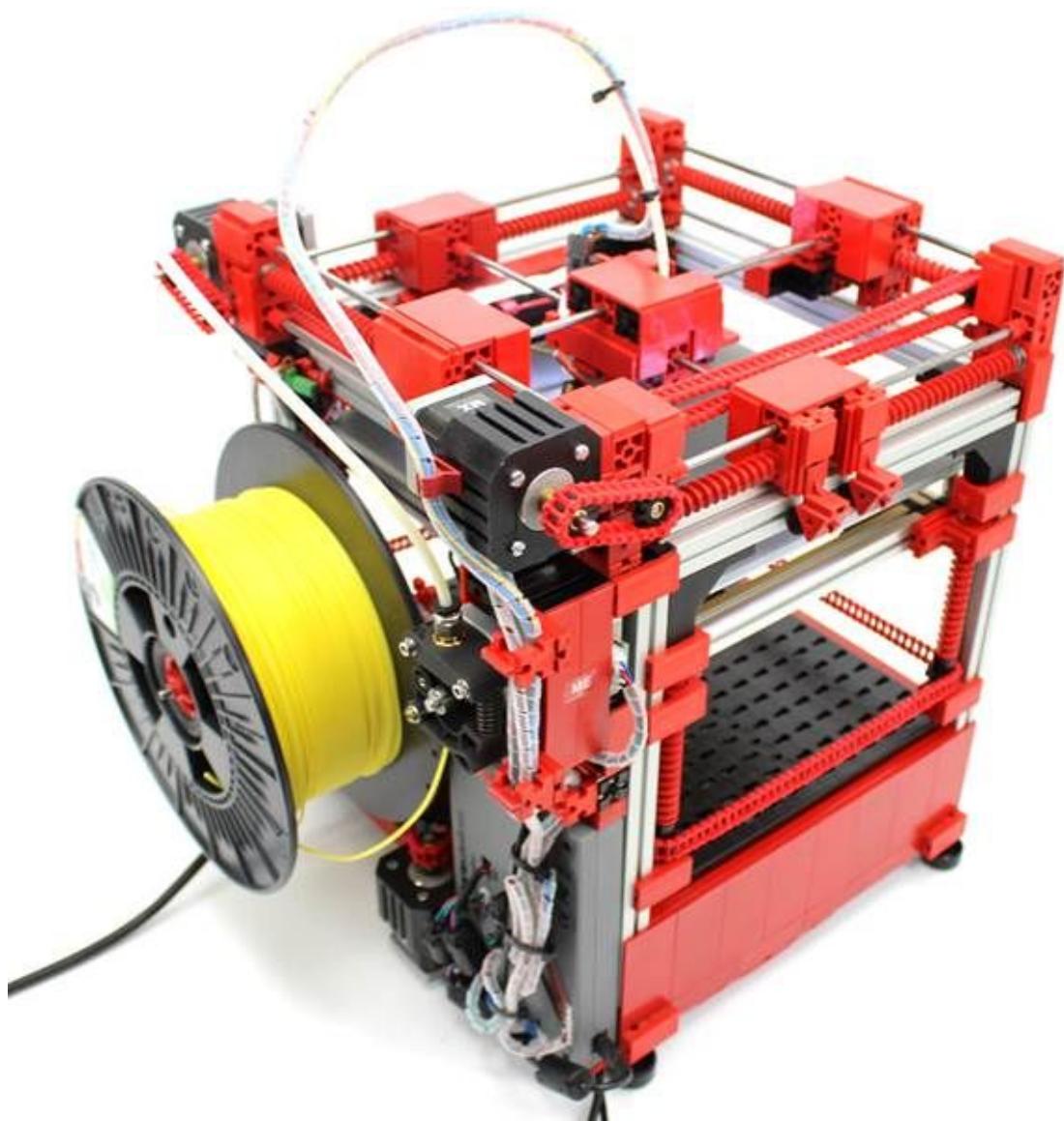


Abb. 23: Rückansicht des 3D-Druckers, mit Filamentrolle

Getriebe

Automatische Differentialsperre

Martin Wanke

Eine ft:pedia ohne Differential? – Nein. Es war bisher in zwei Dritteln aller Ausgaben enthalten, und diese hier soll sich einreihen. fischertechnik-Differentiale wurden bereits in mechanischen Wunderwerken wie Planetarien, Uhren oder Rechenmaschinen verbaut, in anderen Beiträgen wurde ihre Funktion anschaulich erklärt, und es kam – natürlich – in verschiedenen Fahrzeugantrieben zum Einsatz. Hier kommt nun ein weiteres Kapitel hinzu, in welchem dieses vielseitige Bauteil zur automatischen Sperre eines Antriebsdifferentials verwendet werden soll.

Eine Differentialsperre findet man typischerweise in Geländewagen. Sie wird – manuell oder automatisch – aktiviert, um die Wirkung eines Antriebsdifferentials zu unterdrücken, wenn sich auf rutschigem Boden alle Räder zwangsweise gleich schnell drehen sollen.

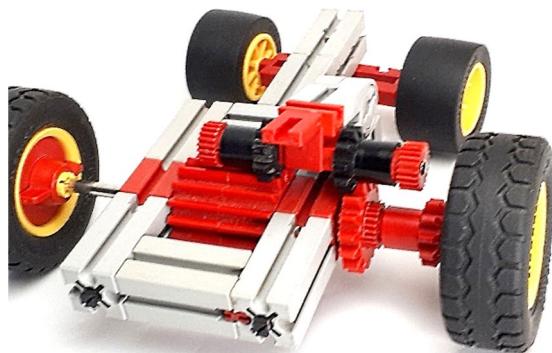


Abb. 1: Manuelle Differentialsperre

In Abb. 1 ist eine nicht-automatische Differentialsperre zu sehen. Durch das manuelle Absenken der schwarzen Zahnräder drehen sich das rechte Rad und das Außenzahnrad des Differentials und folglich auch das linke Rad zwangsweise gleich schnell.

Ohne die Sperre würde sonst die gesamte Leistung an einem durchdrehenden Rad sinnlos verpulvert werden. Im Automobilbau gibt es dafür u. a. automatische hydraulische und elektromechanische Lösungen,

hier soll aber eine rein mechanische gezeigt werden, die automatisch aktiviert wird.

Idee: Drehzahlen differenzieren

Mit einem Differential kann man ja auch rechnen, und zwar, wie der Name schon sagt, beispielsweise Drehzahlen differenzieren. Hier wird ein „Rechendifferential“ verwendet, um den Drehzahlunterschied der zwei Antriebsräder zu ermitteln und um das Antriebsdifferential zu sperren, wenn beim Durchdrehen eines Rades der Unterschied zu groß wird.

Fachleute merken beim Betrachten der Bilder bestimmt gleich, dass es sich im Prinzip um ein Gleichlaufgetriebe handelt, nur dass hier nicht aktiv eine Drehzahl auf ein Rad „aufaddiert“ wird, sondern die Drehzahldifferenz passiv „gemessen“ wird.

Die Differenzdrehzahl wird dann auf eine Fliehkraftbremse geleitet, die ab einem bestimmten Wert blockiert und in diesem Moment zwangsweise die Differenz auf null setzt. Das wirkt wiederum rückwärts über das Rechendifferential auf das Antriebsdifferential, an welchem sich jetzt beide Räder gezwungenermaßen gleich schnell drehen müssen.

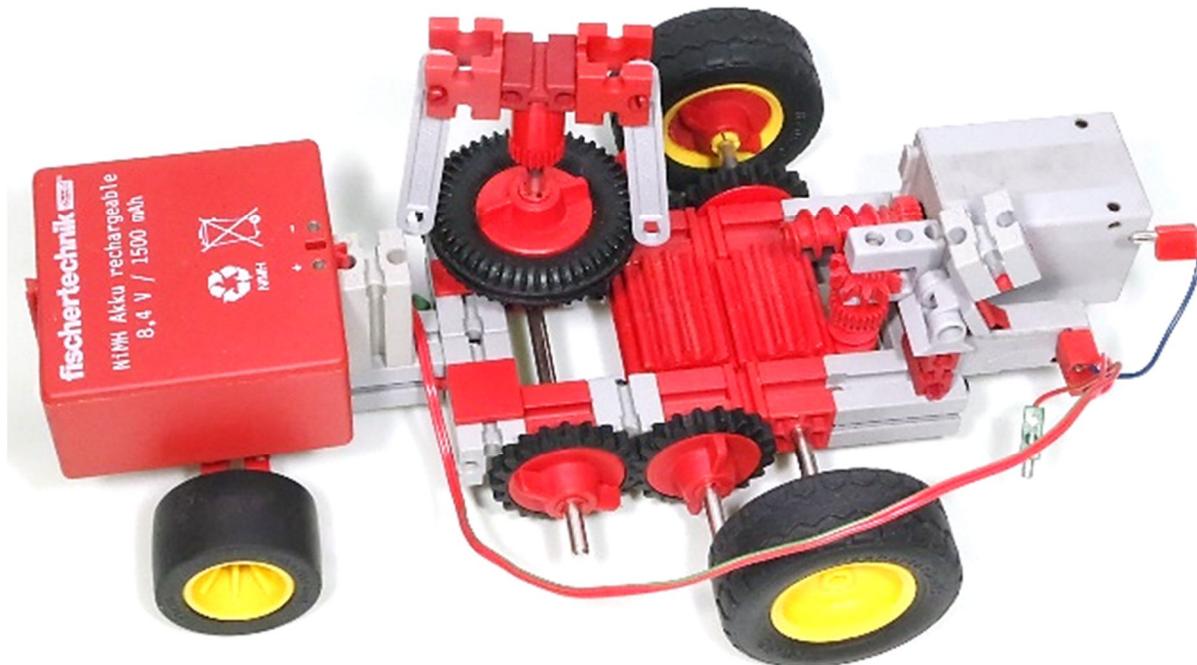


Abb. 2: Modellfahrzeug

Technik: Alte fischertechnik-Differentiale

Die alten fischertechnik-Differentiale eignen sich vorzüglich für diesen Zweck, weil sie mit ihrem Z15-Außenzahnrad in Kombination mit den Zahnrädern Z10 und Z20 passende und gleichzeitig platzsparende Übersetzungsmöglichkeiten bieten.

In Abb. 2 sieht man rechts den Motor im Heck des Modellfahrzeugs. Er wirkt auf das hintere Differential, das ist das Antriebsdifferential. Dessen Außenzahnrad wirkt auf das vordere, das Rechendifferential. Davor befindet sich senkrecht stehend mit den alten Reifen die Fliehkraftbremse. Vorne, unter dem Akku, befindet sich die Vorderachse mit einer simplen Lenkung.

In der Draufsicht in Abb. 3 ist die senkrecht stehende Fliehkraftbremse abgenommen, damit das Getriebe besser sichtbar wird. Die beiden Außenzahnräder der Differentiale drehen sich durch die 1:1-Übersetzung immer mit der gleichen Drehzahl.

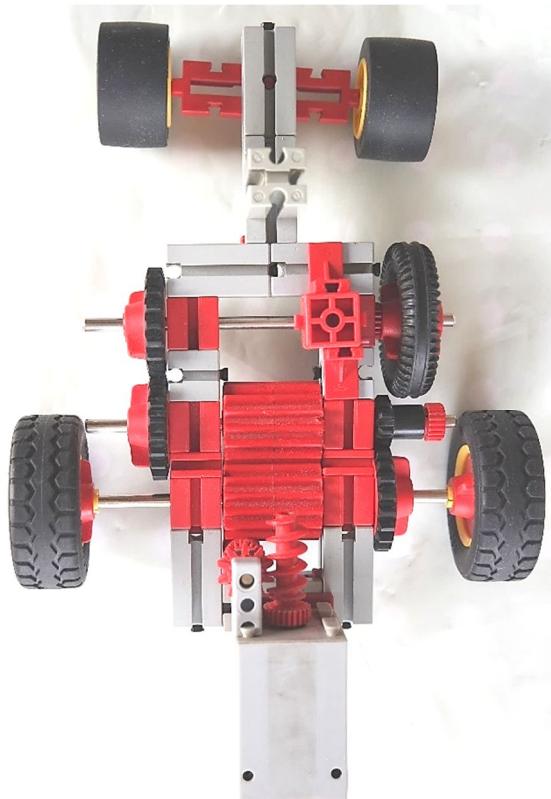


Abb. 3: Fliehkraftbremse abgenommen



Abb. 4: Akku abgenommen

Abb. 4 ist die Ansicht von rechts, der Akku ist abgenommen. Dabei ist vielleicht der Hinweis wichtig, dass das Fahrzeug nur auf den Rädern mit den gelben Felgen steht. Alle Räder mit roten Felgen sind Teil des Getriebes der Differentialsperre, wobei die beiden alten Reifen ein Winkelgetriebe bilden.

Bei normaler Geradeausfahrt dreht sich auch das rechte Z20 mit dieser Drehzahl, das von ihm angetriebene Z10 wegen der 1:2-Übersetzung aber doppelt so schnell. Diese doppelte Drehzahl auf der rechten Seite des Rechendifferentials ergibt zusammen mit der einfachen Drehzahl seines Außenzahnrad ein einen Stillstand auf der linken Seite des Rechendifferentials.

In einer Rechtskurve dreht sich das Z20 langsamer als das Außenzahnrad des Antriebsdifferentials, und folglich das Z10 nicht mehr doppelt so schnell wie das Außenzahnrad des Rechendifferentials.

Dadurch beginnt sich die linke Seite des Rechendifferentials im gleichen Sinn mitzudrehen – das ist dann die gesuchte Drehzahldifferenz der Antriebsräder. Diesen Zusammenhang hat Thomas Püttmann bereits in den Ausgaben 3/2011 und 4/2014 der ft:pedia sehr gut dargestellt [1, 2].

Nun wird diese Drehzahldifferenz nur noch mit Hilfe der beiden Z20 und dem Winkelgetriebe auf die Fliehkraftbremse übertragen.

In Abb. 5 sieht man diese Kurvenfahrt. Die Drehzahldifferenz ist dabei so klein, dass die locker hängenden Statikstreben noch nicht durch die Fliehkraft nach außen getragen werden und sich also noch nicht an dem Grundbaustein, an dem der Akku befestigt ist, verhaken. Die Fliehkraftbremse unterbindet also noch nicht die unterschiedlichen Drehzahlen der Antriebsräder und die Differentialsperre ist noch nicht wirksam.

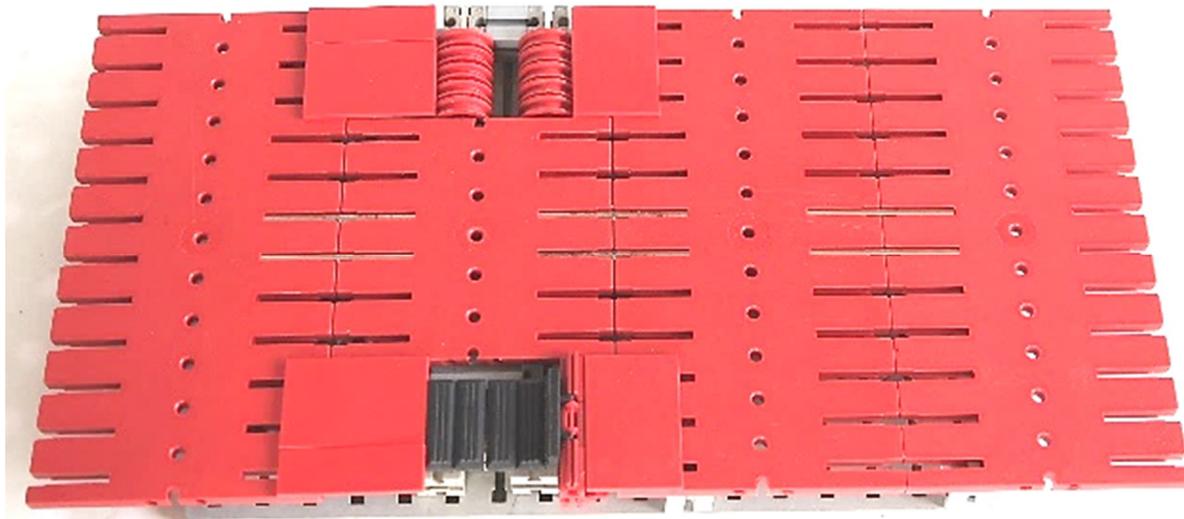


Abb. 6: Differentialfalle



Abb. 5: Rechtskurve

Wirkung bei durchdrehendem Rad

Jetzt kommen wir zu dem Fall, in dem die Differentialsperre wirkt. Dazu ist zunächst in Abb. 6 eine „Differentialfalle“ aufgebaut.

Oben sind Rollen, auf denen das eine Antriebsrad durchdrehen soll, während das andere unten vor dem kleinen Hindernis auf den Gummis blockieren soll. Damit kann die eingangs geschilderte „Durchdreh-situation“ simuliert werden.

In Abb. 7 kommt das Fahrzeug geradeaus angefahren und die Fliehkraftbremse dreht sich nicht.

Dann, in Abb. 8, bleibt es in der Differentialfalle hängen und wie erwartet dreht das obere Rad durch, während das untere blockiert. Die Fliehkraftbremse beginnt sich mit der Differenzdrehzahl zu drehen, welches genau die Drehzahl des durchdrehenden Rades ist.

Da das aber deutlich schneller ist als während der Kurvenfahrt, schlagen jetzt die Statikstreben aus und eine verhakt sich an dem Grundbaustein (Abb. 9).

Die Drehzahldifferenz wird dadurch zwangsweise auf null gesetzt, und durch die dargestellten Zusammenhänge an dem Getriebe wirkt das rückwärts auf die Antriebsachse, wo sich jetzt beide Räder gleichschnell drehen müssen. Die Differentialsperre wirkt also, das untere Rad beginnt sich wieder mitzudrehen und hebt das Auto über das Hindernis, siehe Abb. 10. Voilà!

Wie in Abb. 11 zu sehen ist, löst sich die Statikstrebe durch die nachlassende Kraft und die Vibrationen wieder und die Differentialsperre ist bereit für den nächsten Einsatz.

Leider ist die Idee nicht neu

Wer den Artikel von Thomas Püttmann in der ft:pedia 4/2014 oder die „Technikgeschichte mit fischertechnik“ [3] gelesen hat, weiß, dass bereits die alten Chinesen mit dem Kompasswagen den wichtigsten Teil des hier gezeigten Getriebes erfunden hatten. Sie nutzten die Differenzdrehzahl, um einen Zeiger auf einem Wagen immer in die gleiche Himmelsrichtung zeigen zu lassen. Zur Differentialsperre fehlte ihnen nur noch die Fliehkraftbremse (und der Motor mit Geländewagen drumherum). Wirklich erstaunlich, oder?

Referenzen

- [1] Thomas Püttmann: *Zahnräder und Übersetzungen (Teil 2)*, in: [ft:pedia 2011-3](#), S. 25-28.
- [2] Thomas Püttmann: *Das Differentialgetriebe*, in: [ft:pedia 2014-4](#), S. 12-19.
- [3] Dirk Fox und Thomas Püttmann: [*Technikgeschichte mit fischertechnik*](#). dpunkt-Verlag, Heidelberg 2015.
- [4] Martin Wanke: [*fischertechnik: Automatische Differentialsperre*](#). Modellvideo auf youtube, 2018.



Abb. 7: anrückendes Fahrzeug

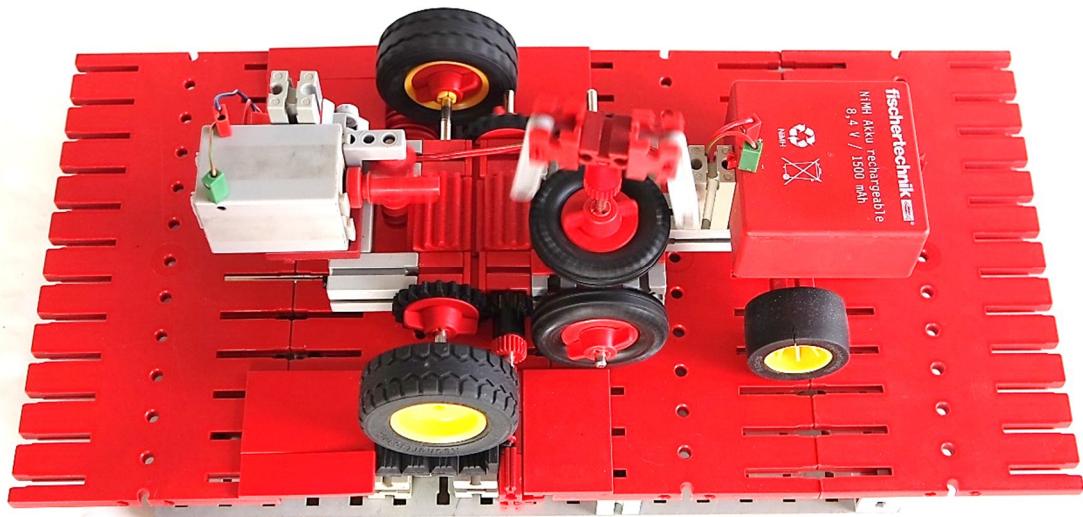


Abb. 8: oberes Rad dreht durch

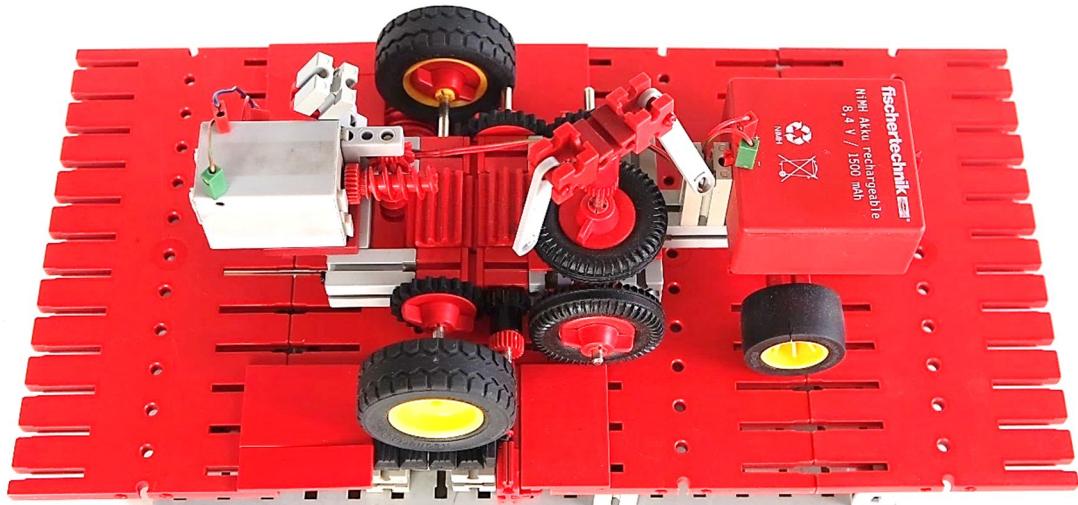


Abb. 9: Fliehkraftbremse wirkt



Abb. 10: Hindernis überwunden

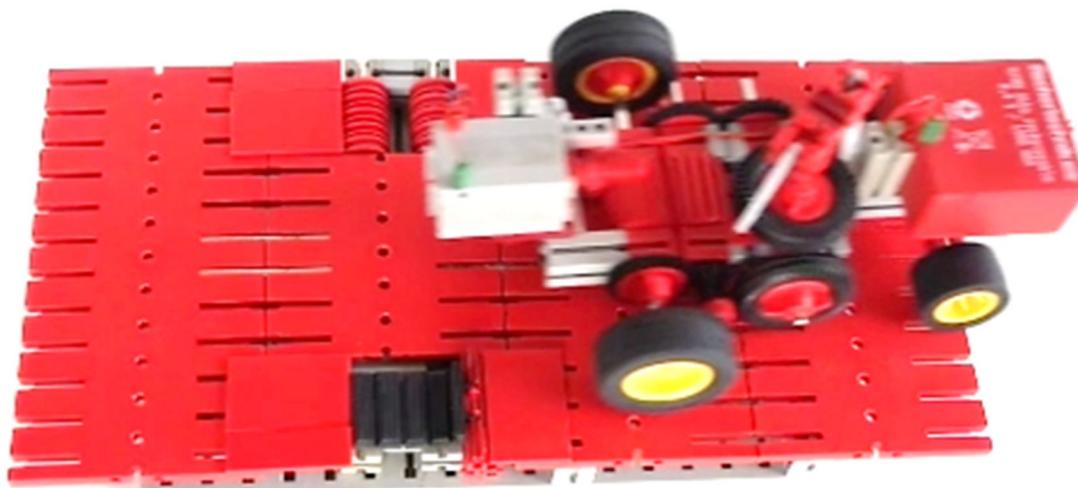


Abb. 11: Fliehkraftbremse wieder gelöst

Elektronik

Neopixel für alle



Christian Bergschneider, Stefan Fuss

Nach der weihnachtlichen Lektüre des Artikels über programmierbare fischertechnik-LEDs in der ft:pedia 4/2017 musste der fischertechniker sofort das coolste und am besten beleuchtete Modell aller Zeiten beginnen. Aber: Muss man jetzt auch noch das Programmieren von Arduinos erlernen? Wie bekommt man das Modell nun vom TX(T) aus angesteuert? Der NeopixelController löst beide Probleme: Die Firmware beherrscht im stand-alone-Betrieb u. a. Lauflicht-, NightRider- und Rainbow-Effekte. Im I²C-Modus können die LEDs vom TX(T) mit Robo Pro gesteuert werden. Es werden nur wenige elektronische Bauteile benötigt, so dass der Nachbau für jeden geeignet ist.

Der NeopixelController ist die Weiterentwicklung der Idee aus [1] und besteht im Wesentlichen aus einem Arduino Nano. Dieser wird auf eine Prototypen-Platine gelötet und um wenige einfache Bauteile ergänzt. Die fertige Firmware wird auf dem Arduino Nano eingespielt – und schon kann es losgehen.

Wer Zugriff auf einen 3D-Drucker hat, kann sich noch ein schickes Gehäuse herunterladen und drucken (Abb. 1). Die Schaltung funktioniert aber genauso gut mit etwas Heißkleber auf einer Bauplatte 30·90 6Z (Artikelnummer [38251](#)).

NeoPixel

Die WS2812B-LEDs erfreuen nicht nur des fischertechnikers Herz – in den Raspberry- und Arduino-Communities werden viele Modelle mit diesen LEDs gepimpt. Neben der in [1] vorgestellten Bauform findet man unter dem Namen NeoPixel viele fertige Platinen mit LEDs in Kreis-, Stick- oder Matrixform, die nur darauf warten, endlich

in ein fischertechnik-Projekt eingebaut zu werden.

Aus Fernost gibt es Nachbauten der NeoPixel-Platinen von Adafruit. Beide Varianten benutzen die gleichen LED-Typen und können deshalb am Controller eingesetzt werden.



Abb. 1: Der Controller im Gehäuse aus dem 3D-Drucker

Die LEDs gibt es als RGB- und RGBW-Bautypen. Die RGB-Baureihe arbeitet mit drei LEDs, die RGBW-Baureihen haben zusätzlich eine weiße LED. Leider sind die Steuerprotokolle der beiden Typen nicht kompatibel, so dass sich an einem Controller gleichzeitig entweder nur RGB- oder nur RGBW-LEDs ansteuern lassen und man sie in einem Modell nicht mixen kann. Bei Adafruit sind die RGBW-Typen an der Bezeichnung zu erkennen: *natural white*, *warm white* und *cool white* sind RGBW-Typen, alle anderen sind RGB-Typen.

Vierpoliger Anschluss

Wie in [1] im Detail beschrieben, werden die LEDs über einen seriellen Bus angeschlossen. Mehrere Platinen bzw. LEDs können einfach hintereinander kaskadiert werden. Grundsätzlich reicht ein dreipoliges Kabel, bestehend aus GND, +5 V und Datensignal, um die Neopixel-Platinen miteinander zu verbinden.

Die eingesetzten einreihigen Stift- und Buchsenleisten sind platzsparend und preiswert. Es fehlt jedoch ein Schutz gegen Verpolung. Wird ein dreipoliges Kabel um 180° verdreht aufgesteckt, so kommt es bei dreipoligen Steckern zum Kurzschluss. Wir haben den vierpoligen Anschluss des Neopixel Sticks übernommen. Werden hier die Kabel falsch zusammengesteckt, so funktioniert nur die Ansteuerung der LEDs nicht; Controller und LEDs nehmen keinen Schaden.

Stromversorgung

Der NeopixelController und die LEDs benötigen eine Versorgungsspannung von 5 V. Diese kann auf zwei Arten bereitgestellt werden: Ein altes USB- oder Handy-Netzteil kann am USB-Anschluss des Nano angeschlossen werden und die ganze Schaltung versorgen. Alternativ verfügt der Nano auch über einen kleinen Spannungsregler, der die fischertechnik-Standardspannung von 9 V auf die benötigten 5 V herunterregelt.

Am USB-Anschluss des PCs dürfen maximal 500 mA abgenommen werden. Der Spannungsregler auf dem Arduino Nano kann aus dem 9 V-fischertechnik-Anschluss bis zu 800 mA zur Verfügung stellen. Die Maximalleistung von USB-Netzteilen ist auf dem Gehäuse ablesbar und liegt im Bereich von 300 mA bis ca. 3 A.

Es gibt USB-Netzteile mit mehr als 3 A Leistung. Diese sollten *nicht* eingesetzt werden, da die Leiterbahnen auf den Nano-Platinen dafür nicht ausreichend dimensioiniert sind.

Maximale LED-Anzahl

Eine RGB-LED hat bei voller Leuchtkraft eine Stromaufnahme von 60 mA. RGBW-LEDs benötigen bis zu 80 mA. Für die Schaltung selbst müssen 30 mA veranschlagt werden. Die maximale Anzahl der betreibbaren RGB-LEDs lässt sich somit mit der folgenden Formel bestimmen:

$$LEDs = \frac{\text{Netzteilstrom} - 30 \text{ mA}}{60 \text{ mA}}$$

Für die Berechnung von RGBW-LEDs müssen die 60 mA auf 80 mA erhöht werden.

	RGB-LEDs		RGBW-LEDs	
Netzteil	100%	50%	100%	50%
ft 9 V	13	26	10	20
1,5 A	25	50	18	36
2 A	33	66	25	50
2,5 A	41	82	31	62
3 A	50	100	37	74

Tab. 1: Berechnung der max. LED-Anzahl

Bei der Stromversorgung mit Standardfischertechnik lassen sich somit 13 RGB-LEDs bei maximaler Helligkeit betreiben. Die LEDs sind jedoch sehr leuchtstark, so dass im realen Modell die maximale Leistung nicht benötigt wird. Dies kann in der Leistungskalkulation berücksichtigt

werden. In der Firmware des Controllers haben wir die maximale Helligkeit sowohl im *Stand Alone* als auch im I²C-Modus auf 12,5% beschränkt.

Schaltplan

Die Schaltung besteht nur aus wenigen Bauteilen (Abb. 2). Der Kern der Schaltung, ein Arduino Nano, ist eine *Open Source Computing Platform*. Die Platine darf lizenzfrei nachgebaut werden, so dass viele verschiedene Hersteller den Nano anbieten. Auch wenn die Preise stark schwanken, die Hardware – und damit die Funktion – ist bei allen Nanos gleich.

Der 10-polige DIP-Schalter SW1 dient der Auswahl des Betriebsmodus und ist an die digitalen Eingänge D3-D12 des Arduino Nano angeschlossen. Durch die internen Pull-Up-Widerstände im Nano ist keine weitere Beschaltung notwendig.

Die Buchse JP1 stellt die Stromversorgung und das Steuersignal für die LEDs von Pin D2 des Arduino zur Verfügung. Hier

werden die Neopixel-Platinen über Kabel angeschlossen.

Auf JP2 wird über einen 10-poligen Stecker der I²C-Bus für den TXT bereitgestellt. Da der TXT mit 3,3 V arbeitet und der Arduino ein 5 V-Modell ist, wird die Logikspannung über die beiden *Level Shifter* (R1 – R4, Q1, Q2) angepasst.

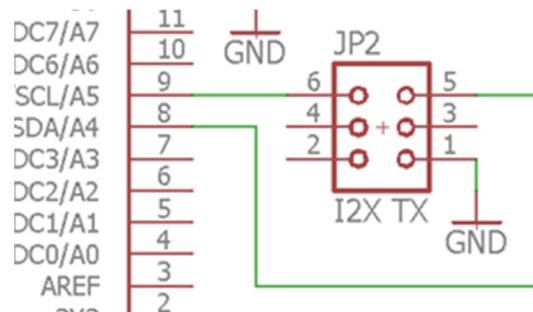


Abb. 3: Vereinfachter Anschluss des TX

Beim Betrieb am TX (Abb. 3) entfallen die *Level Shifter*, da der TX mit 5 V Logikspannung arbeitet. Die Anschlüsse A4 und A5 des TX können dann direkt mit einem 6-poligen Stecker verbunden werden.

