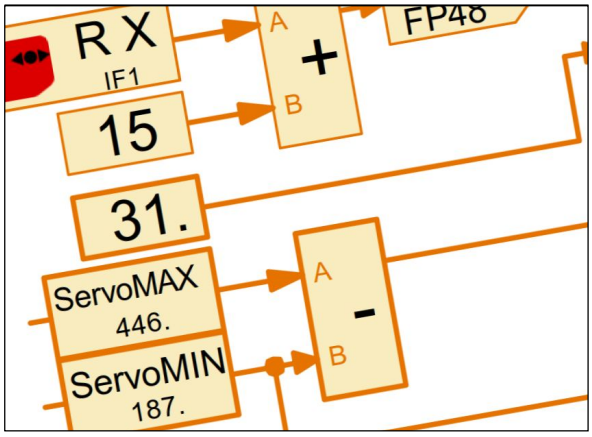
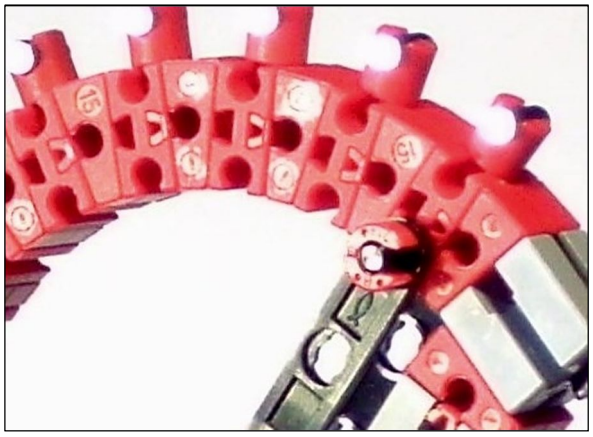
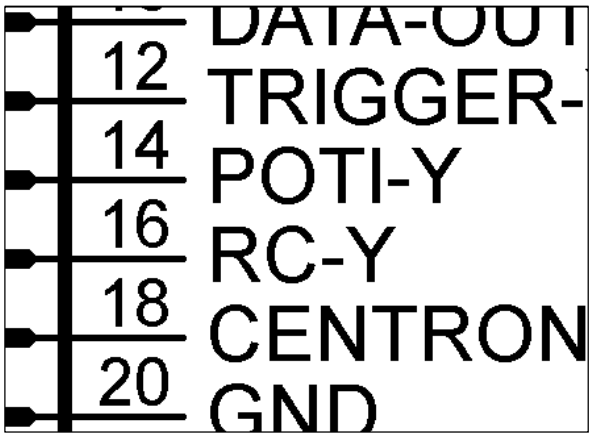
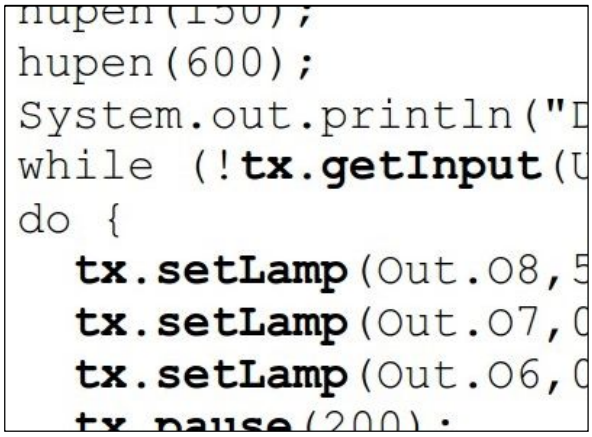
ft:pedia

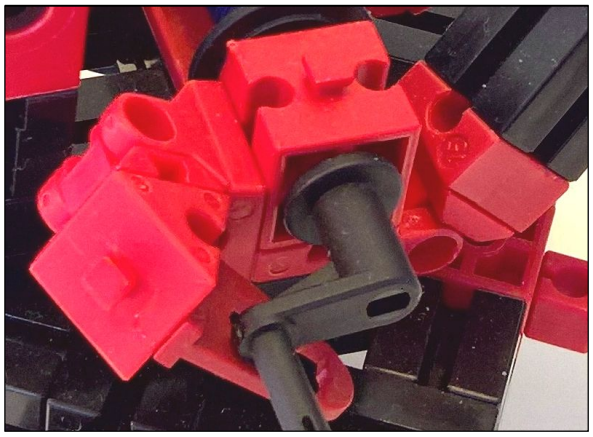
Heft 2/2017











Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

Learning by Doing

Dirk Fox, Stefan Falk

Seit Jahren wird in Deutschland diskutiert, ob Informatik Pflichtfach werden soll. Die Gesellschaft für Informatik e. V., in dieser Frage naturgemäß nicht gänzlich unvoreingenommen, hat bereits 2008 Grundsätze und Standards für den Informatikunterricht in der Sekundarstufe I und 2016 Standards für die Sekundarstufe II definiert.

Ohne Frage: Informatik wird immer wichtiger und ist aus vielen Berufen nicht mehr wegzudenken: Biologen programmieren Analysegeräte, Maschinenbauer entwickeln Systemsteuerungen und Designer codieren Webseiten. Damit gehören die Grundlagen der noch jungen Informatik sicherlich inzwischen zu dem, was wir gerne als „Allgemeinbildung“ bezeichnen.

Aber geht der Vorstoß in die richtige Richtung? In Sekundarstufe I sollen Algorithmen, Datentypen und formale Sprachen vermittelt werden; später sollen die Schüler sich mit Verschlüsselung, HTML-Seiten und E-Mail-Protokollen beschäftigen. Und in den Ministerien wird von App-Programmierung und Simulationen geschwärmt. Hm. Ob das wohl die Themen sind, mit denen sich pubertierende Jugendliche für Informationstechnik begeistern lassen?

Keine Frage, die Vermittlung von Grundlagen ist wichtig – wie in Mathe und Physik. Motivierend ist sie allerdings in der Regel nicht. In der Physik sollte das Experiment zur Formel führen, und erst wenn eine Berechnung eine interessante Frage löst, vermag dies für Mathematik zu begeistern. Die Informatik benötigt daher die Anwendung, denn wie Mathematik und Physik ist sie für die meisten Menschen – von Forschern einmal abgesehen – in erster Linie ein Werkzeug. Ein Werkzeug aber erlernt man am besten durch Benutzung: Den Hammer versteht man nicht über die Erläuterung seiner Komponenten, sondern indem man damit Nägel einschlägt. Daher überlässt man die Erklärung des Hammers auch besser nicht Werkzeugexperten, sondern zieht lieber einen Zimmermann hinzu.

Wer für Informatik begeistern will, sollte Schülerinnen und Schüler so früh wie möglich – also bereits zu Grundschulzeiten – befähigen, ihre fischertechnik-Modelle anzusteuern, seien es nun Kugelbahnen oder später Roboterarme, autonome Fahrzeuge, Drohnen oder 3D-Drucker. Sind die ersten Schritte mit ROBO Pro getan, dann ist es zu einer höheren Programmiersprache nicht mehr weit: Der Hunger danach kommt von selbst. Und die Grundlagen der Informatik sind dann später kein theoretisches Übel mehr, sondern ergänzen das bereits erworbene Praxiswissen.

Daher könnte neben einem Brickly-TXT auch der neue Bluetooth-Controller von fischertechnik zur Einstiegsdroge werden. Wer einmal beobachtet hat, wie leichtfüßig und hartnäckig Kinder – einmal infiziert – das für die Lösung ihrer selbst gewählten Problemstellung benötigte Wissen geradezu aufsaugen, wird sich keine Sorgen mehr um deren (und unser aller) Zukunft machen.

Beste Grüße,
Euer ft:pedia-Team

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

Inhalt

Termine

Was?	Wann?	Wo?
Karlsruher Schul-Robotik-Cup 2017	01.07.2017	Bismarck-Gymnasium
Fan-Club-Tag 4.0	09.-10.09.2017	virtuell
Maker Faire Bergstraße	16.-17.09.2017	Bensheim
Süd-Convention 2017	23.09.2017	Dreieich
Clubdag in Schoonhoven	28.10.2017	Schoonhoven (NL)

Hinweise

Andreas Kempf (kdu@gmx.de) bietet an, weitere fischertechnik-Soccer-Teams für den RoboCup Junior 2018 zu coachen.

Impressum

http://www.ftcommunity.de/ftpedia

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

Autoren: Paul Bataille, Daniel Canonica, Stefan Falk, Dirk Fox, Johann Fox, Helmut Jawutsch, Peter Krijnen, Rüdiger Riedel, Torsten Stuehn, René Trapp, Dirk Uffmann, Dirk Wölffel.

Tipps & Tricks

Fotografieren von Modellen: Klar denken und klar bauen machen es leichter

Paul Bataille

Wie fotografiert man seine Modelle so, dass man damit sein Ziel erreicht? So, dass die Begeisterung, die man über ein gelungenes Modell empfindet, beim Betrachter ankommt? Oder dass man verdeutlichen kann, wie man ein Konstruktionsproblem gelöst hat? Oder so, dass die Bilder sogar als eine richtige Bauanleitung dienen können? Mit ein wenig Aufmerksamkeit ist das gar nicht so schwierig. Es hilft natürlich, wenn das Modell gut gebaut worden ist.

Der Zweck, zu dem man ein Bild oder eine Bilderserie macht, bestimmt, wie man das Fotografieren angehen sollte. Manchmal spielen mehrere oder sogar alle oben gerade genannten Ziele eine Rolle. Betrachten wir einige davon.

Zum Zeigen: Halt es einfach

Ein Totalbild von links, rechts, vorn, hinten, oben (wenn das sinnvoll ist) und unten – das war's. Aber das ist nur der erste, minimal notwendige Schritt. Das Ergebnis ist meistens nicht gerade spannend, aber macht oft schon Spaß. Dazu noch einige Schrägbilder, die die Einheit und allgemeine Form des Modells viel besser zeigen (Abb. 1).

Abb. 1: Ein Schrägbild zeigt die Form des Modells oft am besten

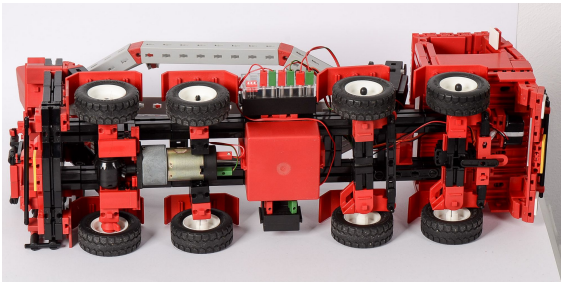


Abb. 2: Vergiss‘ auch die Unterseite nicht, die manchmal zeigt, wie das Modell konstruiert ist

Zur Begeisterung: Schönheit und Essenz

Um für das Modell zu begeistern, muss es gut aussehen. Es soll vielleicht sogar ein bisschen verführen. Dafür sucht man eine Perspektive, die die Stärke des Modells gut herauskommen lässt. Extreme wirken manchmal gut, z. B. höher oder niedriger als gewöhnlich. Ein Halbprofil oder Schrägbild ist manchmal schöner als ein Bild von vorn oder von der Seite. Stelle das Modell auf einen Tisch, gehe rundherum und entscheide, was gut aussieht und wirken wird. Geh‘ mal in die Knie, steige auf einen Stuhl, geh‘ näher heran oder etwas weiter weg. Spiele mit den Möglichkeiten und entdecke, aus welcher Perspektive auffällige Details im Design gleich herausspringen.

Was wirkt, hängt auch vom Wesen des Modells ab. Einige Beispiele.

Radfahrzeuge

Die baue ich am häufigsten. Radfahrzeuge drehen sich um Räder und Reifen, um Fahren und Lenken. Denke an Fotos aus der Autowerbung: Die werden oft von einem sehr niedrigen Standpunkt geschossen. Immer mit leicht eingedrehten Vorderrädern, was Dynamik andeutet. Manchmal mit der übertriebenen Perspektive eines Weitwinkelobjektivs. Wenn Türen, Haube oder (Heck-) Klappen aufgemacht werden können, zeige sie im offenen und geschlossenen Zustand. Wenn das Fahrzeug dem Maßstab der fischertechnik-Männchen entspricht, dann lass‘ die mitmachen: hinter dem Lenkrad, beim Einsteigen oder bei irgendeiner anderen Tätigkeit.

Außer um Räder dreht es sich bei Radfahrzeugen oft entweder um Kraft oder um Geschwindigkeit. Lass‘ es im ersten Fall: groß und mächtig aussehen, aus einer niedrigen Perspektive. Zeige das Fahrzeug in Aktion. Im zweiten Fall solltest du die Aerodynamik betonen, z. B. das Linienspiel, indem du knapp an der Seite entlang fotografierst (Abb. 3).

Abb. 3: Eine klassische Pose für einen PKW: von einem niedrigen Kamerastandpunkt mit leicht eingeschlagenen Rädern und ziemlich knapp an der Seite entlang, um Form und Aerodynamik zu betonen

Wenn Bewegung essentiell ist

Zeige die Mechanik, oder noch besser: zeige die Bewegung selbst. Letzteres gelingt sehr einfach mit einer etwas längeren Verschlusszeit, ab ungefähr 1/30 Sekunde, abhängig von der Geschwindigkeit der Bewegung.

Dafür braucht man entweder eine ruhige Hand oder ein Stativ. Man kann das Modell auch in Arbeitsposition zeigen, z. B. einen Kran mit Last im eingedrehten Zustand, oder noch besser: Zeige den Moment, bevor die Last ergriffen wird.

Abb. 4: Ein Aktionsmodell zeigt man am besten auch gerade bei der Arbeit

Licht

Wenn es um Licht geht, mache dein Foto im Dunkeln oder im Halbdunkel. Licht und Bewegung zusammen, wie z. B. bei Kirmesmodellen, ist natürlich ganz ausgezeichnet, dann bekommt man farbige Streifen oder Flecken und Ähnliches.

Einige gute Tipps zur Ausleuchtung des Modell finden sich auch in [1].

Video

Vergiss‘ nicht die Möglichkeiten eines Videos. Das ist natürlich für Bewegung am besten geeignet.

Zur Verdeutlichung und Erinnerung

Um die Konstruktion eines Modells zu verdeutlichen oder zu dokumentieren, um sich später erinnern zu können, wie es konstruiert ist, sollte man Details fotografieren. Das heißt: dicht heran (aufpassen mit dem Blitz, aber darüber später mehr), nur zeigen, was relevant ist, oder, wenn das nicht möglich ist, später das Bild nachbearbeiten. Ist etwas nicht gut zu sehen, dann muss das Modell vielleicht teilweise demontiert werden. Oder man legt es auf die Seite oder auf den Kopf, vielleicht mit Stützen, oder man bittet jemanden, es gut ins Bild zu halten.

Abb. 5a

Abb. 5b

Abb. 5c: Das Modell kann in einigen charakteristischen Arbeitspositionen gezeigt werden

Man kann auch die Stufen eines Bewegungsvorgangs in einer Bilderserie zeigen oder alle mögliche Positionen eines Modells fotografieren (Abb. 5a-5c).

Während des Bauens eines neuen Modells muss es normalerweise mehrere Male mehr oder weniger vollständig auseinander gebaut werden. Wenn ich vorhabe, etwas zu ändern, und noch nicht genau voraussehen kann, ob die neue Idee klappen wird oder nicht, fotografiere ich schnell den aktuellen Zwischenstand, um später wieder denselben Zustand nachbauen zu können, falls sich das als notwendig erweisen sollte. Selbstverständlich muss ein solches Bild nicht vor allem schön sein. Es reicht, wenn es erkennen lässt, wie das Modell konstruiert wurde.

Zum Nachbauen: beim Zerlegen

Prinzip

Wenn man ein Modell fotografiert, um anderen (oder sich selbst) zu helfen, es nachzubauen, fotografiert man es am bequemsten beim Zerlegen. Wenn man die Bilder später in umgekehrter Reihenfolge betrachtet, hat man seine Bauanleitung. So einfach ist das. Denn wenn man das Modell zum Fotografieren erneut aufbauen muss, ist die Chance, dass man Fehler macht und mehrmals zurück zur vorigen Baustufe muss, ohnehin viel größer.

Mit „Zerlegen“ meine ich hier nicht, dass alle Steine bis auf Einzelteile voneinander

getrennt werden. Wie weit man beim Zerlegen gehen sollte, hängt davon ab, welchen Baumeister man sich vorstellt: Ist er ein erfahrener fischertechniker oder ein Anfänger, den man motivieren will, einmal ein etwas komplizierteres Modell zu bauen? Wenn es beide sind, richte man sich nach dem Anfänger. Der Erfahrene kann schneller durchgehen, aber wenn der Anfänger es nicht versteht, dann kommt er irgendwann überhaupt nicht mehr weiter…

Im Großen und Ganzen ist meine Methode die folgende:

1. Fotografiere das fertige Modell von allen Seiten, so dass man weiß, was man baut;
2. Demontiere ein logisches Teil, entweder als Einzelteil oder als kleine Gruppe von Bauteilen;
3. Erstelle ein Bild des entfernten Teils;
4. Erstelle ein Bild des verbleibenden Modells;
5. Wiederhole die Schritte 2 bis 4, bis etwas übrig geblieben ist, woraus ohne Weiteres deutlich wird, wie es nachgebaut werden kann.

Wo nötig erstellt man zusätzliche Bilder.

Zu beachten

Rechne damit, dass die Bilder später in umgekehrter Reihenfolge betrachtet werden. Daher ist Folgendes wichtig: Das, was vom Modell übrig ist, nachdem ein Bauteil oder eine Bauteilgruppe abmontiert wurde, muss als *Ergebnis des Anbauens* der Teile fotografiert werden, die erst bei dem nächsten Demontier-Schritt entfernt werden – und nicht als „Restprodukt“ des gerade durchgeführten letzten Demontier-Schritts! Da irre ich mich manchmal, und dann muss der letzte Schritt wiederholt werden. Deswegen erstelle ich eher ein paar Bilder zu viel als eines zu wenig. Bilder nicht zu verwenden oder wegzuwerfen ist viel einfacher als das Modell nochmal teilweise demontiert fotografieren zu müssen, vor allem, nachdem es bereits völlig zerlegt oder fertig aufgebaut worden ist.