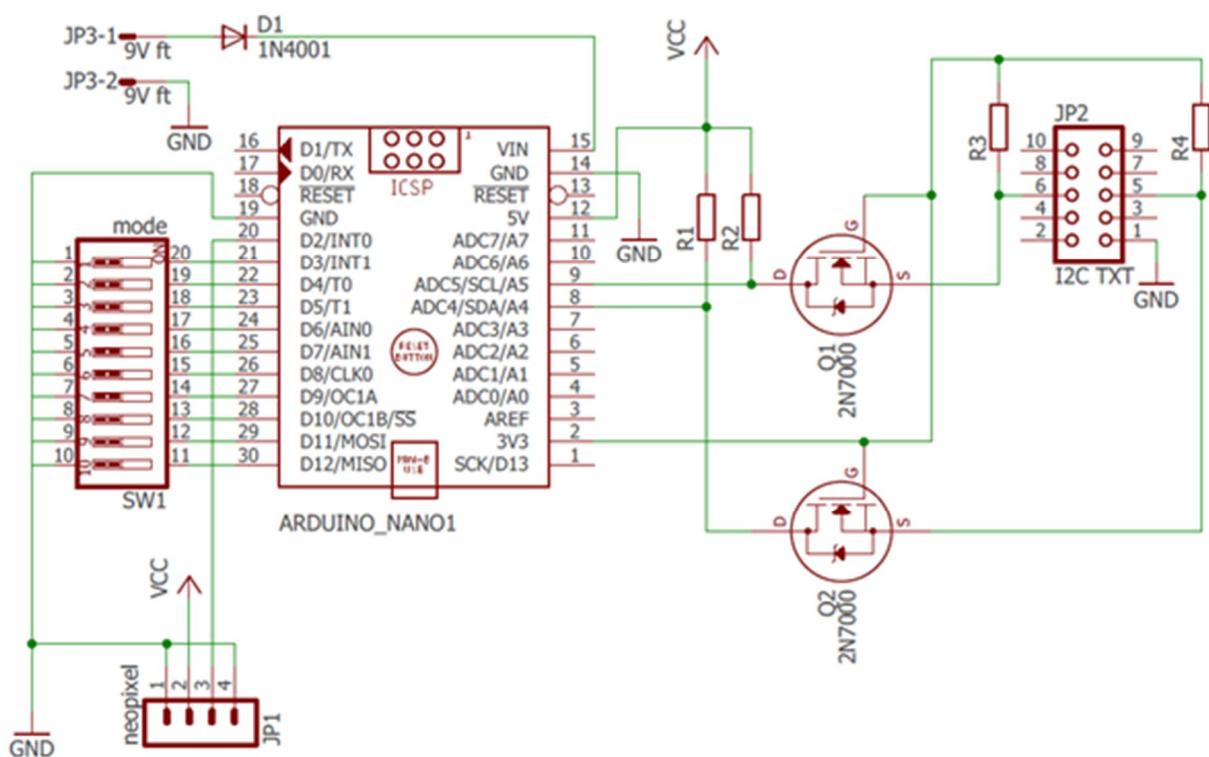


Abb. 2: Der Schaltplan des Controllers

Die fischertechnik-Bordspannung wird an JP3 eingespeist. Die Diode D1 (1N4001) dient als Verpolungsschutz.

Nachbau des Controllers

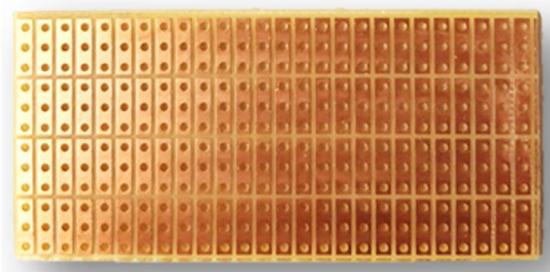


Abb. 4: Experimentierplatine

Der NeopixelController ist sehr einfach aufgebaut, so dass ihn auch Lötanfänger nachbauen können. Wer keine Lötkenntnisse hat, kann sich diese über den kleinen Lötkurs [2] leicht aneignen.

Zunächst muss die Platine ausgeschnitten werden. Wir haben eine Streifenrasterplatine aus Hartpapier verwendet, bei der jeweils drei Lötpunkte miteinander verbunden sind. Hartpapier lässt sich sehr gut bearbeiten, die verbundenen Lötpunkte sparen später beim Löten viele Verbindungen.

Die Platine muss $13 \cdot 25$ Rastereinheiten bzw. Bohrlöcher groß werden. Am einfachsten markiert man zunächst die 14. Lochreihe auf der Kupferseite. Mit einem Cutter-Messer wird nun die Kupferauflage in der Mitte der 14. Lochreihe durchtrennt. Anschließend wird die Platine auf Höhe der eingeschnittenen Lochreihe an der Tischkante angelegt und gebrochen. Das entstandene Werkstück wird nun im gleichen Verfahren an der 26. Lochreihe gebrochen. Mit Hilfe von feinem Schmirgelpapier wird der Rest der angebrochenen Lochreihen entfernt, so dass die Platine nun $68,5 \cdot 32$ mm groß ist (Abb. 4).

Soll die Platine später auf die Bauplatte geklebt werden, so muss die Platine drei Streifen breiter ausfallen, um Platz für die Buchsen für die fischertechnik-Bordspannung zu haben. In diesem Fall muss die Platine nicht an der 26., sondern an der 29. Lochreihe aufgetrennt werden. Die Platine wird dann auf 76,2 mm Breite angepasst.

Die vier Widerstände werden nun für den senkrechten Einbau vorbereitet. Dazu wird ein Drahtende nahe am Bauteil um 180° gebogen.

Q1, Q2	TXT: 2 · MOS-FET 2N7000 TX: keine MOS-FETs
R1-R4	TXT: 4 · Kohleschichtwiderstände 10 kΩ, 1/4W TX: keine Widerstände
D1	1 · Diode 1N4001
SW1	1 · 10-poliger DIP-Schalter
Platine	1 · Streifenrasterplatine aus Hartpapier, mindestens 76,2 · 32 mm
LED-Kabel	1 · 4-polige Buchsenleiste, einreihig 1 · 4-polige Stiftleiste, einreihig 4-poliges Flachbandkabel ggf. Schrumpfschlauch

JP1	1 · 4-polige Buchsenleiste, einreihig
JP2	TXT: 1 · 10-poliger Wannenstecker TX: 1 · 6-poliger Wannenstecker
JP3	2 · Bundhülse 11 mm oder 2 · Lötstift 2 · Zwerghananenbuchse 4 mm
Arduino	1 · Arduino Nano
I2C Kabel	TXT: 2 · 10-polige Pfostenbuchse TX: 2 · 6-polige Pfostenbuchse 10-poliges Flachbandkabel
NeoPixel	Gewünschte NeoPixel-Platine(n). Je Platine: 1 · 4-polige Buchsenleiste, einreihig, evtl. 5 mm-Bauform 1 · 4-polige Stiftleiste, einreihig

Tabelle 2: Bauteilliste

Die Diode wird wie die Widerstände für den senkrechten Einbau vorbereitet. Im Gegensatz zu den Widerständen muss bei der Diode auf die richtige Polung geachtet werden. Dafür muss die Seite ohne Strichmarkierung um 180° umgebogen werden.

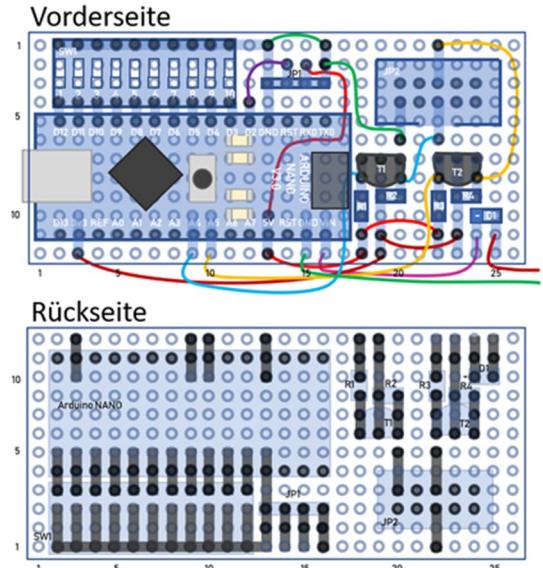


Abb. 5: Bestückungsplan für den TXT

Die Pins der MOS-FETs für die Level Shifter müssen etwas aufgebogen werden, so dass sie in drei nebeneinanderliegende Lötpunkte der Platine passen. Beim späteren Einbau muss drauf geachtet werden, dass die abgeflachte Seite entsprechend dem Bestückungsplan eingebaut wird. Bei manchen Lieferanten sind am Nano bereits die Stiftleisten montiert. Liegen die Stiftleisten dem Nano lose bei, so müssen nun die beiden langen Stiftleisten eingelötet werden. Dabei wird das kurze Ende der Pins von unten in die Nano-Platine gesteckt und von oben verlötet. Um später Platz im Gehäuse zu sparen, wird die beiliegende 2 · 3-polige Stiftleiste nicht am Nano eingelötet; sie wird nicht benötigt. Handelt es sich um einen komplett vormontierten Nano, so werden die sechs Pins der Stiftleiste mit einem Seitenschneider knapp über dem Plastik abgeschnitten.

Auf der Papierseite werden nun die Bauteile zur Probe auf die Platine aufgesteckt. Auf der Lötseite muss kontrolliert werden, dass

die Position der Bauteile auf den Lötstreifen dem Bestückungsplan entspricht. Der Nano wird anschließend wieder entfernt, alle anderen Bauteile werden jetzt verlötet.

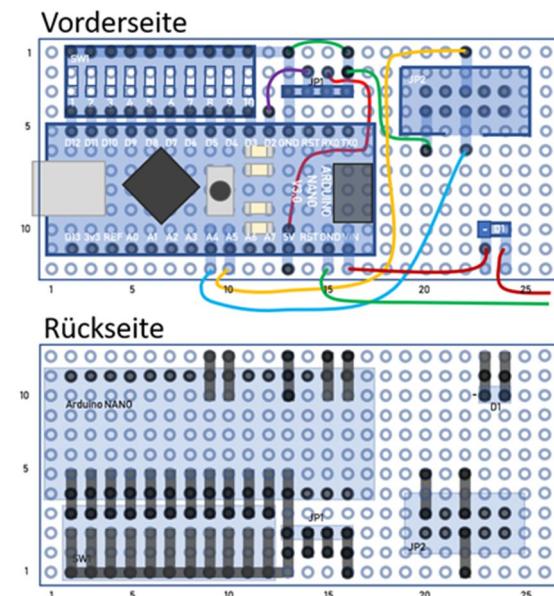


Abb. 6: Bestückungsplan für den TX

Entsprechend dem Bestückungsplan werden jetzt die Bauteile untereinander verbunden. Dazu müssen einerseits reine Lötbrücken auf der Rückseite hergestellt werden und andererseits auf der Vorderseite kurze Kabelbrücken eingelötet werden. Beim Ablängen und Verlegen der Kabel muss der Platz für den Arduino freigehalten werden.

Die Kabel zum Anschluss an die fischertechnik-Stromversorgung werden zunächst nur auf der Platine angelötet. Das andere Ende wird erst beim Einbau in das Gehäuse angeschlossen. Anschließend wird der Arduino selbst direkt auf die Platine gelötet. Das reduziert die Bauhöhe der Platine. Bei den geringen Kosten für einen Arduino machen Buchsenleisten zum Stecken auch keinen Sinn.

Zum Abschluss der Lötarbeiten bitte nochmals genau den Bestückungsplan mit der Platine auf Fehler vergleichen und auf ungewollte Kurzschlüsse und Brücken achten.

Firmware flashen

Die Schaltung wird nun über ein USB-Kabel am PC angeschlossen. Der Arduino Nano bootet, die PWR-LED muss konstant leuchten.

Zum Flashen der Firmware wird die Arduino-IDE benötigt. Dazu zunächst die IDE bei [6] downloaden, auf dem PC installieren und starten.

In der IDE müssen als erstes der Arduino-Board-Typ und der Port eingestellt werden. Dazu muss im Menü *Werkzeuge, Board* die Option *Arduino Nano* ausgewählt werden:

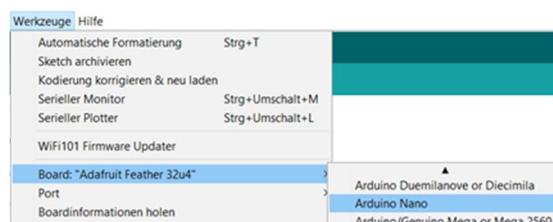


Abb. 7: Auswahl des Board-Typs in der IDE

Für die Kommunikation des PCs mit dem Nano wird am PC eine COM-Schnittstelle emuliert. Die COM-Schnittstelle wird in der IDE im Menü unter *Werkzeuge, Port* ausgewählt. Die Nummer der emulierten Schnittstelle hängt vom verwendeten USB-Port ab. In der Regel ist es die höchste gelistete Nummer.

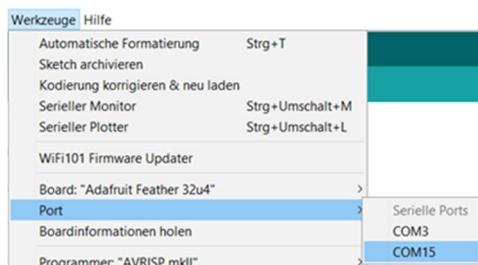


Abb. 8: Auswahl des Ports

Über den Menüpunkt *Werkzeuge, Board-information holen* kann getestet werden, ob die Kommunikation mit dem Nano einwandfrei funktioniert.

Die Firmware benutzt die Adafruit-Neopixel-Bibliothek zur Ansteuerung der LEDs. Die Installation der Bibliothek erfolgt im Bibliotheksverwalter. Er wird im

Menüsysteem über *Sketch, Bibliothek einbinden, Bibliotheken verwalten* aufgerufen:



Abb. 9: Aufruf der Bibliotheksverwaltung

Es erscheint der *Bibliotheksverwalter*. Über die Suche nach *Neopixel* findet man die richtige Library und kann sie automatisch installieren lassen:



Abb. 10: Installation der NeoPixel-Bibliothek

Die Firmware für den Controller kann bei [4] oder [5] heruntergeladen und in *Dokumente\Arduino* als Unterverzeichnis *NEO_I2C* ausgepackt werden. In der Arduino-IDE öffnet man über das Menü mit *Datei, Öffnen* im Verzeichnis *NEO_I2C* die Datei *NEO_I2C.ino*.

Zum Schluss erfolgt das eigentliche Flashen der Firmware. Im Menü geschieht das über *Sketch, Hochladen*:

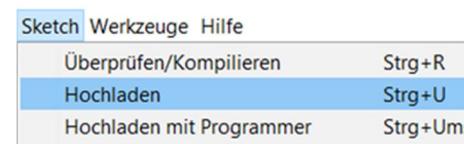


Abb. 11: Flashen der Firmware

Der Prozess dauert einige Minuten, da zunächst die IDE den Sketch (Quelltext) compiliert und dann erst an den Nano überträgt.

Neopixel-Stick

Der Neopixel-Stick (Abb. 12) ist am einfachsten zu verbauen. Er hat auf der Rückseite links und rechts jeweils vier

Lötpads. Die Reihenfolge entspricht der Belegung des Buskabels am Controller.

Die 4-polige Buchsenleiste wird an der rechten Seite angelötet. Hier ist das Datensignal mit DIN beschriftet. An der linken Seite (DOUT) wird die Stifteleiste angebracht.

Die Buchsenleisten gibt es in zwei Bauformen. Die Standardleiste hat 8,5 mm Höhe; die 5 mm-Bauform spart etwas Platz.

Der Stick wird einfach in den 3D-gedruckten fischertechnik-Adapter geklippt und ggf. mit Heißkleber verklebt. Der Neopixel-Stick kann ohne 3D-Druck alternativ mit mehreren Federnocken [31982](#) im Modell festgemacht werden.

Neopixel-Jewel

Der Zusammenbau des Neopixel-Jewels (Abb. 13) benötigt einiges an Fingerspitzengefühl, da mehrere Kabel in einem Lötpad zusammen eingelegt werden müssen. Außerhalb des Gehäuses werden an Stecker und Buchse ca. 4 cm lange, möglichst dünne Kabel angelötet. Danach werden Buchse und Stecker von außen in die passenden Öffnungen am Gehäuse geschoben; die Kabel werden in der Mitte herausgeführt. Mit Heißkleber werden nun Buchse und Stecker inkl. Kabel am Gehäuseboden fixiert.

Die Kabelenden werden jetzt am Jewel angelötet. Durch die dünnen Kabel,



Abb. 12: Neopixel-Stick

lassen sich bei GND und +5 V jeweils zwei Kabel in einem Lötpad zusammenfassen.

Neopixel-Ring

Bei den Lötarbeiten zum Neopixel-Ring aus dem Motorrad auf dem Titelbild muss das Gehäuse von einer dritten Hand stabil gehalten werden.

Es werden zunächst die 4-polige Buchsenleiste und die 4-polige Stifteleiste in das gedruckte Gehäuse gesteckt, so dass die Pins im Gehäuse mit dem Lötkolben gut erreicht werden können.

Die Anschlüsse für +5 V und die beiden für GND werden vom Stecker zur Buchse mit kurzen Kabelstücken verbunden. Anschließend schließt man mit weiteren Kabeln die 5 V- und GND-Pins der Platinen an, verbindet DATA auf der Buchsenleiste mit dem Pin Data Input der Platine sowie DATA auf der Steckerleiste mit Data Output.

Schließlich richtet man Buchse und Stecker im Gehäuse korrekt aus und fixiert sie mit Heißkleber. Der Ring selbst wird nun ins Gehäuse geklippt.

Profi-Lights

Die Profi-Lights aus [1] lassen sich ebenfalls an den Controller anschließen. Es wird nur ein Adapterkabel benötigt, um vom 4-poligen Bus auf den 3-poligen Anschluss der Profi-Lights zu reduzieren.



Abb. 13: Neopixel-Jewel

I²C-Kabel

Das I²C-Kabel verbindet den TX(T) und den Neopixel-Controller. Beim TXT ist es ein 10-poliges Kabel, beim

10-8	Betriebsmodus	7	LED-Typ	6-4	Farbe	3-1	Anzahl LEDs
	I ² C-Modus*				Grün		1
	Night Rider				Rot		2
	Running Light				Blau		4
	Fill				Pink		8
	Blink				Lila		12
	Fader				Gelb		16
	SideStep				Hellblau		32
	Rainbow3 *				Weiß		64
		1	RGB				
		0	RGBW				

* Bei den Betriebsmodi I²C und Rainbow haben die Schalter 1-7 eine andere Bedeutung. Bitte Text beachten!

Tab. 3: Einstellung der DIP-Schalter im stand-alone-Modus

TX ein Sechspoliges. Es kann am einfachsten mit Hilfe einer Wasserpumpenzange hergestellt werden: Das 10-polige Kabel auf die gewünschte Länge zuschneiden, das rot markierte Kabel an der Markierung in die Klemmvorrichtung einlegen und mit der Wasserpumpenzange vorsichtig zusammenpressen. Das Ganze wird am anderen Kabelende wiederholt.

Verbindungskabel

Aus dem Rest des 10-poligen Flachbandkabel werden nun 4-polige Kabel geschnitten. Die Länge kann je nach Modell variiert werden. An die Kabel werden auf einer Seite 4-polige Stecker und auf der anderen Seite 4-polige Buchsenleisten angelötet. Wer mag, kann die Löstellen mit Schrumpfschlauch schützen.

Stand-Alone-Betrieb

Hierfür werden die Neopixel-LEDs angeschlossen, die DIP-Schalter auf das gewünschte Programm eingestellt und zum Schluss die Versorgungsspannung eingeschaltet.

Die Betriebsmodi und deren Parameter können im laufenden Betrieb gewechselt werden. Der Wechsel findet immer am

Ende einer Sequenz statt, so dass es einige Sekunden dauern kann, bis das neue Programm startet. In Tab. 3 sind die einzelnen Funktionen des DIP-Schalters dargestellt.

Beim Rainbow-Modus kann keine Farbe ausgewählt werden, die Schalter 6 – 4 stellen die Geschwindigkeit des Farbwechsels ein.

Programmierung in Robo Pro

Die Programmierung in Robo Pro ist sehr einfach, da nur wenige Kommandos benötigt werden. Die Robo Pro-Bibliothek `neopixel_controller.rpp` arbeitet mit der I²C-Busadresse 0x11.

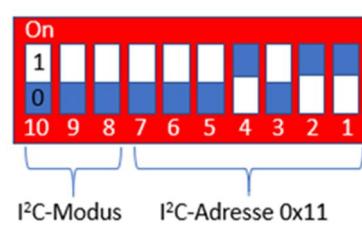


Abb. 14: DIP-Schalter für Betrieb am TX(T)

Der Controller wird über das I²C-Buskabel an den TX(T) angeschlossen. Am DIP-Schalter wird der Betriebsmodus I²C und die Busadresse 0x11 ausgewählt (Abb. 14). Nachdem der Controller über USB oder

fischertechnik-Bordspannung mit Strom versorgt ist, kann es losgehen. Die bereitgestellten Robo Pro-Programme lauten

- *neopixel_controller*: Bibliothek zur Ansteuerung des NeopixelControllers
- *npc_const_3color*: Bibliothek Konstanten für RGB-LEDs
- *npc_const_4color*: Bibliothek Konstanten für RGBW-LEDs
- *neopixel_sample*: Beispieldiagramm

Im Programm *neopixel_sample.rpp* müssen vor der Benutzung ggf. die Parameter der Funktion *NPC_8_init* korrigiert werden. Der Parameter *LE* benötigt die korrekte LED-Anzahl.

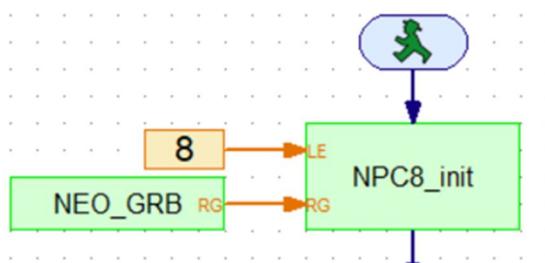


Abb. 15: Anpassung der Init-Funktion im Beispieldiagramm

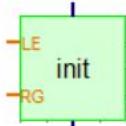
Bei *RG* muss der LED-Typ angegeben werden. Der Parameter kann gefahrlos ausprobiert werden, in Abb. 15 ist es eine 3-Farb-LED mit der Farbreihenfolge GRB.

Das Beispieldiagramm zeigt die Verwendung der Basisfunktionen. Eine komplette Funktionsübersicht ist auf den nächsten Seiten in Tab. 4 dargestellt.

Referenzen

- [1] Harbaum, Till: *Profi-Lights: WS2812B-Vollfarb-Leuchtdioden im fischertechnik-Design*. ft:pedia 4/2017, S. 15-18.
- [2] Bergschneider, Christian: *Löten leicht gemacht*. ft:pedia 1/2018.
- [3] Fuss, Stefan: *Arduino Sensoren am TXT*. ft:pedia 1/2018.
- [4] Christian Bergschneider, Stefan Fuss: [ftcommunity-neopixel](#). GitHub Repo auf GitHub, 2018.
- [5] Christian Bergschneider, Stefan Fuss: [NeopixelController](#). Firmware und Robo Pro-Bibliothek im ftc-Downloadbereich, 2018.
- [6] Arduino: [Download the Arduino IDE](#).

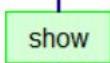
Funktionsreferenz der Robo Pro-Unterprogramme



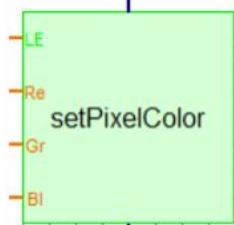
Initialisiert den NeopixelController. Es werden die Anzahl der LEDs *LE* und der RGB-Typ *RG* angegeben.



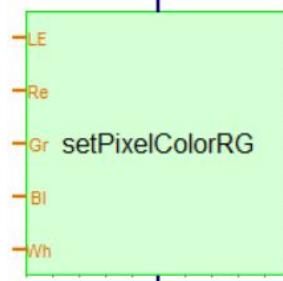
Die Funktion *setBrightness* stellt die Helligkeit der LEDs ein. Alle LEDs bekommen die gleiche Einstellung. Gültige Werte für Brightness sind 0 bis 32. Die Werte 32 bis 255 können zwar übertragen werden, auf dem Controller greift dann aber die Helligkeitsbegrenzung in der Firmware.



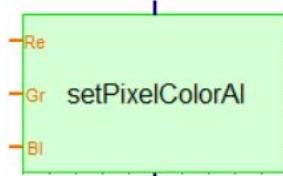
Der NeopixelController arbeitet intern mit einem Pixel-Buffer. Um mehrere Kommandos schnell auszuführen, finden zunächst alle Schreibkommandos im Buffer statt. Danach wird über die Funktion *show* der Buffer an die Pixel übertragen.



Mit *setPixelColor* wird die Farbe einer LED eingestellt. Der Parameter *LED* ist die Nummer der LED. Die erste LED am NeopixelController hat die Nummer 0. Die drei anderen Parameter (*Red*, *Green*, *Blue*) sind die RGB-Werte für die LED.



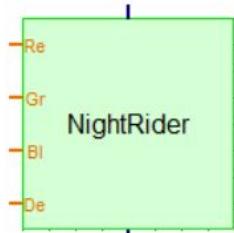
Die Funktion *setPixelColorRGBW* setzt die Pixelfarbe einer RGBW-LED. Der Parameter *LED* ist die Nummer der LED. Die erste LED am NeopixelController hat die Nummer 0. Die vier anderen Parameter (*Red*, *Green*, *Blue*, *White*) sind die RGWB-Werte für die LED.



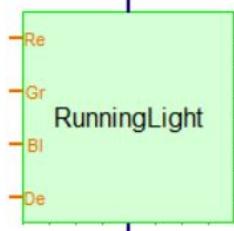
setPixelColorAll setzt die Farbe aller angeschlossenen LEDs auf den übergebenen RGB-Wert.



Regenbogeneffekt. Parameter *In* (Inkrement) gibt an, wie stark der Farbwechsel in einem Schritt ist; *De* die Dauer in ms.



Lauflicht im Night-Rider-Design. *Re*, *Gr* und *Bl* sind die RGB-Werte, *De* die Dauer in ms, bis der Leuchtpunkt auf die nächste LED wechselt.



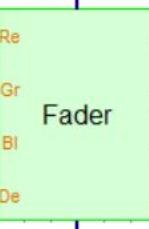
Normales Lauflicht. *Re*, *Gr* und *Bl* sind die RGB-Werte, *De* die Dauer in ms, bis der Leuchtpunkt auf die nächste LED wechselt.



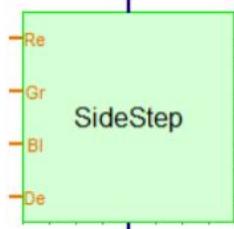
Lässt nacheinander von links nach rechts neue Leuchtpunkte einlaufen, bis alle LEDs an sind. *Re*, *Gr* und *Bl* sind die RGB-Werte, *De* die Dauer in ms, bis der Leuchtpunkt auf die nächste LED wechselt.



Alle LEDs blinken an/aus. *Re*, *Gr* und *Bl* sind die RGB-Werte, *De* die Dauer in ms.



Alle LEDs gehen langsam an und aus. *Re*, *Gr* und *Bl* sind die RGB-Werte, *De* die Dauer in ms.



Zwei nebeneinanderliegende Leuchtpunkte laufen von links nach rechts. *Re*, *Gr* und *Bl* sind die RGB-Werte, *De* die Dauer in ms.

Tab. 4: Funktionsreferenz der ROBO Pro-Unterprogramme

Elektronik

Arduino-Sensoren am TX(T)

Stefan Fuss

Arduinos sind sehr preiswerte Microcontroller, und es gibt für sie jede Menge passender Sensoren. Ob Feuchtesensor, Laserdiode oder Mini-Joystick – die ganze Welt der Arduinos lässt sich mit wenig Soft- und Hardwareaufwand in fischertechnik-Modelle integrieren.

Arduinos sind klein, leicht zu verstehen und aus der Maker-Szene nicht mehr wegzudenken. Dazu gibt es ganze Kisten mit fertigem Zubehör. Nur noch zusammenstecken, Beispielprogramm und Sensor-Bibliothek herunterladen und schon geht es los.

Den integrierten I²C-Bus des Arduino kann man frei programmieren, so dass man den Arduino auch als Bindeglied zum TX(T) benutzen kann. Coole Erweiterungen wie Laserdioden und Joysticks lassen sich so ruckzuck aus Robo Pro ansprechen.

Soweit die Theorie. Die Dokumentation der I²C-Kommandos von Robo Pro ist allerdings nicht gerade vorbildlich. Arduino-als-I²C-Master-Tutorials gibt es viele, doch gute Slave-Tutorials sind rar. Zum Glück ist in [3] sehr gut der Slave-Betrieb des Arduino beschrieben und mit den Protokollbeschreibungen aus [1] und [2] werden auch plötzlich die Felder der Robo Pro-I²C-Bus-Kommandos klar.

Alle Mann ans Steckbrett

Der Laser aus dem „37-in-1 Sensor Kit“ von Elegoo kann über einen PWM-Ausgang des Arduinos in der Helligkeit gesteuert werden. In unserem Testaufbau (Abb. 1) werden wir mit dem Laser das Senden von Kommandos an den Arduino ausprobieren.

Im gleichen Set gibt es einen kleinen Joystick. Die X- und Y-Position können vom Arduino als analoge Werte ausgelesen

werden. Drückt man auf dem Joystick, schaltet zusätzlich noch ein Digitaleingang. Beide Funktionen eignen sich somit, um das Auslesen von Daten am Arduino zu testen.

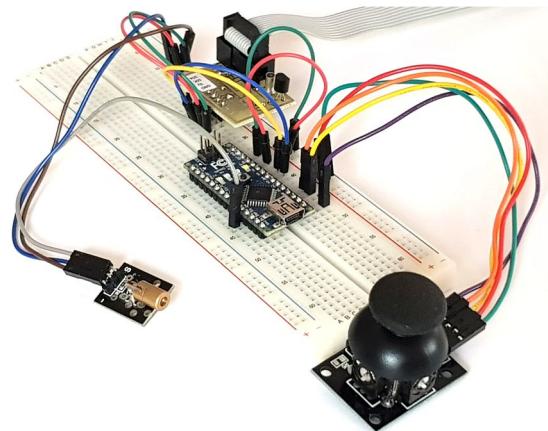


Abb. 1: Testaufbau mit Joystick und Laserdiode

Auf einem Steckbrett ist der Testaufbau schnell realisiert. Basis ist ein Arduino Nano, es eignen sich aber genauso gut alle anderen Arduino Typen.

Die meisten Arduinos arbeiten mit 5V-Logikpegeln. Für den Anschluss an den TXT, der mit 3,3V-Logikpegeln arbeitet, werden deshalb Levelshifter benötigt. Im Testaufbau wurde der Levelshifter aus [4] eingesetzt. Wer es einfach mag, greift auf fertige Levelshifter aus dem Internet zurück. Für die Nutzung am TX wird kein Levelshifter benötigt.

Der Joystick wird an die Anschlüsse A0, A1 und D13 angeschlossen, die Laserdiode wird mit D9 verbunden. Mit oder ohne

Levelshifter werden A4 und A5 mit SDA (Pin 5) und SCK (Pin 6) des TX(T) verbunden.

Wer möchte kann auch das Gehäuse und die Basisschaltung aus [5] übernehmen. Der DIP-Switch wird weggelassen, so dass hier ausreichend Platz entsteht, um Anschlüsse für die Sensoren aus dem Kit bereitzustellen.

Von Herren und Dienern

Der I²C-Bus ist hardwareseitig sehr einfach ausgelegt. Die Idee stammt aus den 80ern und sollte mit preiswerten Mitteln Microcontroller und Sensoren verbinden. Das Protokoll für den Datenaustausch ist fast vollständig über Software gelöst.

Alle Geräte am Bus teilen sich eine Datenleitung. Um Chaos zu vermeiden darf deshalb immer nur ein Teilnehmer am Bus Daten senden. Die Vergabe der Senderechte ist einfach: ein Gerät am Bus – der Master – initiiert jeden Datentransfer. Alle anderen sind Slaves und müssen machen, was der Master ihnen vorgibt.

In der Implementierung von fischertechnik ist der TX(T) immer der Master. Alle anderen Bausteine müssen deshalb Slaves sein. Auf dem TX(T) wird das Steuerprogramm des Modells ausgeführt, so dass der TX(T) auch jede Kommunikation durch sein Programm initiieren will. Ein Interruptverfahren, das den Slaves ermöglicht, spontan Daten zu senden, existiert in der fischertechnik-Welt nicht.

Um mehrere Slaves an einem Bus betreiben zu können, erhält jeder Slave eine 7-Bit-Adresse. Da der Master jede Kommunikation auslöst, braucht er selbst keine eigene Adresse. Die I²C-Adresse ist bei den fertigen I²C-Breakoutboards entweder fest vorgegeben, oder kann in Maßen per Lötjumper eingestellt werden. Bei ganz modernen Bausteinen kann man sogar per Schreibkommando am I²C-Bus die Adresse des Bausteins festlegen. Beim Arduino wird die I²C-Adresse durch die Software gesetzt.

In Robo Pro wird in den I²C-Bus-Kommandos die angesprochene I²C-Busadresse im Feld „Geräteadresse“ angegeben.

Verbotene Bereiche

Die I²C-Adresse eines Bausteins kann nicht frei im 7-Bit-Adressraum gewählt werden. Die jeweils kleinsten und höchsten 16 Adressen (0x00 bis 0x0F sowie 0xF0 bis 0xFF) sind für Sonderfunktionen reserviert und dürfen nicht verwendet werden.

Daten senden

Der einfachste Fall der Datenübertragung ist das Senden von Daten vom Master an den Slave. Der Master beginnt eine Schreibtransmission, indem er zunächst ein Start-Bit – oder auch Startkondition – gefolgt von einem Adressbyte sendet. Darauf folgen ein oder mehrere Datenbytes, die Transmission wird mit dem Stop-Bit oder auch Stop-Kondition beendet (Abb. 2).

	Adresse		R/W	Daten													
Master	START	0	0	1	0	0	1	0	1	x	x	x	x	x	x	x	STOPP
Slave																	ACK

Abb. 2: Senden eines Datenbytes an Adresse 0x12

Der Slave erkennt den Schreibzugriff daran, dass in der Adresse das achte Bit gesetzt ist und der Rest der Adresse seine I²C-Bus-Adresse darstellt. Nach der Adresse und jedem Datenbyte bestätigt der Slave den Empfang mit Acknowledge, beim letzten Byte findet kein Acknowledge statt.

Die Interpretation der Datenbytes erfolgt durch den Slave. Für den Aufbau der Datenbytes gibt es beim I²C-Bus keine Vorgaben.

Daten empfangen

Der Lesezugriff funktioniert nach dem gleichen Prinzip: Der Master initiiert eine Transmission und sendet im einfachsten Fall die Adresse des Slaves. Dabei ist das siebte Bit der Adresse nicht gesetzt; der Slave erkennt daran einen Lesevorgang und beginnt nun seine Datenbytes zu senden (Abb. 3).

	Adresse	R/W	Daten	
Master	START	0 0 1 0 0 1 0	0	STOPP
Slave			ACK X X X X X X X X	

Abb. 3: Auslesen eines Datenbytes an Adresse 0x12

Wie beim Schreiben gibt es beim Lesen keinen vorgegebenen Aufbau der Datenpakete.

Register

Ein Slave hat am I²C-Bus nur eine Adresse. Um komplexe Funktionen bereitzustellen, wird ein Slave häufig in mehrere Register unterteilt. Für die Registeradressierung wird beim Lesezyklus nach dem Adressbyte als nächstes Byte das Register gesendet. Erst danach sendet der Slave seine Datenbytes (Abb. 4).

	Adresse	R/W	Register	Daten	
Master	START	0 0 1 0 0 1 0	0	0 0 0 1 0 0 0 1	STOPP
Slave			ACK	ACK X X X X X X X X	

Abb. 4: Auslesen des Registers 0x09 an Adresse 0x10

Das Lesen aus dem Register funktioniert nach der gleichen Idee. Der Master sendet das Adressbyte gefolgt von der Registernummer. Der angesprochene Slave sendet anschließend den Registerinhalt.

In ROBO Pro wird in den I²C-Bus-Kommandos das angesprochene Register im Feld „Geräteadresse“ angegeben. Der Arduino kennt für die Registeradressierung keine getrennten Prozeduren. Das Register wird als normales Datenbyte empfangen und muss von der Software verarbeitet werden.

Clock stretching

Der angesprochene Slave benötigt bei einem Lesekommando u. U. etwas Zeit, um die angeforderten Daten zusammenzustellen. Ein Lesekommando an einen Register kann z. B. das Auslesen eines analogen Messwertes triggern. Der Messvorgang selbst benötigt jedoch Zeit, bevor der Slave die Antwort senden kann.

Nachdem der Master seine Leseanforderung gesendet hat, wartet er einen kurzen Moment auf die Antwort des Slaves. Nach

Ablaufzeit der Timeoutzeit des Masters gilt der Lesevorgang als gescheitert und der Slave darf seine Antwort nicht mehr senden.

Der Slave kann den Timeout des Masters verhindern, indem er die Clock-Leitung auf Low zieht und damit jeglichen Datentransport blockiert. Der Master erkennt die Situation und wartet ggf. länger als die Timeoutzeit auf das Ergebnis.

Dieser Vorgang wird *clock stretching* genannt. Um den Bus jedoch nicht zu blockieren – während des *clock stretching* kann auch kein Busteilnehmer Daten senden – sollten längere Messungen asynchron zur Datenübertragung gestaltet werden.

Ansteuerung des Lasers

Am TX(T) soll in ROBO Pro nun zunächst der Laser gesteuert werden. Der Arduino reagiert auf die Slave-Adresse 0x10. Über das Register 0x01 wird die Intensität des Lasers als ein Datenbyte geschrieben.

Das ROBO Pro-Programm hierfür ist einfach:

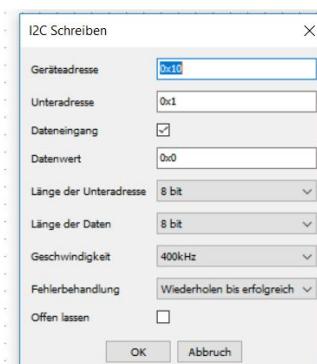


Abb. 5: ROBO Pro sendet die Intensität 42 an die Laserdiode.

Für das Beispiel wird nur ein ROBO Pro-Kommando benötigt, das Adresse, Register und Datenwert überträgt.

Zur Verarbeitung der Daten auf dem Arduino benötigen wir etwas mehr Code. In der Setup-Routine muss zunächst der I²C-Bus initialisiert werden. Außerdem muss der Ausgang D9 für die Laserdiode initialisiert werden (Listing 1):

```
#include <Wire.h>
#define MyI2CBusAddress 0x10
#define LaserPin 9
int Register = 0;

void setup() {
    Wire.begin(MyI2CBusAddress);
    Wire.onReceive(receiveEvent);
    pinMode(LaserPin, OUTPUT);
}
```

Listing 1: Initialisierung des I²C-Busses und des Ausgangs für die Laserdiode

Der Arduino hört nun auf der I²C-Busadresse 0x10. Werden Daten an den Arduino gesendet, d. h. an den Slave mit der Adresse 0x10, so wird beim Arduino ein Interrupt ausgelöst.

Die Ausführung von jedwedem Code in der `loop` wird sofort unterbrochen und es wird die Funktion `receiveEvent` aufgerufen. Diese Prozedur muss ebenfalls implementiert werden:

```
void receiveEvent(int bytesReceived) {
    Register = Wire.read();
    if (Register == 0x01) {
        analogWrite(LASERPIN,
                    Wire.read());
    }
}
```

Listing 2: Setzen der Helligkeit der Laserdiode

Der Arduino liest das Register und das Datenbyte aus und setzt die Helligkeit des Lasers. Damit der erste Testlauf erfolgen kann, wird noch das Hauptprogramm benötigt. Für den Test ist es ausreichend, den Prozessor einfach nur warten zu lassen:

```
void loop() {
    delay(10000);
}
```

Listing 3: Das Hauptprogramm macht nichts

Auslesen des Joysticks

Der Joystick liefert drei Werte zurück: die X- und die Y-Position des Joysticks als Analogwert, den Taster als boolean.

Der ROBO Pro-Beispielcode ist wieder einfach. Der Joystick soll über das Register 0x02 angesprochen werden. Mit drei getrennten Lesekommandos in einer

Transmission werden die Werte für den Taster, X und Y eingelesen:

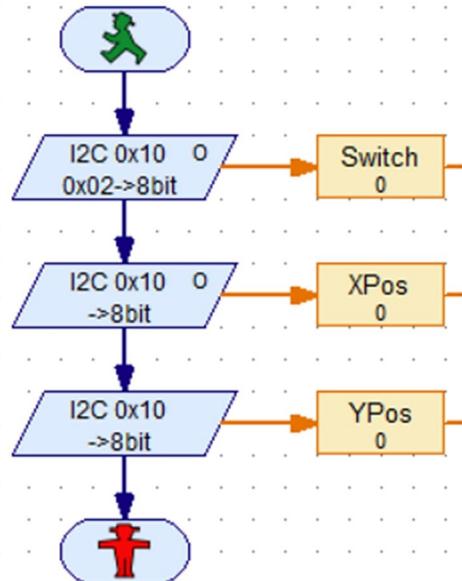


Abb. 6: Aus dem Register 0x02 werden drei Byte gelesen

Die drei Bytes Information für den Taster und die beiden Positionen sollen in einer Transmission ausgelesen werden. Dazu muss in ROBO Pro bei den ersten beiden Lesekommandos ‚Offen lassen‘ gewählt werden.

Sollen nur ein oder zwei Byte gelesen werden, würde wiederum ein Kommando ausreichen. Im Feld ‚Länge der Daten‘ kann zwischen 8 und 16 Bit gewählt werden.

Die Setupprozedur setzt wieder die Funktion der Pins und I²C-Busadresse, registriert aber zusätzlich die Prozedur `receiveRequest`:

```
#include <Wire.h>

#define MyI2CBusAddress 0x10
#define LaserPin 9
#define SWPin 13
#define XPin 0
#define YPin 1

byte Register = 0;

void setup() {
    Wire.begin(MyI2CBusAddress);
    Wire.onReceive(receiveEvent);
    Wire.onRequest(receiveRequest);
```

```
PinMode( LaserPin, OUTPUT);
PinMode(SW_Pin, INPUT);
}
```

Listing 4: In der Setuproutine muss nun zusätzlich receiveRequest definiert werden.

Die drei ROBO Pro-Kommandos lösen nun auf der Arduino-Seite zwei Events aus. Zunächst wird `receiveEvent` durch die Übertragung des Registers ausgelöst. Anschließend wird `receiveRequest` aufgerufen und der Arduino muss die Werte des Joysticks an den TX(T) zurücksenden.

Bei der Laserdiode berücksichtigt bereits die Implementierung von `receiveEvent` die Verarbeitung der beiden Register, so dass die Funktion für den Joystick nicht angepasst werden muss.

In der neuen Funktion `receiveRequest` müssen nun die Werte für den Taster und die beiden Positionen bestimmt und übertragen werden:

```
void receiveRequest() {
    byte Result[2];
    if (Register = 0x02) {
        Result[0] =
digitalRead(SWPin);
        Result[1] = analogRead(XPin);
        Result[2] = analogRead(YPin);
        Wire.write(result, 3 );
    }
}
```

Listing 5: In requestEvent werden die Messwerte übertragen

Als Slave kennt der Arduino keine Kommandos zum Steuern einer Transmission, da

diese vom Master kontrolliert wird. Die Antwort auf einen Request kann deshalb immer nur über ein `wire.write`-Kommando ausgeführt werden.

Im Beispielcode werden dazu die ausgelesenen Daten in einen Array geschrieben. Im `write`-Kommando wird dann das erzeugte Array übertragen.

Referenzen

- [1] Maike Thomas: [Die Zweidrahtbus-systeme I²C-Bus und SPI-Bus: Eigenschaften, Protokolle, Anwendungen im Vergleich der beiden Systeme](#). Seminararbeit, 2008.
- [2] rn-wissen: [I²C, Inter-IC bzw. Inter Integrated Circuit Bus](#).
- [3] dsscircuits: [Step by Step Guide on Making an I²C Slave Device with an Arduino](#).
- [4] Christian Bergschneider, Stefan Fuss: [Ein universeller I²C-Adapter für den TX\(T\)](#). ft:pedia 4/2016, S. 72-79.
- [5] Christian Bergschneider, Stefan Fuss: [Neopixel für alle](#). ft:pedia 1/2018.
- [6] Dirk Fox: [I²C mit TX und Robo Pro – Teil 1: Grundlagen](#). ft:pedia 3/2012, S. 32-27.

Computing

Scratch mit fischertechnik

Dirk Fox

Die am Massachusetts Institute of Technology (MIT) entwickelte Einsteigerprogrammiersprache Scratch wird inzwischen in zahlreichen Schulen eingesetzt. Mit den von fischertechnik im Github zum Download bereitgestellten Programmen „FTScratchTXT“ und „FTScratchLT“ lassen sich seit Ende 2016 auch der Robotics TXT Controller und der LT Controller mit einer entsprechenden Befehlserweiterung direkt aus Scratch(X) ansteuern. Nach Auskunft von fischertechnik soll bald auch der BT Smart Controller mit Scratch genutzt werden können.

Hintergrund

Scratch wurde unter der Leitung von Professor Mitchel Resnick am MIT Media Lab entwickelt und erstmals im Jahr 2007 veröffentlicht. Scratch soll vor allem Kinder ansprechen und spielerisch mit dem Programmieren vertraut machen. Die Programmierungsumgebung (IDE) verfügt über ein Ausgabefenster, „Bühne“ genannt, auf der gezeichnet, aber auch Objekte (wie der Scratch-Kater) bewegt werden können (Abb 1, links). Die 360 x 480 Positionen des Fensters können über ein Koordinatensystem (Mittelpunkt = (0, 0)) angeprochen werden.

Die Programmierung erfolgt mit grafischen Befehlselementen („Skripte“), die in einem Programmfenster zusammengeschoben („gestapelt“) werden. Ein Programm-Block wird entweder durch ein externes Ereignis (z. B. eine Mausbewegung) oder einen Klick auf eine grüne Fahne über der „Bühne“ gestartet und kann mit einem Klick auf einen roten Stopp-Knopf gehalten werden. Die Umgebung unterstützt – ebenso wie ROBO Pro – parallele Prozesse: Im Programmfenster können mehrere Programmblöcke platziert und von unterschiedlichen Ereignissen gestartet werden. Das ermöglicht sehr kurze und elegante Programmsteuerungen.

Scratch

Die [Scratch-Entwicklungsumgebung](#) (IDE, Abb. 1) ist über eine Web-Schnittstelle zugänglich. Um diese zu nutzen benötigt man eine Internet-Verbindung. Es gibt auch einen „Offline Editor“ in der Version 2.0 zum [Download](#), der allerdings nicht mit den fischertechnik-Erweiterungen genutzt werden kann.

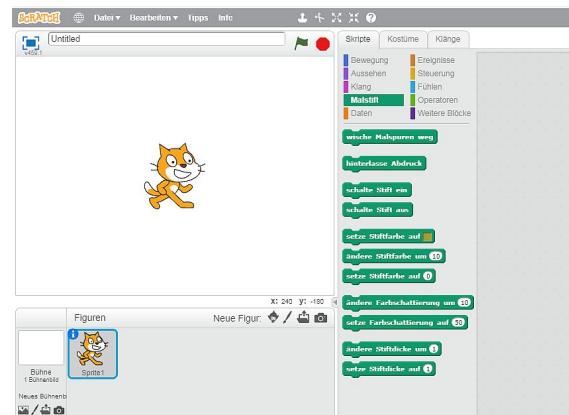


Abb. 1: Scratch-IDE

Scratch bietet alle üblichen Programmstrukturen wie Wiederholungen, Wartezeiten und Verzweigungen („Steuerung“). Daneben unterstützt es logische und mathematische Operationen („Operatoren“), Ereignisse zum Auslösen einer Operation („Ereignisse“) sowie die Auswertung von Eingaben („Fühlen“) wie der Mausposition, den Maustasten oder Timern.

Die Objekte – selbst gezeichnet oder aus einer Bibliothek ausgewählt – die auf der Bühne erscheinen, können mit entsprechenden Skripten bewegt („Bewegung“) oder mit andern Kostümen ausgestattet werden („Aussehen“). Sie können sich sogar Nachrichten zusenden, die auf der Bühne in einer Sprechblase angezeigt werden. Schließlich können Töne erzeugt oder eingespielt („Klang“) und mit einem virtuellen Stift Zeichnungen angefertigt werden („Malstift“).

Eine tiefer gehende Einführung in Scratch spare ich mir hier, da es im Internet zahlreiche hervorragende Einsteiger-Tutorials und Referenzhandbücher gibt, siehe z. B. [1, 2]. Außerdem enthält Scratch Schritt-für-Schritt- und Kurzanleitungen, die direkt aus der IDE aufgerufen werden können.

Die erzeugten Scratch-Programme („Projekte“) können lokal oder in der „Scratch-Cloud“ gespeichert und umgekehrt in die IDE hochgeladen werden.

FTScratch

In der Entwicklerversion [ScratchX](#) wurden fischertechnik-Funktionen als Erweiterung („Weitere Blöcke“) ergänzt. Als „Verbindungsglied“ zwischen ScratchX und dem TXT bzw. LT Controller dienen zwei ausführbare Programme – [FTScratchTXT](#) bzw. [FTScratchLT](#) –, die vor dem Aufruf von ScratchX im Browser gestartet werden müssen und die Koppelung mit dem jeweiligen Controller übernehmen. Sie sind für die Betriebssysteme Windows und Linux/Mac verfügbar – damit können die fischertechnik-Controller ohne „Klimmzüge“ auch aus anderen Betriebssystemen als Windows genutzt werden. Als Browser werden Firefox und Chrome empfohlen.

Unter Windows benötigen [FTScratchLT](#) und [FTScratchTXT](#) das .NET Framework 4 – es muss bei etwas älteren Betriebssystemversionen möglicherweise nachinstalliert werden.

Eine Offline-Version von ScratchX gibt es leider nicht, und die entwickelten Programme können auch nicht in Scratch 2.0 genutzt werden. Einzige Lösung für die Nutzung ohne Internet-Zugang: Die Installation von ScratchX auf einem lokalen Webserver – das ist aber eine für einen Einsteiger kaum geeignete Option, daher gehe ich hier auch nicht näher auf diesen Lösungsweg ein.

Für die Nutzung von FTScratchTXT/LT müssen zuvor lediglich die USB-Treiber für die beiden Controller installiert werden (sofern das nicht bereits geschehen ist). Die Github-Seiten zu [FTScratchLT](#) und [FTScratchTXT](#) [3, 4] enthalten sehr klare (und mehrsprachige) Installationsanleitungen.

Die erweiterte ScratchX-IDE erlaubt allerdings keinen Download von kompilierten Projekten auf den Controller, sondern nur einen Online-Betrieb. In der Praxis ist das allerdings nur eine begrenzte Einschränkung: Der LT Controller unterstützt ohnehin keinen Programm-Download, und wenn man die Verbindung zum TXT via Bluetooth oder WLAN herstellt, ist auch so ein kabelloser Betrieb möglich.

Startet man FTScratchTXT (derzeit v1.23), werden die unterschiedlichen Verbindungsoptionen angezeigt (Abb. 2).

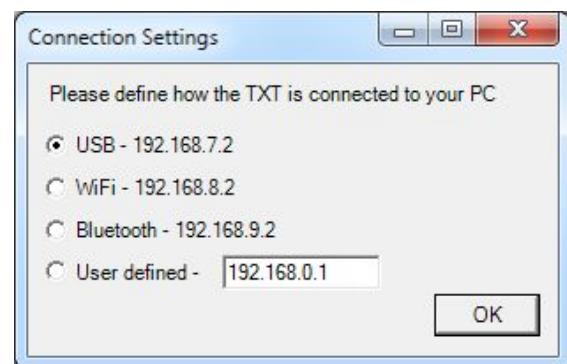


Abb. 2: Verbindungsoptionen mit dem TXT in FTScratchTXT (Windows-Version)

Ist der TXT angeschlossen und die Verbindung aufgebaut, werden in einem Kontroll-Fenster (Abb. 3) die an den acht Eingängen anliegenden Werte bzw. Signale sowie die

Zählerstände der vier schnellen Zählereingänge angezeigt. In der oberen Zeile wird die Version der auf dem TXT installierten Firmware (aktuell: 4.2.4.0) angegeben.

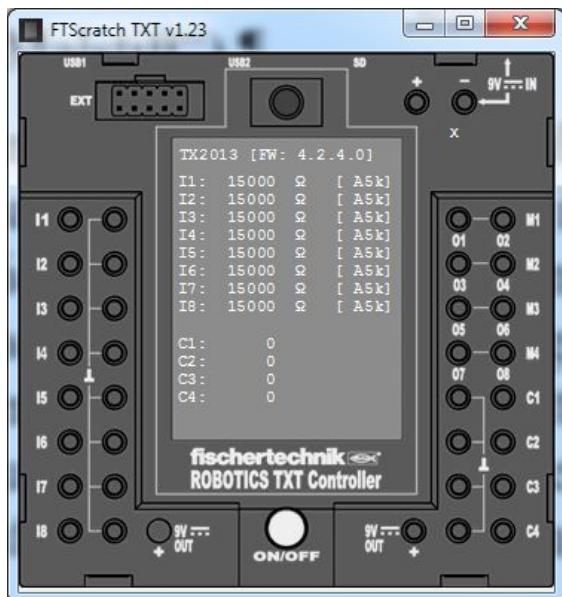


Abb. 3: FTScratchTXT-Kontrollfenster
(Windows-Version)

Das Kontrollfenster von FTScratchLT zeigt ebenfalls die an den Ein- und Ausgängen des LT Controllers anliegenden Werte an (Abb. 4).



Abb. 4: FTScratchLT-Kontrollfenster
(Windows-Version)

Ereignisse

FTScratch für den LT Controller erweitert Scratch um ein neues (Start-) Ereignis: Die Auswertung eines digitalen Inputs – Schalter, Reed-Kontakt oder Fotodiode an I1 bis I3 (beim TXT: I1 bis I8). FTScratch für den TXT kennt zwei weitere Startereignisse:

Die Auswertung eines Zählerwerts C1-C4 und die der analogen Sensoren (Farbsensor, Abstandssensor, NTC-Widerstand und Fotowiderstand) an I1 bis I8 (Abb. 5).



Abb. 5: Ereignisse in FTScratch für den TXT

Operatoren

Scratch-Operatoren umfassen einfache und komplexe arithmetische Operationen, Vergleiche, logische Verknüpfungen, die Erzeugung von Zufallszahlen sowie die Verarbeitung von Zeichenfolgen (Text).

FTScratch für den LT ergänzt die Operatoren um die booleschen Operatoren „Schalter geschlossen?“ und „Lichtschranke geschlossen?“ an I1-I3. Außerdem kann der aktuelle Ausgabewert der beiden Motorausgänge abgefragt werden. FTScratch für den TXT kann über den booleschen Operator außerdem einen Reed-Kontakt abfragen, den Wert der Zählereingänge C1-C4 sowie den eines analogen Sensors (Farbsensor, Abstandssensor, NTC-Widerstand und Fotowiderstand) an I1 bis I8 auslesen (Abb. 6).



Abb. 6: FTScratch-Operatoren für den TXT

Klang

FTScratch für den TXT bringt auch zusätzliche Klang-Befehle mit: Damit lassen sich bis zu 29 auf dem TXT installierte Sounddateien abspielen (Abb. 7). Beim zweiten Befehl wird das Programm erst fortgesetzt, wenn der Klang fertig abgespielt wurde.

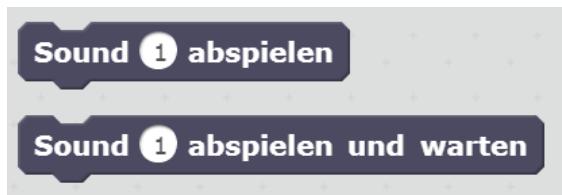


Abb. 7: FTScratch-Klang-Befehle für den TXT

Ausgangssteuierung

FTScratch erweitert die Befehls-Skripte von Scratch vor allem um die Möglichkeit, die PWM-Ausgänge (Pulsweitenmodulation) – beim LT: M1 und M2, beim TXT: M1-M4 bzw. O1-O8 – für die Motor- bzw. Lampensteuerung mit einem Wert von 0 bis 8 zu belegen (Abb. 8). Die feinere Steuerung über 512 Schritte wird von FTScratch nicht unterstützt.

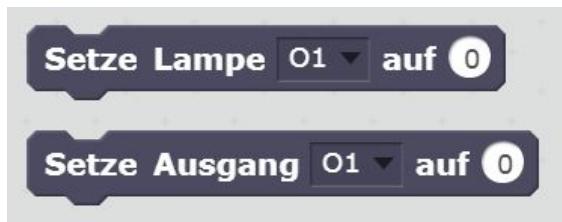


Abb. 8: FTScratch-Lampensteuerung

Bei den (einfachen) Motorbefehlen kann zusätzlich die Bewegungsrichtung (vorwärts/rückwärts) gewählt werden (Abb. 9).



Abb. 9: FTScratch-Motorsteuerung

FTScratch bietet für den TXT zusätzlich erweiterte Motorbefehle, mit denen eine Ansteuerung der Encoder-Motoren möglich ist (Abb. 10).

Schließlich können beim TXT noch die Eingänge auf „digital“ (Voreinstellung: „analog“) und die Zähler C1 bis C4 zurückgesetzt werden. Mit einem einzigen Befehl kann der gesamte TXT auf die Anfangseinstellung (Eingänge: analog, Ausgänge: 0) gesetzt werden (Abb. 11).



Abb. 11: Setzen der Eingänge, Rücksetzen der Zählereingänge und des gesamten TXT



Abb. 10: Erweiterte Motorbefehle von FTScratch für den TXT

Beispielprogramme

Ampelsteuerung

Versuchen wir nun, eine erste einfache Programmieraufgabe mit FTScratch zu lösen: Die Steuerung einer Fahrzeug-Lichtzeichenanlage mit Fußgänger-Bedarfsampel.

Wir brauchen insgesamt fünf Ausgänge, um die Lichtzeichenanlage der Fahrzeuge (rot, gelb, grün) und die Fußgängerampel (rot, grün) anzusteuern – dafür benötigen wir einen TXT. Wer einen LT verwenden möchte, muss die Steuerung auf die Fußgängerampel (zwei Ausgänge) beschränken.

Die Fußgänger-Bedarfsampel soll drei Sekunden nach dem Drücken des Bedarfschalters auf Grün wechseln und nach weiteren fünf Sekunden wieder auf Rot. Die Verzögerungszeiten zwischen den Wechseln der Phasen der Lichtzeichenanlage (Abb. 12) soll dabei eine Sekunde betragen.

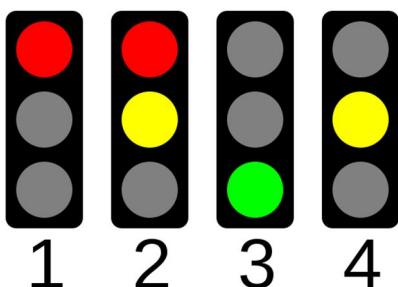


Abb. 12: Phasen der Lichtzeichenanlage

Abb. 13 zeigt ein einfaches Modell einer solchen Lichtzeichenanlage (vulgo Ampel). Die (grünen) GND-Anschlüsse sind, um Kabel zu sparen, über eine Verteilerplatte verbunden; nur die 9V-Anschlüsse haben eine direkte Verbindung zu den Lampenausgängen des TXT.

Der Anschluss der Lämpchen an den TXT ist einfach: Für die in Abb. 14 gezeigte Steuerung wurde die Lichtzeichenanlage für die Fahrzeuge an die Ausgänge O1 (rot), O2 (gelb) und O3 (grün) angeschlossen, die Fußgängerampel an O4 (rot) und O5 (grün).

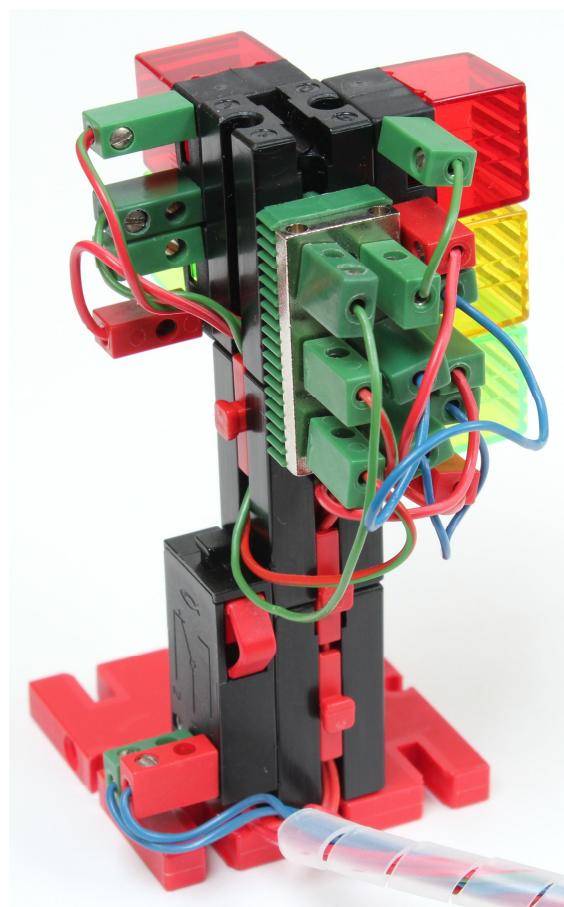


Abb. 13: Fußgänger-Bedarfsampel



Abb. 14: Steuerung der Bedarfsampel

Wer über ausreichend viele Lämpchen verfügt, kann auch gleich das Gegenstück für die andere Straßenseite hinzubauen und die Lämpchen parallel schalten; weitere Ausgänge werden dafür nicht benötigt.

Parkhausschranke

Mit der zweiten Programmieraufgabe wollen wir eine einfache Schranke z. B. an einer Parkhauseinfahrt steuern: Kommt ein Fahrzeug vorgefahren und drückt der Fahrer eine Taste, soll sich die Schranke öffnen. Sie soll so lange geöffnet bleiben, bis der Fahrer eine Lichtschranke (alternativ einen Reed-Kontakt in der Fahrbahn) durchfahren hat, dann soll sie schließen.

Damit die Steuerung auch mit einem LT-Controller gelingt, können wir das Anhalten der Schranke in der jeweiligen Endlage durch eine einfache Motorsteuerung lösen (wie das geht, lässt sich in [5] nachlesen). Dafür genügen uns ein Motorausgang und zwei Eingänge (Taster, Lichtschranke).