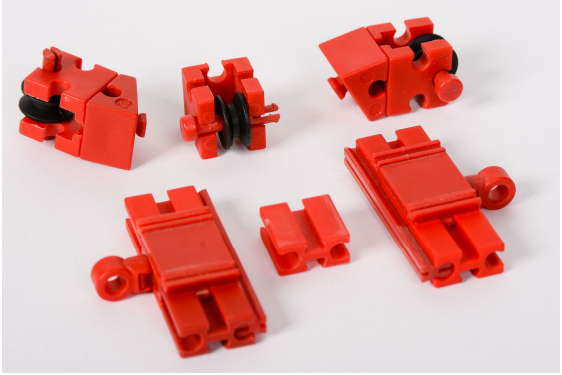


Abb. 6a

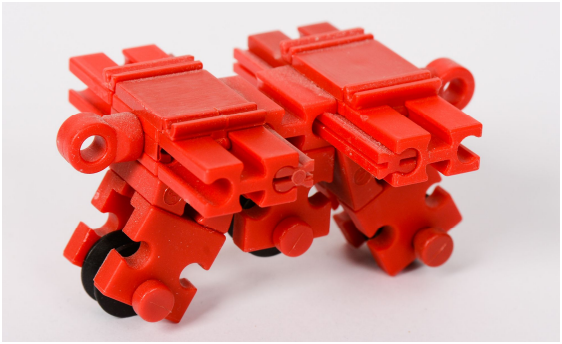


Abb. 6b

Abb. 6c

Abb. 6d: Minibauanleitung für eine Laufkatze

Beim Fotografieren von bereits entfernten Teilen sollte man am besten eine Perspektive wählen, bei der deutlich wird, wo und wie diese befestigt werden sollen. Zum Beispiel in einer Position, die zu der nächsten vorgesehenen Phase des Bauprozesses passt.

Wenn ein Modell gut modular aufgebaut ist, kann man die einzelnen Module auf diese Weise festlegen, nachdem man fotografiert hat, wie es zusammengebaut wurde. Zuletzt sind dann Detailaufnahmen von den Verbindungen der Module im gesamten Modell erforderlich, die man besser am Anfang des ganzen Foto-Prozesses macht.

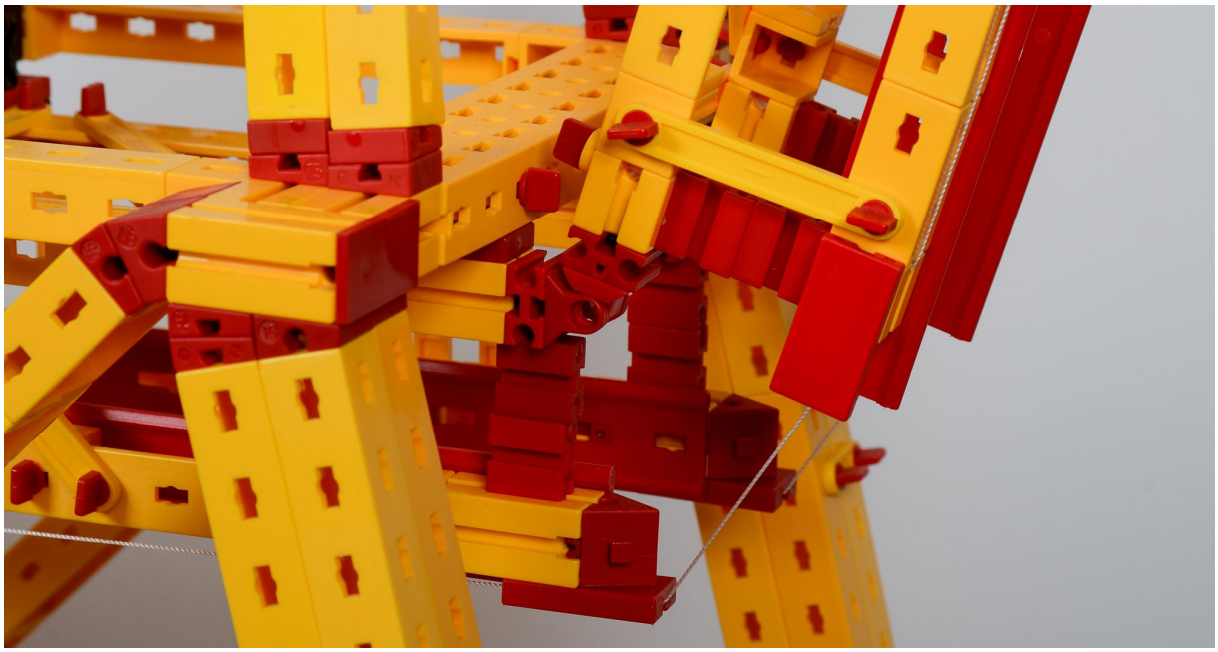


Abb. 7: Bevor das Modell in Module getrennt wird, sollte man fotografieren, wie die Module verbunden worden sind, so wie hier, wo die Verbindung zwischen Ausleger und Basis eines Containerkrans gezeigt wird. Wie das Seil beim Heben und Senken des Auslegers auf dem rechten Platz gehalten wird, wird gleich mit gezeigt

Große Modelle

Bei sehr großen Modellen, die man längere Zeit aufbewahren will, ist demontieren kaum möglich oder erwünscht. Dann muss man eben improvisieren: Zum Beispiel kurzzeitig doch einige Module demontieren, um etwas zeigen zu können und etwas mehr Detailaufnahmen herzustellen.

Besser noch ist es, während des Bauens damit zu rechnen, dass man das Modell später fotografiert haben will. Wenn dann ein Modul fertig ist, das später wahrscheinlich nicht mehr gut zu sehen ist, dieses gleich in Bildern sorgfältig festhalten, solange man noch alles gut sehen kann oder noch bequem das eine oder andere zeigen kann, ohne vieles zerlegen zu müssen.

Gleich wieder zusammenbauen

Ich zerlege meine Modelle oft gleich nachdem ich sie fertig gebaut habe, nur mit dem Ziel, sie zu fotografieren. Oft bin ich sogar zu ungeduldig: Ich ändere später noch das eine oder andere am Modell und muss das alles dann später noch ein zweites Mal machen. Enthusiasmus kann eine Last sein…

Nach dem Zerlegen und Fotografieren baue ich die Module gleich wieder zusammen, noch während der Arbeit des Fotografierens. Das erneute Zusammenbauen des gesamten Modells ist auf diese Weise nach dem Foto-Prozess oft in einigen Minuten geschehen. Umso mehr, als es gerade erst gebaut worden ist und man noch exakt

weiß, wie die Teile ineinander zu stecken sind.

Klarer Bau – ein Credo

Es hilft natürlich, wenn das Modell klar gebaut worden ist. Das macht alles einfacher. Was „klar bauen“ heißen soll, ist allerdings nicht so eindeutig. Ich meine, jeder stellt sich selber anderen Herausforderungen beim Bauen und hat Freude an anderen Dingen. Für mich heißt es vielleicht etwa dieses (ihr bemerkt, dass ich hier behutsam formuliere):

- dass das Modell einen eindeutigen Rahmen hat;
- dass es logisch ineinander steckt (was damit auch immer genau gemeint sein soll);
- dass es meistens im fischertechnik-Raster liegt;
- dass Lagen mit „Verschieberisiko“ gut gesichert sind;
- dass Teile nicht unnötig mit anderen verbunden sind;
- dass das Modell tatsächlich symmetrisch ist, wo es symmetrisch sein soll;
- dass eine Verkleidung nur Verkleidung ist und nicht in den Rahmen oder in Funktionen eingreift.

Natürlich will man solche „Regeln“ gerne in den Müll schmeißen, wenn ein großer genialer Plan entsteht oder wenn man um jeden Preis eine inspirierende Idee realisieren will, besonders im Kleinen. Und Regeln gibt es in normalen Leben schon genug – beim Hobby sollten sie keine Rolle spielen. Gut, klar. Aber wenn man die „Regeln“ ignoriert und später etwas am Modell ändern möchte oder sogar muss (wegen irgendeines Problems), stürzt möglicherweise alles auf einmal ein und man kann komplett von vorne beginnen.

Was man übrigens nicht schlimm finden muss und was sogar Spaß machen kann. Das ist die andere Seite der Geschichte. So, wie es Spaß machen kann, eine Funktion um jeden Preis einzubauen, wenn das eigentlich gar nicht möglich ist, und dabei dann alles zu tun, was verboten ist. Das letzte mögen wir alle manchmal, oder?

Integrität des Rahmen

Jedenfalls tendiere ich selber dazu, dafür zu sorgen, dass zum Beispiel ein Antriebsblock, ein Lenksystem oder ein Dach mehr oder weniger als zusammenhängendes Teil entfernt werden kann. Und dass Antriebswellen, Kabel und Seile in dieser Hinsicht klug geführt worden sind. Dass die gesamte Verkleidung gewechselt werden kann, ohne dass an Funktion oder Rahmenwerk etwas Fundamentales geändert werden muss.

Wie das zu erreichen ist, wäre einen eigenen Beitrag wert. Es hat etwas damit zu tun, zuerst auf möglichst einfache Weise den Rahmen und die Funktionen unabhängig voneinander aufzubauen. Und mit einer prinzipiellen Weigerung, die Integrität des Rahmens zu kompromittieren, um irgendwelche Funktion realisieren zu können. Und mit der Selbstverpflichtung, das Problem auf eine andere Weise zu lösen. Und wenn nötig, d. h. wenn man einfach keine Lösung findet, bereit zu sein, die Idee einfach aufzugeben.

Sehr oft entsteht gerade dann der Raum für eine neue, vielleicht sogar viel bessere Idee. Eine Idee aufgeben tut weh, aber es ist – wie im Leben – manchmal die einzige und damit die beste Lösung. „Schaffen“ bedeutet manchmal auch abziehen und abkratzen, so zu sagen „ins eigene Fleisch schneiden“, und entdecken, was darunter an Essentiellem verborgen ist.

Getrennt, nicht geschieden

Erst wenn Rahmen und Funktionen (des ganzen Modells oder eines Moduls davon) realisiert worden sind, wird verkleidet, dekoriert und garniert, wie in der Küche oder beim Hausbau. Obwohl manchmal beim ersten das zweite schon „mit einen schiefen Auge“ betrachtet werden soll, wie wir in Holland sagen. Und man sollte Schmuckelemente, die man unbedingt anbauen will, besser schon früh ausprobieren, um später keine großen Konstruktionsänderungen vornehmen zu müssen.

Manche Äußerlichkeiten können essenziell sein für den Gesamtanblick des Modells, für dessen Erkennbarkeit und damit für die Möglichkeit, ein Aha-Erlebnis beim Zuschauer zu erzeugen. Gerade darum kann es auch anders herum wirken: Manchmal – aber seltener – merkt man beim Verkleiden, dass die ganze Idee des Rahmens oder eine oder mehrere der Funktionen geändert werden müssen.

Alles beeinflusst alles andere, im Modellbau wie im Leben. Die gegenseitige Einstellung der Dinge ist gerade die Essenz, vielleicht mehr als die Dinge selber, das ist zumindest meine Erfahrung. Mit dieser gegenseitigen Beeinflussung aller Aspekte des Modells sind wir in den Grenzbereich von Technik, Handwerk, Kreativität und Ästhetik (oder vielleicht sogar Kunst) geraten. Es könnte sein, dass es gerade das ist, was mich an fischertechnik fasziniert.

Referenzen

[1] Thomas Püttmann, *Modellfotografie*. ft:pedia 1/2016, S. 21-23.

Abb. 8: Formel-1-Bolide

Modellideen

Glücklich

Peter Krijnen

Vor wenigen Wochen hat mich jemand gefragt, ob ich glaubte, nochmals glücklich sein zu können. Da habe ich ihr gesagt: Glücklich wäre ich nur in meiner eigenen Welt.

In meiner Welt bin ich der Präsident. Der König, der Heizer und, obwohl es in meiner Welt keine Marine gibt, auch der Admiral. Ich bin auch der Boss aller meiner Firmen. Aber ich bin ein mitarbeitender Boss. Meine Mitarbeiter und ich erdenken, entwickeln und konstruieren Sachen, von denen wir glauben, dass sie für die Leute in meiner Welt von Vorteil sind. Braucht der Bauer einen neuen Traktor, liefern wir ihm einen neuen Traktor. Den werde ich ihm mit einem von meinen Firmen gebauten LKW mit Satteltiefanhänger liefern.

Im Frühling muss gesät werden. Aber zuvor muss der Boden noch mit dem Wendepflug bearbeitet werden. Dabei wird auch gleich Dünger in den Boden eingebracht. Im Sommer muss das Getreide gemäht werden. Dazu braucht der Bauer einen Mähdrescher oder einen Rübenroder. Später benutzt er auch die Heuballenpresse, um das Heu einfacher in der Scheune zu lagern.

In meinen Firmen werden allerlei Geräte, Maschinen und Gebäude entwickelt und gebaut. Zum Transport all dieser Güter und Materialien braucht man, neben LKW, Schiff und Eisenbahn, auch die dazu gehörenden Wege. Flüsse müssen ausgebaggert werden, damit die Schiffe weiter fahren können. Dazu benutzt man Löffelbagger auf einem Ponton. Für PKW und Eisenbahn bauen wir, neben Straßen und Bahndämmen, auch die Brücken. Da das doch erheblich große Konstruktionen sind, kann man das am besten in Modulbauweise realisieren. Wir haben deswegen ein Spezialgerüst konstruiert, mit dem die Module gehoben und platziert werden können. Zur Unterstützung aller Arbeiten wurden auch mehrere große Krane eingesetzt. Wir haben dazu eine große Auswahl von Kranen mit Raupen oder Rädern, die alle Arten von Lasten heben können. Betonmischer bringen den Beton von der Mühle zum Bauplatz.

Abb. 1: Betonmischer von Paul Bataille

In den Niederlanden brauchen wir immer mehr Platz, um zu bauen. Schon Jahrhunderte lang erobern wir Land vom Meer. ‚Einpoldern' oder ‚eindeichen' nennen wir das. Dazu werden Hoppersauger benutzt, die den Sand vom Meeresboden aufsaugen und vor der Küste wieder ausspucken.

Um all diese Maschinen bauen zu können, braucht man sehr viel Stahl. Das Erz kommt aus Skandinavien oder Australien. In Schweden wird, unter Zuhilfenahme von

Spezial-Radladern, das Erz unter Tage abgebaut. In Australien wird das Erz hingegen mit Löffelbaggern auf Roadtrains geladen.

Seit einer Weile wohne ich in einem neuen Haus. Ein altes Schloss, das ich restaurieren ließ. Für die Arbeiten wurde ein Faltkran benutzt. Die Bäume für das Holz wurden durch Forwarder aus dem Wald befördert und im Bahnhof auf Eisenbahnwaggons verladen, zum Transport in eine Sägemühle. Abfall wird nicht einfach entsorgt. Nein, das wird mit speziell einwickelten LKW mit Containern erledigt.

Abb. 2: Langholzwagen von Magnus und Dirk Fox

Um all das möglich zu machen, braucht man Energie. Alles soll elektrisch betrieben werden. Dazu braucht man natürlich Kraftwerke. Die kann man mit Öl, Kohle oder Braunkohle befeuern. Um die Braunkohle abzubauen benutzt man sehr große Schaufelradbagger, die das dann auch gleich mit 240.000 Kubikmeter am Tag tun. Öl oder Gas pumpt man am besten durch ein Rohr. Wir haben für das Verlegen der Rohre einen Raupenfahrzeug mit einem seitlich angesetzten Kran entwickelt. Aber auch Windkraftanlagen und Solaranlagen werden wir vermehrt aufbauen.

Ich bin auch ein weltbekannter Rennfahrer. 20 Mal im Jahr reise ich für ein Wochenende um die ganze Welt und rase in meinem roten Formel-1-Rennwagen zum Sieg.

Im Winter bin ich aber Skifahrer. Mit der Seilbahn fahre ich zur Spitze, von wo ich auf meinen Schiern die mit einem Pistenbully präparierte Piste herunterjage.

Abb. 3: Pistenbully von Martin Westphal

Nach der Arbeit darf entspannt werden. Dazu schaue ich mir im Fernsehen am liebsten die US-Krimis an. Das mache ich, weil ich selber die Hauptrollen spiele. In Miami spiele ich CSI Horatio Caine, in Washington bin ich der Special Agent Anthony DiNozzo Junior und in New York spiele ich den „Mystery writer“ Richard Castle. In Las Vegas bin ich CSI Gil Grissom und in Los Angeles LAPD Detektiv Marty Deeks. In Hollywood aber bin ich der Kapitän der Enterprise und fliege mit meiner Besatzung zum Planeten Tatooine, wo ich als Han Solo auf den Millennium Falcon umsteige und zum Planeten Abydos fliege, wo ich als Kolonel Jack O’Neill mit der Stargate wieder zur Erde transportiert werde.

Abb. 4: Enterprise von Harald Steinhaus

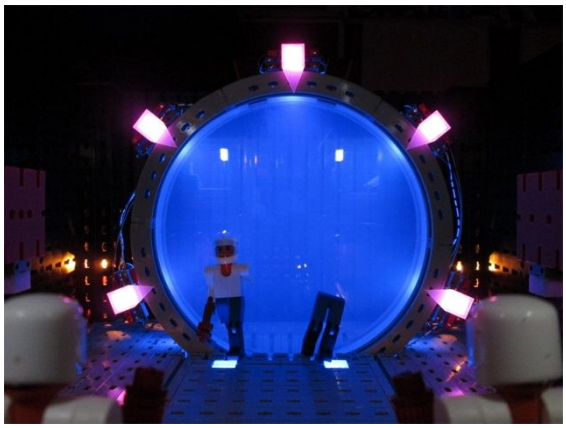


Abb. 5: Stargate von Jonathan Baur

Zurück auf der Erde gehe ich zum nächst gelegenen Bahnhof, wo ich auf meine schon bereitstehende Lokomotive steige – zur Fahrt nach New York, wo ich auf ein Flugzeug umsteige und nach Hause fliege. In Amsterdam werde ich am Flughafen von einem Bekannten abgeholt, der mich mit seinem Landrover nach Hause bringt.

Erst zu Hause merke ich, wie müde ich bin. Deswegen lege ich mich ins Bett. Nach einer Viertelstunde schlafe ich ein und träume von Autos, die vom Weg abkommen und gegen Bäume fahren, und von Kunden, die nicht zufrieden sind. Und von Stress und wie übel ich mich deswegen fühle. Als ich am nächsten Morgen aufwache, ist die Welt aber wieder heil.

Abb. 7: Bett von Evert Hardendood

Wo ist meine Welt? Meine Welt beschreibt einen elliptischen Kreis um die Erde: Tagsüber auf 1,8 m und in der Nacht auf 0,7 m über der Erdoberfläche.

Wie der Name meiner Welt ist? Das ist doch klar: fischerwelt.

Alle Links verweisen auf Bilder von www.ftcommunity.de sowie von www.fischertechnikclub.nl.

Abb. 6: Lokomotive von Peter Krijnen

Baukasten

Die Staubschutz-Stückliste für den Urlaubskasten

Stefan Falk

Im Wohnzimmer-Dienstreisen-Urlaubs-Notfallkasten aus der ft:pedia 1/2016 [1] wird eine Bauplatte 500 als Deckel benutzt. Durch deren Nuten fällt bei längerem Herumstehen gerne Staub ins Innere des Kastens. Im Download-Bereich der ft:community gibt es deshalb ein Staubschutz-Einlegeblatt, das gleichzeitig einen Sortierplan mit Stückliste darstellt und einige Modellanregungen zeigt.

Der Download [2] enthält eine zweiseitige PDF-Datei zum beidseitigen Ausdrucken auf ein einziges Blatt. Für eigene Veränderungen ist auch das originale Microsoft Word-Dokument im Download enthalten. Auf der Rückseite der Stückliste sind neben Beispiel-Modellen auch zwei vielleicht nützliche 25-cm-Lineale angebracht. Deshalb enthält eine ebenfalls im Download enthaltene Fassung des Word-Dokuments das VBA-Makro, mit dem diese Lineale erzeugt wurden – für Interessierte.

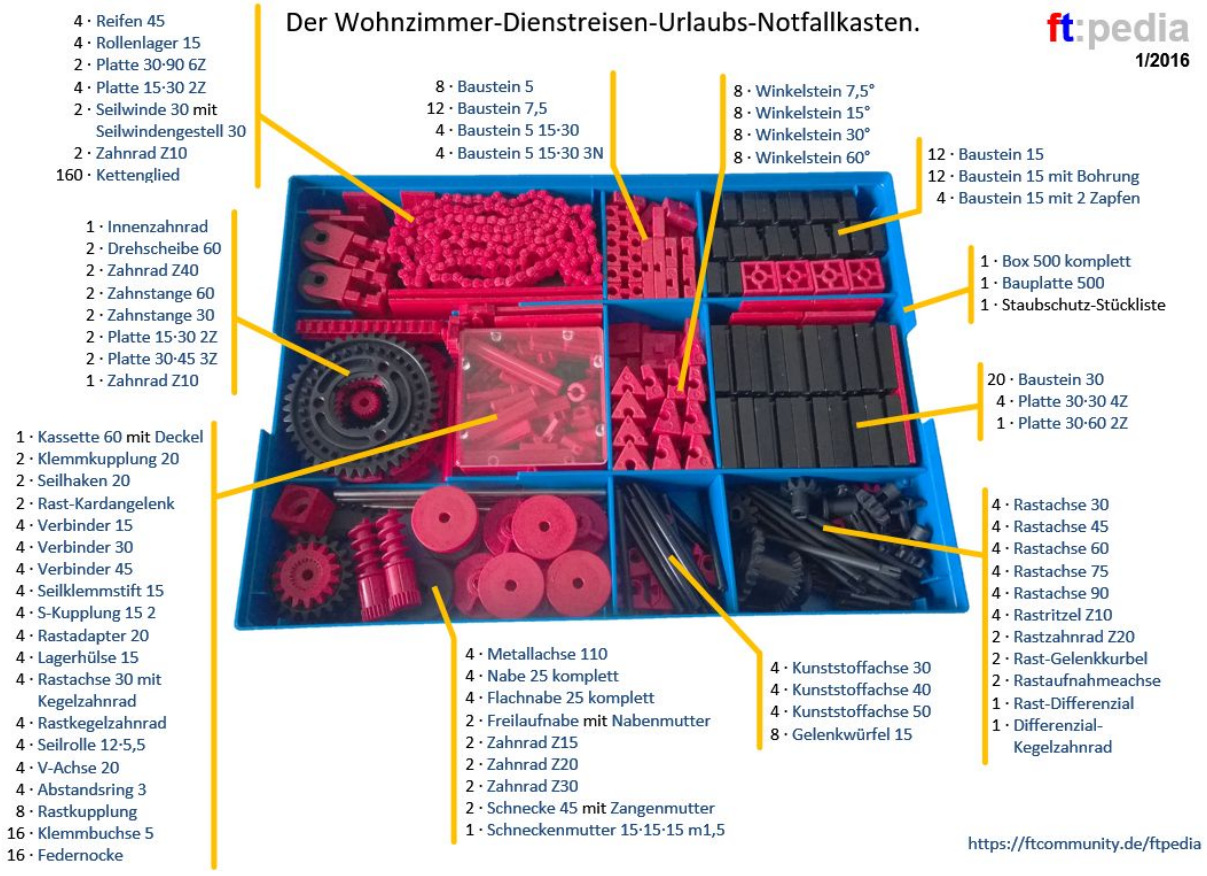


Abb. 1: Die Stücklisten-Seite

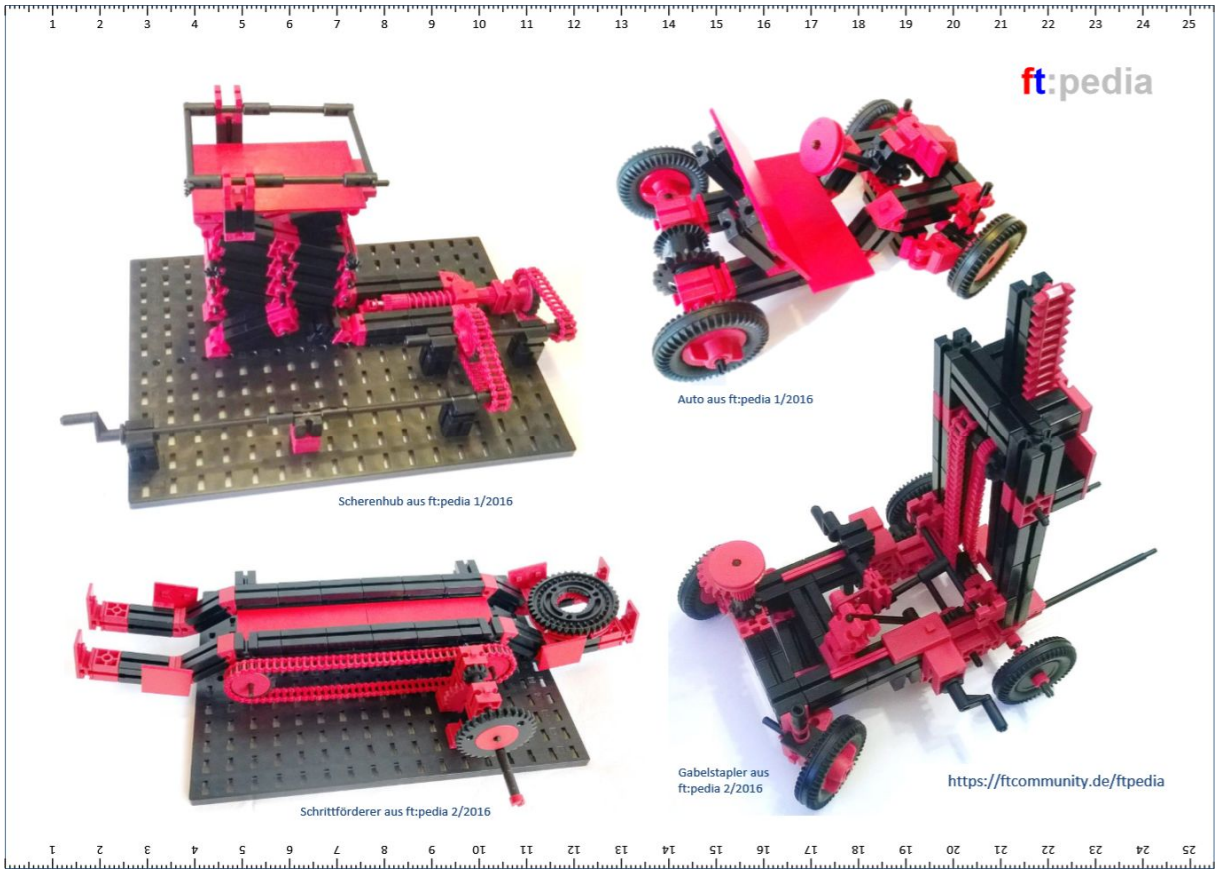


Abb. 2: Die Rückseite

Für die Benutzung am Bildschirm wimmelt es nur so von Hyperlinks: Sämtliche Teilebezeichnungen führen direkt in die fischertechnik-Datenbank [3]; die Modellbilder auf der Rückseite verlinken in die ft:pedia-Ausgaben, in denen das jeweilige Modell enthalten ist.

Ein Ausdruck in schwarz/weiß auf einem DIN-A4-Blatt und Beschneiden entlang der gepunkteten Linie der Rückseite genügt schon. Achtet beim Druck je nach PDF-Reader darauf, dass der gepunktete Rahmen 25,5 cm · 18,1 cm groß ist. Tests mit Microsoft Edge, Microsoft Reader und Google Chrome als PDF-Reader verliefen leider enttäuschend: All diese Programme passen die ganze A4-Seite, also einschließlich des weißen Randes darum, in den Druckbereich ein und ergeben also einen zu klein skalierten Ausdruck. Der Ausdruck des Word-Dokumentes funktioniert gut. Beim Adobe Reader muss die Option zur Verkleinerung „übergroßer Seiten“ deaktiviert sein; dann kommt auch hier die korrekte Größe zu Papier.

Eine edle und langlebigere Hochglanzfassung erhält man unter Verwendung von hochwertigem Papier und anschließendem Laminieren [4].

Quellen

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

[2] Stefan Falk: *Staubschutz-Stückliste für den Urlaubskasten* im ftc-Downloadbereich.

[3] ft Community: *fischertechnik-Datenbank*.

[4] Wikipedia: *Lamination*.

Modell

Urlaubskasten-Modell 4: Kranwagen

Stefan Falk

Zum Bau dieses Kranwagens mit Lenkung, Hinterachsfederung sowie schwenk- und neigbarem Kranarm genügen die Bauteile des Wohnzimmer-Dienstreisen-Urlaubs-Notfallkastens aus ft:pedia 1/2016 [1].

Überblick

Der kleine Kranwagen (siehe Abb. 1 und 15) bietet eine ganze Menge:

- Achsschenkellenkung
- Mitdrehendes Lenkrad
- Gefederte Hinterachse
- Drehbares Krangestell
- Neigbarer Kranarm
- Sperrklinke für das Kranseil
- Flaschenzug für den Kranhaken
- Kupplungshaken für Anhänger oder zur Einhängen des Kranhakens bei der Fahrt

Das Modell ist spielerprobt und hält sicher auch härtesten Einsätzen im Kinderzimmer und draußen stand. Die folgenden Bilder und Texte sollten den Nachbau ganz einfach machen.

Das Fahrgestell

Abb. 2 und 3 zeigen das Chassis von oben und von unten. Der schwenkbare Kranaufsatz wird später einfach in den zentralen BS15 mit Bohrung im Heckteil des Fahrzeugs eingesteckt. Fangen wir aber vorne an:

Der Vorderbau

Der vordere Teil des Fahrwerks ist aus Standardbauteilen schnell zusammengesteckt, wie Abb. 4 zeigt. Auf dem Verbinder 45 sitzen zunächst zwei Winkelsteine 15° und auf diesen zwei BS5.