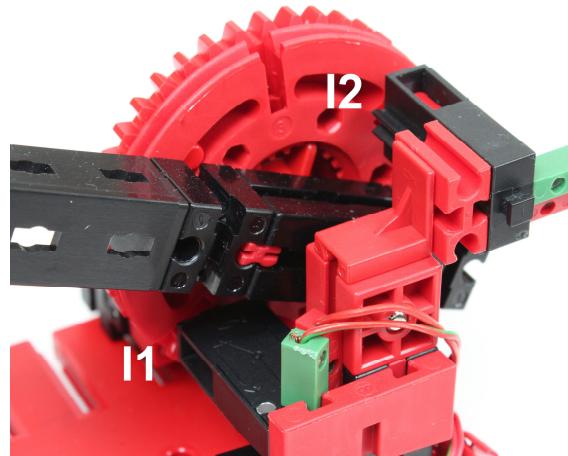


Abb. 15: Endlagentaster der Schranke

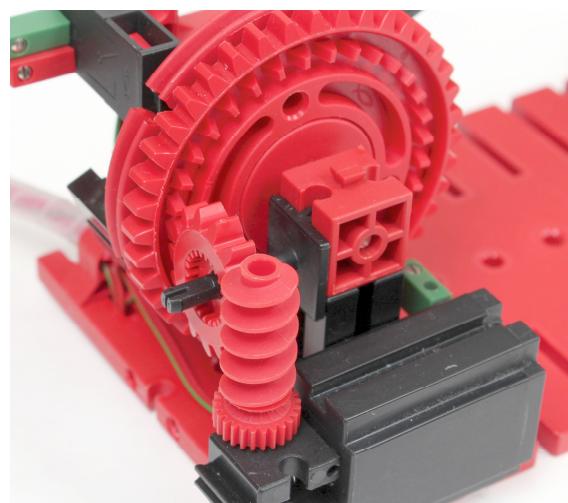


Abb. 16: Schrankenantrieb

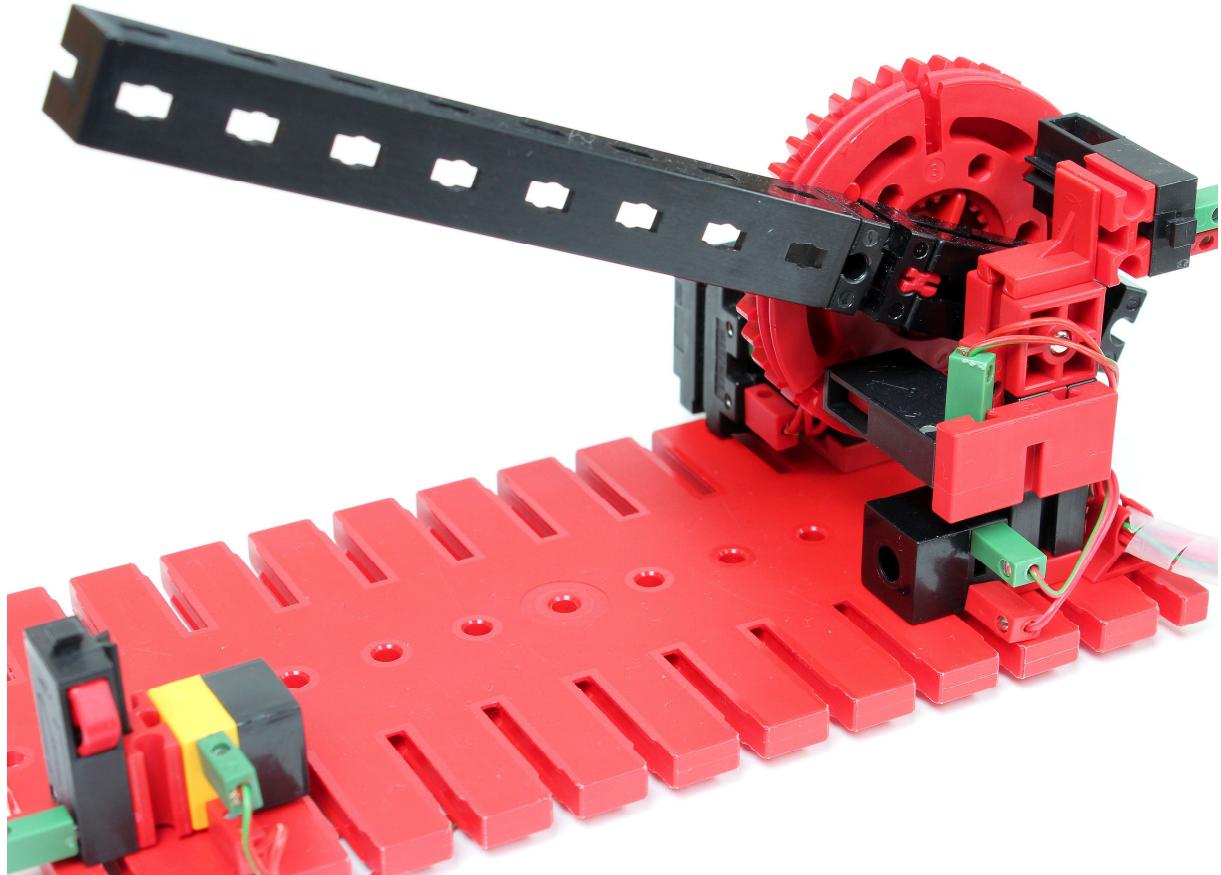


Abb. 17: Gesamtansicht der Schranke

Abb. 15-17 zeigt unser Modell der Parkhausschranke mit Endlagentastern und Lichtschranke. Wem der Nachbau nicht anhand der Fotos gelingt, kann sich eine [Designer-Datei des Modells](#) herunterladen.

In unserem Modell fragen wir die Endlagentaster am TXT ab. Dafür benötigen wir vier digitale Eingänge. In Abb. 15 sind die beiden an I1 und I2 angeschlossenen Endlagentaster gut zu erkennen, die in geöffneter bzw. geschlossener Stellung schalten; der Taster zur Aktivierung der Schranke wird an I3, die Fotodiode (Lichtschranke) an I4 angeschlossen.

Abb. 18 zeigt eine Lösung für eine FT-Scratch-Steuerung mit dem TXT – die Vereinfachungen für den LT-Controller lassen sich leicht umsetzen: Der Anschluss und die Auswertung der beiden Endlagentaster entfallen; der Taster und die Fotodiode können dann an den Eingängen I1 resp. I2 ausgewertet werden.



Abb. 18: Steuerung der Schranke mit FTScratch

Buggy

Als dritte Aufgabe wollen wir einen einfachen kleinen Buggy mit Scratch steuern.

Unser Buggy-Modell (Abb. 19) geht zurück auf den fischertechnik-Buggy der BBC aus dem Jahr 1983, der um 2003 von Economics zum PIC-Buggy weiterentwickelt wurde. Der Buggy hat auch für den Kasten *Mini Bots* aus dem Jahr 2015 Pate gestanden.

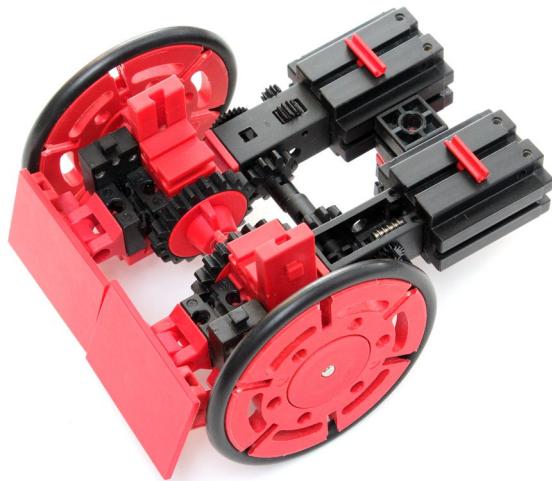


Abb. 19: Kleiner Buggy (von oben)

Der TXT Controller kann über die Federhöckern in den Nuten der beiden Mini-Motoren befestigt werden; die beiden Bausteine 15 x 30 x 5 mit Nut und Zapfen in Abb. 19 dienen der Aufnahme eines fischertechnik-Akkus. Die beiden Mini-Motoren treiben das linke bzw. rechte Rad an; vorne sind zwei Mini-Taster hinter beweglichen Bauplatten befestigt, über die Hindernisse erkannt werden (Abb. 20). Das dritte Rad ist über eine Aufnahmehaube in einem Stein 15 mit Ansenkung drehbar und über einen Winkelstein 7,5° leicht schräg gelagert.

In die „Felgen“ der Drehscheiben 60 sind Dichtungsringe („O-Ringe“) eingelegt; es passen die Größen 54 x 3 und 60 x 5. Alternativ kann der Buggy auch fischertechnik-Reifen erhalten. Um den Nachbau zu erleichtern gibt es eine [Designer-Datei des Modells](#) zum Download.



Abb. 20: Kleiner Buggy (von unten)

Mit der Steuerung wollen wir erreichen, dass der Buggy geradeaus fährt, bis er auf ein Hindernis trifft. Ist es links, setzt er ein kurzes Stück zurück, dreht sich nach rechts und fährt geradeaus weiter; ist das Hindernis rechts, macht er dasselbe in umgekehrter Richtung. Sind beide Taster gedrückt, soll er sich zufällig nach rechts oder links drehen, bevor er seine Fahrt fortsetzt. Abb. 21 zeigt ein entsprechendes FTScratch-Programm.

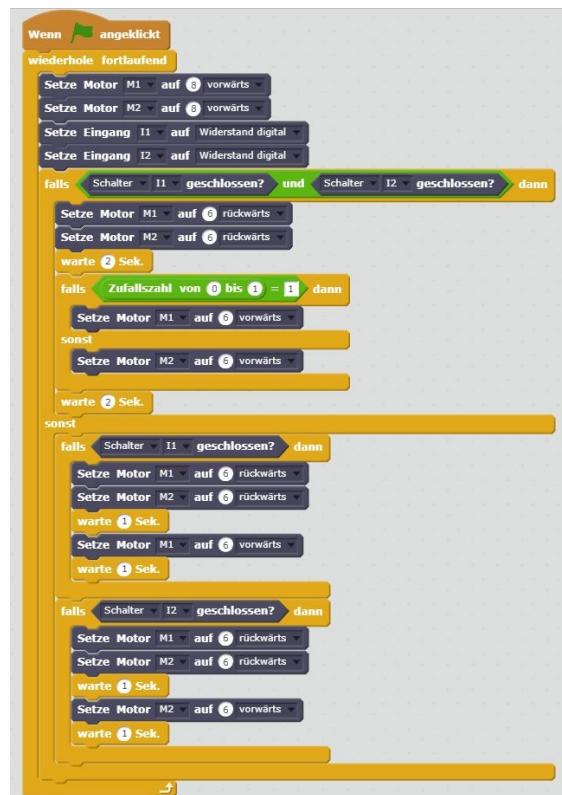


Abb. 21: Steuerung des Buggys mit Tastern

Alternativ kann man statt der Taster auch eine berührungslose Hinderniserkennung mit dem Ultraschallsensor realisieren. Auch den können wir mit der FTScratch-Extension abfragen (Abb. 22).



Abb. 22: Steuerung des Ultraschall-Buggy mit FTScratch

Fazit

Scratch ist eine ansprechende Einsteiger-Programmiersprache, die zu Recht an immer mehr Schulen Einzug hält. Damit erlernen Kinder spielerisch Programmstrukturen und logische Bedingungen. Allerdings sind die Ergebnisse rein virtuell – gesteuert werden Objekte oder Figuren auf der „Bühne“. Und gerade bei der grafischen Darstellung bleiben natürlich auch viele Programmdetails in den Skripten verborgen.

Mit FTScratch lässt sich Scratch mit der „wirklichen Welt“ verbinden und ist nicht länger Simulation, sondern die Ergebnisse sind „be-greifbar“. Für Kinder, die bereits in einer fischertechnik-AG Modelle konstruiert haben, ist das eine faszinierende Erweiterung ihrer Möglichkeiten. Damit ist FTScratch eine ernsthafte Alternative zum Einstieg mit ROBO Pro.

Kurz zusammengefasst sehe ich die folgenden Vorteile und Grenzen:

Vorteile von (FT)Scratch

- Scratch-Programme sind sehr kompakt.
- Das Zusammenfügen der Befehlsblöcke ist zumindest für Einsteiger deutlich einfacher als das Einfügen der Verbindungslien bei ROBO Pro.
- Die „Bühne“ erlaubt die Anzeige umfangreicher Kommentare und Schaltelemente und ist flexibler als die Anzeigen in ROBO Pro.
- Die Bearbeitung von Texten und deren Ausgabe ist eleganter und leistungsfähiger als in ROBO Pro.
- Zur Steuerung von Modellen können Maus und Tastatur verwendet werden.

Grenzen von (FT)Scratch

- Scratch-Programme können nur im Online-Betrieb genutzt werden.
- Variablen und Datenflüsse, Unterprogramme und Operatoren erlauben in ROBO Pro deutlich komplexere Algorithmen.
- Die Erweiterung FTScratch umfasst – zumindest beim TXT – nur einen Teil der ROBO Pro-Kommandos. So fehlen z. B. alle I²C-Befehle, die Bluetooth-Kommunikation oder die Kamera-Ansteuerung; bei komplexen Roboter-Anwendungen stößt man mit FTScratch daher schnell an Grenzen.

Die Begrenzungen von FTScratch schränken Einsteiger jedoch kaum in ihrer Kreativität und ihren Möglichkeiten ein. Wer alle Möglichkeiten des TXT nutzen will, wird auf die Dauer auch mit ROBO Pro nicht vollends glücklich werden – und muss daher ohnehin auf eine „ausgewachsene“ Programmiersprache wie C# oder Python umsteigen.

Referenzen

- [1] Thomas Buesser: [Scratch 2.0 Tutorial](#).
- [2] Alex Olinger (Übersetzung): [Scratch Referenzhandbuch](#).
- [3] [FTScratchLT](#), Installation und Blockbeschreibung, Github.io.
- [4] [FTScratchTXT](#), Installation und Blockbeschreibung, Github.io.
- [5] Stefan Falk: [Motorsteuerungen \(Teil 2\)](#). ft:pedia 2/2011, S. 19-25.

Computing

fischertechnik meets BASCOM

Christian Riebeling

Das fischertechnik-System war in den 80er und frühen 90er Jahren Pionier bei der Nutzung von Mikroprozessoren für Steuerungen. Inzwischen gibt es sehr viele preiswerte Mikroprozessor-Boards, mit denen sich fischertechnik-Modelle steuern lassen – auch aus anderen Programmierumgebungen heraus. Eine dieser Umgebungen, die sich auch für den Einstieg in die Programmierung in Schulen eignet, ist BASCOM.

Hintergrund

Dass fischertechnik hervorragend für die Erstellung eigener Funktionsmodelle geeignet ist, steht sicherlich außer Frage. Auch die Verknüpfung der geschaffenen Modelle mit elektrischen Komponenten war fast von Anfang an fester Bestandteil des Baukastensystems von Arthur Fischer. Immer der technischen Entwicklung folgend erschienen mit den „Silberlingen“ schon bald elektronische Komponenten, die sich ebenfalls wieder durchdacht kompatibel in das bereits bestehende System einfügten.

Eingebettet in einer umfangreichen Reihe von Begleitbüchern waren diese nicht nur für ambitionierte Hobbybauer eine wahre Fundgrube technischen Wissens, sondern mit den UT-Kästen, der Schulvariante der Hobbykästen, auch an den allgemeinbildenden Schulen in Deutschland Bestandteil des technischen Unterrichts, wie ich ihn erleben durfte.

Es verwundert daher nicht, dass mit dem Aufkommen der Homecomputer auch dieses neue, zukunftsweisende Medium recht schnell Bestandteil des fischertechnik-Programms wurde und hier sogar eine Vorreiterrolle einnahm. Unzählige Anwender aller Altersstufen wagten mit dieser Technik die ersten Schritte in die computergesteuerte Automation, und dies wiederum

mit einem so präzise aufeinander abgestimmten Bauteilprogramm, dass sogar Industriebetriebe Fischertechnik für die Ausbildung oder die Präsentation von Kundenprojekten nutzten. Es erschienen – der technischen Entwicklung folgend – mehrere Generationen von Controllermodulen, die den immer leistungsfähiger werdenden Mikrocontrollern gerecht wurden.

Seit einiger Zeit machen sich Mikrocontrollerboards wie beispielsweise die Arduinos in der technischen Welt breit, weil sie vielfältig einsetzbar und inzwischen enorm preiswert geworden sind. Mit dem Aufkommen erschwinglicher 3D-Drucker befassen sich mehr und mehr Maker mit dem Herstellen fischertechnik-kompatibler Gehäuse. Und auch das Aufkleben mit doppelseitigem Klebeband ist für all jene, die nicht 3D-Drucken können, eine gängige Praxis geworden – erlaubt diese Technik doch bei richtiger Auswahl des Klebebandes das sichere Befestigen der Platinen, ohne den fischertechnik-Bauteilen dauerhaft zu schaden.

So praktisch die Arduinos sind, gibt es auch Schattenseiten, denn diese Module lassen sich nicht mit der gewohnten Umgebung der fischertechnik-Controller programmieren. Für die Programmierung von Arduinos ist es notwendig, sich intensiver mit dieser Technik zu befassen. Einer der größten

Vorteile von fischertechnik, die Möglichkeit des intuitiven Umgangs mit der Technik, geht dabei etwas verloren.

Basis für die Arduinos sind 8-Bit-Mikrocontroller der Firma ATMEL. Mit diesen Controllern führe ich bereits seit einigen Jahren Computerkurse in der 7. und 8. Klasse einer öffentlichen Schule durch, was mich dazu bewog, jetzt hier einen Brückenschlag zwischen fischertechnik und ATMEL-Controllern zu wagen.

Ich verwende im Schulunterricht für die Programmierung das Programm BASCOM-AVR, welches ich seit vielen Jahren auch für private und berufliche Zwecke mit großem Erfolg einsetze. BASCOM leitet sich ab aus einer Zusammensetzung von BASIC und Compiler. BASIC ist dabei eine traditionelle, leicht zu erlernende Programmiersprache, die es ermöglicht, den Controller angelehnt an die englische Sprache zu programmieren. Compiler nennt man ein

Programm, das ein geschriebenes Programm in den Code umwandelt, den der Mikrocontroller „versteht“.

BASIC steht bei einigen Programmierern in einem weniger guten Licht, da es eng mit dem Aufkommen der ersten Homecomputer in Verbindung steht, wobei die heutigen BASIC-Interpreter mit denen der damaligen Zeit vom Codeaufbau her nicht mehr viel gemeinsam haben, außer vielleicht die enorm einfache Erlernbarkeit der Programmiersprache, die – wie erwähnt – eng an die englische Sprache angelehnt ist.

Bei den Schülern habe ich die Erfahrung gemacht, dass es mit BASCOM eine steile Lernkurve gibt, die schnell zu Erfolgen und der Umsetzung eigener Ideen führt, was wiederum motivierend wirkt, weil es schnell vorwärts geht. Der Fokus liegt von Anfang an auf der Programmierung und nicht auf dem Erlernen undurchsichtiger Programmierroutinen.

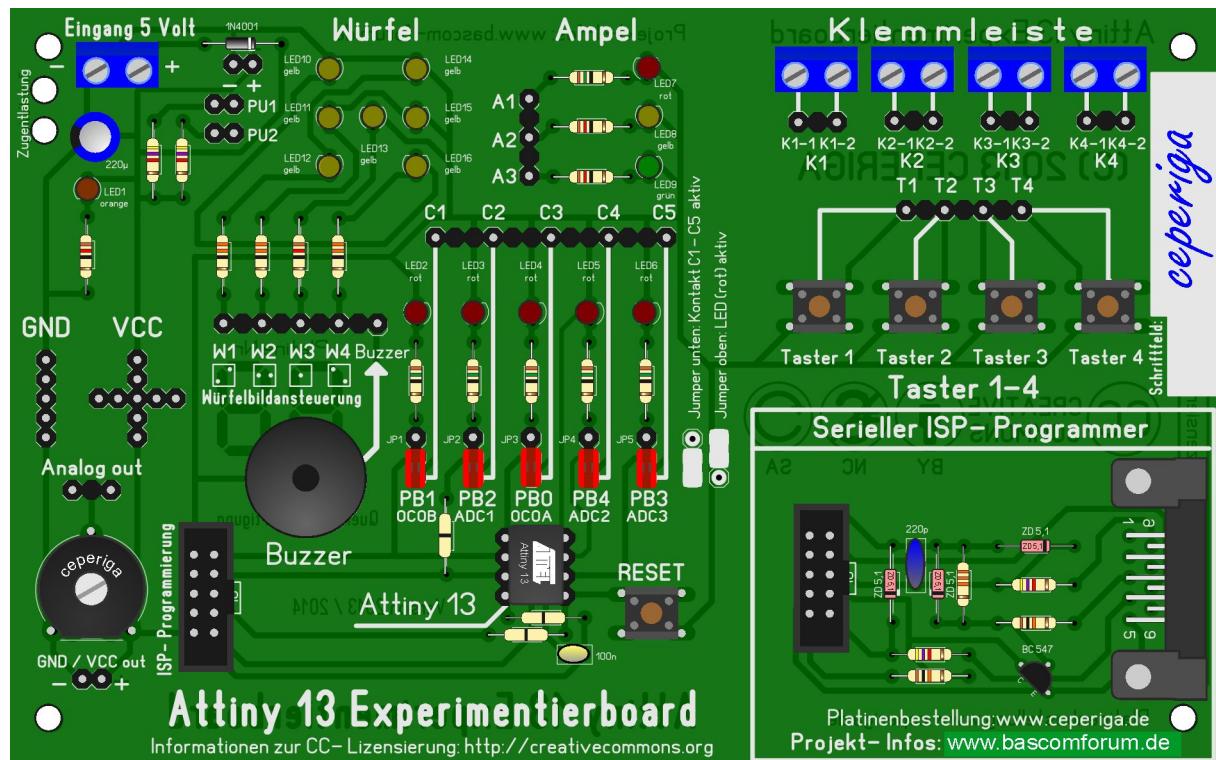


Abb. 1: Experimentierplatine

Auch bei mir gab es (allerdings im Bereich der Unterrichtsvorbereitung an der Schule) eine Lernkurve, welche dazu führte, dass ich zu dem Basiskurs eine eigene Experimentierplatine entworfen habe, auf deren Basis ich einen BASCOM-Anfängerkurs geschrieben habe, der keinerlei Vorkenntnisse erfordert. Zweck dieser Dokumentation war zunächst die Vorbereitung der Unterrichtsstunden, schnell wurde daraus aber ein Begleitwerk, welches den Schülern ein nachträgliches Aufarbeiten der im Unterricht behandelten Themen ermöglichte.

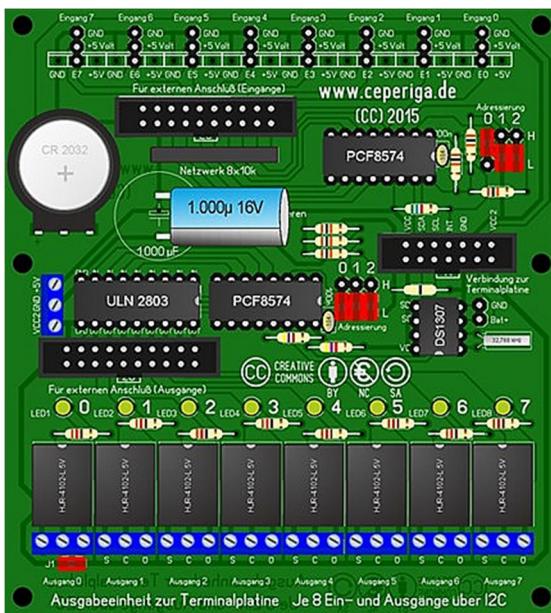


Abb. 2: I²C-Ein-Ausgabeplatine

Während ich anfangs noch mit Overheadfolien im Unterricht gearbeitet habe, konnte ich nach Installation von Multimediatafeln in der Schule im Unterricht auf das Internet zurückgreifen. Ich nutze dieses „überdimensionale Tablet“ für die Präsentation der Lerninhalte und übertrug den BASCOM-Anfängerkurs auf die Dokumentationsseiten (in der Forum-Software als „Lexikon“ bezeichnet) des offiziellen deutschen BASCOM-Forums, wo er mir seither im Unterricht zur Verfügung steht.

Durch die Veröffentlichung im Forum (und parallel dazu auf meiner Homepage) stand er nunmehr auch allen Interessierten außerhalb meines Schullehrgangs zur Verfügung

und erfreut sich regen Interesses, was ich an den Nachfragen zum Kurs und zu den Platinen erkennen kann.

In den bereits veröffentlichten Lektionen wird das Basiswissen für den Umgang mit den ATMEL-Controllern und somit auch für die Arduinos vermittelt. Der Kurs wird bedarfsorientiert fortgesetzt und widmet sich zukünftig auch einer zweiten bereits entwickelten Platine, die nicht mehr nur das reine Experimentieren zum Ziel hat, sondern auch den praktischen Einsatz (Abb. 3). Dafür verfügt die Platine über ein Text-LCD und eine Tastatur. Über den I²C-Port lässt sich im Sandwich eine (oder bis zu acht) eigens entwickelte Zusatzplatine anbauen, welche über acht Eingänge und acht Ausgänge verfügt. Es ist somit ein Leichtes, fischertechnik-Modelle damit zu steuern und über Display und Tastatur zu bedienen.



Abb. 3: Platine mit LCD und Tastatur für den praktischen Einsatz

Ebenso wie diese Platinen lassen sich aber auch die Arduinos mit BASCOM programmieren. Während man für die von mir entwickelten (und viele andere käuflich zu erwerbenden) Platinen einen Programmer für die ISP-Programmierung (*In-System-Programming*, das heißt, der Controller kann beim Beschreiben (*Flashen*) in der Schaltung verbleiben) benötigt, nutzen die Arduinos alternativ einen sogenannten *Bootloader*, der es ermöglicht, die Platinen einfach mittels USB-Kabel zu programmieren. BASCOM unterstützt dabei die Programmierung mittels Bootloader direkt aus dem Programm heraus.

Was macht BASCOM so anders?

Der eindeutige Vorteil von BASCOM ist, dass viele externe Komponenten, die in anderen Programmiersprachen mit vielen zu erlernenden Einzelbefehlen erst zusammengestellt werden müssen, mit wenigen Sonderbefehlen aufgerufen werden. Die Arbeit im Hintergrund wird von BASCOM beim Compilieren eigenständig erledigt. So kann man sich bei der Programmierarbeit auf die eigentliche Lösung der Probleme konzentrieren, ohne tiefgreifend in die Controllerstruktur blicken zu müssen. Dies ist sozusagen das „fischertechnik-Prinzip“ für die Controllerprogrammierung.

Ein Beispiel...

Ihr möchtet in Eurem Modell einen Text auf dem LCD anzeigen? Das ist normalerweise mit einer ganzen Reihe von Befehlszeilen verbunden. BASCOM hingegen möchte von Euch nur wissen an welcher Stelle der Text und natürlich was ausgegeben werden soll. Selbstverständlich wird im Programm zuvor mit einem Deklarationsbefehl beschrieben, welches Display verwendet wird und wie dieses an den Controller angeschlossen ist.

So sehen die beiden Programmzeilen für das LCD in BASCOM aus:

```
Locate 1,1
Lcd „Position:“ ; x
```

Dabei ist `Locate 1,1` die Anweisung, in die erste Zeile an Position 1 beginnend die nachfolgenden Informationen (Text, Variableninhalte, Sonderzeichen oder bis zu acht selbst definierbare Zeichen) auszugeben, ist somit also nichts weiter als der Positionierungszeiger, welcher auf jede Anzeigestelle eines Textdisplays gesetzt werden kann.

`Lcd „Position:“ ; x` bedeutet: Schreibe den Text „Position“ auf das LCD und anschließend den Wert der Variablen x (zum Beispiel 546). So sieht dann die Ausgabe auf dem LCD aus:

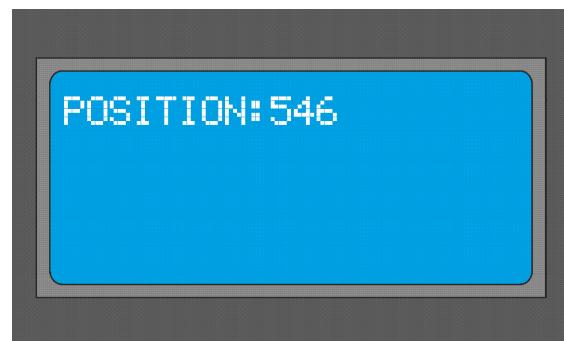


Abb. 4. Textausgabe auf LCD-Display

Genauso einfach lassen sich in BASCOM aber auch Servos, 1wire-Sensoren, I²C-Komponenten, Grafikdisplays, Infrarotfernsteuerungen oder auch die in [ftpedia 4/2017](#) beschriebenen Mehrfarb-LEDs WS 2812B ansteuern. Sie sind zugleich ein gutes Beispiel für die stetige Weiterentwicklung des Programmes. Dieses recht junge Bauteil wurde über eine sogenannte „LIB“ ergänzt, über die BASCOM-AVR einfache Befehle zur unkomplizierten Ansteuerung der LEDs zur Verfügung stellt.

Auch die „normale“ Ansteuerung von Ausgängen oder die Abfrage von Tastern und Sensoren gestalten sich sehr einfach und sind mit ein wenig Übung bereits am geschriebenen Code nachvollziehbar. Hier als weiteres Beispiel der Code für eine blinkende LED, für viele Programmieransteiger der erste Schritt. In grün die Kommentare, wie sie nach Eingabe durch den User in BASCOM dargestellt werden:

Do	\` Schleifenanfang
Toggle LED	\` LED wechselseitig ein- und ausschalten
Wait 1	\` 1 Sekunde warten
Loop	\` Schleifenende, zurück zu Do

LED ist dabei ein Alias, das bedeutet, wir haben der LED zuvor einen Ausgang des Controllers zugewiesen (`Led alias PortB.0`). Beim Programmieren müssen wir uns nun aber nicht mehr merken, dass die LED an PortB.0 angeschlossen ist, sondern müssen nur LED schreiben, wenn wir dieser einen Befehl widmen. Das ist bei wenigen Aus- und Eingängen sicherlich nicht zwingend erforderlich, aber bei umfangreichen Programmen eine große Hilfe, weil man Sensoren und Aktoren immer beim Namen nennen kann. Versieht man das Programm dann noch mit entsprechenden Kommentaren, die sich überall im Programm platzieren lassen, ist die Programmfunction für andere Nutzer und auch für einen selber (auch noch nach mehreren Wochen) leicht nachvollziehbar.

`Toggle` ist der Befehl für eine binäre Negation. Heißt hier ganz einfach: wenn die LED ausgeschaltet ist, schalte sie ein und schalte sie aus, wenn sie eingeschaltet ist.

`Wait 1` weist das Programm an, eine Sekunde zu warten. Dies wird man als Fortgeschrittenen in komplexeren Programmen sicher mit einem Timer lösen, aber bei einfachen Programmstrukturen ist dieser Befehl durchaus legitim.

`Do` und `Loop` sind Anfang und Ende einer Schleife und umschließen fast immer ein sich ständig wiederholendes Hauptprogramm.

Wer neugierig geworden ist, kann einmal einen Blick in den Mikrocontrollerkurs werfen. Dort gibt es ausreichend „Input“, was hier sicherlich zu weit führen würde. Dazu muss man noch nicht einmal über Hardware verfügen, denn die entwickelten Programme sind (soweit möglich) mit

animierten Grafiken zur Verdeutlichung der Programmfunction ergänzt.

Ein weiterer Vorteil insbesondere für junge Nutzer und Schulen ist, dass die [Demo-version von BASCOM-AVR](#) ohne Einschränkungen der Befehle wie die Vollversion verwendet werden kann. Die einzige Beschränkung liegt in der Programmgröße bei 4k. Das reicht aber für Controller wie den oft verwendeten Attiny 2313 oder den ATMEGA 48 aus, um diese vollständig zu nutzen. Auch alle anderen Controller wie der ATMEGA 328 auf dem Arduino lassen sich mit Code bis zu 4k verwenden. Wem das nicht mehr reicht, der muss die [Vollversion erwerben](#) (ca. 75 €). Zukünftige Updates stehen Besitzern einer Vollversionslizenz übrigens kostenlos zur Verfügung.

fischertechnik meets BASCOM

fischertechnik und BASCOM sind nach meiner Erfahrung eine gelungene Kombination für Maker, die Mikrocontroller nutzen möchten ohne sich langwirig mit Controllerstrukturen und Befehlsketten auseinanderzusetzen zu müssen. Und bei Problemen stehen Euch Internetforen zur Verfügung, in denen erfahrene User Euch bei der Lösung vom Problemen behilflich sind, wie beispielsweise das [BASCOM-Forum](#), in dessen Lexikon ich auch den Mikrocontrollerkurs veröffentlicht habe.

Ich wünsche Euch viel Spaß beim Ausprobieren. Individuelle Fragen zu BASCOM, dem Mikrocontrollerkurs oder erhältlichen Platinen beantworte ich Euch gerne über kontakt@ceperiga.de.

Auch Vorschläge zu wünschenswerten Platinen zur Verknüpfung von fischertechnik mit den 8-Bit-Controllern von ATTEL oder Zusatzplatinen (beispielsweise über I²C) nehme ich gerne entgegen. Vielleicht ergibt sich daraus ja ein neues Projekt als Ergänzung zum Mikrocontrollerkurs.

Referenzen

- [1] Christian Riebeling: [Der Mikrocontrollerkurs](#).
- [2] Christian Riebeling: [Die Terminalplatine zum Mikrocontroller-Kurs](#).
- [3] Christian Riebeling: [Die PC-Ein-Ausgabeplatine zum Mikrocontroller-Kurs](#).
- [4] Stefan Hoffmann: [Einfacher Einstieg in die Elektronik mit AVR-Mikrocontroller und BASCOM](#), 2010.

Computing

ftDuino – Open-Source trifft Konstruktions-Baukasten

Till Harbaum

Ein fischertechnik-Controller, der das Universum der erfolgreichen Arduino-Plattform mit der mechanischen und elektrischen Welt von fischertechnik verknüpft – genau das hat bisher gefehlt. Ein Community-Projekt hat dies nun Wirklichkeit werden lassen...

Teil 1: Der ftDuino im Schnellüberblick

Der ftDuino ist ein brandneuer Robotics Controller für fischertechnik. Er ist ein Verwandter des TXT und des BT Smart Robotics Controller. Das Besondere: Er ist kein Produkt von fischertechnik, sondern wurde von der Community für die Community entwickelt. Seit Mitte Februar 2018 gibt es die ersten [ftDuinos](#) zu kaufen. Aktuelle Informationen zur Verfügbarkeit gibt es unter info@ftduino.de.

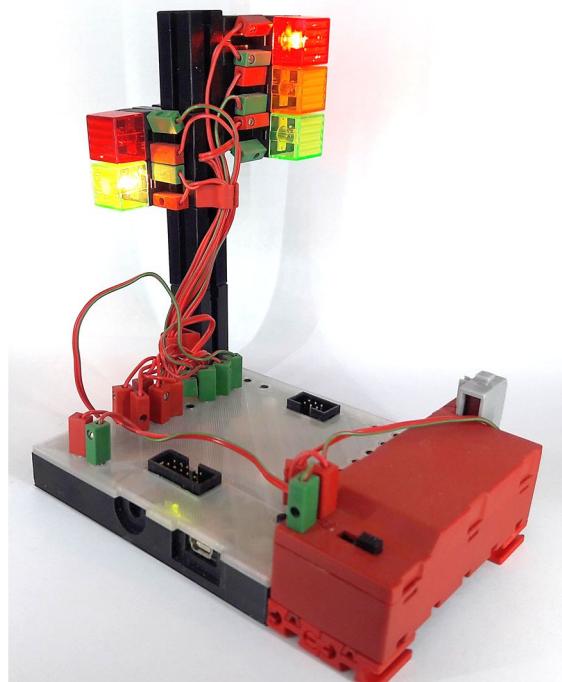


Abb. 1: Der ftDuino in einem Ampel-Modell

Der ftDuino gehört zur Familie der so genannten [Arduino-Experimentier-Boards](#), wurde aber speziell für den Einsatz in fischertechnik-Modellen entworfen. Er bietet die gleichen Anschlüsse wie ein TXT und kann diesen bei einem Preis von ca. 50 Euro in den meisten Modellen direkt preisgünstig ersetzen.