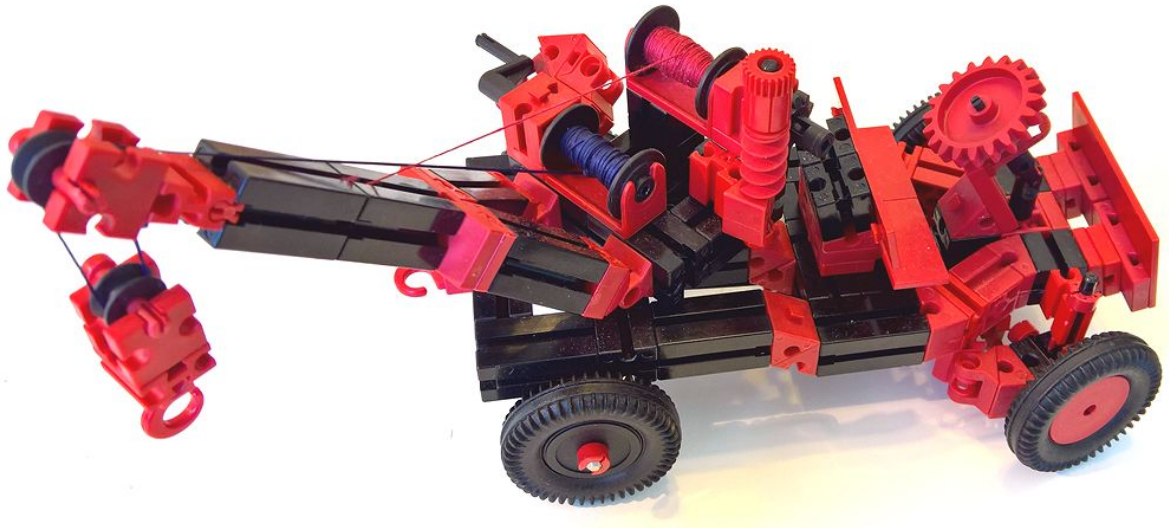


Abb. 1: Gesamtansicht von rechts

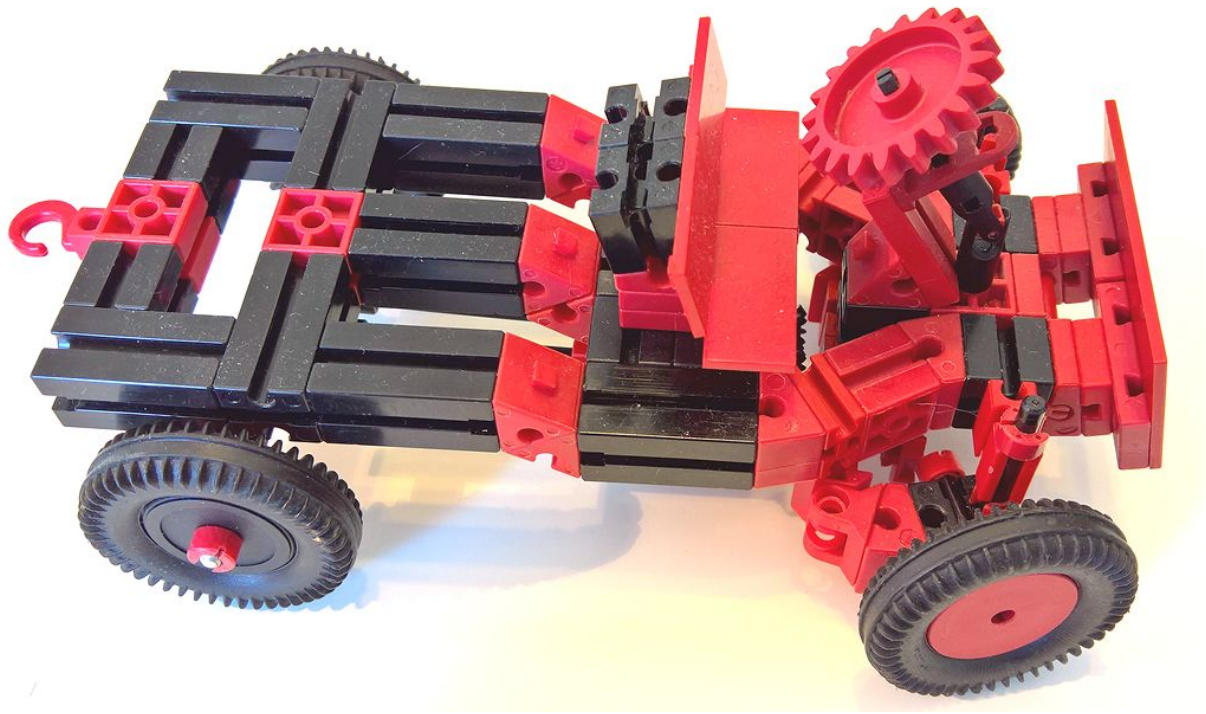


Abb. 2: Das fertige Fahrgestell von oben

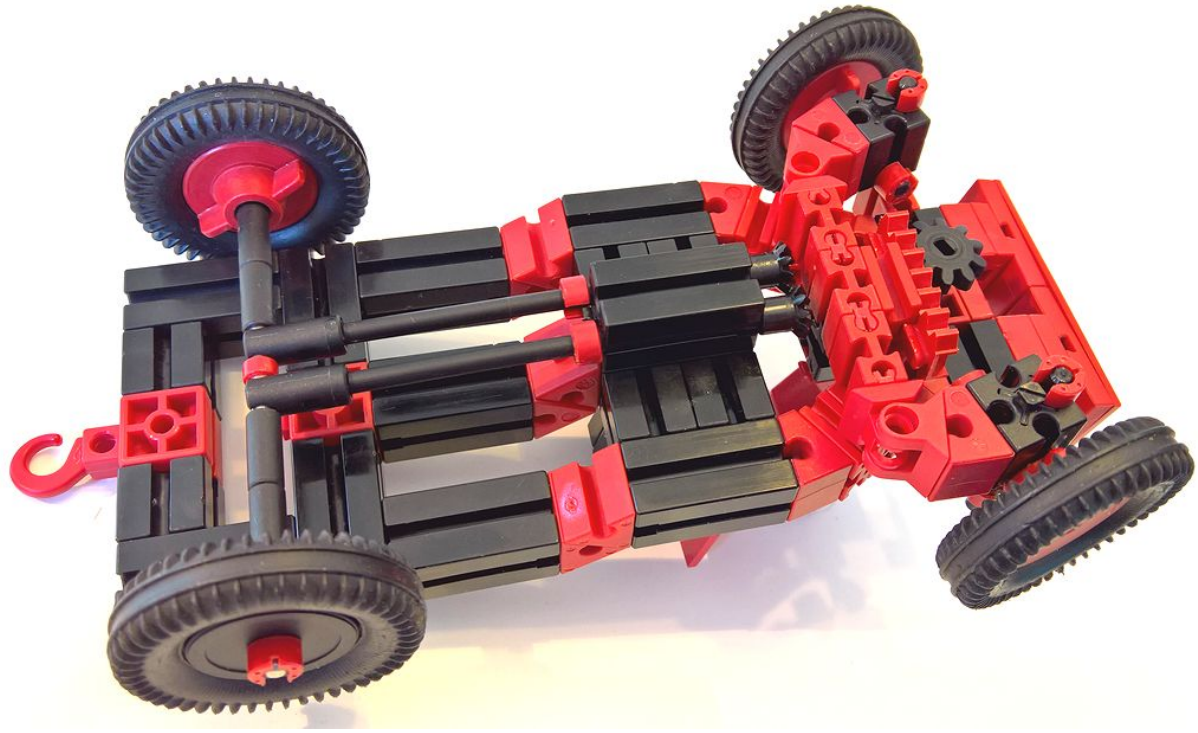


Abb. 3: Das fertige Fahrgestell von unten

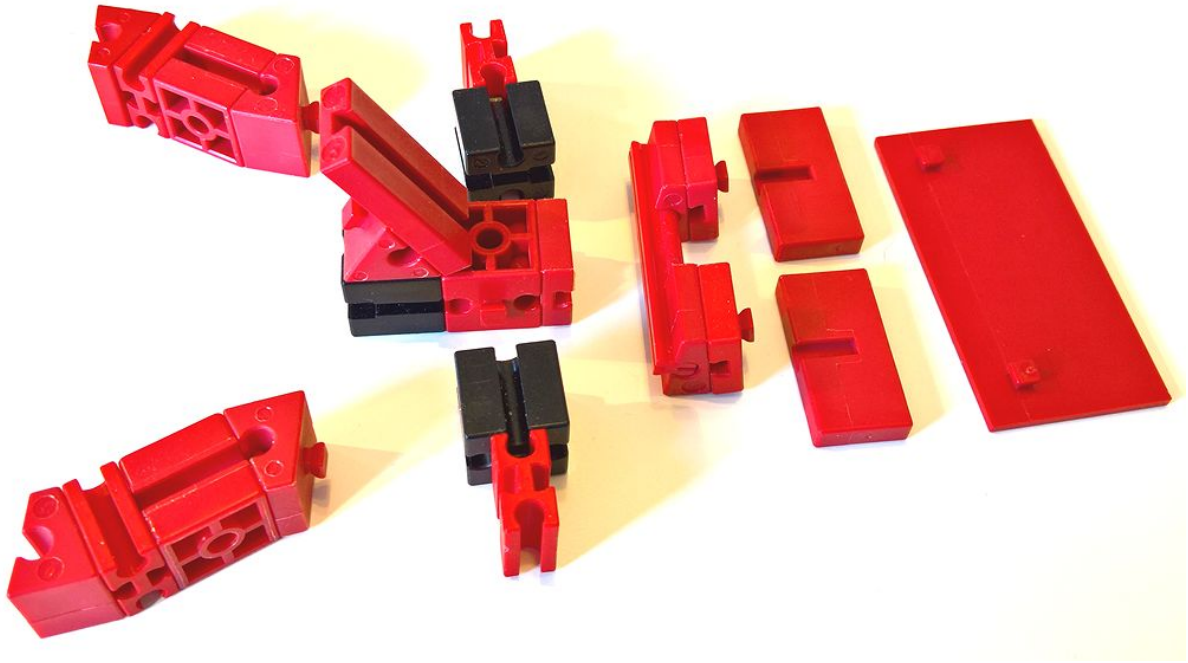


Abb. 4: Aufbau der Frontpartie

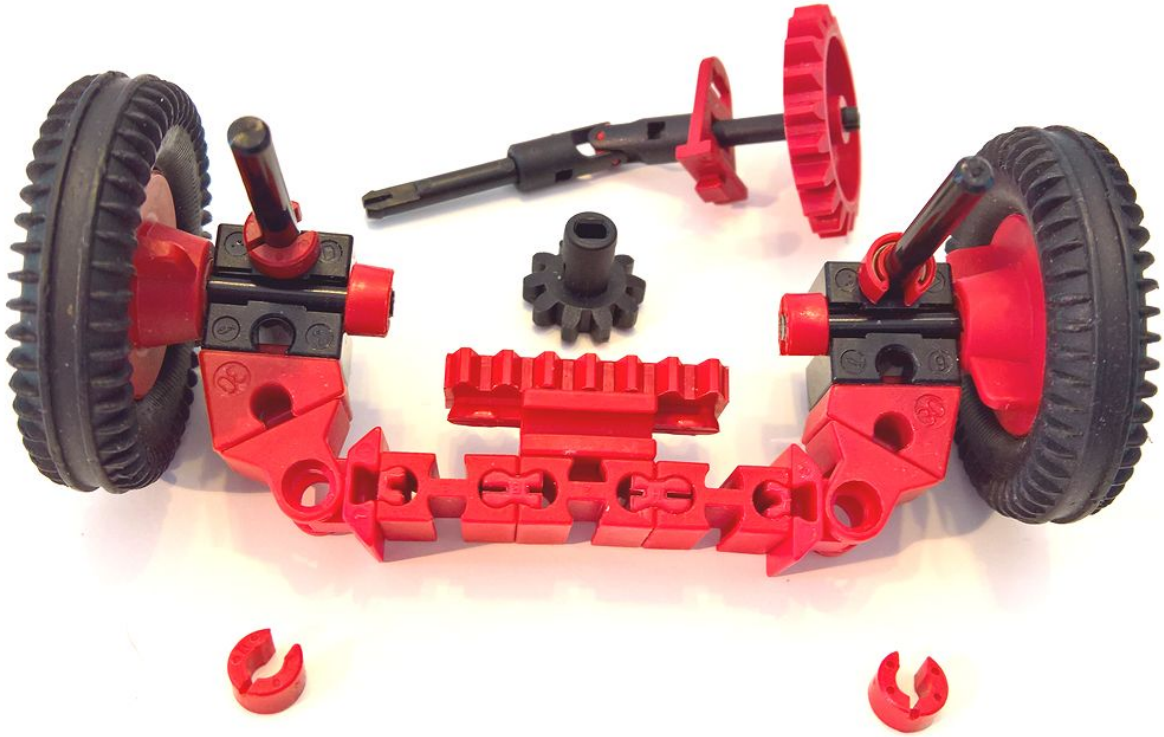


Abb. 5: Die lenkbare Vorderachse

Vorderachse und Lenkung

Die lenkbare Vorderachse ist aufgebaut wie in Abb. 5 gezeigt. Der BS7,5 der Zahnradstange 30 ist mit einem Federnocken mit der quer verlaufenden Spurstange verbunden. Die senkrecht stehenden Kunststoffachsen sind auf beiden Seiten der BS15 mit Klemmringen gesichert (die untere Seite seht ihr in Abb. 3). Je ein dritter Klemmring ist separat abgebildet und kommt später aufs obere Ende der Lenkachsen. Wenn die

Vorderachse fertig eingebaut ist, muss das so aussehen:

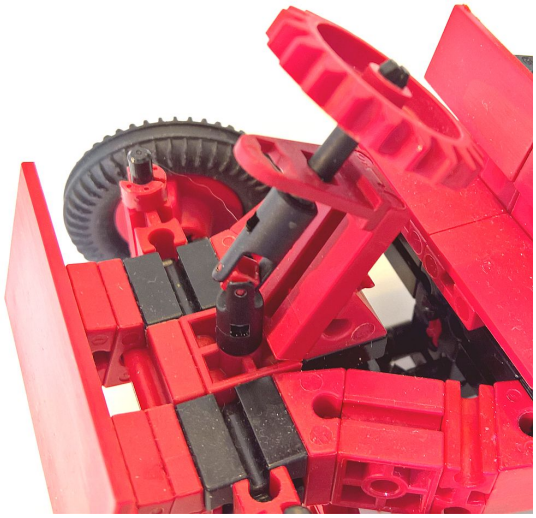


Abb. 6: Die eingebaute Vorderachse und Lenkung

Mittelteil und Sitze

Der mittlere Teil des Fahrzeugrahmens besteht ebenfalls nur aus einigen Standardteilen. Beachtet, dass die drei WS60° unterschiedlich ausgerichtet sind. Der Federnocken in der Mitte trägt den Sitz.

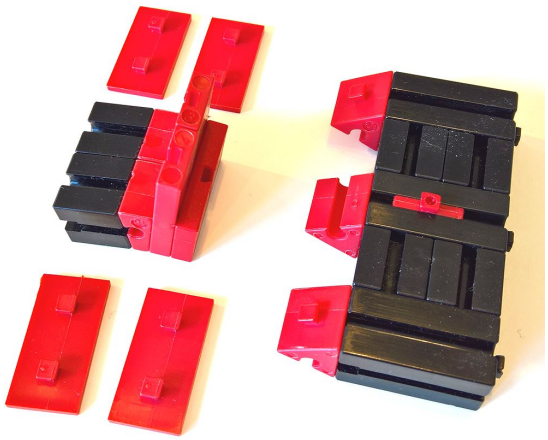


Abb. 7: Aufbau der Sitze

Der Sitz besteht aus einem Baustein 5 15·30 3N (der auf den Federnocken geschoben wird), je zwei Bausteinen 5 15·30, Winkelsteinen 15° und BS15. Die Sitzflächen werden aus vier Platten 15·30 gebildet die auf die BS15 als Lehne und BS5 15·30 als Sitzfläche gesteckt werden.

Die gefederte Hinterachse

Die Hinterachse ist eine Starrachse [2] unter Verwendung einer Metallachse 110. Auf ihr sitzen, von innen nach außen:

- ein Abstandsring 3 mm,
- Rastadapter 20,
- auf jeder Seite zwei Lagerhülsen 15 als Abstandshalter,
- die Reifen 45 auf Freilaufnaben (damit sich Kurveninneres und -äußeres Rad in Kurven unterschiedlich schnell drehen können) und schließlich
- Klemmringe zur Fixierung der Räder, da die ja nur auf Freilaufnaben sitzen.

Abb. 8: Hinterachskonstruktion

In den Rastadaptern stecken lange Rastachsen, auf denen ein BS30 sitzt, der auf der einen Seite mit Klemmringen und auf der anderen einfach mit zwei Rastkegelzahnrädern gesichert ist. Die langen Rastachsen sind das federnde Element und erfüllen ihren Zweck recht gut.

Der BS30 wird über den darauf angebrachten Verbinder 30 einfach auf das Fahrzeugmittelteil aufgeschoben, und zwar von unten auf den mittleren BS30 aus Abb. 7. Wählt am besten einen fest sitzenden Verbinder 30 dafür aus, damit die Hinterachse sich auch bei rauer Fahrt nicht verschieben kann.

Das Heck

Die Heckpartie ist ganz einfach aufgebaut:

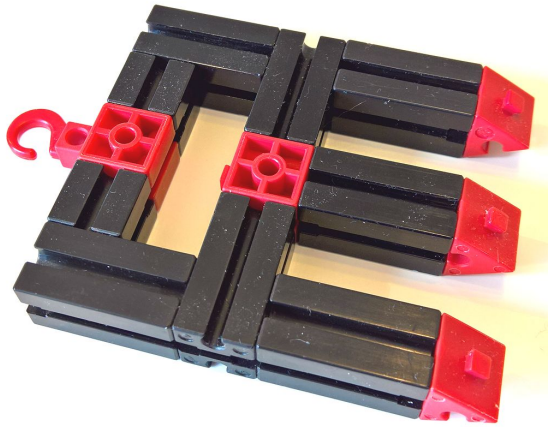


Abb. 9: Heckpartie

Ein paar BS30, BS15 und BS15 mit Bohrung bilden den größten Teil. Beachtet wieder die Ausrichtung der Winkelsteine 60°. In den in der Mitte sitzenden BS15 mit Bohrung wird später der Kranaufsatz eingesteckt.

All die einzelnen Baugruppen lassen sich leicht zum gesamten Fahrzeugrahmen laut Abb. 2 und 3 zusammenfügen.

Der Kranaufsatz

Nun zum drehbaren Kranarm. Abb. 10 und 12 zeigen ihn von oben und von unten. Wie ihr seht steckt unten eine Rastaufnahmeachse 22,5. Die ist es, die in den mittleren BS15 mit Bohrung aus Abb. 9 eingesteckt wird. Mehr ist zur Befestigung des drehbaren Aufsatzes gar nicht nötig. Im Einzelnen wird der Kranaufsatz so gebaut:

Drehgestell und Neigungskontrolle

Aus folgenden Bauteilen ist der drehbare Aufsatz hergestellt:

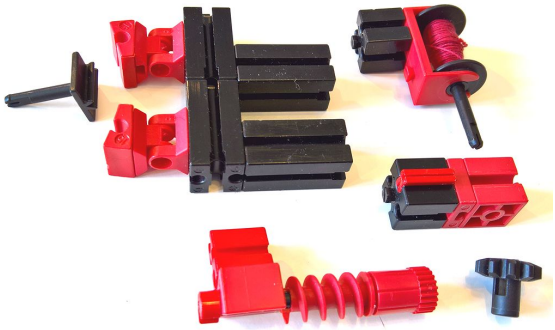


Abb. 11: Drehgestell und Antrieb für die Armneigung

Die Rastaufnahmeachse, links im Bild, kommt auf die Unterseite des Gestells in der Mitte des Bildes. Das Gestell selbst besteht

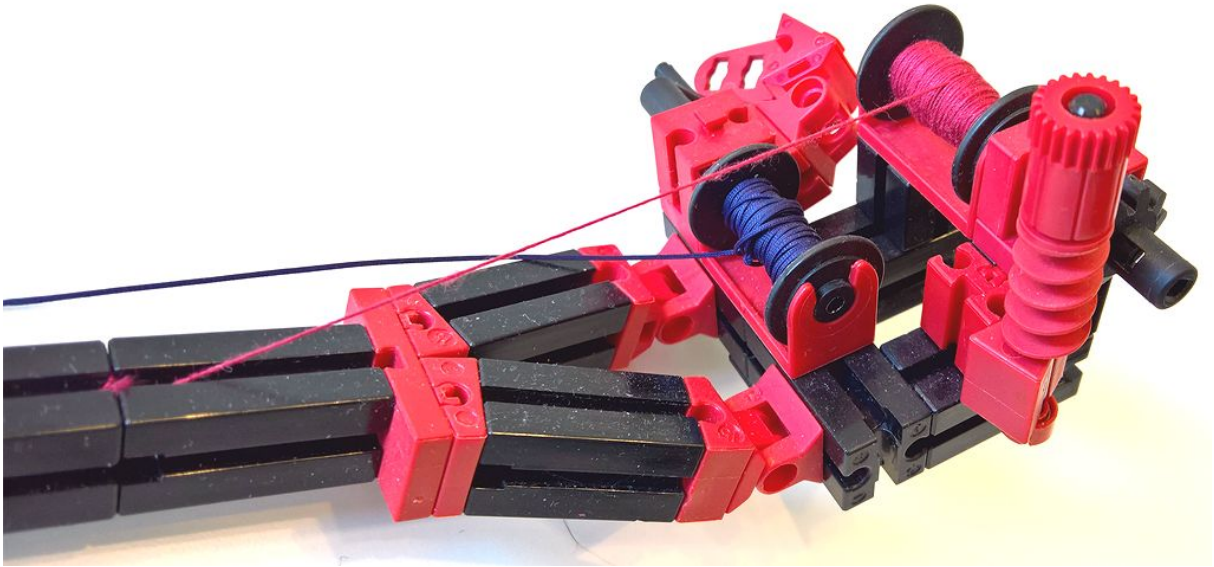


Abb. 10: Der Kranaufsatz von oben

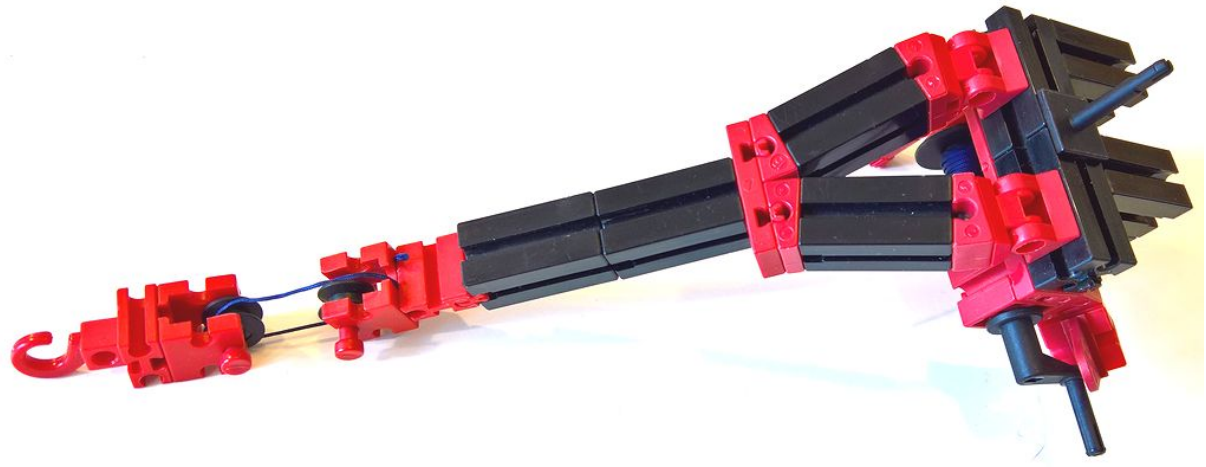


Abb. 12: Der Kranaufsatz von unten

einfach aus vier BS30, zwei Gelenkbausteinen und zwei Winkelsteinen 15°.

Rechts im Bild seht ihr die Seiltrommel für die Neigung des Kranarms. Die Seilwinde wird über eine Rastachse 30 vom Z10 angetrieben. Vergesst nicht den BS5 zwischen den BS15 und BS15 mit Bohrung.

Damit später eine angenehme Drehrichtung entsteht, achtet darauf, dass das Seil von *oben* an der Seiltrommel ankommt (wickelt es ggf. andersherum auf).

Das Z10 wird von der Schnecke angetrieben, deren Achse, mit einem Klemmring gesichert, in einem Winkelstein 30° sitzt, der an einem BS7,5 angebracht ist. Letzterer wird einfach auf den Verbinder 15 rechts im Bild aufgeschoben.

Da der Schneckenantrieb selbsthemmend ist, wird sich der Kranarm auch unter Last nicht von alleine absenken – ganz wie gewünscht.

Zu guter Letzt wird der fertige Seilantrieb auf die beiden herausstehenden BS30 des Drehgestells aufgeschoben (siehe Abb. 10).

Der Antrieb des Kranseils

Wenige Bauteile genügen für das Kranseil selbst:

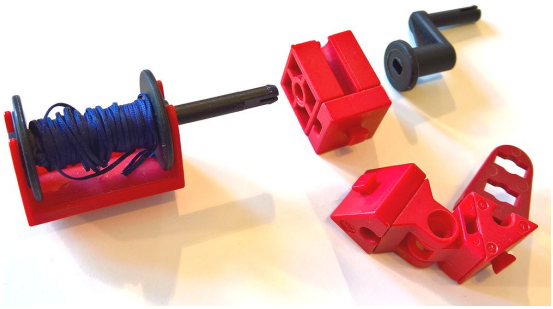


Abb. 13: Antrieb des Kranseils

Wieder steckt in der Seiltrommel 30 eine Rastachse, auf die eine Rastkurbel kommt. Die Mimik aus WS30°, Gelenkstein, WS60° und S-Kupplung 15 2 dient als Sperrklinke [3] für die Kurbel: Man kann das Kranseil immer hochkurbeln, aber durch die Sperrklinke ist es vor unerwünschtem Absenken geschützt, weil die Kurbel beim Zurückdrehen an ihr hängen bleibt. Will man das Seil tatsächlich herablassen, muss man die Sperrklinke lösen – also wegklappen.

Damit das richtig funktioniert muss das Kranseil an der Seiltrommel von *unten* ankommen – ggf. das Seil also bitte andersherum aufwickeln.

Der Kranarm

Auch recht einfach ist der Kranarm selbst aufgebaut:

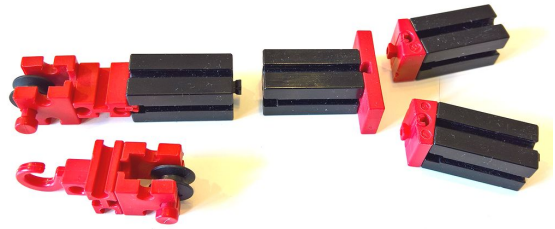


Abb. 14: Aufbau des Kranarms

Bemerkenswert hier ist die Befestigung der zwei Seile: Das Seil für die Kranneigung wird zwischen den beiden BS30 des Arms eingeklemmt, die in Abb. 14 extra deswegen aufgetrennt gezeigt sind. Das Kranseil selbst hingegen läuft über die Seilrolle am Ende des Kranarms, durch den Kranhakenaufbau und schließlich in Richtung Arm-Inneres wieder hoch. Das Ende des Kranseils klemmt ihr zwischen dem BS7,5 und dem Rollenlager 15 am Ende des Arms ein. Es empfiehlt sich, das Seil einmal um den Zapfen des Rollenbocks zu schlingen, damit es gut hält.

Lasset das Spielen beginnen!

Fügt die einzelnen Baugruppen also noch zusammen, um das Gesamtmodell wie in den großen Abbildungen gezeigt fertigzustellen. Und dann krant, was das Zeugs hält – viel Spaß!

Quellen

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

[2] Wikipedia: *Starrachse*.

[3] Wikipedia: *Sperrklinke*.

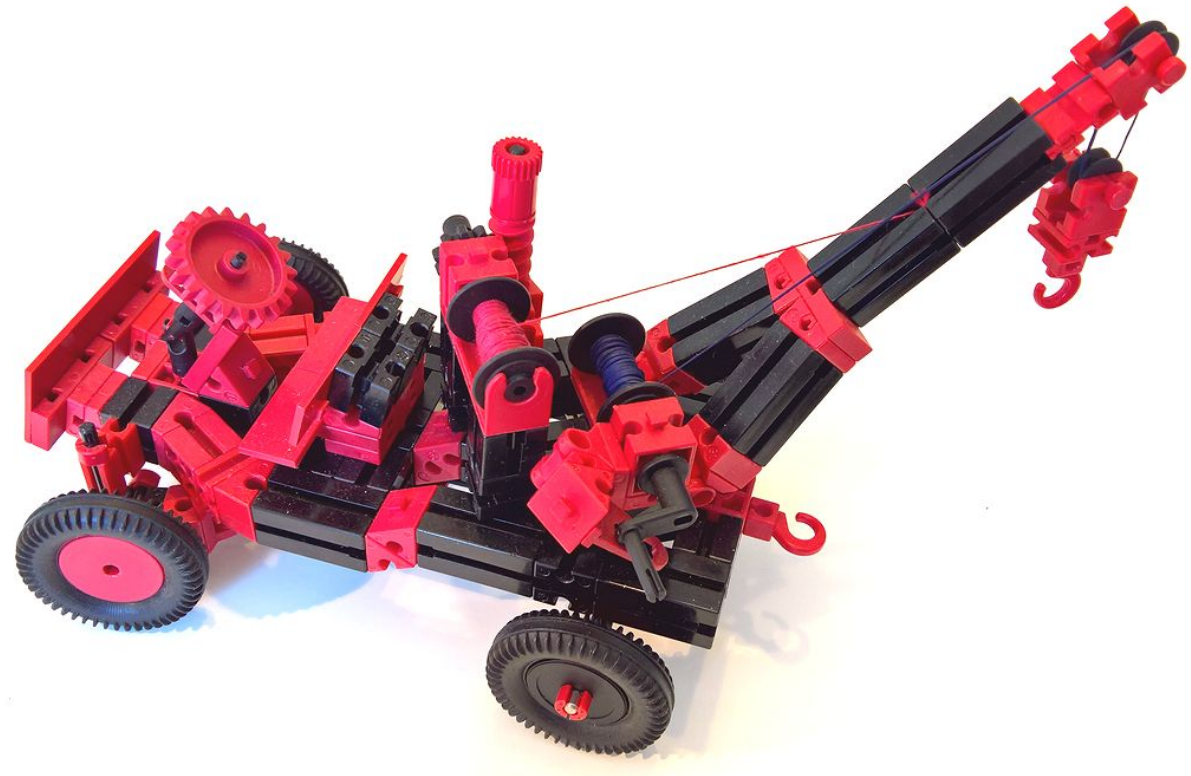


Abb. 15: Gesamtansicht von links

Elektromechanik

Neue Synchronmotoren

Rüdiger Riedel

In den vergangenen Ausgaben der ft:pedia haben sich mehrere Beiträge mit Synchronmotoren beschäftigt – eingeleitet von Matthias Dettmers Beitrag in ft:pedia 2/2016 [2]. Diesmal wird die Liste der mit fischertechnik konstruierbaren Synchronmaschinen um weitere Varianten mit konstanten Drehzahlen von u. a. 429, 333 oder 214 U/min und bis zu 15 Polpaaren erweitert.

Aber immer das Anwerfen!

Synchronmaschinen laufen nicht von alleine los, sie müssen angeworfen werden. Doch das müssen auch Automotoren und Flugzeugturbinen. Der Unterschied besteht darin, dass diese eine Mindestdrehzahl benötigen, die man erreichen oder überschreiten muss. Synchronmaschinen wollen dagegen recht genau auf ihre konstruktiv festgelegte Drehzahl gebracht werden.

Das Anwerfen gelingt aber sehr einfach mit einem Stroboskop als Kontrolle [1]. Damit habe ich die gewünschte Drehzahl sofort im Blick und kann nach Bedarf schneller oder langsamer andrehen.

Die nach meiner Kenntnis bisher erreichten Synchron-Drehzahlen mit fischertechnik-Synchronmotoren sind in Tabelle 1 zusammengestellt. Darunter finden sich einige neue Varianten, die nachfolgend mit Abbildung vorgestellt werden. Alle von mir konstruierten Synchronmaschinen laufen mit den ‚alten' fischertechnik-Elektromagneten an 6 V Wechselstrom. Bei einigen Maschinen habe ich zwei Elektromagnete verwendet. Das verbessert teilweise die Laufruhe, ist für die Funktion aber nicht erforderlich – mit einer Ausnahme, wie noch gezeigt wird.

Zur besseren Ansicht habe ich die für das Stroboskop benötigte Flachnabe mit Markierung auf den meisten Fotos weggelassen.

Die Drehzahlen in Tabelle 1 sind teilweise gerundet. Die Anzahl der Pole ist gleich Polpaarzahl *p* mal 2.

p	U/s	U/min
1	50,00	3000
2	25,00	1500
3	16,67	1000
4	12,50	750
5	10,00	600
6	8,33	500
7	7,14	429
8	6,25	375
9	5,56	333
10	5,00	300
11	4,55	273
12	4,17	250
13	3,85	231
14	3,57	214
15	3,33	200
...	...	...
20	2,50	150

Tabelle 1: Polpaare und erreichbare Drehzahlen der Synchronmaschinen

Polpaarzahl *p* = 1

Der Läufer besteht aus einer Strebe 30, die von einem Mitnehmer (31712) und den Klemmhülsen an der Achse fixiert wird. Statt der Strebe 30 können auch Laschen 15 oder 21,2 verwendet werden. Die Stabmagnete (d = 4 mm, l = 10 mm) sind entgegengesetzt magnetisiert. Der Motor läuft problemlos mit 6 V Wechselstrom; angeworfen wird die Achse mit Daumen und Zeigefinger. Kräftig drehen!

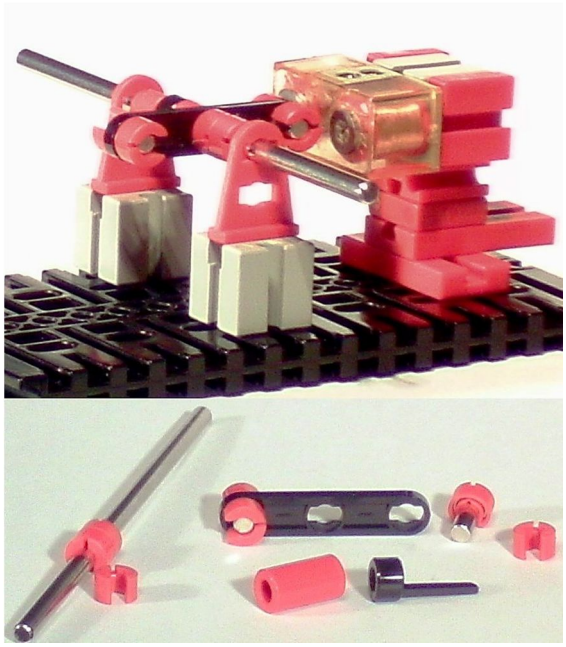


Abb. 1: Oben der Motor, darunter die Einzelteile des Läufers

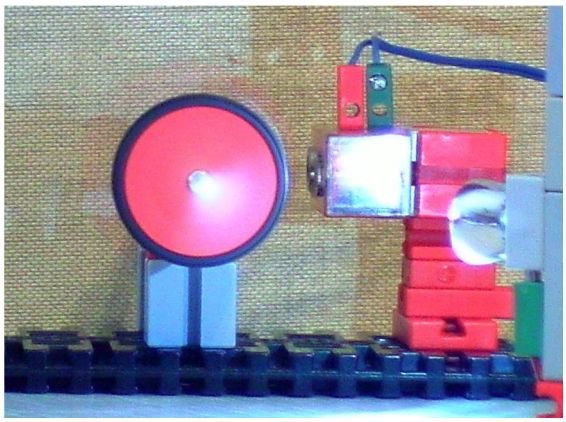


Abb. 2: Stroboskopbild für 3000 U/min; Läufer ist wg. der kleinen Kontur kaum zu sehen

Das Stroboskop zeigt ein Segment auf der Flachnabe (Erläuterungen findet ihr in [1]).

Polpaarzahl *p* = 2

Die Achsbefestigung in Abb. 3 entspricht der von p = 1, ebenso wie nachfolgend alle Achsbefestigungen von Statikteilen.

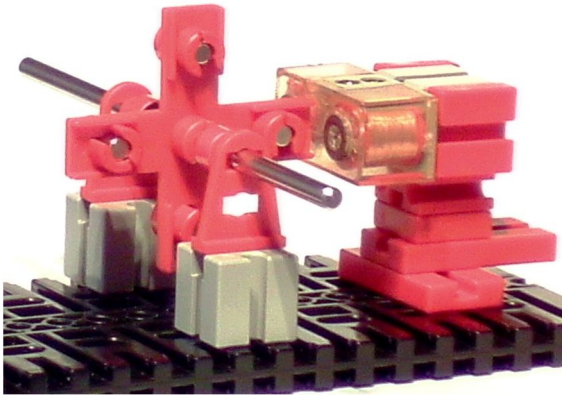


Abb. 3: Motor mit Statik-Kreuzlasche

Grundsätzlich ist es bei den Polpaarzahlen p = 1 und 2 günstig, das Trägheitsmoment oder Schwungmoment des Läufers möglichst klein zu halten

Polpaarzahl *p* = 3

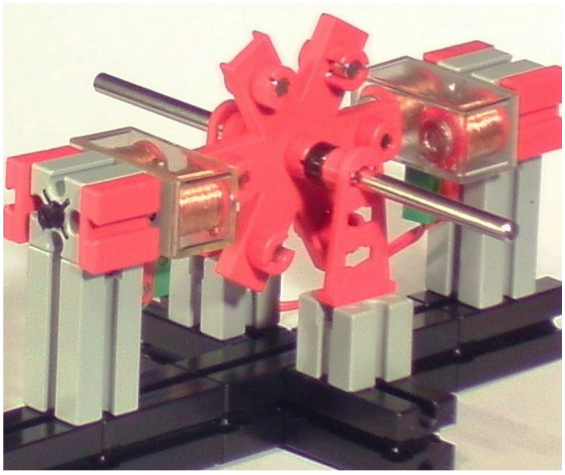


Abb. 4: Hier kommt die Sternlasche zum Einsatz

Weitere Synchronmaschinen mit p = 3 finden sich in [2, 3].

Polpaarzahl *p* = 4

Der Läufer für die Polpaarzahl 4 in Abb. 5 besteht aus vier Baugruppen (Winkelsumme 360°) WS60 + BS5 + WS7,5 + WS15 + WS7,5 + BS5, verbunden mit der Achse über vier Streben 15 und zwei Laschen 15.

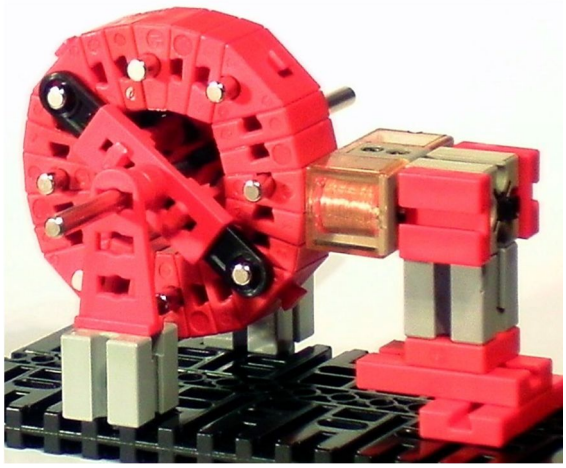


Abb. 5: Der Läufer ist nicht ganz kreisförmig

WS steht für ‚Winkelstein', WS60 für ‚Winkelstein 60°', BS steht für ‚Baustein', also BS5 für ‚Baustein 5'.

Die Magnete sind mit wechselnder Polung von vorne und von hinten in die Löcher der WS60 und WS15 gesteckt (es sind also 16 Magnete) und halten durch die gegenseitige Anziehung. Das ist absolut zuverlässig.

Nicht verschweigen möchte ich, dass alle Motoren, bei denen sich die Magnete durch die gegenseitige Anziehung in den Löchern halten, zum Klappern neigen.