Die Programmierung des ftDuino erfolgt textbasiert in der Arduino-IDE (Abb. 2) in den Programmiersprachen C und C++. Er ermöglicht damit einen einfachen Einblick in die professionelle Programmierung, wie sie auch in der Industrie zum Einsatz kommt.

```
#include <FtduinoSimple.h>

void setup() {}

void loop()
{
    ftduino.output_set(Ftduino::O1, Ftduino::HI);
    delay(1000);
    ftduino.output_set(Ftduino::O1, Ftduino::LO);
    delay(1000);
}
```

Abb. 2: Arduino-IDE

Wie alle Arduinos richtet sich der ftDuino an Personen, die hinter die Kulissen schauen wollen. Er wendet sich an Schüler

weiterführender Schulen, Berufsschüler, Studenten und alle anderen, die einen Einblick in die die professionelle Entwicklung sogenannter eingebetteter Geräte erhalten wollen.

Der ftDuino ist keine Black-Box, sondern vollständig dokumentiert, sowohl was die Hardware als auch die Software angeht. Wer verstehen will, wie das Herz seines Roboters tatsächlich funktioniert, erhält mit dem ftDuino die Möglichkeit, sämtliche Aspekte zu entdecken und zu begreifen. Alle Software liegt entsprechend der Open-Source-Philosophie vollständig im so genannten Quellcode vor und lädt dazu ein, entdeckt, modifiziert und angepasst zu werden.

Wie wertet der Controller Eingangsspannungen aus? Wie lässt er einen Motor mit einer bestimmten Geschwindigkeit drehen? Diese Fragen und viele mehr beantwortet die ausführliche Bedienungsanleitung. Keine Geduld, auf ein neues Software-Feature zu warten? Hilf Dir selbst, dank Open-Source! Open-Source bedeutet unter anderem „Jeder kann mitmachen“. Die Grenze zwischen Hersteller und Anwender verwischt und jeder kann auf Seiten wie [Github](#) zum Projekt beitragen.

Der ftDuino kostet trotz der viel geringeren Stückzahl nur ein Bruchteil des TXT. Trotzdem kann er in fast allen TXT-basierten Modellen als vollwertiger Ersatz eingesetzt werden.

fischertechnik hat die Zeichen der Zeit erkannt und setzt neuerdings ebenfalls mit dem Calliope auf einen vollständig dokumentierten Open-Source-Controller. Aber während sich fischertechnik an den Calliope anpassen muss und spezielle Sensoren und Motoren zum Einsatz kommen, geht der ftDuino einen Schritt weiter und passt sich an fischertechnik an, indem er die üblichen fischertechnik-Aktoren und Sensoren verwendet, vom XS-Motor bis zum

Kompressor, vom Fototransistor bis zum Ultraschallsensor.

Informationen zur Verfügbarkeit gibt es unter info@ftduino.de, alles weitere im [Web](#) oder im Computing-Bereich des [fischertechnik-Forums](#).

Happy Hacking mit dem ftDuino!

Teil 2: Der ftDuino im Detail

Mit dem TXT und dem BT Smart Controller hat fischertechnik zwei recht unterschiedliche Robotics Controller auf den Markt gebracht. Nimmt man die gelegentlich noch erhältlichen TX und Robo LT Controller hinzu, so hat man die Wahl zwischen vier ganz verschiedenen Geräten zu unterschiedlichen Preisen und mit unterschiedlicher Ausstattung.

Diese fischertechnik-eigenen Controller folgen wie z. B. auch die Lego-Controller seit über 30 Jahren klassischen Entwicklungsmodellen und bilden jeweils in sich geschlossene Systeme. Der Anwender kauft den Controller im Paket mit einer passenden Programmiersoftware für den heimischen PC. Das Ganze ist dabei in erster Linie auf einfache Benutzbarkeit ausgelegt, und das komplexe Innenleben der Controller bleibt soweit möglich verborgen. Während bei den Robotermustern anschauliche und begreifbare Mechanik im Vordergrund stehen, bleibt der informationstechnische Aspekt dahinter weitgehend im Dunklen. Aus Benutzersicht sind diese Geräte daher „Black-Boxes“, deren Arbeitsweise nur oberflächlich erkennbar ist.

Parallel hat sich seit der Jahrtausendwende mit Geräten wie dem [Raspberry-Pi](#) und dem [Arduino](#) die Gerätefamilie der „Open Source Controller“ etabliert, die die Veranschaulichung von Informationstechnologie in den Vordergrund stellt. Diese billigen und handlichen Geräte laden dazu ein, ihr Innenleben zu erkunden und zu verstehen. Der Aufbau von interessanten Robotermustern scheitert hier eher an der Mechanik,

die man sich im Zweifelsfall selbst bauen muss.

Diese beiden so unterschiedlichen Welten zu kombinieren, jeweils das Beste zu nehmen und zu vereinen ist keine neue Idee. Viele Projekte im Internet kombinieren die Mechanik der fischertechnik- und Lego-Baukästen mit dem Raspberry-Pi, dem Arduino oder ähnlichen Projekten. Die mechanische und elektrische Anpassung der Open-Source-Controller an die Sensoren und Aktoren des jeweiligen Baukastensystems ist aber selbst keine triviale Aufgabe, so dass relativ viel Know-How benötigt wird, bevor ein einfache zu verwendendes System entsteht.

Genau diese Lücke zu schließen ist das Ziel des ftDuino.

Die Hardware



Abb. 3: Der ftDuino
im transparenten Gehäuse

Der ftDuino ist ein vollständiges, betriebsbereites Gerät und beinhaltet bereits die komplette mechanische und elektrische Anpassung an das fischertechnik-System sowie die softwareseitige Integration in das Arduino-Ökosystem. Ein passendes Gehäuse schützt die Elektronik und macht sie spielzeugtauglich.

Damit ist der ftDuino genauso einfach in einem fischertechnik-Modell einsetzbar wie jeder andere fischertechnik-Controller, bietet aber gleichzeitig die Offenheit und Flexibilität der Arduino-Controller. Und

endlich wird so nicht nur der mechanische Aspekt sondern auch der informations-technische Aspekt eines Modells im Wort-sinne begreifbar.

Vom Konzept ...

Der ftDuino wurde anders als die meisten Arduino-basierten Bastelprojekte von vornherein mit dem Ziel einer Serienproduktion entworfen. Dieser Anspruch hatte massive Auswirkungen auf das Konzept des ftDuino und bedeutete u. a.:

- die Hardware muss so einfach wie möglich sein, um auch bei geringer Stückzahl einen vertretbaren Preis zu ermöglichen
- alle verwendeten Komponenten müssen zuverlässig erhältlich sein
- das Gerät muss mechanisch und elektrisch robust sein
- das Gerät muss zu aktuellen fischertechnik-Baukästen kompatibel sein
- die Inbetriebnahme muss auch für Laien möglich sein

... über die ersten Prototypen ...

Die Einfachheit der Hardware lag beim allerersten Prototyp Ende 2016 am stärksten im Fokus. Der ftDuino 1.0 basierte auf der Technik des einfachsten Arduino, dem Arduino-Nano, und kombinierte diesen mit den preisgünstigsten für fischertechnik-Motoren geeigneten sogenannten H-Brücken als Motor-Treibern.

Er verzichtete bewusst auf den Einsatz eines echten Arduino, war also kein so genannter „Shield“, sondern selbst ein kompletter Arduino und kam daher mit sehr wenigen Bauteilen aus. Sämtliche Ein- und Ausgänge waren bereits Kurzschluss- und Überspannungsfest, sodass Verdrahtungsfehler zu keinen Beschädigungen führten. Die gesamte Elektronik basierte auf robusten industriell hergestellten Platinen und es gab von vornherein ein stabiles 3D-gedrucktes Gehäuse. Der ftDuino wurde

mechanisch und elektrisch so nah wie möglich an den TXT-Controller angelehnt, um diesen in sämtlichen fischertechnik-Modellen direkt ersetzen zu können.

Dadurch, dass der erste ftDuino auf dem Arduino-Nano basierte, ließ er sich wie dieser sofort aus der Arduino-IDE direkt ansprechen und konnte zum Aufbau erster einfacher Modelle verwendet werden.

... zum Seriengerät

So hat der erste ftDuino bereits die wichtigste Aufgabe eines Spielzeugcontrollers im Hause Harbaum erfüllt: Als Teil von „Idas Ampel“ hat er Schleichtiere sicher durch den Playmobil-Straßenverkehr gebracht. Die prinzipielle Machbarkeit und Alltagstauglichkeit des Konzepts war damit bewiesen. Ein paar Dinge erwiesen sich aber recht schnell als verbesserungswürdig.

Als Benutzer der fischertechnik-Controller ist man es zum Beispiel gewohnt, dass man an einen Motorausgang alternativ auch zwei Lampen oder Ähnliches anschließen kann, da die beiden einen Motorausgang bildenden Anschlüsse hardwareseitig komplett unabhängig sind. Das ist bei üblichen Motortreiber-Chips, wie sie auch der erste ftDuino verwendete, leider nicht der Fall, und die beiden Einzelausgänge ließen sich eben doch nicht vollständig unabhängig steuern.

Das führte bereits bei „Idas Ampel“ dazu, dass die Zuordnung der Lampen zu den Ausgängen des ftDinos nicht frei wählbar war. Diese Beschränkungen wären einem unerfahrenen Benutzer nur schwer zu erklären gewesen.

Ein zweiter Schwachpunkt waren die fehlenden Erweiterungsmöglichkeiten. Die vielleicht größte Stärke der Arduino-Plattform ist, dass man fast beliebige Hardware anschließen kann, allem voran billige Sensoren aller Art. Dadurch, dass der ftDuino alle Ein- und Ausgänge mit Schutzschaltungen versieht bzw. die Ausgänge mit

fischertechnik-tauglichen Motorleistungsstufen versehen sind, verlieren die Anschlüsse leider ihre Universalität. Man kann nicht mehr wie bei einem echten Arduino einen Eingang mal eben zum Ausgang umdefinieren. An dieser Tatsache lässt sich bei einem für den fischertechnik-Einsatz entworfenen Gerät leider wenig ändern.

Allerdings gibt es einen in der Arduino-Welt für den Einsatz von Sensoren sehr verbreiteten Anschluss, den sogenannten I²C-Bus. Da auch der TXT-Controller und der TX-Controller über einen I²C-Anschluss verfügen, lag es nahe, auch dem ftDuino einen solchen I²C-Anschluss zu verpassen. Es wurde dabei die 5V-Variante des TX-Controllers übernommen und der I²C-Anschluss auch mechanisch an diesen angelehnt, so dass für den TX entworfene Geräte auch direkt an den ftDuino anschließbar sind und umgekehrt. Der I²C-Anschluss des TXT wird im Gegensatz zu dem des ftDuino und dem des TX Controllers nicht mit 5 Volt, sondern mit 3,3 Volt betrieben und ließ sich daher nicht ohne zusätzlichen Aufwand direkt am ftDuino nachbilden. Entsprechende [Adapter](#) schlagen allerdings auch diese Brücke und verbinden den ftDuino mit dem TXT sowie mit für diesen angebotenen I²C-Sensoren.

Die dritte große Änderung betrifft die Arduino-Technik, auf der der ftDuino basiert. Der erste Prototyp basierte auf dem Arduino-Nano, dem kleinsten und günstigsten Mitglied der Arduino-Familie. Dieser Arduino ist allerdings relativ beschränkt, was seine USB-Fähigkeiten angeht, und wird lediglich als sogenannter COM-Port am PC erkannt. Der etwas neuere Arduino-Leonardo ist hier sehr viel flexibler und lässt USB-seitig sehr viel mehr Raum für Kreativität. Er kann sich gegenüber einem PC (oder Smartphone) u. a. als USB-Tastatur, USB-Joystick oder gar als Audio-Gerät ausgeben. Für den Serien-ftDuino wurde daher auf die gleichen Komponenten gewechselt, auf denen auch der Arduino-

Leonardo basiert, wodurch sich auf Basis des ftDuino z. B. auch [Joysticks und Gamepads](#) bauen lassen. Beibehalten wurde der bewährte Formfaktor des ersten ftDuino, so dass sich der ftDuino weiterhin direkt als TXT-Ersatz eignet.

Die Software

Die Programmierung des ftDuino erfolgt in der Regel textbasiert in der offiziellen Arduino-IDE auf einem Windows-, Linux- oder MacOS-PC. Die Arduino-IDE ist

schnell installiert und bietet eine schlichte, aber praktische Oberfläche rund um den eigentlichen Programmeditor.

Ab Werk bringt die Arduino-IDE alle nötigen Zusatz-Werkzeuge inklusive Code-Bibliotheken und -Beispiele für die Original-Ardudos mit. Zusätzlich lässt sich die Arduino-IDE mit wenigen Klicks für weitere Arduino-basierte Geräte wie den ftDuino erweitern.

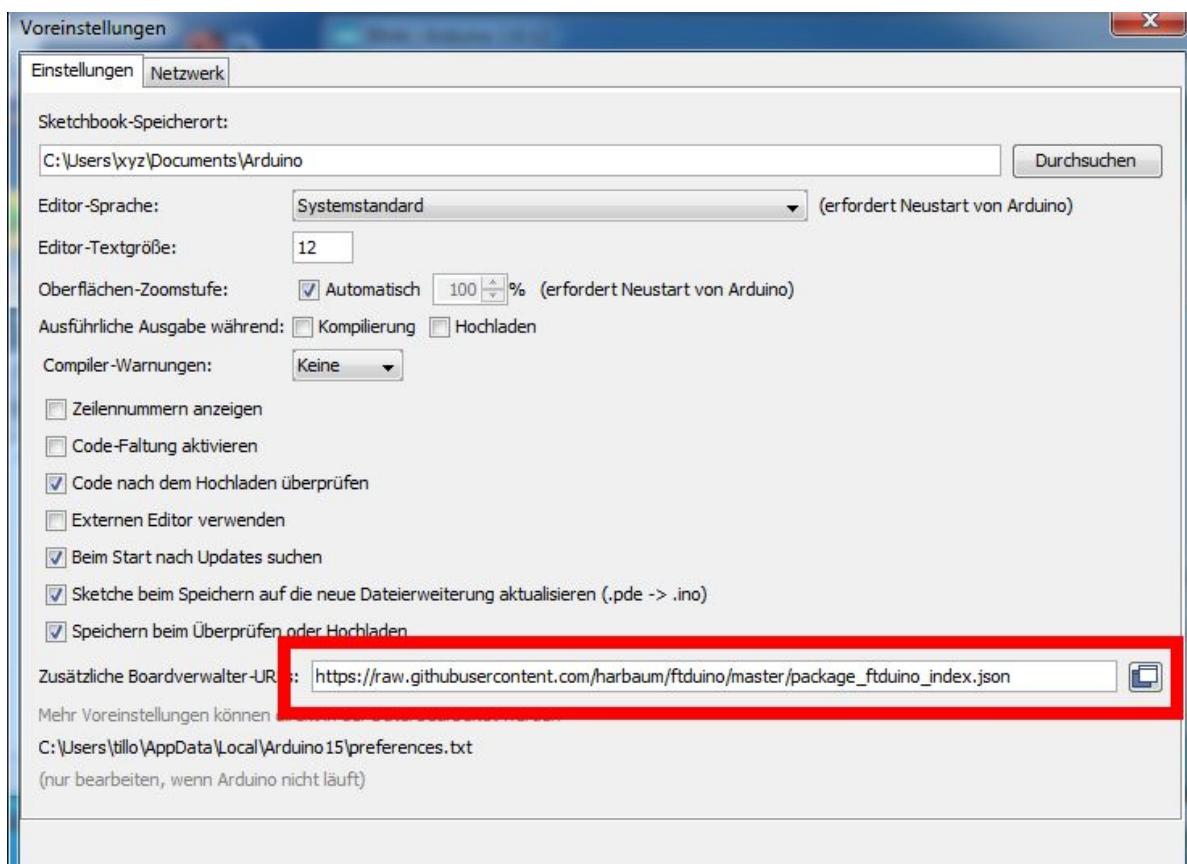


Abb. 4: Die ftDuino-Konfiguration in den Voreinstellungen der Arduino-IDE

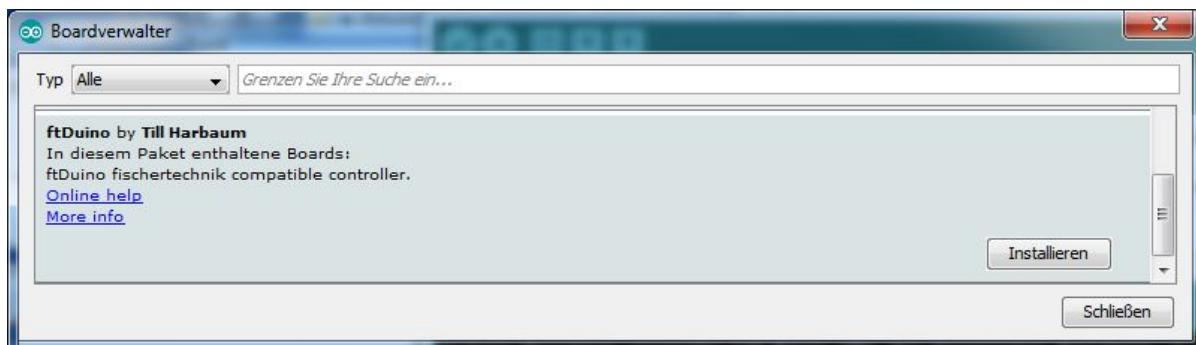


Abb. 5: Der ftDuino im Boardverwalter der Arduino-IDE

Die [Arduino-IDE](#) kann kostenlos heruntergeladen werden und ist mit wenigen Klicks installiert. Alles was es zum ftDuino an Dokumentation, Software und sonstigen Informationen gibt, findet sich [auf Github](#).

Vor allem findet sich dort die [ftDuino-Konfiguration](#) für die Arduino-IDE. Die Konfigurationsdatei wird einfach in den Voreinstellungen der Arduino-IDE einge tragen wie in Abb. 4 dargestellt. Der ftDuino taucht daraufhin im sogenannten Boardverwalter der Arduino-IDE auf, wie in Abb. 5 zu sehen.

Im Boardverwalter lässt sich die eigentliche ftDuino-Installation mit einem Klick starten. Die Arduino-IDE installiert automatisch die neueste Version aller nötigen Dateien inklusive diverser Beispiele und hält den Benutzer in Zukunft über Aktualisierungen auf dem Laufenden. Nach der ftDuino-Installation kann man unter anderem (wie in Abb. 6 dargestellt) direkt im Datei-Menü die Beispiele auswählen, die auch in der ausführlichen Anleitung im Detail beschrieben sind.

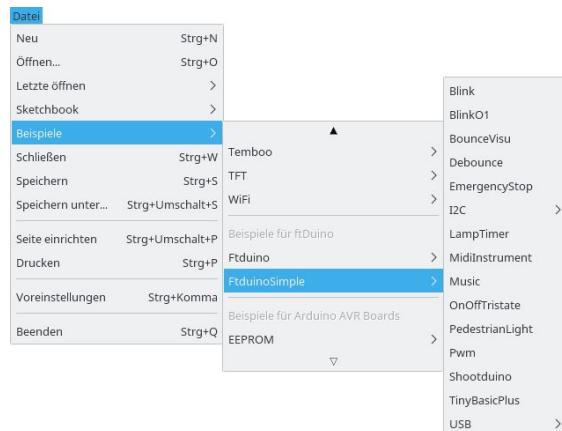


Abb. 6: Die ftDuino-Installation bringt viele Beispiele gleich mit

Die [komplette Anleitung](#) findet sich im Github. Sie wird wie die ganze ftDuino-Installation regelmäßig aktualisiert.

Integration in das fischertechnik-System

Der ftDuino wurde auf maximale Kompatibilität zu den aktuellen fischertechnik-Robotics-Kästen ausgelegt. Seine mechanische Größe entspricht den TX und TXT Controllern, und er ist auch elektrisch zu beiden Geräten weitgehend kompatibel. Unterstützt werden unter anderem die folgenden Aktoren und Sensoren:

- sämtliche Motoren von XS bis XM
- alle Encoder-Motoren aus den TXT- und TX-Baukästen
- sämtliche Lampen und Leuchtdioden
- sämtliche Schalter und Taster
- Farbsensor, Magnetventile und Kompressor aus „Electro Pneumatic“
- NTC-Temperatursensor, Potis, Fototransistoren und Fotowiderstände
- Ultraschall-Abstandssensor und IR-Spursensor aus „Robo TX Explorer“ (Abb. 7)
- Kombisensor (Orientierungssensor) aus „Robotics Competition Set“ (Adapter nötig)

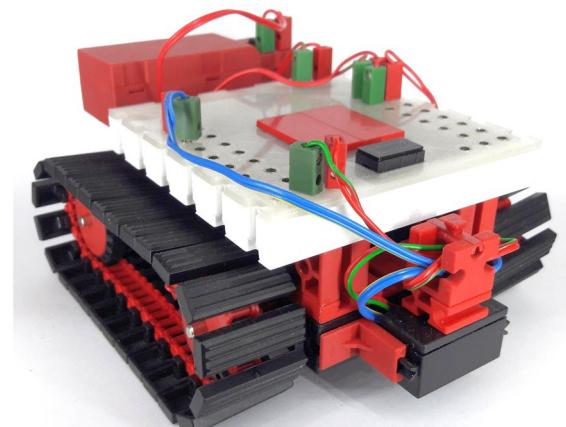


Abb. 7: ftDuino im Linienfolge-Roboter

Und falls mit den fischertechnik-Aktoren und -Sensoren wirklich die Grenzen erreicht werden, dann bietet der ftDuino mit seinem I²C-Anschluss eine flexible und

einfach zu benutzende Erweiterungsschnittstelle, an die sich kleine Bildschirme (siehe Abb. 8) genauso anschließen lassen wie z. B. Servomotor-Treiber oder unzählige Sensoren vom einfach Temperaturfühler bis zum Umwelt- oder Laser-Distanz-Sensor.

Die Arduino-Plattform spielt hier gegenüber z. B. ROBO Pro ihre besondere Stärke aus, denn mit ein wenig Suche im Internet findet man zu praktisch jedem I²C-Sensor oder -Aktor den passenden Treiber und Programmbeispiele.



Abb. 8: Der ftDuino mit I²C-OLED-Display

Kontaktfreudig

Der ftDuino ist zwar in sich bereits ein komplettes Produkt und lässt sich direkt zur Steuerung diverser Modelle einsetzen. Aber hier endet es nicht. Wie es sich für ein Open-Source-Projekt gehört ist der ftDuino sehr kommunikationsfreudig.

Via USB lässt sich der ftDuino nicht nur mit dem PC verbinden. Auch am Raspberry-Pi, dem fischertechnik-TXT, dem Lego-EV3 oder einem Smartphone lässt er sich betreiben und diesen Geräten den Zugriff auf seine fischertechnik-Anschlüsse ermöglichen. Und via I²C-Bus findet der ftDuino Anschluss an den TXT, den TX oder gar an seinesgleichen. Bis zu 126 ftDinos lassen sich so koppeln.

Auch auf der Softwareseite setzt sich die Kontaktfreudigkeit fort. So kann sich der

ftDuino bei entsprechender Programmierung auch als USB-Joystick oder USB-Audiogerät ausgeben und sich damit auch an Geräten betreiben, die sehr wählerisch beim Umgang mit USB-Peripherie sind. Als USB-Gamepad wird der ftDuino sogar von einigen Spielkonsolen als Eingabegerät akzeptiert und auch Smartphones oder Tablets können oft überraschend viel mit solchen ftDuino-basierten Eingabegeräten anfangen.



Abb. 9: Computing-Power satt: Der ftDuino am Raspberry-Pi inkl. Display

Ausblick

Der ftDuino soll erst der Anfang sein. Er ist der Versuch, die Maker-Idee in die fischertechnik-Welt zu tragen und soll beweisen, dass professionelles fischertechnik-Zubehör nicht unbedingt vom Originalhersteller kommen muss. So wie sich die Arduino-Welt weiter entwickelt öffnen sich auch für Projekte wie den ftDuino neue Möglichkeiten und ich bin gespannt, wo die Reise hingehet.

Referenzen

- [1] Till Harbaum: [ftDuino-Seite](#).
- [2] Till Harbaum: [ftDuino-Handbuch](#).
Github.com
- [3] [Arduino-Präsenz](#) im Internet
- [4] [fischertechnik Community Forum](#)

Computing

Hacker im Kinderzimmer: Remote-Angriff auf den TXT

Till Harbaum

Wie sicher ist eigentlich der TXT? Eine Frage, der mit der WLAN-Fähigkeit und der Nutzung des Controllers als Smart Home Device neue Bedeutung zukommt.

Hintergrund

Der fischertechnik-TXT-Controller war zu seiner Markteinführung 2014 unter den Konstruktionsspielzeugcontrollern unangefochtener Spitzenreiter. Auch wenn die Hardware inzwischen etwas betagt ist, so erlaubt sie nach wie vor kleine und große Softwareerweiterungen. Eine dieser Erweiterungen war Mitte 2016 die Aufrüstung der WLAN-Fähigkeiten des TXT um die Möglichkeit, sich in das private Heimnetzwerk einzuhüpfen. Bis dahin stellte der TXT immer ein eigenes WLAN zur Verfügung, in das man sich mit dem Computer einwählen und so insbesondere die Verbindung zum Internet kappen musste, solange man mit dem TXT verbunden war.

Was auf den ersten Blick als lange fällige Erweiterung erscheint, hat leider gleichzeitig gravierende negative Konsequenzen. Um diese geht es in diesem Beitrag.

Das ursprüngliche Konzept, dem TXT nur zu erlauben, sein eigenes lokal begrenztes WLAN zu betreiben, brachte vor allem eines: Sicherheit. Weder der TXT noch der mit ihm verbundene PC waren mit dem Internet verbunden. Bösartige Angreifer waren damit ausgeschlossen, und die jugendlichen Nutzer konnten auch nicht versehentlich Kameradaten vom TXT aus dem heimischen Kinderzimmer ins Internet übertragen. Die kleine Online-Welt des TXT war sicher und verschlossen und

garantiert auf das Kinderzimmer beschränkt.

Bald bewies aber die Community Firmware, dass man den TXT auch mit dem Heim-WLAN und damit auch mit dem gesamten Internet verbinden kann. Den meisten Beteiligten war schnell klar, dass damit auch neue Risiken eingegangen wurden, und die Community Firmware erhielt neben aktuellen Software-Komponenten auch eine eigene Firewall. Einige besonders empfindliche Dienste (wie der ROBO Zugriff per fischertechnik-GUI) laufen inzwischen nicht mehr unbeaufsichtigt im Hintergrund, sondern müssen explizit durch den Anwender gestartet werden.

Kurze Zeit später zog fischertechnik nach und rüstete 2016 auch in der offiziellen Firmware 4.2.4 des TXT die Möglichkeit nach, den TXT mit dem Heim-WLAN und somit indirekt mit dem Internet zu verbinden. Die übrige Software des TXT und auch die Sicherheitsmechanismen blieben davon unberührt.

Einer dieser Sicherheitsmechanismen, dem dieser Beitrag besondere Beachtung schenkt, ist der Schutz des so genannten Root-Passworts. In der Linux-Welt des TXT ist das Root-Passwort das Gegenstück zum Administrator-Passwort unter Windows. Erlangt man Kenntnis dieses Passworts und hat man Netzwerkzugang zum TXT, so kann man aus der Ferne die volle

Kontrolle über das Gerät übernehmen. Man kann beliebige Software installieren und starten und hat freien Zugriff auf alle angeschlossene Hardware wie Sensoren und Kameras. Das Root-Passwort eines mit dem Internet verbundenen Gerätes sollte daher auf keinen Fall einem Angreifer aus dem Internet zugänglich sein.

Nun ist es meist nicht direkt möglich, aus dem Internet auf heimische Netzwerkgeräte zuzugreifen. Normalerweise leitet der DSL-Router daheim keine Zugriffe aus dem Internet ins Heimnetz weiter. Softwarefehler und Fehlkonfigurationen sind aber hier an der Tagesordnung, daher ist es fahrlässig, sich auf die Sicherheit des Heimnetzwerks zu verlassen. Als Betreiber eines privaten Netzwerks muss man leider davon ausgehen, ungebettete Gäste zu haben – insbesondere wenn man videofähige Spielzeuge nutzt ist eine gewisse Vorsicht geboten.

Tatsächlich war sich fischertechnik der Risiken von vornherein bewusst: Im Dokument „ROBOTICS TXT Controller Security Information Firmware Version 4.1.4“ werden sie recht deutlich beschrieben, auch wenn einige kritische Aspekte dort keine Erwähnung finden. Allerdings gab es zu jener Zeit noch keine Möglichkeit, den TXT mit dem Internet zu verbinden. Und wenn die neuen Möglichkeiten bisher kaum beworben und vor allem von Kindern wohl kaum genutzt werden, so ist doch davon auszugehen, dass sich das mit dem Erscheinen des *Smart Home* Kastens ändert, da dessen Funktion auf eine bestehende Internet-Verbindung des TXT angewiesen ist.

Spätestens im Sommer dieses Jahres werden sich also massenhaft TXTs ins Internet verbinden. Wenn es nach fischertechnik geht, werden diese Geräte vor allem auch unbeaufsichtigt und über einen längeren Zeitraum laufen und automatisch Video- und Sensordaten sammeln und im Internet hinterlegen.

Wie aber sieht es dabei mit dem Root-Passwort aus? Um zu verstehen, was es mit dem Passwort auf sich hat und warum es wo hinterlegt ist, muss man verstehen, wie die Netzwerkkommunikation des TXT funktioniert.

Jedes mit einem Netzwerk verbundene Gerät verfügt über eine eigene IP-Adresse. Diese Adresse identifiziert es eindeutig im Netzwerk. Man kann sich die IP-Adresse wie eine Telefonnummer vorstellen. Darüber hinaus gibt es sogenannte Ports; ihre Funktion kann sich man wie die einer Durchwahl-Nummer veranschaulichen.

Unter Telefonnummer und Durchwahl erreicht man dann einen so genannten Dienst. Dienste können völlig unterschiedlicher Natur sein. Einige nehmen zum Beispiel Befehle von außen entgegen und führen diese aus, andere liefern auf Nachfrage lediglich bestimmte Informationen.

Ein Web-Service ist z. B. ein solcher Dienst. Er hat die „Durchwahl“ (Port) 80, und wenn man sich von seinem PC aus mit diesem Dienst verbindet, dann kann man die Webseiten dieses Dienstes abrufen. Genau das machen Internet-Browser. Ein TXT besitzt mehrere solcher Ports. Uns interessieren zwei davon: Die Ports 22 und 65.000. Unter der Durchwahl 65.000 ist ein spezieller fischertechnik-Dienst zu erreichen. Er wird von ROBO Pro verwendet, um Daten im Online-Betrieb mit dem TXT auszutauschen oder ROBO Pro-Programme auf den TXT zu laden. Unter der Durchwahl 22 erreicht man den sogenannten SSH-Dienst, die „Secure Shell“ oder „Sichere Eingabeaufforderung“. Hier erwartet der TXT beliebige Linux-Kommandos. Damit nun nicht einfach jeder Nutzer beliebige Kommandos ausführen kann ist der SSH-Dienst passwortgeschützt. Nur wer sich mit einem Namen identifiziert und das dazugehörige Passwort kennt darf Befehle erteilen.