Polpaarzahl *p* = 5

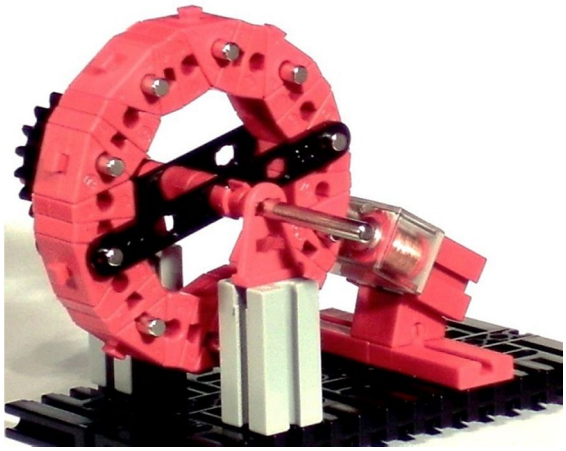


Abb. 6: Das Zahnrad hinten hat keine Bedeutung

Der Läufer in Abb. 6 besteht aus einem „fast-Kreis" mit zehn Baugruppen WS60 - WS7,5 - WS15 (Winkelsumme 375°). Das Minuszeichen soll andeuten, dass dieser Baustein mit seiner Schmalseite nach außen eingefügt wird. Die Streben des Läufers werden wieder durch Klemmbuchsen und einen Mitnehmer kraftschlüssig mit der Achse verbunden.

Zu weiteren Synchronmaschinen mit $p = 5$ siehe [2, 4, 5].

Polpaarzahl *p* = 6

Der Läufer in Abb. 7 besteht aus 12 Baugruppen WS60 - WS30. Für weitere Synchronmaschinen mit $p = 6$ siehe [6, 7, 8, 9, 11].

Abb. 7: Läufer für 500 U/min

Polpaarzahl *p* = 7

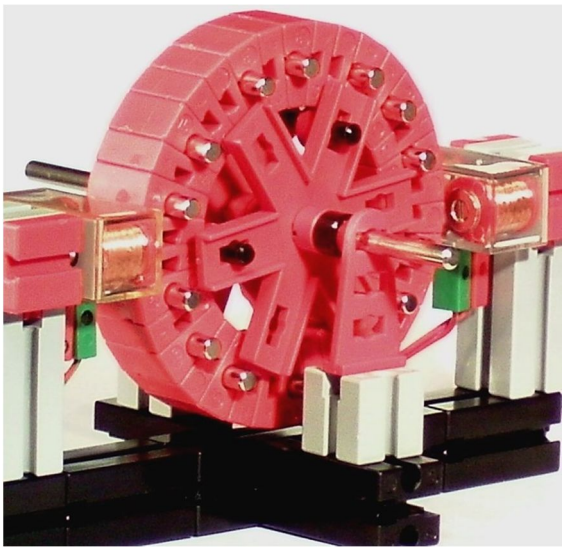


Abb. 8: Die Sternlaschen halten den Läufer

Obwohl die Abstände der Magnete ungleichmäßig sind, dreht sich dieser Läufer einwandfrei. Er besteht aus 14 WS15 und 20 WS7,5 (Winkelsumme 360°), eingeklemmt zwischen zwei Sternlaschen mit drei Kunststoffachsen und Klemmbuchsen.

Polpaarzahl *p* = 8

Der Läufer besteht aus 16 Baugruppen WS30 - WS7,5 (Winkelsumme 22,5° x 16 = 360°), eingeklemmt zwischen zwei Drehscheiben 60.

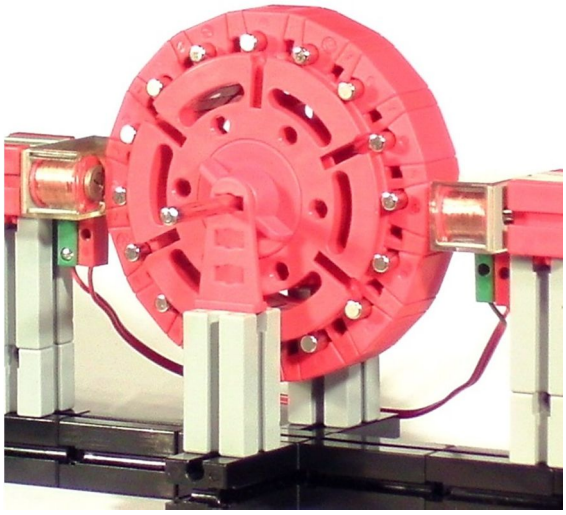


Abb. 9: Die Drehscheiben passen genau in den Magnetkreis

Polpaarzahl *p* = 9

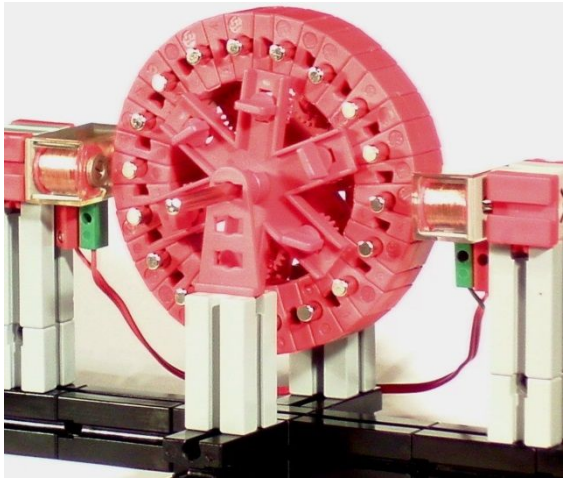


Abb. 10: Die Polpaarzahl ist 9

Der Läufer besteht aus sechs Baugruppen WS15 + WS7,5 + WS15 + BS5 + WS15 + WS7,5 (Winkelsumme 270° + 90° = 360°), eingeklemmt zwischen zwei Sternlaschen mit Riegelscheiben.

Polpaarzahl *p* = 10

Der Läufer besteht aus zehn Baugruppen WS60 + BS5 - WS30 + WS7,5 (Winkelsumme 375°, also ein „fast-Kreis“), verbunden mit der Achse über vier Streben 30 und zwei Laschen 21,2.

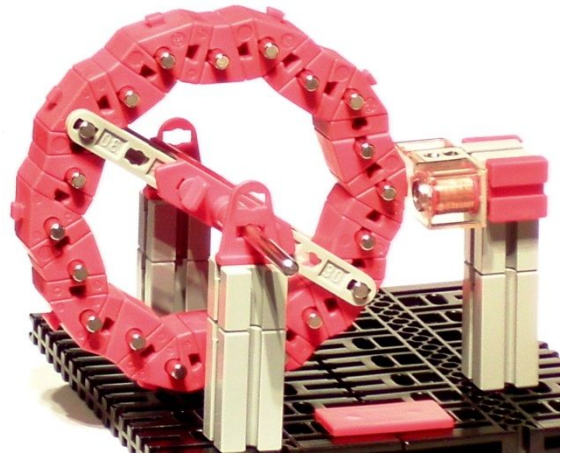


Abb. 11: Läufer als „fast-Kreis“

Polpaarzahl *p* = 11

Der Läufer besteht aus 22 WS15, 18 BS5 und 4 WS7,5 (Winkelsumme 330° +30° = 360°), verbunden mit der Achse über vier Streben 30 und zwei Laschen 15.

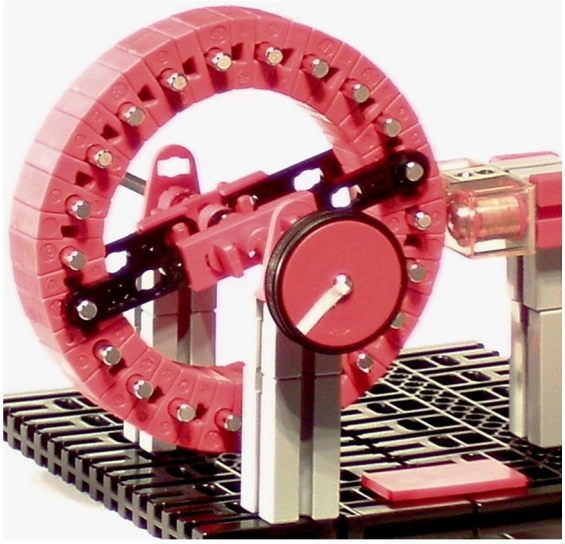


Abb. 12: Vorne die Flachnabe mit Markierung zur Drehzahlkontrolle mit dem Stroboskop

Polpaarzahl *p* = 12

Der Läufer in Abb. 13 besteht aus 24 WS15 und 24 BS5, verbunden mit der Achse über vier Streben 30 und zwei Laschen 21,2.

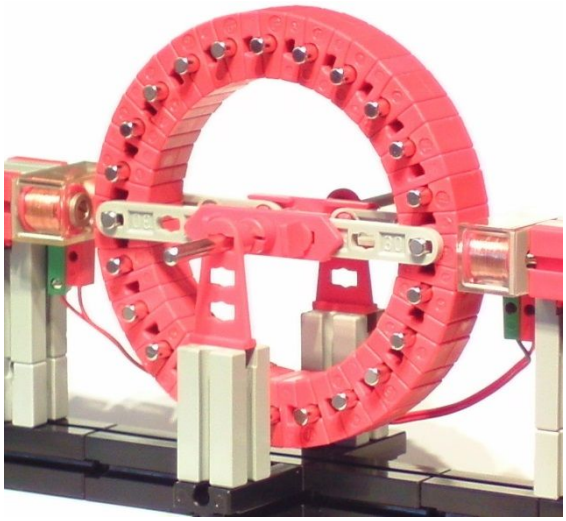


Abb. 13: Dieser Motor dreht mit 250 U/min

Polpaarzahl *p* = 13

Der Läufer besteht aus 26 Baugruppen WS30 - WS15 (Winkelsumme 390°, also ein „fast-Kreis“), verbunden mit der Achse über vier Streben 45 und zwei Laschen 21,2.

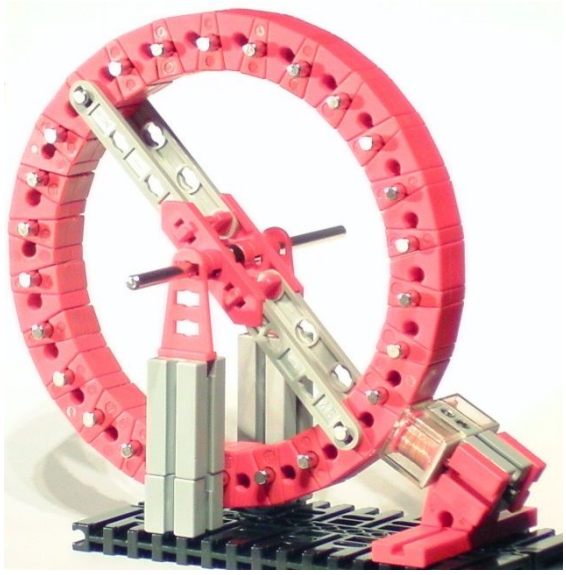


Abb. 14: Recht filigran, dieser Motor

Polpaarzahl *p* = 14

Der Läufer besteht aus vier Baugruppen WS15 - WS7,5 + WS15 + BS5 + WS15 + BS5 + WS15 - WS7,5 + WS15 + BS5 + WS15 +BS5 + WS15 + BS5 (Winkelsumme 360°), verbunden mit der Achse über vier Streben 30 und zwei Sternlaschen.

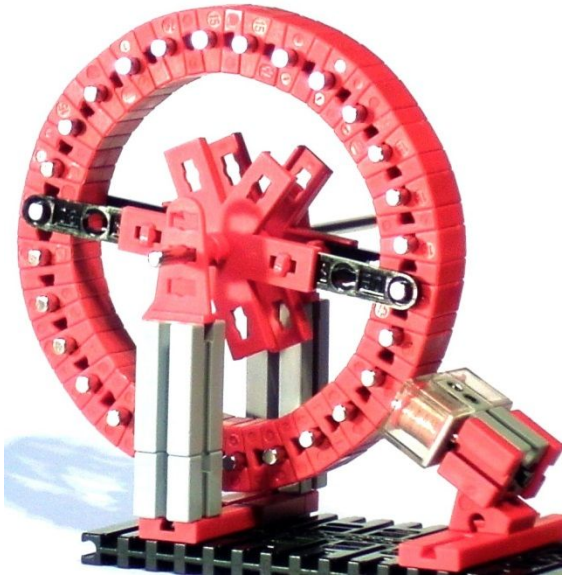


Abb. 15: Motor mit Statik-Sternlaschen

Polpaarzahl *p* = 15

Abb. 16: Mein größter Motor, Drehzahl 200 U/min

Der Läufer besteht aus sechs Baugruppen WS15 + BS5 - WS7,5 + WS15 + BS5 + BS5 + WS15 + BS5 + BS5 + WS15 + BS5 - WS7,5 + WS15 + BS5 + BS5 (Winkelsumme 360°), verbunden mit der Achse

über sechs Streben 60. Wegen der großen Speichenlänge sind noch zwei Kunststoffachsen 30 mit Klemmbuchsen angebracht.

Weiterhin unerreicht ist die Polpaarzahl 20, realisiert von Thomas Püttmann [10, 11].

Die halbe Synchronmaschine

Bis auf die ersten drei Maschinen haben alle einen Schwachpunkt: Sie benötigen durch die Konstruktion doppelt so viele Magnete wie entsprechend der Polpaarzahl erforderlich wären. Die Maschine in Abb. 17 hat 24 Magnete bei $p = 12$, dagegen benötigt die Maschine in Abb. 13 ganze 48.

Abb. 17: Polpaarzahl p = 12

Aber wenn wir schon beim Magnete-Sparen sind, dann unternehmen wir auch den nächsten Schritt: Wir halbieren den Läufer.

Zum Ausbalancieren habe ich vier Magnet-Rückschlussplatten verwendet, aber man kann auch Metallachsen nehmen. Es übernimmt immer einer der Elektromagnete den Antrieb, nach einer halben Umdrehung der andere, ohne irgendwelche Umschaltungen. Die Elektromagnete sind parallel an 6 V Wechselstrom angeschlossen. Der Motor dreht einwandfrei mit 250 U/min, entsprechend einer Polpaarzahl 12.

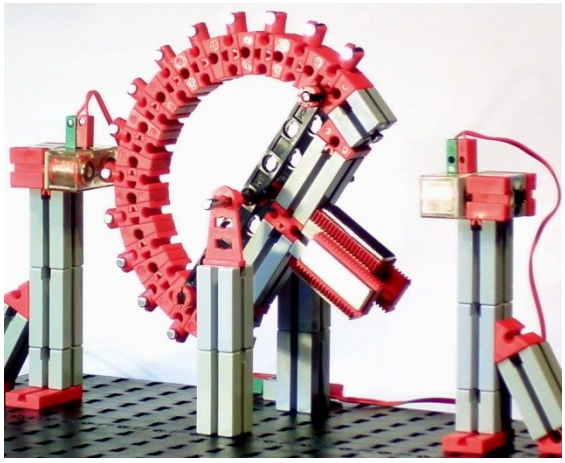


Abb. 18: Der halbe Läufer braucht ein Gegengewicht

Rechts im Vordergrund von Abb. 19 ist das LED-Stroboskop zu sehen. Der Reflektor stammt von einer kleinen Taschenlampe.

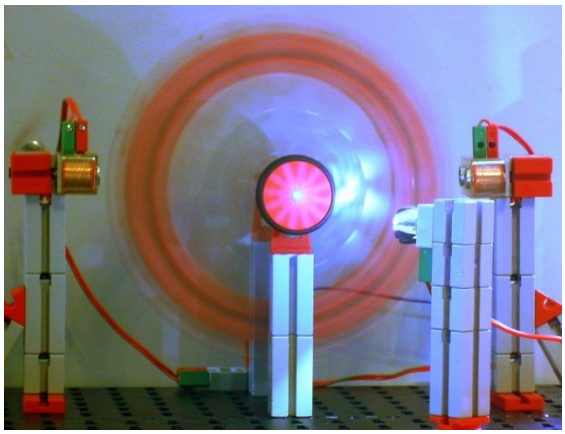


Abb. 19: Die „halbe“ Synchronmaschine

Zum Schluss eine halbe Synchronmaschine mit Polpaarzahl 24 (Abb. 20).

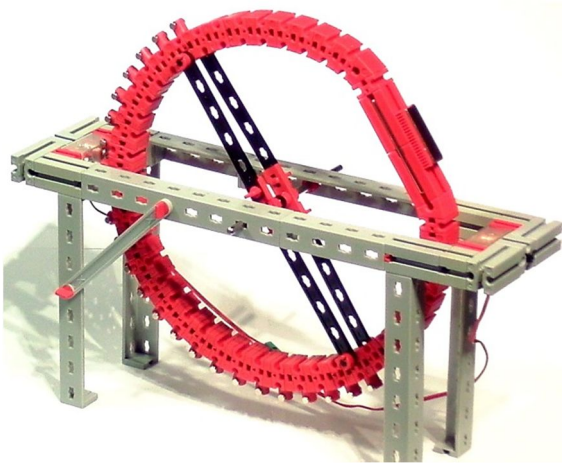


Abb. 20: Dreht mit 125 U/min

Und auch wenn sie niemand braucht: Zu zeigen, dass „es geht“, gibt ein prima Gefühl!

Quellen

[1] Rüdiger Riedel: *Funktionsmodelle von Gleich- und Wechselstrom-motoren*. ft:pedia 4/2016, S. 52-58.

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

[3] Stefan Falk: *Flüsterleise 50 Hz-Zeigeruhr*. Bilderpool der ftcommunity, 2016.

[4] Thomas Püttmann: *Synchronmotor mit 600 U/min mit Ludger Mäsings Rollenblock-10-Eck*. Bilderpool der ftcommunity, 2017.

[5] Thomas Püttmann: *Synchronmotor mit 600 U/min aus drei Drehscheiben*. Bilderpool der ftcommunity, 2017.

[6] Thomas Püttmann: *50 Hz-Motor*. Bilderpool der ftcommunity, 2011.

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

[8] david-ftc: *Synchronmotor mit Anlasser*. Bilderpool der ftcommunity, 2017.

[9] Helmut Schmücker: *50 Hz-Uhr*. Bilderpool der ftcommunity, 2013.

[10] Thomas Püttmann: *50 Hz-Uhr*. Bilderpool der ftcommunity, 2011.

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

Modell
Tunnelbohrmaschine

Daniel Canonica

Ohne die gewaltigen Tunnelbohrmaschinen wären die heutigen enormen Tunnelprojekte nicht machbar. Diese Maschinen sind technische Meisterwerke. Sie erlauben – verglichen mit herkömmlichen Baumethoden – einen weitgehend sicheren Vortrieb in verschiedenartigem Gestein.

Der Gotthard-Basistunnel

Im Sommer 2016 wurde mit dem Gotthard-Basistunnel der längste Eisenbahntunnel der Welt eröffnet. Seit dem 11.12.2016 ist er für den fahrplanmäßigen Personenverkehr mit Tempo 200 freigegeben. Der Hauptnutzen besteht aber darin, dass lange und schwere Güterzüge und Huckepackwagen in wenigen Stunden die Nord-Süd-Achse Basel-Mailand befahren können, um einen Teil des alpenquerenden Güterverkehrs von der Straße auf die Schiene zu bringen.

Für den 57 km langen Basistunnel mit zwei parallelen Röhren und diversen Zusatzstollen wurden von 2003 bis 2011 vier Tunnelbohrmaschinen mit den schönen Namen „Sissi“, „Heidi“, „Gabi-I“ und „Gabi-II“ eingesetzt. Insgesamt wurden 28 Millionen Kubikmeter Material ausgebrochen [1, 3].

Der Vortrieb gelang weitgehend nach Plan. Zwischendurch mussten aufgrund des Gesteins längere Pausen eingelegt werden. Insbesondere der wie Zucker aussehende Dolomit, welcher stellenweise mit Wasser unter hohem Druck austreten kann, war eine besondere Herausforderung.

Die Tunnelbohrmaschine

Eine Tunnelbohrmaschine (TBM) besteht aus einem rotierenden Bohrkopf mit Rollmeißeln und dahinter liegenden Geräten für den Abtransport des Ausbruchmaterials, für die Errichtung der Tunnelwände und für die Energie- und Wasserversorgung.

Der Bohrkopf hat einen Durchmesser bis etwa 14 Meter, die gesamte Maschine ist bis zu 400 Meter lang [2].

Abb. 1: Tunnelbohrmaschine

Der Bohrkopf wird mit bis zu 8000 kN gegen den Fels gedrückt. Dabei hält sich der hintere Teil der Maschine mit einer Art Stempel (Gripperplatten) an den Felswänden fest und der Bohrkopf wird mit Hydraulikzylindern vorgeschoben. Nach einem Stück Vortrieb werden die Gripper zurückgezogen, der hintere Teil der Maschine fährt vor und wird von neuem an der Felswand fixiert.

Je nach Gestein wird im Idealfall ein Vortrieb von bis zu 50 Metern pro Tag

erreicht; dabei wird normalerweise im Drei-Schichtbetrieb gearbeitet.

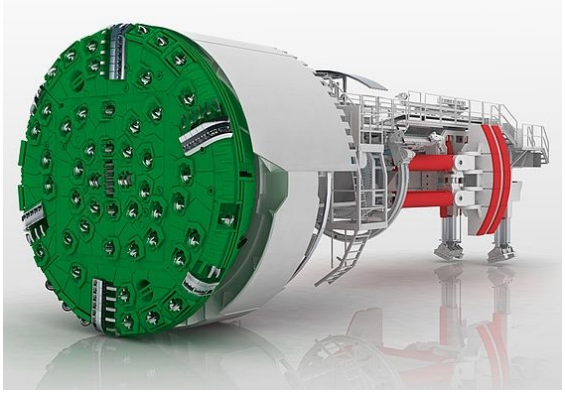


Abb. 2: Gripper-TBM [3]

Im Gotthard-Basistunnel wurden etwa 150-500 Meter pro Monat und 3-12 mm pro Umdrehung des Bohrkopfs erreicht [3].

Hinter dem Bohrkopf wird die Felswand entweder mit Spritzbeton gesichert, oder es werden sogenannte Tübbinge angebracht, das sind vorgefertigte, abgerundete Elemente, welche etwa einem Siebtel eines Kreises entsprechen. Die Tübbinge werden ringsum an die Tunnelwand gepresst und mit einem Schlussstück wie beim klassischen Bogenbau gesichert.

Abb. 3: Tübbing [4]

Die elektrische Leistung einer TBM liegt bei 250-950 kW.

Das Modell

Bei meinem Modell einer fischertechnik-Tunnelbohrmaschine (Abb. 4) handelt es sich noch um einen Prototypen – da kann noch vieles ergänzt und verbessert werden.

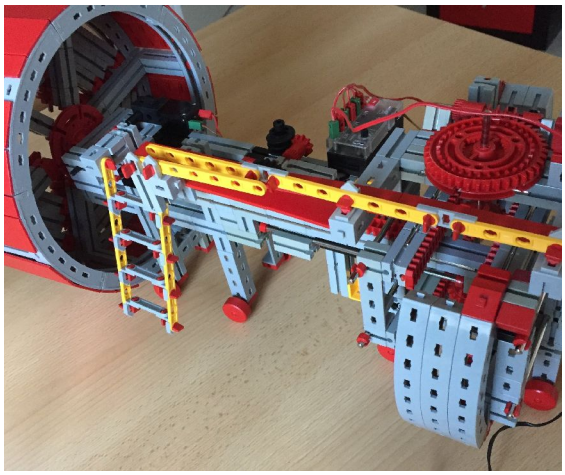


Abb. 4: Das TBM-Modell, vorne der Bohrkopf, hinten die Gripper

Hier sieht man den Bohrkopf mit den Rollmeißeln:

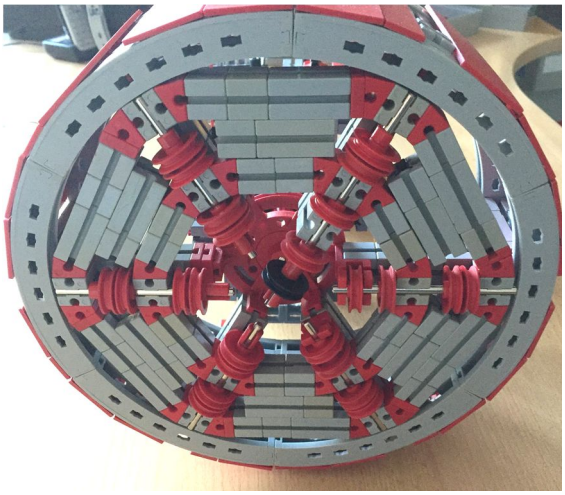


Abb. 5: Bohrkopf

Der Bohrkopf wird, wie schon erwähnt, mit enormem Druck gegen den Fels gepresst. Durch die Rotation und die Rollmeißel wird der Fels beinahe pulverisiert, kleine Stücke fallen ab und werden durch Führungen gegen die Mitte des Bohrkopfs auf ein Förderband geleitet.

Auf dem folgenden Bild sieht man den Antrieb des Bohrkopfs. Dank der starken Untersetzung kann problemlos ein kleiner Motor verwendet werden (solange man nicht auf allzu hartes Gestein trifft …).

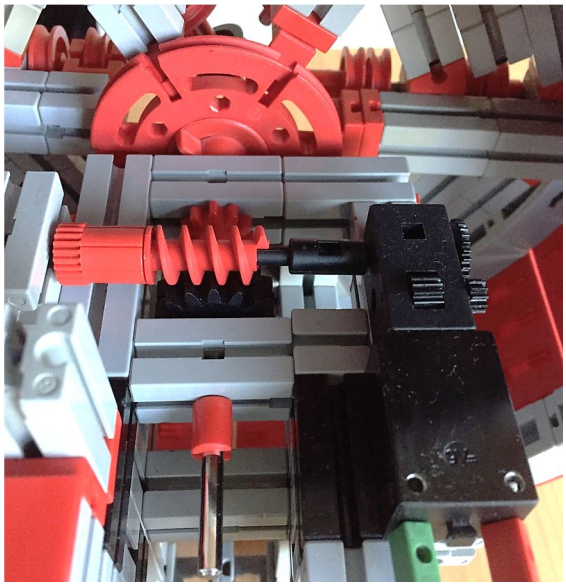


Abb. 6: Bohrkopf-Antrieb

Für den Vortrieb habe ich eine Lösung mit Zahnstangen und einem zweifachen Schneckengetriebe gewählt.

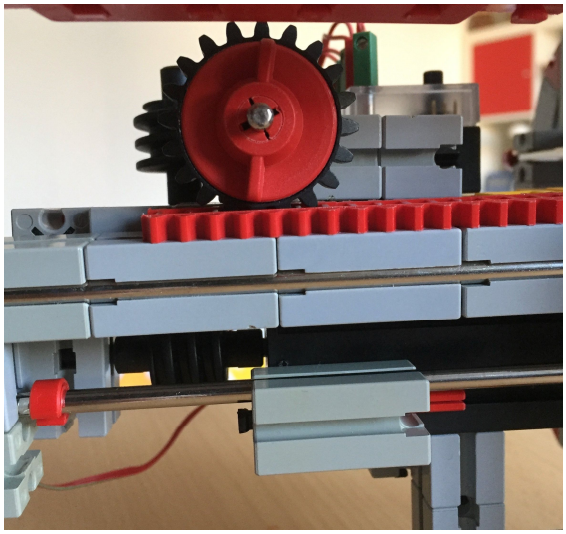


Abb. 7: Vortrieb

Die Einheit mit der Zahnstange und der unteren Stange ist mit dem Bohrkopf verbunden, der Motor ist mit den Grippern verbunden. Durch Abrollen des Zahnrads und Gleiten der Stange im BS30 am Motorblock ergibt sich ein maximaler Vortrieb von etwa 10 cm.

Hier noch der Vortrieb von oben gesehen (Abb. 8).

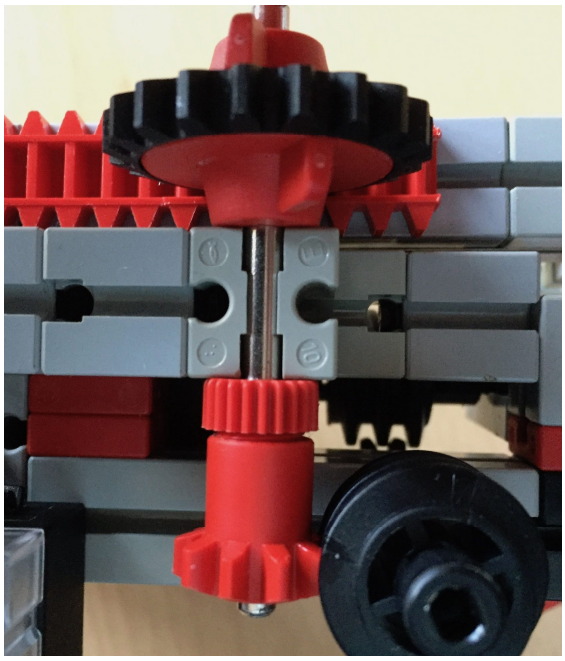


Abb. 8: Vortrieb von oben

Und hier nun die Gripper:

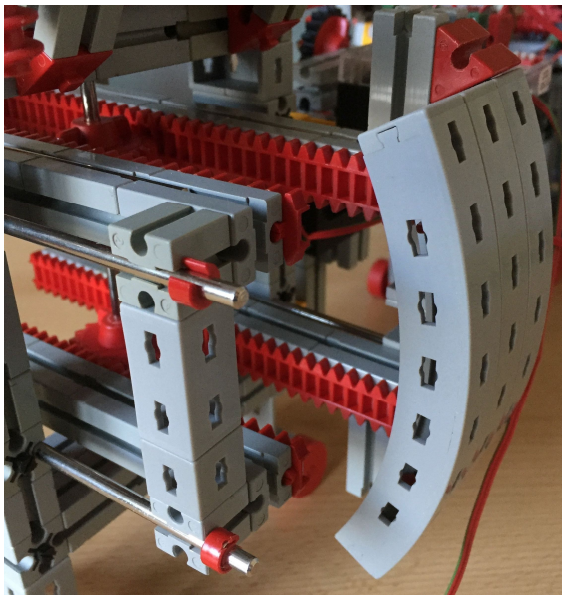


Abb. 9: Gripper

Durch die beiden mittigen Z20 werden die Gripperplatten mit je zwei Zahnstangen gegenläufig nach außen oder innen bewegt. Die Verankerung mit dem Rest der Maschine geschieht auch hier mit langen Stangen, welche in den Nuten der BS30 laufen.

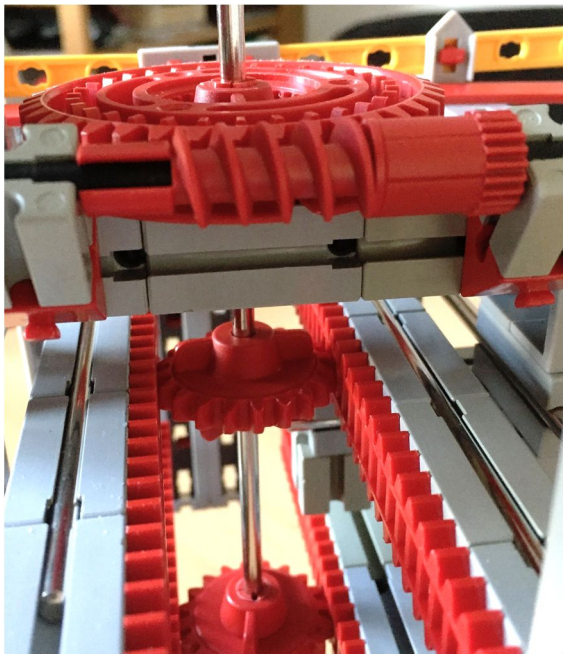


Abb. 10: Gripper von der Seite

Schließlich musste ich noch ein Stück Tunnel bauen, damit sich die Gripperplatten irgendwo abstützen können (Abb. 11).

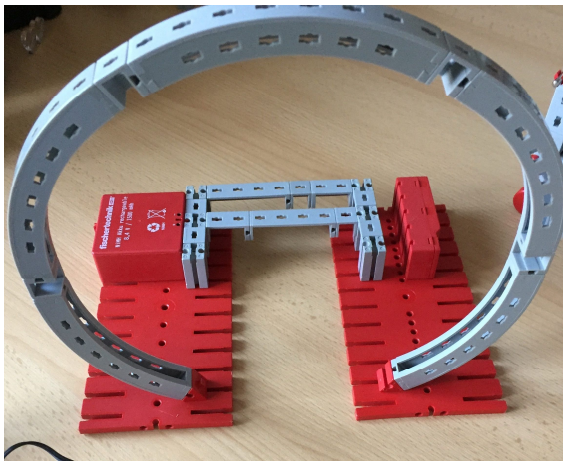


Abb. 11: Tunnelbogen

Wenn nun die Gripper ausgefahren sind, funktioniert das tatsächlich (Abb. 12)

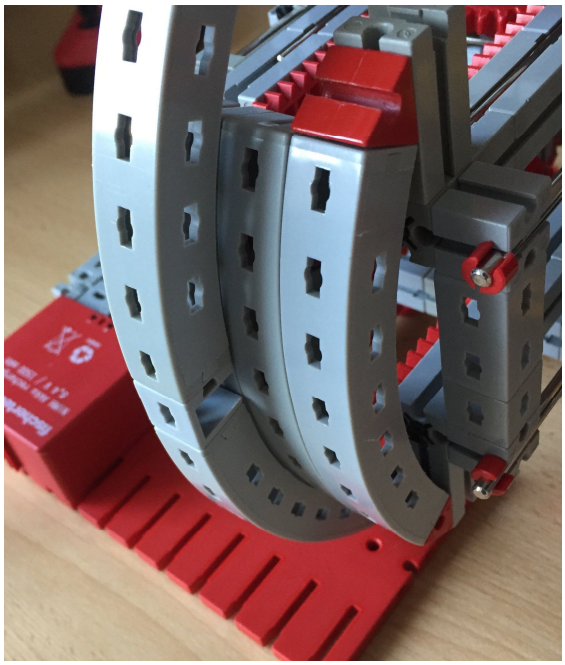


Abb. 12: Gripper an Tunnelwand

Verbesserungsmöglichkeiten

Mangels Hydraulik- bzw. Pneumatic-Teilen habe ich die Hauptelemente mit Zahnrädern und Zahnstangen gebaut. Hier gibt es viele Möglichkeiten, das Modell noch wesentlich wirklichkeitsnäher zu bauen.

Der Vortrieb mit nur einer Zahnstange funktioniert zwar, hier können aber wesentlich robustere Lösungen entwickelt werden.

Ich wünsche viel Freude beim Nachbauen und Austüfteln von noch ausgereifteren Lösungen, und bin gespannt, wann sich das erste Modell durch echten Sandstein arbeitet!

Literatur

[1] Wikipedia: Gotthard-Basistunnel

[2] Wikipedia: Tunnelbohrmaschine

[3] Herrenknecht: Gotthard-Basistunnel

[4] Wikipedia: Tübbing

Tipps & Tricks

Impulsmessung mit dem TX(T)

Dirk Fox, Johann Fox

Wer präzise Steuerungen (Plotter, Roboterarme, Einparkassistenten etc.) mit fischertechnik realisieren will, benötigt möglichst exakte Motorbewegungen. Für diesem Zweck hat fischertechnik Encodermotoren eingeführt, die je Achsumdrehung 75 Impulse liefern und über die Zähleingänge des TX(T) Controllers ausgewertet werden können. Leider sind die Motoren etwas schwach auf der Brust – wer neben Präzision auch Power braucht, muss sich daher etwas einfallen lassen.