Eine weitere Unterscheidung findet auf Basis des Namens statt, den man bei der

Anmeldung am SSH-Dienst angeben muss. Einige Nutzer haben weniger Rechte, andere mehr. Auf dem TXT sind die Namen zweier Nutzer voreingestellt. Der erste Nutzer heißt „ROBOPro“ und sein Passwort lautet „ROBOPro“. Das ist von fischertechnik offiziell so dokumentiert, und dieser Zugang ist von Jedermann über das Netzwerk nutzbar. Der Nutzer ROBOPro darf nur sehr wenige Befehle auf dem TXT ausführen und hat vergleichsweise wenig Möglichkeiten, so dass fischertechnik es als sicher genug erachtet hat, diesen Benutzernamen und das Passwort öffentlich zu dokumentieren. Dass wir der Meinung sind, dass das schon ein viel zu großes Risiko darstellt, wird uns später noch interessieren. Der zweite voreingestellte Benutzer hört auf den Namen „root“. Ein Root-Benutzer existiert üblicherweise auf jedem Linux- bzw. Unix-System und verfügt über maximale Rechte. Er kann auf dem Gerät praktisch tun und lassen was er will. Das Passwort des Root-Benutzers darf daher niemals in die Hände eines Angreifers fallen und z. B. nicht öffentlich dokumentiert sein. Das hat fischertechnik sichergestellt, indem sie für jeden TXT ein eigenes, zufälliges Passwort generieren. Dieses Passwort lässt sich nur über den Bildschirm des TXT ablesen. Da ein Angreifer aus dem Netzwerk aber den Bildschirm des TXT nicht sehen kann, ist es ihm unmöglich, an das Root-Passwort zu kommen.

Leider musste fischertechnik einen zweiten Weg vorsehen, um an das Passwort des Root-Benutzers zu kommen. Im normalen Betrieb wird der SSH-Zugang des TXT zwar nicht benutzt, wohl aber für das Update der Controller Firmware. Das passiert alles „magisch“ im Hintergrund. Der Benutzer merkt nicht, dass Robo Pro einen SSH-Kanal zum TXT öffnet und hier seine Magie werkeln lässt. Und da ein komplettes Update des TXT ein massiver Eingriff in das System darstellt, hat fischertechnik festgelegt, dass dafür Root-Rechte erforderlich sein sollen. Das wiederum hat natürlich

zur Folge, dass ROBO Pro für ein Update des TXT eine Möglichkeit erhalten muss, den Root-Zugang zum TXT zu nutzen.

Wir schreiben an dieser Stelle bewusst nicht, dass ROBO Pro dazu das Root-Passwort benötigt – es gibt technisch andere Lösungen, die an dieser Stelle ggf. mehr Sicherheit gebracht hätten. Um ROBO Pro den Root-Zugang zu ermöglichen, ist das Root-Passwort auf dem TXT verschlüsselt hinterlegt. Klingt soweit nicht schlecht, hat aber leider ein paar fundamentale Lücken.

Das fischertechnik-Root-Zugangs-Dokument erwähnt selbst bereits, dass man das ROBO Pro-Programm analysieren und so an den Mechanismus zur Entschlüsselung des Root-Passworts gelangen könnte. Das ist korrekt. Nun besteht ein Programm wie ROBO Pro aus einigen Millionen Befehlen und die Analyse dieses Codes mit dem Ziel, die Entschlüsselungsbefehle zu extrahieren, ist ein relativ aufwändiges Unterfangen. Aber diese Arbeit muss man sich noch nicht einmal machen.

Um zu verstehen, wie man relativ einfach an das Root-Passwort eines TXTs kommt, ohne physischen Zugang zu ihm zu haben, müssen wir etwas mehr ins Detail gehen und die Abläufe während des Update-Vorgangs etwas genauer beleuchten.

ROBO Pro verwendet im Normalfall nur die Verbindung über Port 65.000. Stellt ROBO Pro dabei fest, dass der TXT keine aktuelle Firmware verwendet, so startet es den Update-Prozess, in dessen Verlauf irgendwann das Root-Passwort Verwendung findet. Wollen wir also einen entfernten TXT angreifen, dann müsste ROBO Pro nur feststellen, dass die Software auf dem entfernten TXT veraltet ist. Daraufhin würde ROBO Pro alle nötigen Schritte unternehmen, um sich mit dem Root-Passwort am entfernten TXT anzumelden. Wenn wir diesen Vorgang beobachten könnten, dann müsste uns dabei das Root-Passwort über den Weg laufen.

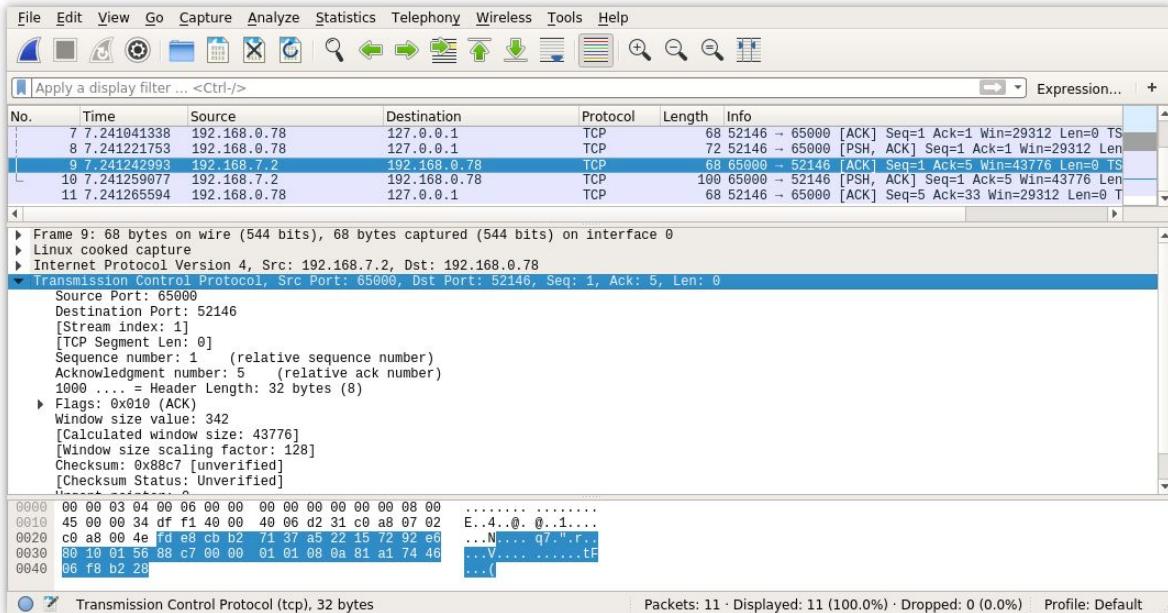


Abb. 1: Wireshark-Mitschnitt

Zunächst müssen wir also die Kommunikation zwischen ROBO Pro (auf dem PC) und dem TXT beobachten. Dazu gibt es Softwarewerkzeuge wie [Wireshark](#). Damit kann man auf seinem PC allen ein- und ausgehenden Netzwerkverkehr aufzeichnen und ihn sich später übersichtlich aufbereitet anzeigen lassen. Beobachtet man die Kommunikation mit einem nicht aktuellen TXT auf diese Weise, so fällt auf, dass die Kommunikation bereits nach wenigen Datenpaketen beendet wird.

Welche Kommunikation genau über diesen Port stattfindet ist nicht dokumentiert, und wie man im Wireshark-Trace sieht, sieht man nicht viel. Die Kommunikation scheint in keiner Weise textbasiert oder gar menschenlesbar zu sein. Dokumentiert ist das Protokoll im „[TXT C-Programming Expert Kit](#)“. Praktischerweise hat uns Torsten Stuehn mit seiner [ftrobopy-Bibliothek](#) diese Lektüre und Auswertung bereits abgenommen: ftrobopy macht nichts anderes, als die ROBO Pro-Seite der PC-TXT-Kommunikation nachzubilden. ftrobopy kann sich an Stelle von ROBO Pro mit dem TXT auf Port 65.000 verbinden und ihm Befehle erteilen. Und da ftrobopy offene Software

ist, kann man im Code von ftrobopy leicht ablesen, wie die Kommunikation mit einem TXT ablaufen sollte. Mit Hilfe dieser Informationen lässt sich in Erfahrung bringen, was die beobachtete Kommunikation bedeutet.

Leider funktioniert dieser erste Schritt nur, wenn der TXT eine veraltete Software nutzt. Meldet er die aktuelle Version, dann macht ROBO Pro einfach mit seiner normalen Kommunikation auf Port 65.000 weiter und versucht gar nicht erst, das Root-Passwort zu verwenden. Da die Kommunikation an dieser Stelle aber so simpel ist, bietet sich eine einfache Möglichkeit: Wir zweigen an dieser Stelle den Netzwerkekehr zum TXT ab, leiten ihn um auf den PC zurück und lassen dort ein Programm namens `txt_dummy.c` laufen, das eine TXTartige Kommunikation auf Port 65.000 anbietet, aber im Gegensatz zum TXT immer eine veraltete Versionsnummer liefert. Der nötige Programmcode ist lediglich 72 Codezeilen lang.

Betriebssysteme wie Linux bringen auch gleich die nötigen Werkzeuge mit, um die Kommunikation umzulenken:

```
iptables -t nat -A OUTPUT -d
192.168.0.123 -j DNAT --to-
destination 127.0.0.1 ^
```

Dieses kryptische Linux-Kommando leitet die Kommunikation, die eigentlich an den TXT unter der IP-Adresse 192.168.0.123 gerichtet ist, zurück auf den eigenen PC unter der IP-Adresse 127.0.0.1. Dazu muss man auch ROBO Pro auf einem Linux-PC laufen lassen, was dank [wine](#), einer Art Windows-Umgebung für Linux-PCs, kein Problem ist.

Sobald ROBO Pro nun den TXT ansprechen will, wird die Kommunikation auf den PC zurückgeleitet. Dort beantwortet statt des TXTs nun unser kleines `txt_dummy.c`-Programm die Versionsanfrage auf Port 65.000 mit einer älteren Versionsnummer. ROBO Pro schließt darauf die Verbindung und möchte den TXT aktualisieren. Die erste Hürde auf dem Weg zum Root-Passwort ist damit genommen.

Im nächsten Schritt sollte ROBO Pro nun versuchen, den TXT zu aktualisieren. Hier waren wir kurz überrascht, den erwarteten SSH-Datenverkehr an unseren TXT unter 192.168.0.123 nicht finden zu können. Eine kurze Analyse des gesamten Netzwerkverkehrs ergab, dass ROBO Pro den TXT per SSH immer unter der IP-Adresse 192.168.7.2 anspricht. Wer mit dem TXT im Netzwerk etwas experimentiert hat, wird vielleicht wissen, dass das die IP-Adresse eines per USB an den PC angeschlossenen TXT ist. ROBO Pro scheint den TXT also fest als USB-Gerät zu erwarten, um ein Update zu machen. Dies ist der Grund, warum man den TXT nicht über WLAN oder Bluetooth aktualisieren kann. Uns kommt (als Angreifer) diese Tatsache entgegen, da wir nun die SSH-Kommunikation anhand der IP-Adresse ebenfalls leicht umleiten, aufzeichnen und analysieren können.

Jetzt behindert uns noch die Kernfunktion von SSH: Die Datenpakete sind verschlüsselt. Wir können zwar sehen, dass ROBO Pro irgendetwas vom TXT will,

aber wir können den Inhalt der verschlüsselten Kommunikation nicht lesen. Genau das ist der Zweck der Verwendung des SSH-Protokolls, und deshalb sollte man SSH auch nutzen, wenn man z. B. über das Internet ein entferntes Gerät steuern möchte. Auch HTTPS-Verbindungen nutzen diese Verschlüsselung, um zu verhindern, dass Angreifer die Kommunikation z. B. beim Online-Banking belauschen oder gar PINs und TANs mitlesen können. Die SSH-Verschlüsselung ist für uns nicht zu knacken – was also tun?

Einen Teil der Kommunikation müssen wir gar nicht unbedingt mitlesen. Der fischertechnik-Dokumentation kann man entnehmen, dass das verschlüsselte Passwort in der Datei `/etc/rootpwd.enc` auf dem TXT liegt. ROBO Pro wird nun irgendwie versuchen, diese Datei zu lesen. Lassen wir ROBO Pro genau das einfach tun. Viel interessanter ist für uns, was danach passiert – da müssen wir eingreifen und mitlesen. Aber leider ist die Kommunikation ja auch verschlüsselt.

Um zumindest mal den ersten Schritt abzukürzen haben wir uns kurzerhand manuell per SSH (unter Windows z. B. per [Putty](#)) auf dem entfernten TXT eingeloggt. Das hat auch den Vorteil, dass wir uns nicht daran stören müssen, dass ROBO Pro darauf besteht, den TXT unter 192.168.7.2 anzusprechen. Befindet sich der TXT in einem WLAN, hat er eine andere Adresse und ROBO Pro würde ihn nicht ansprechen können.

Wenn wir uns die Datei mit dem verschlüsselten Root-Passwort aber per SSH oder Putty holen wollen geht das auch mit einem entfernten TXT, der z. B. in ein WLAN eingebucht ist. Folgendes Kommando gibt das verschlüsselte Passwort eines entfernten TXTs aus:

```
ssh ROBORro@<IP-Adresse des
anzugreifenden TXT> cat
/etc/rootpwd.enc
```

Wir werden dann nach einem Passwort gefragt, aber das ist ja das wohlbekannte und von fischertechnik dokumentierte „ROBOPro“:

```
$ ssh ROBORro@<TXT> cat  
/etc/rootpwd.enc ROBORro@<TXT>'s  
password: ROBOPro  
fJDmTWaoHDadxqnR4J1WnhvaGub6KADtxZ  
7/leggriY=  
$
```

Wie man sieht rückt der TXT anstandslos das verschlüsselte Passwort raus. Es lautet

```
fJDmTWaoHDadxqnR4J1WnhvaGub6KADtxZ  
7/leggriY=
```

Wir haben als nächstes versucht, den gesamten SSH-Verkehr ebenfalls auf den lokalen PC umzuleiten. Dazu haben wir auf dem lokalen PC einen SSH-Server installiert, wie er auch auf dem TXT läuft. Wir haben einen Benutzer namens ROBO Pro auf dem lokalen PC angelegt, die Datei /etc/rootpwd.enc hinterlegt und wieder ROBO Pro gestartet. Und tatsächlich, in der ROBO Pro-Fortschrittsanzeige schreitet der Update-Fortschrittsbalken bis zum eigentlichen Update fort, bleibt dort aber stehen.

Zu diesem Zeitpunkt war in unsere Experimente kein echter TXT mehr involviert: Wir kannten das verschlüsselte Root-Passwort und ROBO Pro war mit unserer vorgedankten Kommunikation zufrieden genug, um den Updatevorgang starten zu wollen. Wir wussten also, dass ROBO Pro zu diesem Zeitpunkt versuchte, sich mit dem geheimen Root-Passwort anzumelden.

Nun mussten wir uns das Passwort nur noch anzeigen lassen. Das hat sich aber als unerwartet aufwändig herausgestellt. Könnte man die Ausgabe des Passworts auf üblichen Linux-Servern mal eben aktivieren, dann könnte zumindest ein Administrator, der diese Option aktiviert, schnell die Passwörter seiner Nutzer in Erfahrung bringen. Um das zumindest zu erschweren sehen SSH-Server-Programme, wie auch der TXT eines enthält, nicht vor, diese

Daten zu protokollieren oder anderweitig preiszugeben.

Wir sind aber nicht die ersten, die den Inhalt von SSH-Verbindungen sehen wollen. Einen ähnlichen Bedarf haben z. B. so genannte „Honeypots“, also Honigtöpfe. So nennt man im Internet leicht zu erreichende Systeme, die aufgestellt werden, um Hackern eine Falle zu stellen. Findet der Hacker einen solchen Honeypot und probiert er, diesen anzugreifen, so führt der Honeypot die vom Hacker empfangenen Befehle nicht aus, sondern protokolliert sie für eine spätere Analyse. Auf einige Befehle wird er ggf. mit Antworten reagieren, die einem echten SSH-Server entsprechen, um den Hacker etwas länger bei Laune zu halten und ihn zu verleiten, mehr seiner Angriffstechniken auszuprobieren, so dass der Betreiber des Honeypots später bei der Analyse der Protokolle möglichst viel über den Angriffsversuch in Erfahrung bringen kann. Und genau das wollen wir auch!

Eine kurze Suche im Internet brachte diverse Honeypots zu Tage. Wir haben uns willkürlich für ein System namens [Cowrie](#) entschieden. Cowrie kann alle möglichen Dienste (einschließlich SSH) nachbilden. Wir haben also eine Cowrie-Installation so angepasst, dass sie einem Benutzer namens ROBOPro einen scheinbaren Login sowie den Abruf der verschlüsselten Root-Passwort-Datei erlaubt. Außerdem haben wir den Root-Login mit beliebigem Passwort erlaubt und ROBO Pro's SSH-Verbindungsanfragen auf unser Cowrie-Setup umgeleitet. Wieder kam ROBO Pro bis zu Schritt drei des Update-Vorgangs. Im Gegensatz zu einem echten SSH-Server schreibt Cowrie aber alle Eingaben in eine Log-Datei. Ein Teil dieser Log-Datei ist im Anhang abgebildet.

Interessant ist vor allem die Zeile, die mit 2018-02-16T14:42:29+0100 beginnt. Hier hat sich laut Cowrie ein Benutzer namens root mit dem Passwort xZ6TXLDBULYV eingeloggt.

Und das ist es: Das Root-Passwort unseres angegriffenen TXT lautet

XZ6TXLDBULYV

Wir haben den TXT dazu nicht berühren müssen; letztlich mussten wir nur per SSH oder Putty und mit Hilfe des allgemein bekannten ROBO Pro-Benutzerzugangs und dem ebenfalls bekannten Passwort „ROBO Pro“ die verschlüsselte Datei abrufen und von ROBO Pro entschlüsseln lassen.

Damit haben wir volle Kontrolle über den entsprechenden TXT. Wir können beliebig Hardware steuern, wir können eine ggf. angeschlossene Kamera auslesen, und sobald wir das Interesse verloren haben, könnten wir den internen Flash-Speicher des TXT löschen und ihn damit so schwer beschädigen, dass er ein Service-Fall für fischertechnik würde. Das können wir alles ohne den TXT jemals gesehen oder berührt zu haben.

Lösungsvorschläge

Das klingt alles irgendwie böse und gefährlich und man kann fischertechnik nicht guten Gewissens das Problem aussitzen lassen. Bisher war die Gefahr eher gering, da die normale Verwendung des TXT keine Online-Verbindung beinhaltete. Die dazu nötigen Informationen musste man sich separat bei fischertechnik besorgen und die allermeisten Nutzer dürften von dieser Möglichkeit gar nicht erst erfahren haben. Das wird sich grundlegend ändern, wenn der Smart-Home-Baukasten auf den Markt kommt und der TXT für die dort zu bauenden Modelle mit dem Internet verbunden werden muss. Noch ist etwas Zeit, das Problem zu adressieren.

Was kann man aber gegen dieses Problem tun? Der erste Gedanke ist natürlich, dass man die vorhandenen Dienste irgendwie besser absichern muss. So könnte man die Root-Passwort-Verschlüsselung durch einen neuen und besseren Mechanismus

ersetzen. Da diese und generell der Umgang mit dem SSH-Zugang den gesamten Vorgang des Firmware-Updates beeinflusst, betreffen solche Änderungen den TXT und auch ROBO Pro, so dass Updates und Modifikationen an vielen Stellen nötig werden. Je nach Art der Änderung wird auch der Nutzer sich mit einem veränderten Update-Mechanismus auseinander setzen müssen. Und letztlich werden Up- und Downgrades über mehrere Versionssprünge hinweg immer irgendwie mit dem geänderten Update-Mechanismus umgehen müssen.

Aber es geht auch einfacher. Zunächst lohnt es sich, sich ganz prinzipiell vor Augen zu führen, auf welche Dinge ein Hacker Zugriff erlangen kann und auf welche nicht. Zugreifen kann er auf die ROBO Pro-Software und auf die Software eines TXT, und hier krankt schon die bisherige Lösung: Die Ver- und Entschlüsselung des Passworts findet auf dem TXT und in ROBO Pro statt. Entsprechend schwierig ist es, das vor einem Hacker zu verbergen.

Auf was hat der Hacker aber garantiert keinen Zugriff? Er hat keinen physischen Zugang zum angegriffenen TXT. Wenn man es also schafft, dass man für den Angriff physischen Zugriff auf den TXT haben muss, dann hat der Hacker keine Chance. Für den physischen Zugriff bieten sich die Anschlüsse des TXT und der Touchscreen an. Die fischertechnik-Anschlüsse sind eher ungeeignet, aber eine USB-Verbindung z. B. kann der Hacker nicht herstellen. Und der Touchscreen lässt sich natürlich sehr gut für Nachfragen und Bestätigungen nutzen.

Ich sehe vor allem zwei Lösungen. Beide nutzen aus, dass der gesamte SSH-Zugang nur für das Update sowie ggf. für seltene Administrationsaufgaben benötigt wird. Das bedeutet, dass wir den gesamten SSH-Dienst im Normalbetrieb eigentlich komplett abschalten können. Das hat den wunderbaren Effekt, dass fischertechnik sich

über Passwörter oder veraltete SSH-Versio-
nen und ähnliches keinerlei Gedanken mehr
machen muss. Wenn der SSH-Dienst nur
für diese seltenen, speziellen Aufgaben
aktiviert wird, ist der TXT im Normalbe-
trieb auch nicht über diesen Dienst angreif-
bar. Das löst praktischerweise auch gleich
das Problem des gering privilegierten, aber
dafür mit Name und Passwort öffentlich
dokumentierten Benutzerkontos ROBOPro.
Schon dieses Konto gibt z. B. Zugriff auf
die angeschlossene Kamera, damit ist das
Abgreifen der Videodaten aus dem Kinder-
zimmer auch ohne Knacken des Root-Pass-
worts möglich.

Die Lösung besteht darin, den SSH-Dienst
bei Systemstart des TXT nicht automatisch
zu starten. Die dazu nötigen Änderungen
sind trivial und betreffen nur die Konfigura-
tion des TXT.

Die nächste Frage ist natürlich, wann denn
stattdessen der SSH-Dienst gestartet wird.
Für Updates wird er ja nach wie vor benötigt.
Offensichtlich kann man im Touch-
screen-Menü des TXT dafür einen Menü-
punkt ergänzen. Steht ein Update an, dann
muss der Benutzer manuell am TXT den
SSH-Dienst starten. Man könnte den Menü-
punkt „Update zulassen“ nennen. Erfahrene
Nutzer wüssten, dass sich hinter dieser
Option der SSH-Dienst verbirgt und könnten
diesen Menüpunkt dann nutzen, um sich
selbst manuell Zugang z. B. für Konfigura-
tionsänderungen wie das „[Freischalten des
Bootloaders](#)“ verschaffen. Eine solche Ände-
rung würde nur den TXT betreffen und
alle in diesem Beitrag beschriebenen Un-
sicherheiten beseitigen.

Es geht sogar noch einfacher. Wie wir
während unserer Experimente ja festgestellt
haben, kann ROBO Pro den SSH-Zugang
nur dann nutzen, wenn der TXT per USB
mit dem PC verbunden ist. Also wäre es
sinnvoll, den SSH-Dienst überhaupt nur
dann zu starten, wenn eine USB-Verbin-
dung zum PC besteht, was im Falle der zu
erwartenden IoT-Experimente z. B. nicht

der Fall ist. Das TXT-Linux bietet spezielle
Aufhänger (Hooks), um Dinge zu erledigen,
wenn bestimmte Netzwerkverbindungen
aktiviert oder deaktiviert werden. Die
Community-Firmware nutzt dies, um z. B.
für eine WLAN-Verbindung ein paar nötige
Hilfsprogramme im Hintergrund zu starten
und zu stoppen. Auf dem TXT lassen sich
in der Datei `/etc/networks/interfaces`
mit den „pre-up“- und „post-down“-Auf-
hängern beliebige Dienste starten und stop-
pen. Genau dieser Aufhänger bietet sich an,
um nur für die USB-Verbindung den SSH-
Dienst zu aktivieren. Das erfordert nur
wenige Änderungen am TXT und löst das
Problem unsicherer SSH-Zugänge kom-
plett. Für normale Benutzer ändert sich
nichts, denn das Update hat ja schon immer
nur dann funktioniert, wenn der TXT per
USB mit dem PC verbunden war. Lediglich
erfahrenere Benutzer würden in Zukunft
ggf. vermissen, dass sie über WLAN per
SSH auf ihren TXT zugreifen können.
Diese Nutzer werden in Zukunft vorher das
USB-Kabel einstecken müssen – ein gerin-
ger Preis für den erzielten Sicherheits-
gewinn.

Ich habe fischertechnik Anfang Februar
über das Problem informiert – zusammen
mit einem mir eigentlich unbekannten Pass-
wort – und ihnen diese konkreten und leicht
umsetzbaren Lösungsvorschläge unterbrei-
tet.

Fazit

Vor allem bei Systemen wie dem TXT, bei
denen sich das Nutzungsverhalten über die
Zeit und mit den diversen Updates und
Erweiterungen stark ändert, ist es wichtig,
immer wieder zu prüfen, ob die ursprüng-
lichen Sicherheitsvorkehrungen noch aus-
reichen. Dabei muss das Ziel nicht immer
sein, bestehende Sicherheitssysteme zu ver-
stärken; oft ist es viel sinnvoller, über kon-
zeptionelle Lösungen nachzudenken. Ein
SSH-Dienst, der gar nicht aktiviert ist, muss
auch nicht abgesichert und kann trotzdem
nicht angegriffen werden.

```

2018-02-16T14:42:29+0100 [SSHChannel session (0) on SSHService 'ssh-
connection' on HoneyPotSSHTransport,0,127.0.0.1] remote close
2018-02-16T14:42:29+0100 [HoneyPotSSHTransport,0,127.0.0.1] avatar ROBOPro
logging out
2018-02-16T14:42:29+0100 [HoneyPotSSHTransport,0,127.0.0.1] connection lost
2018-02-16T14:42:29+0100 [HoneyPotSSHTransport,0,127.0.0.1] Connection lost
after 1 seconds
2018-02-16T14:42:29+0100 [cowrie.ssh.factory.CowrieSSHFactory] New
connection: 127.0.0.1:47672 (127.0.0.1:2222) [session: 8c3d6924bb23]
2018-02-16T14:42:29+0100 [HoneyPotSSHTransport,1,127.0.0.1] Remote SSH
version: SSH-2.0-PuTTY_Local:_Feb_22_2015_12:23:14
2018-02-16T14:42:29+0100 [HoneyPotSSHTransport,1,127.0.0.1] kex alg, key
alg: 'diffie-hellman-group-exchange-sha256' 'ssh-rsa'
2018-02-16T14:42:29+0100 [HoneyPotSSHTransport,1,127.0.0.1] outgoing:
'aes256-ctr' 'hmac-sha1' 'none'
2018-02-16T14:42:29+0100 [HoneyPotSSHTransport,1,127.0.0.1] incoming:
'aes256-ctr' 'hmac-sha1' 'none'
2018-02-16T14:42:29+0100 [HoneyPotSSHTransport,1,127.0.0.1] NEW KEYS
2018-02-16T14:42:29+0100 [HoneyPotSSHTransport,1,127.0.0.1] starting
service 'ssh-userauth'
2018-02-16T14:42:29+0100 [SSHService 'ssh-userauth' on
HoneyPotSSHTransport,1,127.0.0.1] 'root' trying auth 'none'
2018-02-16T14:42:29+0100 [SSHService 'ssh-userauth' on
HoneyPotSSHTransport,1,127.0.0.1] 'root' trying auth 'keyboard-interactive'
2018-02-16T14:42:29+0100 [SSHService 'ssh-userauth' on
HoneyPotSSHTransport,1,127.0.0.1] login attempt [root/XZ6TXLDBULYV]
succeeded
2018-02-16T14:42:29+0100 [SSHService 'ssh-userauth' on
HoneyPotSSHTransport,1,127.0.0.1] 'root' authenticated with 'keyboard-
interactive'
2018-02-16T14:42:29+0100 [SSHService 'ssh-userauth' on
HoneyPotSSHTransport,1,127.0.0.1] starting service 'ssh-connection'
2018-02-16T14:42:29+0100 [SSHService 'ssh-connection' on
HoneyPotSSHTransport,1,127.0.0.1] got channel 'session' request
2018-02-16T14:42:29+0100 [SSHChannel session (0) on SSHService 'ssh-
connection' on HoneyPotSSHTransport,1,127.0.0.1] channel open
2018-02-16T14:42:29+0100 [SSHChannel session (0) on SSHService 'ssh-
connection' on HoneyPotSSHTransport,1,127.0.0.1] unhandled request for
simple@putty.projects.tartarus.org
2018-02-16T14:42:29+0100 [SSHChannel session (0) on SSHService 'ssh-
connection' on HoneyPotSSHTransport,1,127.0.0.1] asking for subsystem
"sftp"
2018-02-16T14:42:29+0100 [SSHChannel session (0) on SSHService 'ssh-
connection' on HoneyPotSSHTransport,1,127.0.0.1] {'sftp': <class
twisted.conch.ssh.filetransfer.FileTransferServer at 0x7f2150bcbef0>}
2018-02-16T14:42:29+0100 [SSHChannel session (0) on SSHService 'ssh-
connection' on HoneyPotSSHTransport,1,127.0.0.1] SFTP getAttrs:
/opt/knobloch/update.sh
2018-02-16T14:42:29+0100 [SSHChannel session (0) on SSHService 'ssh-
connection' on HoneyPotSSHTransport,1,127.0.0.1] SFTP openFile:
/opt/knobloch/update.sh
2018-02-16T14:42:29+0100 [SSHChannel session (0) on SSHService 'ssh-
connection' on HoneyPotSSHTransport,1,127.0.0.1] Unhandled Error

```

Listing 1: Cowrie-Log

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <ctype.h>
#define BUFFER_SIZE 1024
#define on_error(...) { fprintf(stderr, __VA_ARGS__); fflush(stderr); exit(1); }
int main (int argc, char *argv[]) {
    if (argc < 2) on_error("Usage: %s [port]\n", argv[0]);
    int port = atoi(argv[1]);
    int server_fd, client_fd, err;
    struct sockaddr_in server, client;
    char buf[BUFFER_SIZE];
    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd < 0) on_error("Could not create socket\n");
    server.sin_family = AF_INET;
    server.sin_port = htons(port);
    server.sin_addr.s_addr = htonl(INADDR_ANY);
    int opt_val = 1;
    setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &opt_val, sizeof
opt_val);
    err = bind(server_fd, (struct sockaddr *) &server, sizeof(server));
    if (err < 0) on_error("Could not bind socket\n");
    err = listen(server_fd, 128);
    if (err < 0) on_error("Could not listen on socket\n");
    printf("Server is listening on %d\n", port);
    while (1) {
        socklen_t client_len = sizeof(client);
        client_fd = accept(server_fd, (struct sockaddr *) &client,
&client_len);
        printf("Accepted\n");
        if (client_fd < 0) on_error("Could not establish new connection\n");
        while (1) {
            int read = recv(client_fd, buf, BUFFER_SIZE, 0);
            if (!read) break; // done reading
            if (read < 0) on_error("Client read failed\n");
            // build simple reply header
            buf[0] = 0x3e;
            buf[1] = 0x72;
            buf[2] = 0xc9;
            buf[3] = 0xba;

            // Version 4.2.3
            buf[23] = 4;
            buf[22] = 2;
            buf[21] = 3;
            buf[20] = 0;
            // reply with 32 bytes packet
            err = send(client_fd, buf, 32, 0);
            if (err < 0) on_error("Client write failed\n");
        }
    }
    return 0;
}
```

Listing 2: `txt_dummy.c` - Tool to simulate a TXT with outdated firmware version 4.2.3

Computing

startIDE für die Community Firmware – Programmieren direkt auf dem TXT oder TX-Pi

Peter Habermehl

Ziel bei der Entwicklung von startIDE waren schnelle und einfache Umsetzung von Mess-, Steuer- und Regelungsaufgaben im fischertechnik-Umfeld sowie eine einsteigergerechte Handhabung. Der Beitrag stellt einige Grundlagen sowie Hintergrundinformationen zu startIDE vor.

Motivation und Entwicklungsgeschichte

Sowohl der aktuelle TXT-Controller als auch die älteren fischertechnik-Robotik-Interfaces benötigen zusätzliche Hard- und Software zur Programmierung von Modellen. Mit der Community Firmware und deren Adaption für den Raspberry Pi in Form des TX-Pi-Projektes lag jedoch der Gedanke nahe, dass man unter der leistungsfähigen grafischen Benutzeroberfläche auf dem Gerät selbst Programmabläufe erstellen, d. h. programmieren könnte.

Der Besitz von Robo Interface und Robo I/O Extension veranlasste den Autor zusätzlich, diese Hardware einsteiger- und auch kindgerecht für erste Schritte auf dem Feld der Robotik nutzbar zu machen. So sollten z. B. die Modelle des Robo LT Beginner Lab steuerbar sein.

Erste Versuche mit einer grafischen Programmierung durch Anordnen von entsprechenden Icons auf dem Touchscreen des TXT ergaben schnell, dass der Screen schlichtweg zu wenig Auflösung bietet, um eine grafische Programmierumgebung zu realisieren. Dies führte zu der Idee, zwar textuellen Programmcode zu verwenden, diesen aber nicht über eine Tastatur, sondern durch die Auswahl von Befehlen

und Eingabe der Parameter über ein Touchscreen-Interface zu erzeugen.

Das hat den Vorteil, dass die Syntax der Befehlszeilen durch die automatische Generierung per se korrekt ist und die Parametrierung schon bei der Eingabe überwacht und ggf. korrigiert werden kann. Damit entfällt ein Großteil der ansonsten notwendigen Überwachung zur Laufzeit des erstellten Programmes, was wiederum die Programmierung des startIDE-Code-interpreters vereinfachte.

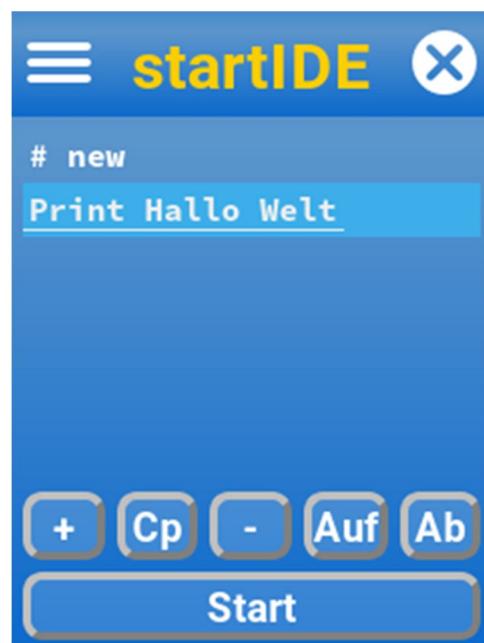


Abb. 1: startIDE-App

startIDE selbst ist als App für die Community Firmware in Python3 programmiert. Abb. 1 zeigt das Benutzerinterface von startIDE.

Das erste startIDE-Projekt – „Hallo Welt“

Nach dem Start von startIDE zeigt sich zunächst die bis auf den Kommentareintrag „# new“ leere Codeliste, unter der sich sechs Buttons finden. Nach Betätigen des „+“-Buttons öffnet sich eine Auswahlliste, die verschiedene Gruppen von Befehlen zur Auswahl anbietet (Abb. 2).

Aus der Gruppe „Interaktion“ wird nun der „Print“-Befehl ausgewählt und als auszugebender Text „Hallo Welt“ über die On-Screen-Tastatur eingegeben. Damit ist das „Hallo-Welt“-Programm schon erstellt. Ein Klick auf den Start-Button startet die Ausführung, die Benutzeroberfläche wechselt in den Ausführungsmodus und zeigt nun ein Log-Fenster sowie statt des „Start“-nun einen „Stop“-Button.

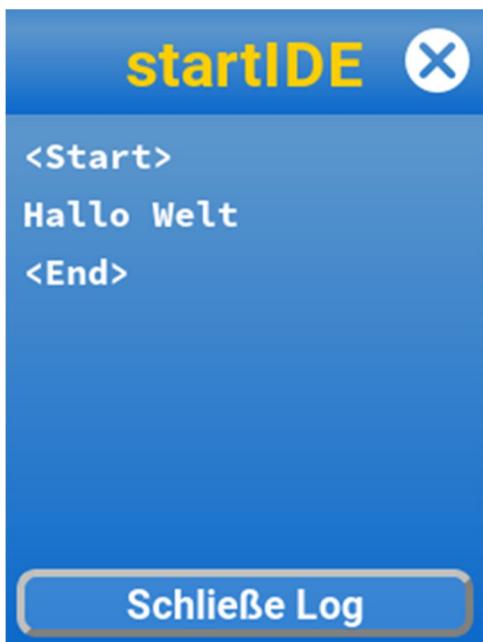


Abb. 2: „Hallo Welt“

Der Button unter dem Logscreen dient nach dem Programmende zum Schließen des Logscreens und damit zur Rückkehr zum Programmcodeeditor.

Der Funktionsumfang

Gegenwärtig kann startIDE, installiert auf dem TXT oder TX-Pi, die I/O des TXT-Controllers und jeweils eines daran über USB angeschlossenen Robo Interfaces, einer I/O Extension, eines Robo LT Controllers oder eines ftDuinos (wenn dieser unter `ftduino_direct` läuft) ansprechen. Ein experimenteller Support für das Intelligent Interface ist auch vorhanden.

startIDE bietet in der Gruppe „Input“ Befehle zum Abfragen und Auswerten der Eingänge des Controllers (beim TXT) und des daran angeschlossenen Interface. Die Eingänge können je nach Fähigkeiten der verwendeten Hardware digital oder analog ausgewertet werden, bei Analogbetrieb ist die Erfassung von Spannung bzw. Widerstand möglich. Die Ultraschallsensoren für TXT und ROBO Interface werden ebenfalls unterstützt.

In der „Output“-Gruppe sind die Funktionen zur Ansteuerung der Ausgänge zusammengefasst. Neben dem einfachen Ein- und Ausschalten einzelner Ausgänge gibt es Funktionen zum Betrieb von Encodermotoren (an dazu geeigneten Interfaces) oder Motoren mit Impulsgeber und Endschalter, womit die Ansteuerung z. B. der älteren und des aktuellen Industrieroboter-Baukastens möglich ist.

In der „Variable“-Gruppe können Variable definiert und mit den Befehlen der gleichnamigen Gruppe verarbeitet werden. Es steht eine Arithmetik-Funktion „Calc“ zur Verfügung, die zwei Operanden mittels eines Operators verknüpft. Operatoren sind z. B. die vier Grundrechenarten, Ganzzahldivision und Modulo (Divisionsrest), es können Minimum- und Maximumauswahl aus den Operatoren getroffen werden, zur Erzeugung von Zufallszahlen gibt es einen „Random“-Operator, es sind boole'sche And- und Or-Verknüpfung möglich, ebenso wie es eine Reihe von Vergleichsoperatoren (größer, kleiner, gleich, ungleich usw.) gibt.

Alle Rechenoperationen der „Calc“-Funktion werden als Integerrechnungen durchgeführt, wobei Python und damit startIDE keine Begrenzung des Datentyps kennen, die Zahlen können also „unendlich“ groß werden.

Eine weitere mathematische Funktion ist der „FromPoly“-Befehl, der als einzige Funktion in startIDE Fließkommawerte (auch in Exponentialschreibweise) als Parameter erlaubt. Mit den vier Koeffizienten A-D wird ein Polynom dritten Grades in der Form $f(x) = A*x^3 + B*x^2 + C*x + D$ berechnet und das Ergebnis als Integerwert in eine startIDE-Variable geschrieben.

Neben vielfältigen anderen Möglichkeiten kann durch die Wahl entsprechender Koeffizienten einer Näherungskurve die Kennlinie z. B. eines NTC-Widerstands abgebildet werden, so dass mit dem gemessenen Widerstand als Eingangsgröße die äquivalente Temperatur errechnet werden kann.

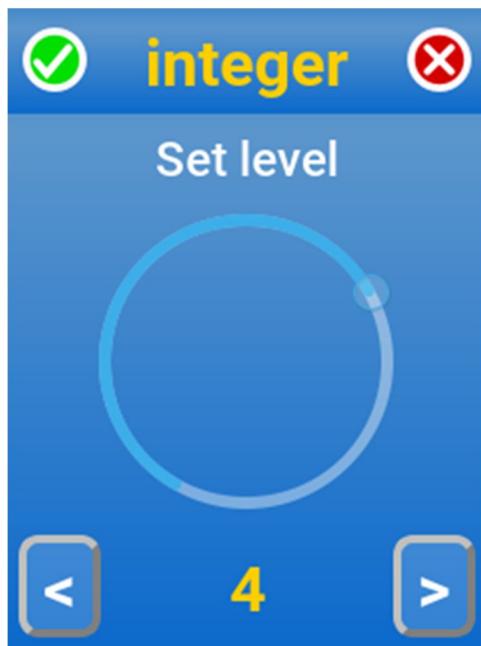


Abb. 3: Eingabefunktion „FromDial“, mit der der Wert einer Variable interaktiv während des Programmablaufs gesetzt werden kann

Die Gruppe „Steuerung“ bietet Elemente zur Programmablaufsteuerung, wie z. B.

den Sprungbefehl „Jump“ und die Definition von Sprungzielen mit dem „Tag“-Befehl. Ebenso finden sich hier die Funktionen zur Definition von Schleifen (LoopTo) und zur Zeitsteuerung (Ermittlung der Laufzeit, Zeitmessung, Interrupt).

In der Gruppe „Module“ sind die Funktionen für Unterprogramme, Module genannt, zusammengefasst. Module können entweder Bestandteil des Programmcodes (internes Modul) oder auf SD-Karte gespeichert (externes Modul) sein. Externe Module können von mehreren startIDE-Programmen verwendet werden.

Befehle für die Bildschirmausgabe von Text, das Löschen des Logscreens sowie zum Aktivieren des Loggens der Ausgaben in eine Logdatei finden sich in der Gruppe „Interaktion“.

Über das Webinterface von startIDE können Programme und Module vom Interface zum Computer gesendet, dort evtl. bearbeitet und zurück übertragen werden. Weiterhin besteht die Möglichkeit, Logfiles zur weiteren Bearbeitung als Rohdaten oder csv-konvertiert herunterzuladen.

Einsteiger-Projekte

Der Händetrockner

Seit vielen Jahren ist der „Händetrockner“ ein beliebtes Modell in den fischertechnik Robotik-Baukästen: Ein mit Luftschaube versehener Motor dient als Gebläse, das immer dann laufen soll, wenn eine davor befindliche Lichtschranke unterbrochen wird (Abb. 4).

Im Beispiel wird als Hardware ein Robo LT Controller verwendet, an Motorausgang M1 ist der Gebläsemotor angeschlossen, an Motorausgang M2 die Lampe der Lichtschranke. Die Fotodiode der Lichtschranke ist mit Eingang I1 des Interface verbunden.

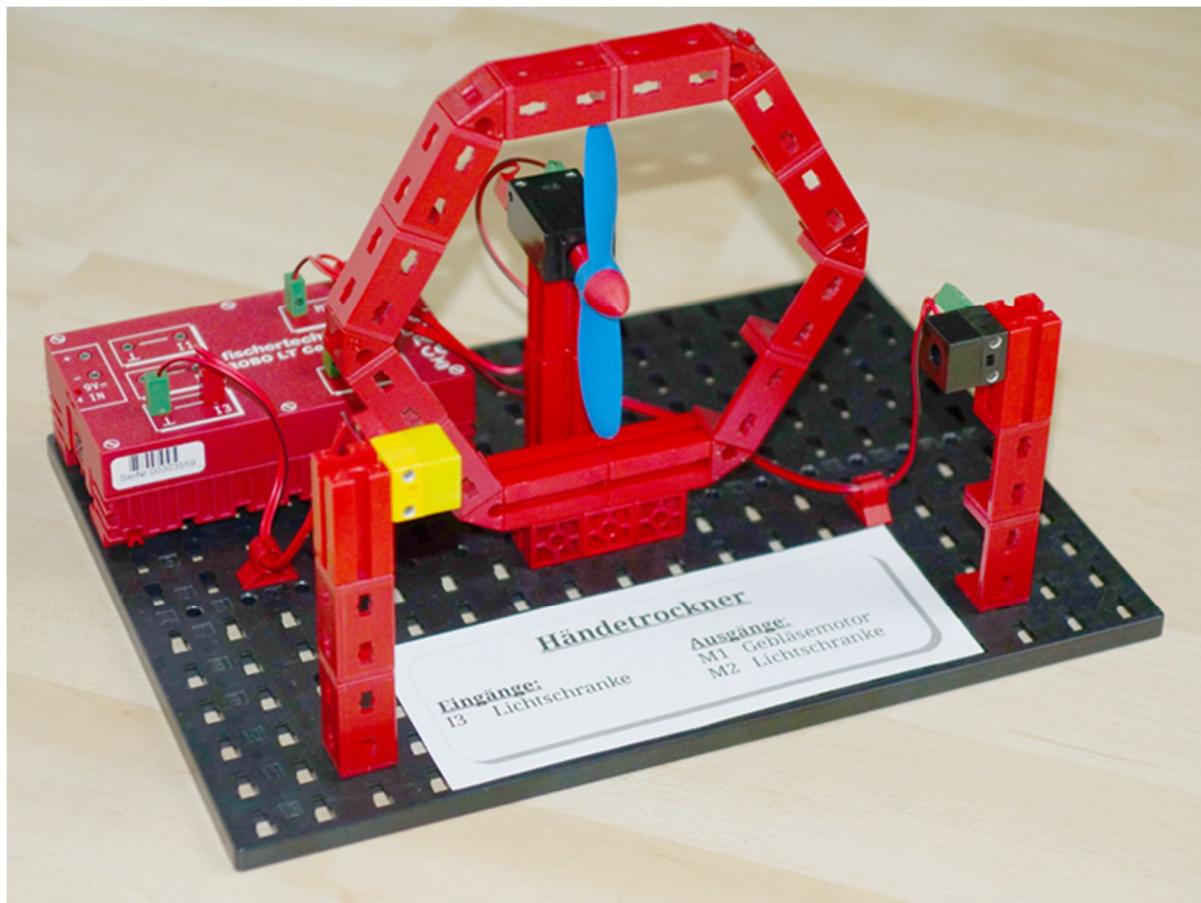


Abb. 4: Der Robo LT Händetrockner

Der dazugehörige startIDE-Code sieht so aus:

```

01 # Haendetrockner
02 Motor RIF 2 1 7
03 Delay 100
04 Tag START
05 IfInDig RIF 1 False FAN_ON
06 WaitInDig RIF 1 Falling 0
07 Tag FAN_ON
08 Motor RIF 1 r 7
09 WaitInDig RIF 1 Raising 0
10 Delay 1000
11 Motor RIF 1 s 0
12 Jump START
    
```

Listing 1: Händetrockner

Das gesamte Projekt umfasst also lediglich 11 Zeilen Programmcode. In Zeile 2 wird die Lampe der Lichtschranke eingeschaltet, es folgt eine 100 ms lange Pause in Zeile 3, damit die Fotodiode schalten kann. In Zeile 4 wird die Sprungmarke „START“ definiert. Falls die Lichtschranke nun bereits unterbrochen wurde (Digitaleingang 1 =

False), wird in Zeile 5 zur Sprungmarke „FAN_ON“ (Zeile 7) verzweigt.

Andernfalls wird in Zeile 6 auf eine fallende Signalflanke – also das Unterbrechen der Lichtschranke – gewartet. Nach dem Unterbrechen der Lichtschranke bzw. nach der Verzweigung in Zeile 5 wird in Zeile 8 der Gebläsemotor eingeschaltet und danach in Zeile 9 wiederum auf die Freigabe der Lichtschranke, also steigende Signalflanke an Eingang 3, gewartet.

Nach einer Verzögerung von 1000ms – Zeile 10 – wird der Gebläsemotor wieder ausgeschaltet und zurück zu Zeile 4 gesprungen, womit der Zyklus neu beginnt.

Temperaturregler

Als etwas komplexere Aufgabe sei hier exemplarisch ein Temperaturregler dargestellt. Ein NTC1k5 aus dem fischertechnik-Sortiment als Temperatursensor wurde

direkt über einer Linsenglühlampe montiert. Der NTC ist an Eingang I1 eines ftDuino angeschlossen, die Lampe an Motorausgang M1.

Die Programmieraufgabe ist nun, die Temperatur am NTC auf 30° Celsius einzuregeln; dies soll mit einem P-Regler geschehen.

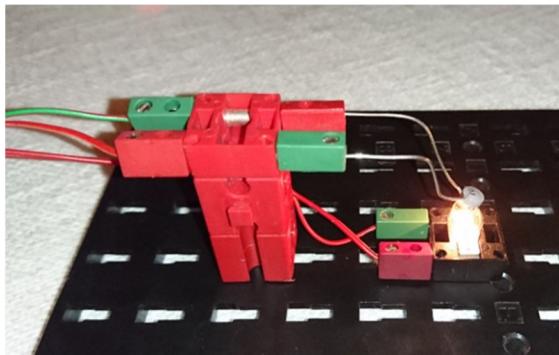


Abb. 5: Temperaturregelung mit NTC

Der Programmcode dazu:

```

01 # Temperaturregler
02 Init R_NTC 0
03 Init PWM_OUT 0
04 Init Kp 2
05 Init T_SOLL 30
06 Init T_Ist 0
07 Init T_DELTA 0
08 Init X 0
09 #
10 Log 1
11 TimerClear
12 Tag LOOP
13 FromIn FTD 1 R R_NTC
14 FromPoly T_Ist R_NTC -
1.19005379623e-09 1.236196074067e-
05 -0.047586949030459 15
72.5923764402556
16 Calc T_DELTA T_SOLL - T_Ist
17 Calc X T_DELTA * Kp
18 Calc PWM_OUT PWM_OUT + X
19 Calc PWM_OUT PWM_OUT max 0
20 Calc PWM_OUT PWM_OUT min 512
21 Motor FTD 1 1 PWM_OUT
22 Clear
23 TimerQuery
24 QueryVar T_SOLL
25 QueryVar T_Ist
26 QueryVar PWM_OUT
27 Delay 100
28 Jump LOOP

```

Listing 2: Temperaturregler

In den Zeilen 2 bis 8 werden die verwendeten Variablen deklariert und initialisiert; in Zeile 10 wird das Datenlogging in Datei aktiviert. In der Schleife zwischen Zeile 12 und 28 wird zunächst die Ist-Temperatur ermittelt. Dazu wird in Zeile 13 der Widerstand des NTC in die Variable R_NTC eingelesen und der Widerstandswert in Zeile 14 über eine Transferfunktion dritten Grades in die äquivalente Temperatur umgerechnet und diese in die Variable T_IST geschrieben.

Die Regelabweichung T_DELTA als Differenz aus Soll- und Istwert wird in Zeile 17 mit dem Verstärkungsfaktor Kp multipliziert und in die Hilfsvariable X gespeichert. In Zeile 18 wird die Stellgröße PWM_OUT mit dem Wert der Hilfsvariable X nachgeführt, in den Zeilen 19 und 20 wird PWM_OUT auf einen Wert zwischen 0 und 512 begrenzt und schließlich der Ausgang M1 auf den errechneten Wert gesetzt.

In den Zeilen 23 bis 26 werden Zeitstempel sowie Soll- und Ist-Temperatur und PWM-Wert auf den Logscreen und in die Logdatei ausgegeben, dann erfolgt nach 100 ms Wartezeit der Rücksprung zu Zeile 12. Der Programmablauf muss manuell über den Stop-Button auf dem Logscreen beendet werden.

Ein exemplarischer Einregelvorgang ist in Abb. 6 dargestellt; die Werte wurden der vom Programm erzeugten Logdatei entnommen.

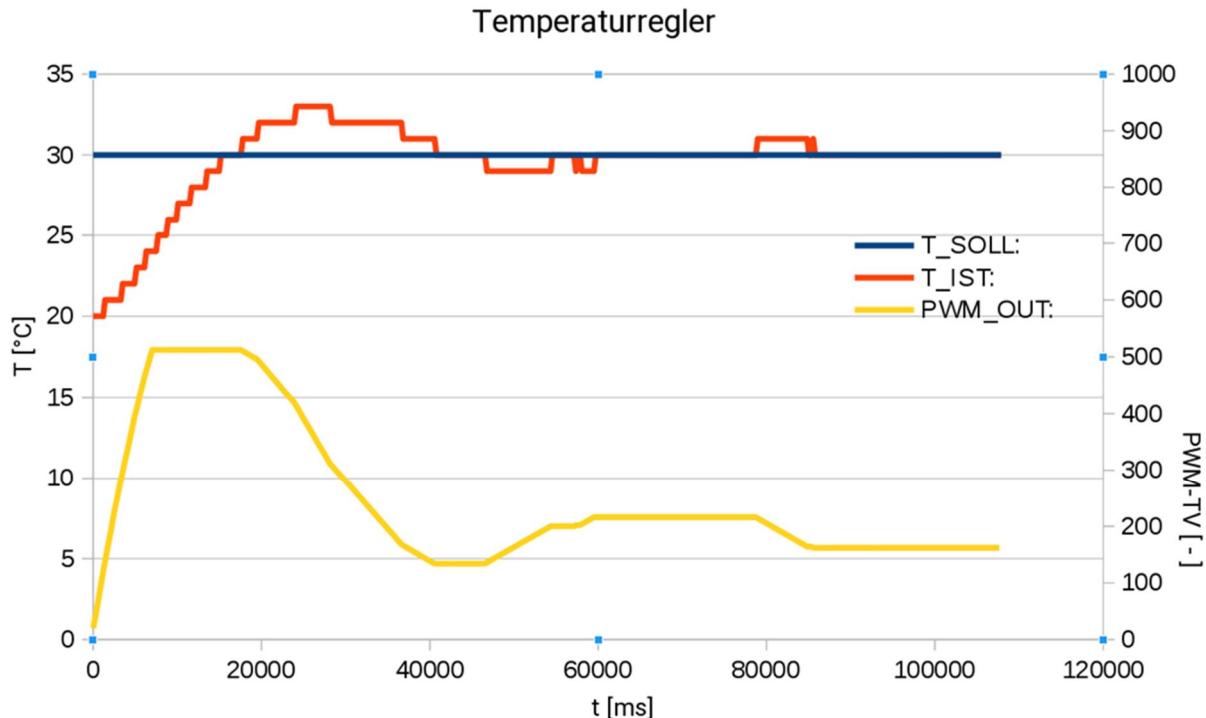


Abb. 6: Soll- und Istwertverlauf des Temperaturreglers

Ausblick

Mit startIDE steht für unter der Community Firmware betriebene TXT-Controller und TX-Pis eine App zur Verfügung, mit der sich Mess-, Steuer- und Regelungsaufgaben einfacher bis mittlerer Komplexität lösen lassen, wobei auf eine Vielzahl aktueller und älterer Hardware-Interfaces zurückgegriffen werden kann.

Ein umfangreiches Handbuch steht über das Webinterface der App unter „Get more application info“ bzw. im GitHub-Repository [1] zur Verfügung.

Für die Zukunft sind noch einige funktionale Erweiterungen angedacht, explizit Indexvariable (Arrays) sowie die Möglichkeit, Variableninhalte speichern und wieder laden zu können.

Angedacht ist auch, in Ergänzung zum Handbuch noch ein Trainingsdokument zu erstellen, das insbesondere jungen Einsteigern anhand einfacher Beispiele analog der fischertechnik-Einsteigerbaukästen (Händetrockner, Lichtsignalanlagen etc. bis hin zu einem einfachen Spurensuch-Roboter) erlaubt, spielerisch erste Erfahrungen auf dem Gebiet der Robotik zu sammeln.

Dabei gilt jedoch, wie bei allen Community-Projekten, dass die Ressourcen des Autors begrenzt sind und Mitarbeit jederzeit gern gesehen ist.

Referenzen

- [1] Peter Habermehl: [Handbuch startIDE](#). Github.

Computing

startIDE (1): Messen und Experimentieren

Rolf Meingast

Schon bald nach Einführung der ersten persönlichen Computer (PC) in den Jahren 1976/1977 (Apple I und Commodore PET 2001) wurden diese für Messaufgaben eingesetzt. Was damals nur wenigen „Profis“ möglich war, kann heute jedes Kind.

Mit dem TXT hat man ein Gerät in der Hand, mit dem man – besonders unter der TXT Community Firmware – viele Möglichkeiten zum spielerischen Messen und Experimentieren zur Verfügung hat. Ich schlage deshalb vor, in der ft:pedia in loser Reihenfolge Beispiele zu diesem Thema vorzustellen und beginne mit der

Prüfung der langen Verschlusszeiten einer alten Kamera

Eine entsprechende Messung ist sehr einfach. Eine Lichtquelle beleuchtet bei abgenommener Rückwand einen Fototransistor, während der Verschluss offen ist. Die Dauer dieser Hellphase wird mit dem TXT (oder einem ftDuino) registriert. Das Auslesen des *timers* ergibt die verstrichene Zeit in Millisekunden.

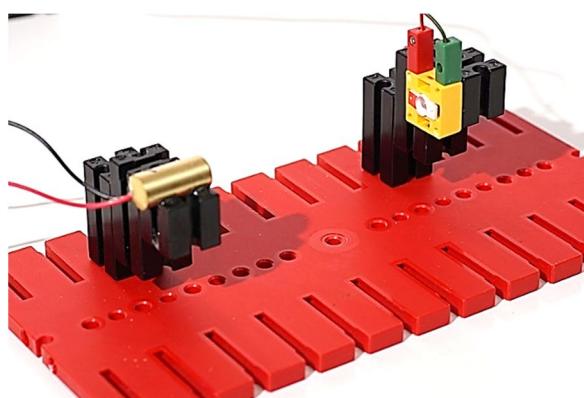


Abb. 1: Der Messaufbau

Als Lichtquelle verwende ich eine Laserdiode, die direkt mit 3 bis 12 V betrieben werden kann (Conrad 816304-62). Als Empfänger dient der fischertechnik-Fototransistor, angeschlossen an Eingang I1.

Auf der SD-Karte des freigeschalteten TXT befindet sich die Community Firmware, mit der die Programmierumgebung startIDE [2] im Store geladen wurde. Nach Aufruf der startIDE tippt man mit dem Finger auf dem Display des TXT die folgenden Befehle an:

```
+  
Eingänge  
WaitForInputDig  
Raising  
Bestätigung  
+  
Steuerung  
Time  
TimerClear  
+  
Eingänge  
WaitForInputDig  
Falling  
Bestätigung  
+  
Steuerung  
Time  
TimerQuery
```

Listing 1

Auf dem Display sieht man nun dieses Programm:

```
#new  
WaitInDig TXT 1 Raising 0  
TimerClear  
WaitInDig TXT 1 Falling 0  
TimerQuery
```

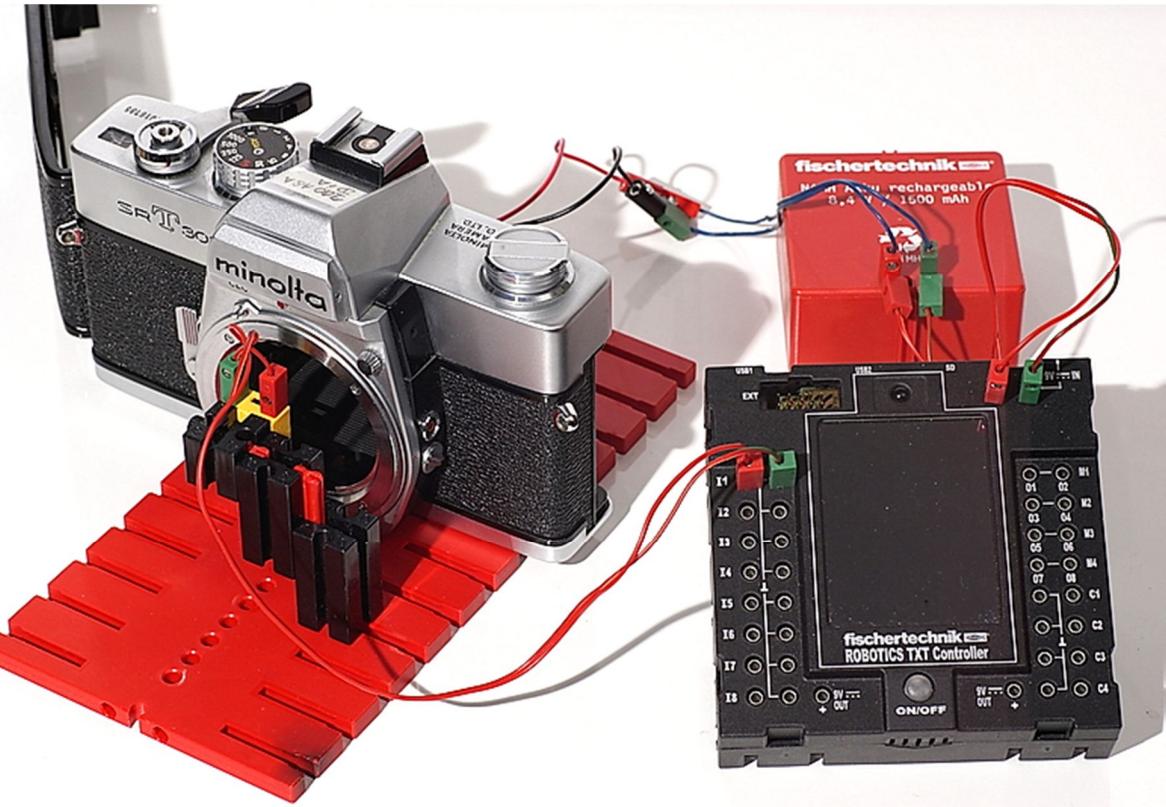


Abb. 2: Die Messung

Startet man das Programm beginnt eine Zeitmessung. Es lassen sich Zeiten bis zu $\frac{1}{30}$ s erfassen.

Je länger die Verschlusszeiten, desto genauer sind die Ergebnisse. Die Fehler liegen in der Größenordnung von wenigen Millisekunden.

Um das Programm nicht ständig manuell starten zu müssen, empfiehlt sich eine Endlosschleife. Dazu wird vor den ersten Befehl

+
Steuerung
Tag
"A"

das Sprungziel „A“ gesetzt und am Schluss mit

+

Steuerung

Jump

"A"

Bestätigung

der Sprungbefehl *Jump A* erzeugt.

Alle Befehle von *startIDE* hat ihr Entwickler Peter Habermehl in [1] ausführlich dokumentiert.

Referenzen

- [1] Peter Habermehl: [startIDE Referenzhandbuch v1.3x](#) auf GitHub, 2018.
- [2] Peter Habermehl: *startIDE für die Community Firmware – Programmieren direkt auf dem TXT oder TX-Pi*. ft:pedia 1/2018.

Computing

startIDE (2): Seilwinde

Rolf Meingast

In dem Experimentierhandbuch COMPUTING EXPERIMENTAL [1] wird als einführendes Experiment eine einfache Seilwinde behandelt. Hier zeige ich eine Übertragung auf die Programmierumgebung startIDE.

Eine Seilwinde soll über zwei Taster gesteuert werden, die die Drehrichtung vorgeben. Im Hinblick auf die Anwendung bei einem Aufzug werden mit einem dritten Taster die halben Umdrehungen der Seilwinde gezählt.

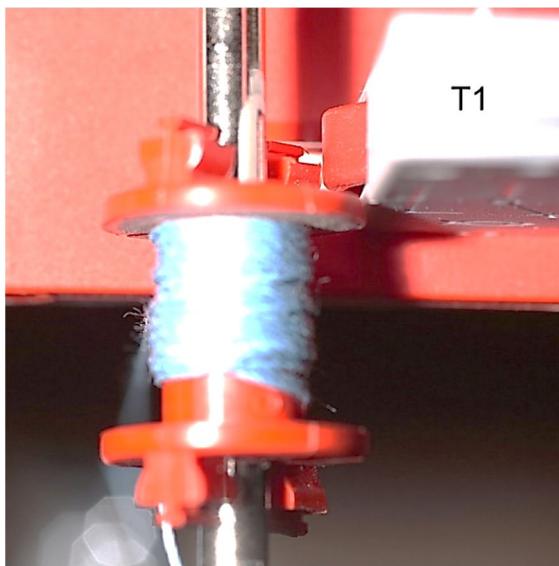


Abb. 1: Zähltaster T1 an der Seiltrommel

Ein TX-Pi steuert mit der App startIDE ein über USB angeschlossenes ROBO-Interface mit folgendem Programm:

```

1. # new
2. Interrupt Every 100 interrupt
3. Init counter 0
4. Init richtung 0
5. Tag Start
6. Call COUNTER 1
7. QueryVar counter
8. Delay 50
9. IfVar counter >= 50 Ende
10. Clear
11. QueryVar richtung

```

```

12. IfVar richtung == 1 rechts
13. IfVar richtung == 2 links
14. Motor RIF 1 s 0
15. Jump Start
16. Tag rechts
17. Motor RIF 1 r 4
18. Jump Start
19. Tag links
20. Motor RIF 1 l 4
21. Jump Start
22. Tag Ende
23. Stop
24. Module COUNTER
25. IfInDig RIF 1 True ohne
26. IfVar richtung == 0 ohne
27. Calc counter counter + 1
28. Delay 250
29. Tag ohne
30. MEnd
31. Module interrupt
32. IfInDig RIF 2 True A
33. IfInDig RIF 3 True B
34. Jump D
35. Tag A
36. Calc richtung 1 + 0
37. IfInDig RIF 3 True C
38. Jump D
39. Tag B
40. Calc richtung 1 + 1
41. Jump D
42. Tag C
43. Calc richtung 1 - 1
44. Tag D
45. MEnd

```

Listing 1

Zeile 2 bewirkt, dass das Modul *Interrupt* (ab Zeile 31) alle 100 ms aufgerufen wird, mit dem die Drehrichtung (0/1/2) durch Abfrage der Tasten 2 und 3 festgelegt wird.

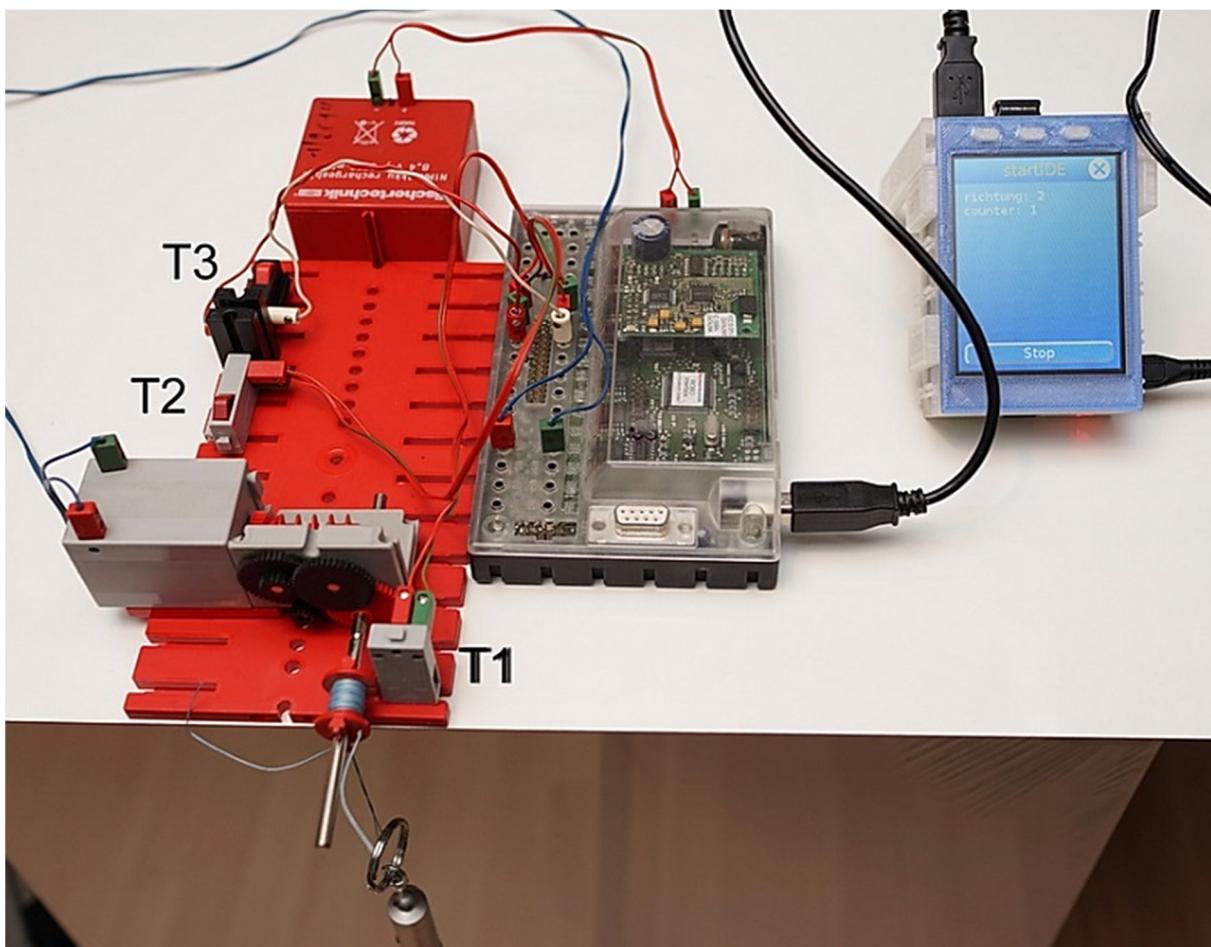


Abb. 2: Das fertige Modell

Die Auswertung der Variablen *richtung* erfolgt in den Zeilen 12 bis 14.

Nach Start erfolgt immer der Aufruf des Moduls *COUNTER* (Zeile 24). Wenn Taster 1 durch die Nocken der Seiltrommel betätigt wird, wird der Zähler um 1 erhöht und 250 ms gewartet (zum Entprellen des

Tasters). Zeile 26 soll ein Zählen bei Stillstand und gedrückter Taste verhindern.

Referenzen

- [1] fischertechnik: *Computing Experimental* (für C64).

Computing

startIDE (3): TXT im freien Fall

Rolf Meingast

Ein Fallexperiment mit Datengewinnung via startIDE und Auswertung: Zur Untersuchung der Fallbewegung lasse ich einen TXT mit Ultraschallsensor auf eine sehr weiche Unterlage fallen. Gemessen wird dabei die Entfernung zur Decke.

Aufbau

Am TXT wird oben der Ultraschall-Sensor befestigt, außerdem zwei T-Füße zur besseren Positionierung an der Zimmerdecke (Abb. 1, 2).

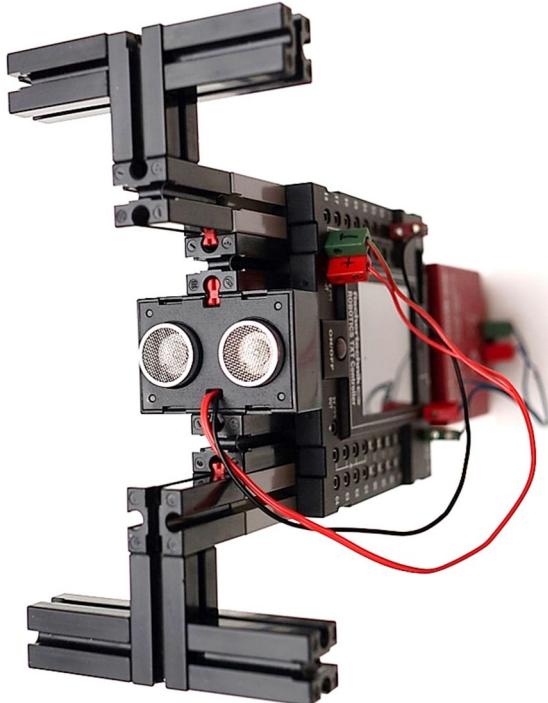


Abb. 1: Der Aufbau

Unten am TXT befinden sich ein oder mehrere Akkus.

Eine dicke und weiche Polsterung auf dem Boden oder Bett verhindert Beschädigungen beim Aufprall.

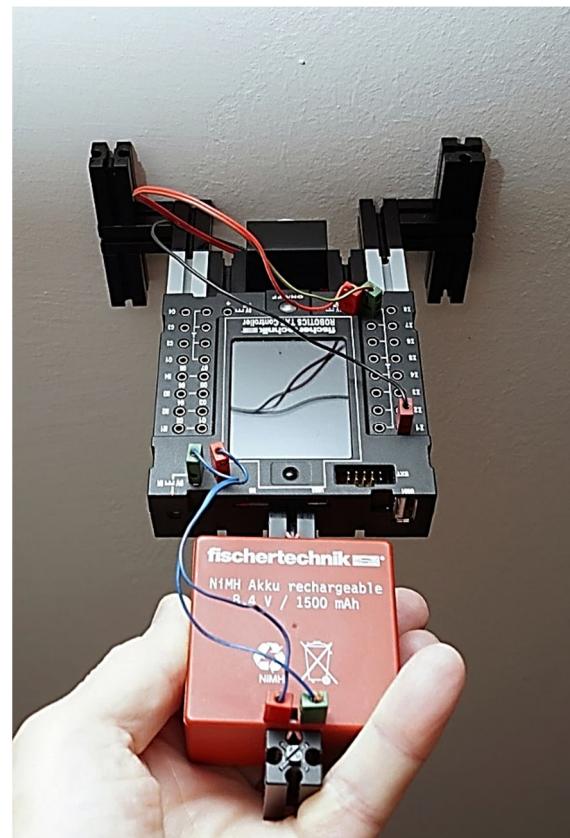


Abb. 2: Der TXT an der Zimmerdecke

Durchführung

Das Messprogramm wird gestartet, die gesamte Einrichtung gegen die Zimmerdecke gehalten und nach einigen Sekunden Beruhigungszeit losgelassen. Die Achse Ultraschall-Sensor-Akku darf sich beim Fall nicht neigen.

Das Programm

```
# new
Delay 5000 // Zeit zum
Positionieren
Print los!!!!!!
Log 1 // Datenloggen einschalten
Log silent // Datenloggen ohne
Bildschirmausgabe
TimerClear
Tag A // Sprungmarke
TimerQuery // Ausgabe des
timerwertes in msec
QueryIn TXT 1 D h: // Ausgabe der
gemessenen Entfernung
LoopTo A 500
```

Listing 1

Auswertung der Daten

Die so gewonnenen Datenpaare (*timer, h*) werden auf dem PC (z. B. unter <http://192.168.2.101/>) im startIDE-.CSV-Format ausgegeben. Im folgenden Beispiel sind das die ersten beiden Spalten:

Timer	h	Sekunden	Meter
4982	3	0,000	0,00
4998	3	0,016	0,00
5014	3	0,032	0,00

Tabelle 1

Das Diagramm mit Hilfe der Spalten 3 und 4 zeigt, dass die Auslesung der Daten häufiger als die Bereitstellung der Messwerte erfolgt (Abb. 3). Für das Diagramm in Abb. 4 wurden immer nur die ersten Datenpaare berücksichtigt, wenn die gleiche Entfernung angezeigt wird.

Zusammenfassung

Unter startIDE kann mit dem TXT der Bewegungsablauf beim freien Fall auf einfache Weise erfasst werden. Ein spannendes Experiment für jedes fischertechnik-Zimmer!

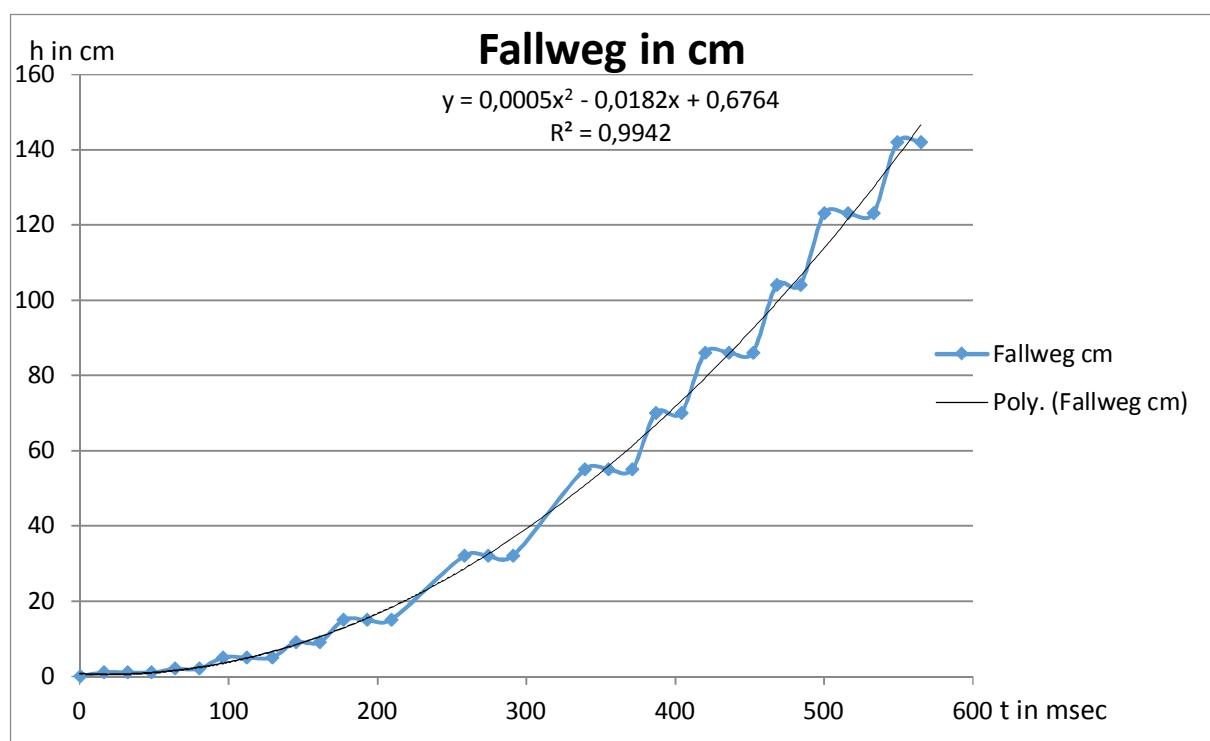


Abb. 3: Darstellung der Messergebnisse (blaue Linie mit Messpunkten) und per Excel-Trendlinienfunktion angepasster quadratischer Gleichung (schwarze Kurve) des freien Falls

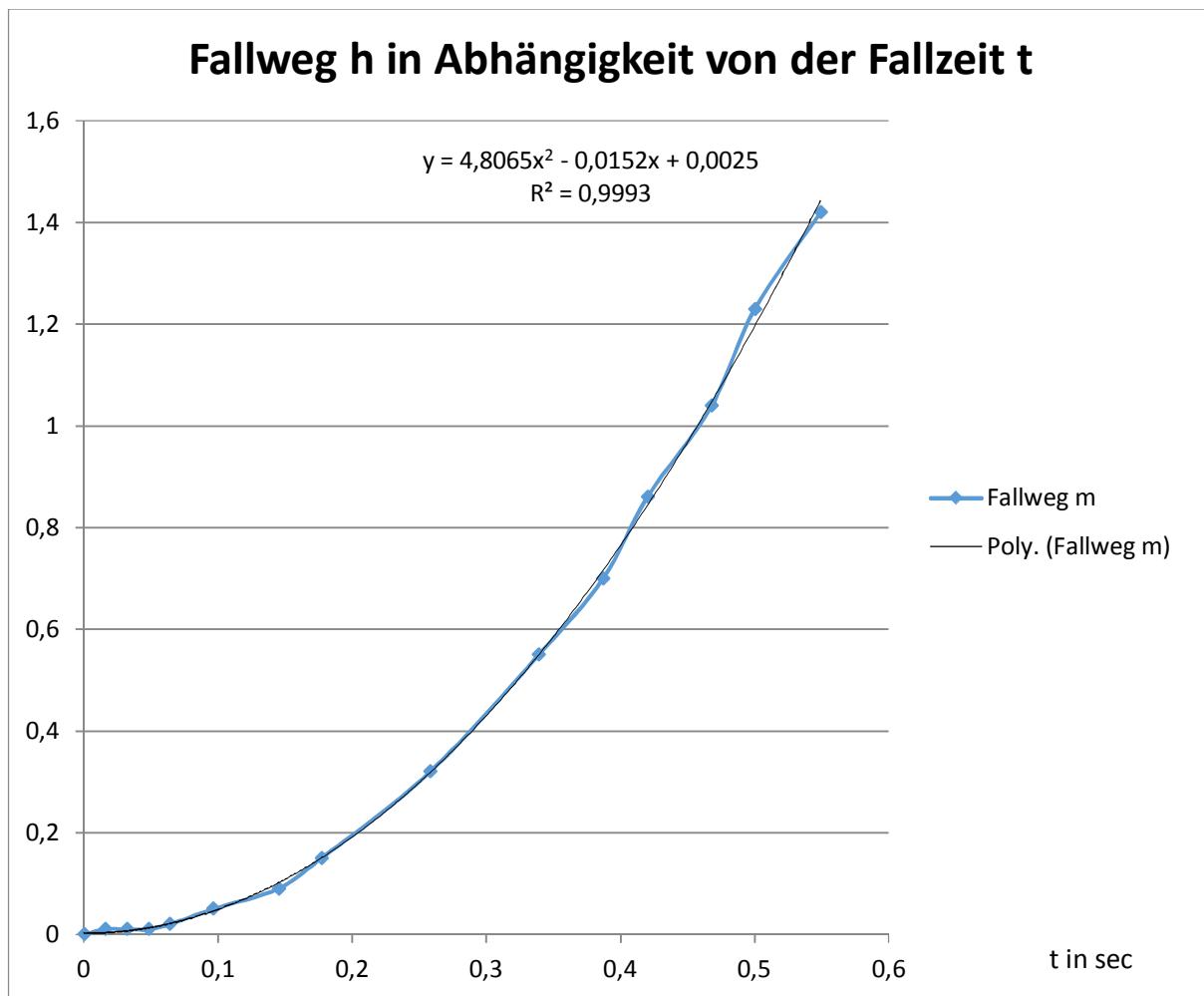


Abb. 4: Auswertung unter Berücksichtigung nur der jeweils ersten Messpunkte gleichen Abstands

Computing

startIDE (4): Focus Stacking

Rolf Meingast

Unter dem Begriff *Focus Stacking* versteht man Verfahren zur Erweiterung der Schärfentiefe beim Fotografieren durch Kombination mehrerer Bilder, die aus unterschiedlichen Gegenstandsfernungen entstanden sind. Dieser Beitrag beschäftigt sich mit dem Aufbau eines Makroschlittens und dem Programm zur Steuerung des ROBO Interfaces, das auf dem TXT oder dem TX-Pi mit der App startIDE [1] läuft.

Aufbau des Makroschlittens

Die Anforderungen an den Makroschlitten waren u. a.:

- stabile Geradführung
- kleinste Schrittweiten im μm -Bereich

Nach ersten Versuchen nur mit Bauteilen von fischertechnik habe ich zur exakten

Führung des Objekts bzw. der Kamera zunächst ein Mikroskopstativ [2] und dann den auf den ersten beiden Bildern zu sehenden Schlitten mit Mikrometer-Schraube verwendet (Abb. 1). Als Antrieb dient ein kleiner aus einem Disketten-Laufwerk [4] ausgebauter Schrittmotor, der mit ausreichendem Drehmoment am ROBO

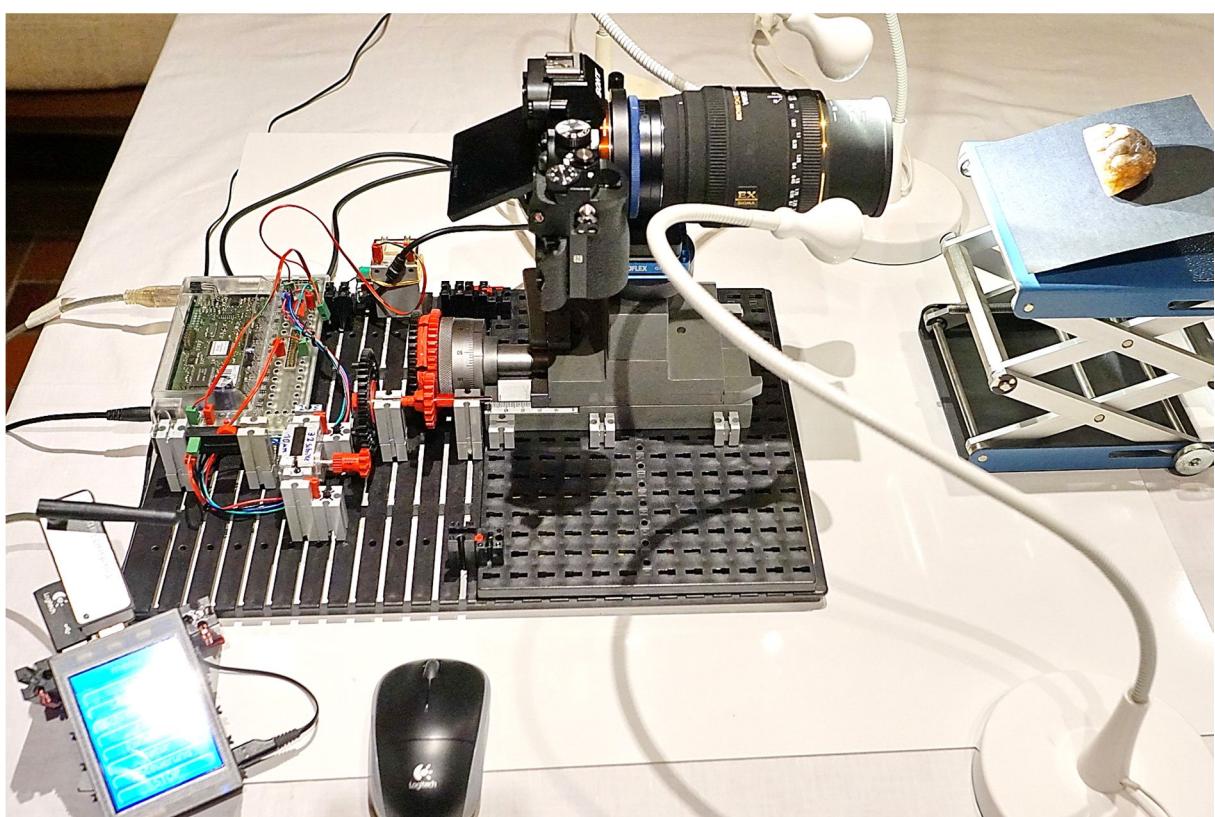


Abb. 1: Überblick über den Aufbau

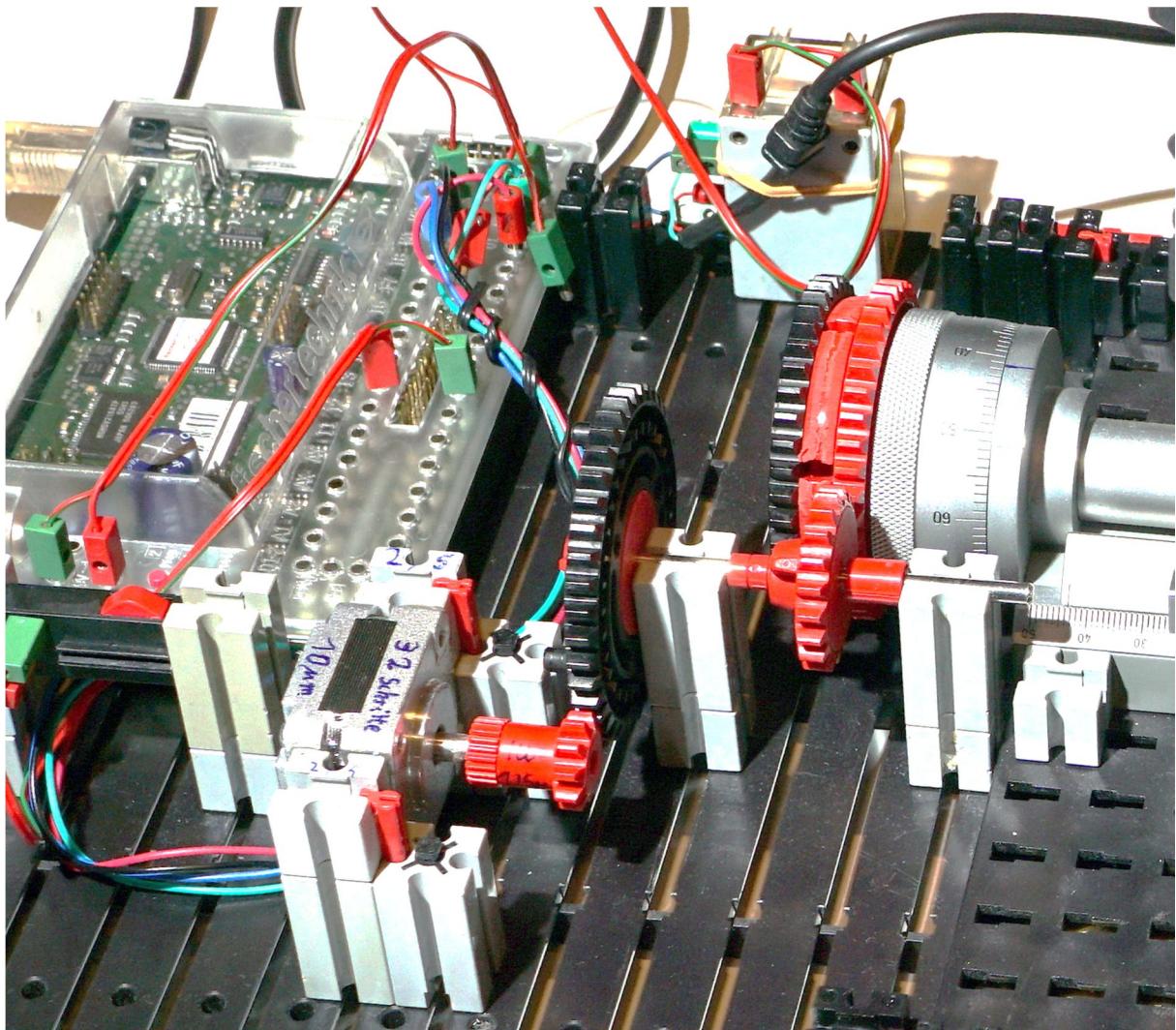


Abb. 2: Detailblick auf die Steuerung

Interface (M1 und M2) betrieben werden kann. 32 Halbschritte des Schrittmotors ergeben mit der gezeigten Zahnradkombination einen Vorschub von $10 \mu\text{m}$. Der Schlitten kann von Hand 50 mm weit bewegt werden und lässt sich alle 0,5 mm an die Mikrometerschraube ankoppeln.

Die Auslösung der Kamera erfolgt über ein fischertechnik-Relais, das an M3 angeschlossen ist.

Das ROBO Interface ist über USB-Kabel entweder mit einem TXT oder mit dem TX-Pi [3] verbunden. Letzterer kann komfortabel über eine Funkmaus bedient werden.

Steuerprogramm

Die Anforderungen an das Steuerprogramm lauten: Der Schlitten soll mit definierter Schrittweite zwischen den Bildern vorwärts oder rückwärts von Aufnahme zu Aufnahme bewegt werden. Diese Parameter, die Anzahl der Bilder und die Dauer der Pausen gegen Verwacklung werden vom Benutzer eingegeben oder bei der Initialisierung der Variablen vorgegeben.

Zur genauen Justierung der Startposition soll der Schlitten zurück oder vorwärts gesteuert werden können. Dies kann auch über die fischertechnik-IR-Steuerung erfolgen. Darüber gelangt man in eine Testroutine, mit der die für einen ruhigen Lauf

des Schrittmotors erforderliche Pausenlänge zwischen den Halbschritten ermittelt werden kann.

Programmiert wurde mit den Befehlen der startIDE.

Initialisierung der Variablen:

```
# Schrittmotor
```

```
Init Position 0
Init Richtung 1
Init Vorzeichen 1
Init Unterbrechung 4
Init Testschritte 512
Init schritte 32
Init kl-schritte 8
Init Bilder 10
Init summe 0
Init noch Bilder
Init Pause 3000
Init menu 0
Init ir 0
```

```
# Hauptmenue
```



```
Tag mainloop
Init ir 0
FromButtons menu Start Schritt_vor
Schritt_zurueck Foto Parameter
retour irSteuerung STOP
IfVar menu == 1 start
IfVar menu == 2 Schrittvor
```

```
IfVar menu == 3 Schrittzur
IfVar menu == 4 Foto
IfVar menu == 5 parameter
IfVar menu == 6 retour
IfVar menu == 7 AIR
Stop
Jump mainloop

# IR Fernbedienung

Tag AIR
Clear
Print Foto: M3r
Print Test: M3l
Print Abbruch: 1
Print oder Taster I3
Call Motor-aus 1
Tag AIR1
Init ir 0
FromRIIR ir
IfVar ir == 11 mainloop
IfVar ir == 8 Schrittvor
IfVar ir == 7 Schrittzur
IfVar ir == 1 Foto
IfVar ir == 2 test
IfIn RIF 3 S == 1 mainloop
Jump AIR1
```



```
# alle Schritte zurueck

Tag retour
IfVar summe == 0 mainloop
Clear
Calc Vorzeichen 0 - Richtung
```

```

Call SM summe
Calc summe 0 + 0
Call Motor-aus 1
Jump mainloop

# Schritt vorwärts

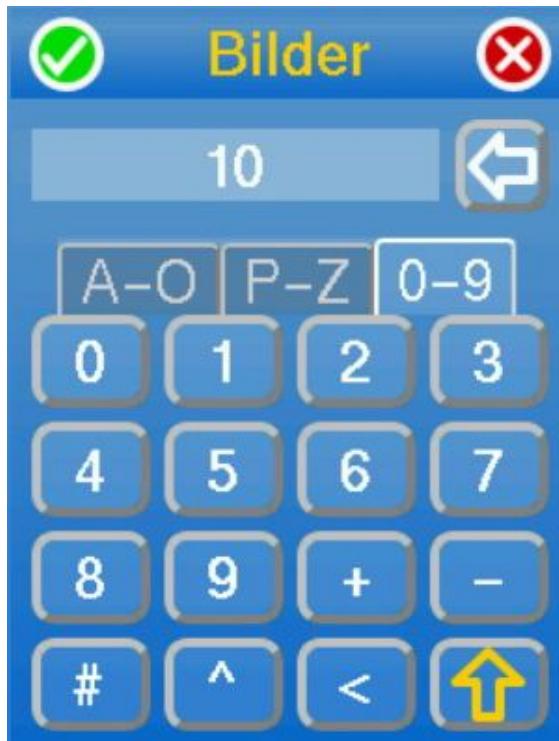
Tag Schrittvor
Calc Vorzeichen 1 + 0
Call SM kl-schritte
Call Motor-aus 1
IfVar ir != 0 AIR
Jump mainloop

# Schritt rückwärts

Tag Schrittzur
Calc Vorzeichen 0 - 1
Call SM kl-schritte 1
Call Motor-aus
IfVar ir != 0 AIR
Jump mainloop
Calc schritte 8 + 0
Call SM schritte
Jump mainloop

# Aufnahme

```



```

Tag Foto
Motor RIF 3 l 7
Delay 500
Motor RIF 3 s 0
IfVar ir != 0 AIR
Delay 1000

```

```

Jump mainloop
# Parameter
Tag parameter
# Bilderzahl:
FromKeypad Bilder 1 32768
# Schrittzahl:
FromKeypad schritte 0 32768
# Pausenlaenge:
FromKeypad Pause 0 32768
Jump mainloop
# stacking-Start
Tag start
IfVar schritte == 0 mainloop

```



```

# Richtungswahl
Tag neu1
Calc Richtung Vorzeichen + 0
FromDial Richtung -1 1 nur -1 oder
+1
IfVar Richtung == 0 neu1

```

```

# Zahl der restlichen Bilder           Jump mainloop
Calc noch Bilder + 0                 # Schrittfolge
Tag Wdh
Call Aufnahme 1
Calc noch noch - 1
Clear

# Anzeige der restlichen Bilder       # Berechnung des nächsten Schritts

QueryVar noch
Calc summe summe + schritte
Calc Vorzeichen Richtung + 0
Call SM schritte
TimerClear
Print Abbruch: I3

# Pause

Tag warten
IfIn RIF 3 S == 1 mainloop
IfTimer < Pause warten
IfVar noch > 0 Wdh
Call Aufnahme 1
Jump mainloop

# Testlauf

Tag test
FromDial Unterbrechung 0 20 msec
FromDial Testschritte 1 4096 Set
level
Call SM Testschritte
IfVar ir != 0 AIR1

```



```

Jump mainloop
Module SM
Calc Position Position +
Vorzeichen

Calc Position Position mod 8
IfVar Position == 0 S0
IfVar Position == 1 S1
IfVar Position == 2 S2
IfVar Position == 3 S3
IfVar Position == 4 S4
IfVar Position == 5 S5
IfVar Position == 6 S6
IfVar Position == 7 S7
Tag S0
Motor RIF 1 r 7
Motor RIF 2 l 7
Jump SM_END
Tag S1
Motor RIF 1 s 0
Motor RIF 2 l 7
Jump SM_END
Tag S2
Motor RIF 1 l 7
Motor RIF 2 l 7
Jump SM_END
Tag S3
Motor RIF 2 s 0
Motor RIF 1 l 7
Jump SM_END
Tag S4
Motor RIF 2 r 7
Motor RIF 1 l 7
Jump SM_END
Tag S5
Motor RIF 1 s 0
Motor RIF 2 r 7
Jump SM_END
Tag S6
Motor RIF 1 r 7
Motor RIF 2 r 7
Jump SM_END
Tag S7
Motor RIF 1 r 7
Motor RIF 2 s 0
Jump SM_END
Tag SM_END
TimerClear
Tag Z
IfTimer < Unterbrechung Z
MEnd

```

```
# Strom aus
Module Motor-aus
Motor RIF 1 s 0
Motor RIF 2 s 0
MEnd

# Einzelbild
Module Aufnahme
# Kameraauslösung
Motor RIF 3 1 7
Delay 500
Motor RIF 3 s 0

# Belichtungszeit maximal 1 sec
Delay 1000
MEnd
```

Listing 1

Eine Achatdruse [5] ($51 \text{ mm} \times 35 \text{ mm}$) wurde mit dieser Vorrichtung im Maßstab 2:3 mit einer Schrittweite von $150 \mu\text{m}$ 164-mal fotografiert. Die nachfolgenden Abbildungen zeigen das Ergebnis.

Referenzen

- [1] Peter Habermehl: [CFW: startIDE v1.0, Modelle direkt auf dem TXT programmieren](#) im ftc-Forum, 2018.
- [2] Rolf Meingast: [ROBO + TX-Pi mit startIDE im Einsatz](#) im ftc-Forum, 2018.
- [3] Till Harbaum: *tx-pi* auf GitHub: <https://github.com/harbaum/tx-pi>
- [4] Wikipedia: [Diskette](#).
- [5] Wikipedia: [Achat](#)

*Abb. 3: Einzelbild*



Abb. 4: Ergebnis



Abb. 5: Ausschnitt