Hintergrund

Auch schon vor der Einführung der Encodermotoren konnten mit fischertechnik Drehimpulse gemessen werden. So finden sich schon in den Hobby-Bänden der 70er Jahre interessante Ideen für eine berührungslose Impulsmessung (Abb. 1 [1]): Eine Drehscheibe 60 unterbricht eine Lichtschranke am Außenrand bis zu sechs Mal je Umdrehung.

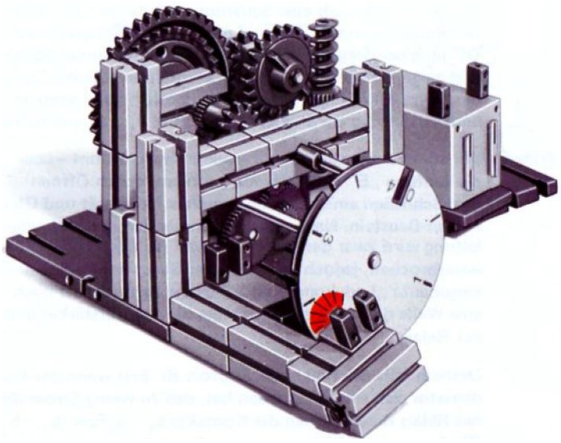


Abb. 1: Lichtschranke mit Drehscheibe 60 als Segmentscheibe (hobby 4 Band 1) [1]

Eine sehr elegante Lösung war die 5-V-Gabellichtschranke im fischertechnik-Gehäuse (Abb. 2), die allerdings nur in einem der ersten Computing-Kästen – dem Trainings-Roboter von 1985 – enthalten war. Durch sie wurde ein mit 32 senkrechten schwarzen, im gleichmäßigen Abstand aufgetragenen Linien beschriftetes Walzenrad geführt. Damit ließen sich je Achsumdrehung 32 Impulse oder 64 (fallende oder ansteigende) Impulsflanken zählen.

Abb. 2: fischertechnik-Gabellicht schranke (32357), 1985

Eine mechanische Lösung nahm fischertechnik 1997 mit den Rast-Impulszahnrädern Z4 ins Programm auf. Damit lassen sich (Mini-)Taster je Achsumdrehung viermal auslösen – vier Impulse oder acht Flanken (Abb. 3). Von Andreas Tacke stammen einige spezielle Anwendungen des Rast-Impulszahnrads [2, 3] – die Auflösung ist jedoch für viele Präzisionssteuerungen nicht fein genug.

Abb. 3: Rast-Impulszahnrad Z4 (37157), 1997

Lösungsideen

Die fischertechnik-Controller TX und TXT verfügen beide über vier schnelle Zählereingänge, die mehrere 100 Hz verarbeiten – genug, um auch deutlich schnellere Impulsfrequenzen als die der Encodermotoren verarbeiten zu können.

Für die Messung schneller Impulse benötigen wir auch schnelle Sensoren. Mechanische (wie z. B. Taster) scheiden dafür aus, da sie einerseits keine sehr hohen Schaltgeschwindigkeiten erreichen und andererseits selbst einen erheblichen Widerstand darstellen.

Eine Alternative bilden die optischen (Fototransistoren) und magnetischen Sensoren (Reed-Kontakte).

Impulsmessung mit Reed-Kontakten

Mit Reed-Kontakten lassen sich Umdrehungszahlen schnell drehender Achsen sehr zuverlässig berührungslos bestimmen. Eine solche Lösung habe ich 2013 zur Drehzahlmessung meiner Elektromotoren eingesetzt [4]: Am Ende der (Rast-) Achse steckte ich einen Baustein 15 mit Magnet (108278) auf einen Rastadapter und richtete einen Reed-Kontakt (36120) darauf (Abb. 4). Die Konstruktion lieferte bei 600 U/min stabil jede Umdrehung einen Impuls.

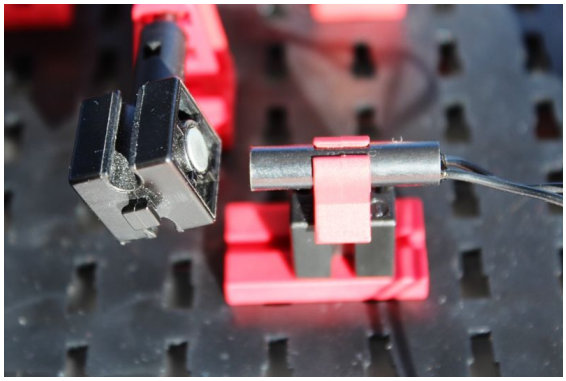


Abb. 4: Drehzahlmessung mit Reed-Kontakt

Für eine präzise Messung ist ein Impuls je Umdrehung jedoch nicht genug; auch ist der Baustein 15 zusätzliche Masse, die vom Motor bewegt werden muss. Bleiben also die optischen Lösungen.

Optische Impulsmessung

Ganz ohne Fremdteile kommt man bei der optischen Messung nicht aus, sofern man nicht eine der seltenen fischertechnik-Gabellichtschranken sein eigen nennt.

Naheliegend ist zunächst die Verwendung einer Standard-Gabellichtschranke wie z. B. der CNY 37, die für kleines Geld zu haben ist. Herausfordernder ist jedoch der Unterbrecher: Bemalte oder bedruckte Folien sind zu lichtdurchlässig, zugeschnittene Pappscheiben zu ungenau. Nicht verwunderlich, dass Andreas Tacke auch für dieses Problem im Bilderpool der fischertechnik-Community eine Lösung vorgestellt hat (Abb. 5): Fünf Segmentscheiben mit 4, 8, 19, 31 und 47 gleichmäßigen Schlitzen (die „krummen“ Werte ergeben sich aus Andreas‘ spezieller Aufgabenstellung).

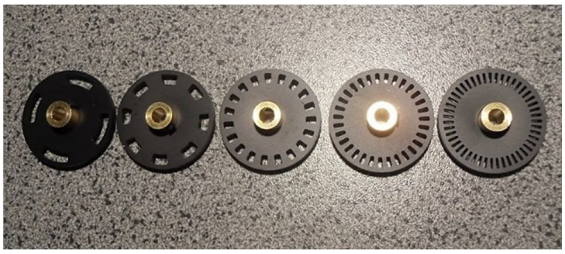


Abb. 5: Segmentscheiben von Andreas Tacke

Warum aber eine schlecht verbaubare Gabellichtschranke einsetzen, wenn man fischertechnik-Fototransistoren besitzt? Auch damit lassen sich Lichtschranken zur Impulsbestimmung konstruieren. Allerdings ist die Eignung der fischertechnik-Linsenlampen für diesen Zweck begrenzt: Damit weder Streulicht der Lampe noch externes Störlicht das Verhalten des Fototransistors beeinträchtigt und dennoch viele schnelle Unterbrechungen gezählt werden können, ist viel Justage erforderlich (Abdeckkappen, Abstände, Abschirmung).

Eine Alternative ist die Verwendung einer Laserdiode, wie von Andreas und Joachim Gail vorgeschlagen [5]. Ein Laser wirft einen so stark gebündelten Lichtstrahl, dass man sogar auf die Segmentscheibe ver-

zichten kann: Die Zähne eines fischertechnik-Zahnrads genügen. Klasse-1-Laser mit einer Leistung von weniger als 0,4 mW sind nahezu ungefährlich – und für wenig Geld in Gehäusegrößen erhältlich, die sich hervorragend in fischertechnik-Bauteile einfügen lassen. Ich verwende einen roten Klasse-1-Punktlaser (3-12 V, 5-25 mA, ca. 10 €) von Picotronic in einem Messinggehäuse von 9 mm Durchmesser und etwa 2,1 cm Länge (Abb. 6).

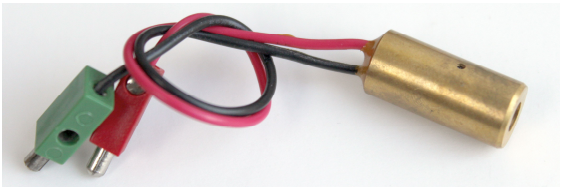


Abb. 6: Klasse-1-Laser von Picotronic

Der Laser lässt sich exakt in die Öffnung einer Schneckenmutter m=1 hineinschieben. Mit der Laser-Schneckenmutter, einem Fototransistor und einem Rast-Z20 lässt sich ein ziemlich kompaktes, berührungsloses Impulsmessmodul konstruieren:

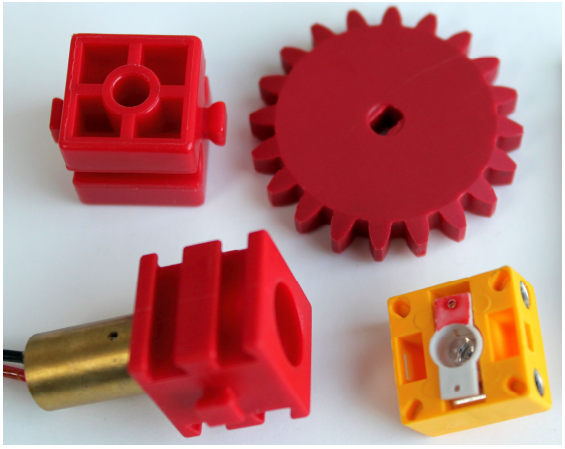


Abb. 7: Bauteile des Laser-Moduls

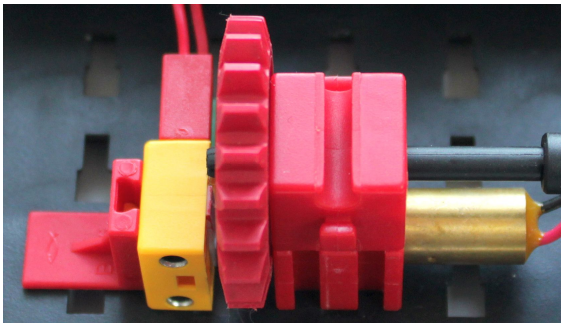


Abb. 8: Fertiges fischertechnik-Laser-Modul

Wie präzise das Modul funktioniert, lässt sich leicht mit dem folgenden Versuchsaufbau nachweisen (Abb. 9): Unsere „Segmentscheibe“, das Rast-Z20, stecken wir auf die Abtriebsachse eines Encodermotors.

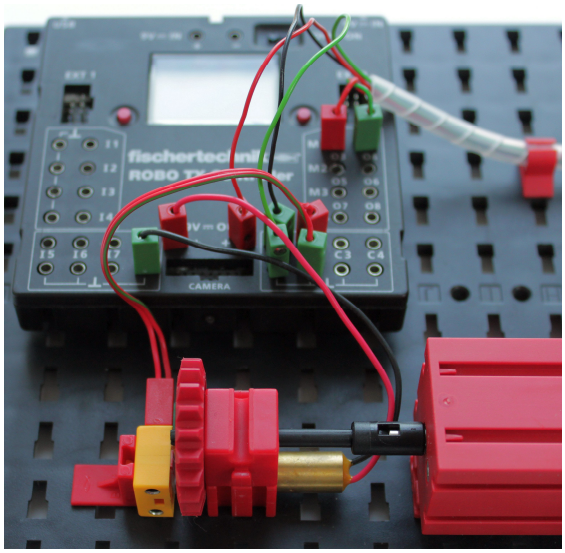


Abb. 9: Testaufbau zur Impulsmessung

Dann schließen wir den Encoder an C1, den Encodermotor an M1 und unser Impulsmessmodul an C2 an.

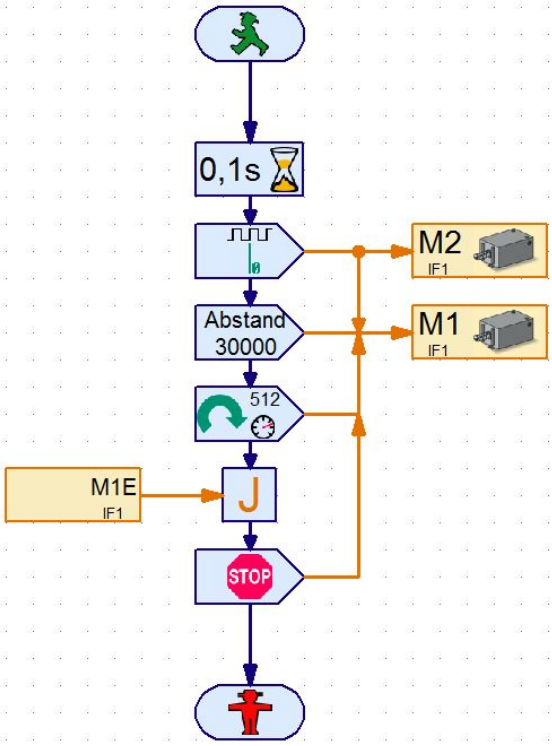


Abb. 10: ROBO Pro-Testprogramm (alter Encoder-Motor: 30.000 Impulse)

Wenn wir den Encodermotor nun einen vorgegebenen Abstand, möglichst eine feste Anzahl an Umdrehungen, „fahren“ lassen, lässt sich leicht prüfen, ob das Laser-Modul die richtige Anzahl an Impulsen erfasst hat.

Dabei liefern die „alten“ Encodermotoren (die mit dem TX vertrieben wurden) 75, die „neuen“ (erkennbar an einer „Wulst“ auf der Rückseite) 63⅓ Impulse je Umdrehung.

Mit einem kleinen ROBO Pro-Testprogramm stoppen wir den alten Encodermotor nach 30.000 Impulsen (≡ 400 Umdrehungen, Abb. 10); für den neuen wählen wir 19.000 Impulse (≡ 300 Umdrehungen). Unser Laser-Modul am Zählereingang C2 sollte dann 400 x 20 = 8.000 bzw. 300 x 20 = 6.000 Impulse gezählt haben. Der Interface-Test (Abb. 11) zeigt das Ergebnis.

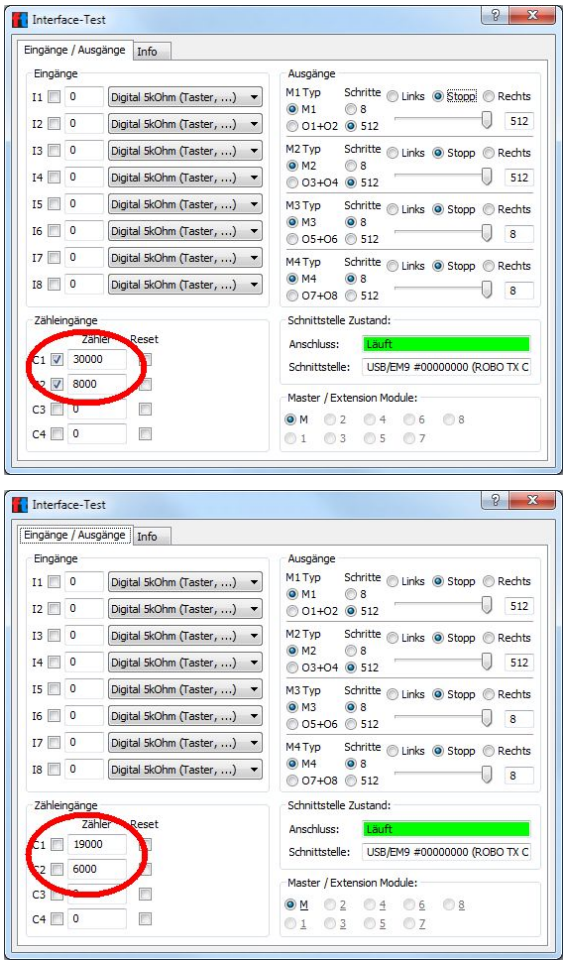


Abb. 11: Korrekte 8.000 (alter Encoder-Motor, oben) bzw. 6.000 Impulse (neuer Encoder-Motor, unten) an C2

Mit unserem Impulsmessmodul wird so aus jedem fischertechnik-Motor ein Encodermotor mit 20 Impulsen je Achsumdrehung. Zur Ansteuerung in ROBO Pro können wir die entsprechenden Motorsteuerungsbefehle verwenden.

Zwar erreicht die Auflösung des Moduls nur knapp ein Drittel der fischertechnik-Encoder. Für die meisten Anwendungen genügt das jedoch – wie zum Beispiel für die Realisierung eines präzisen Einparkassistenten (Abb. 12).

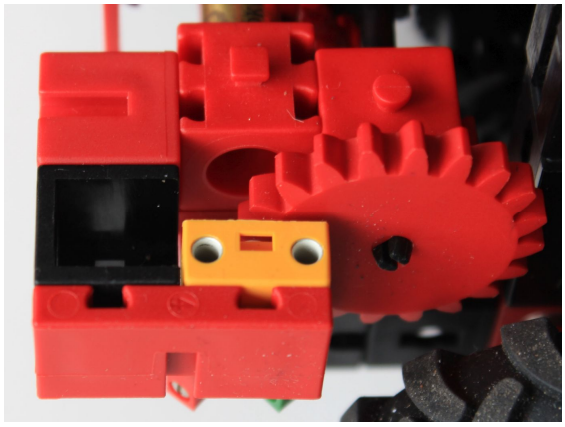


Abb. 12: Einparkassistent mit Laser-Modul zur Motorsteuerung

Um die Auflösung zu erhöhen kann man vor dem Modellantrieb oder dem Messmodul noch eine Unter- oder Übersetzung einfügen. Alternativ könnte man auch ein größeres Zahnrad wie z. B. ein Z40 als Segmentscheibe verwenden [5]: Es liefert 40 Impulse je Umdrehung. Damit wird das Laser-Modul jedoch deutlich voluminöser.

Es geht jedoch noch einfacher: Wir können am Zählereingang des TX(T) sowohl steigende als auch fallende Impulsflanken auswerten. Damit lässt sich die Auflösung des Moduls verlustfrei auf 40 Impulse je Umdrehung verdoppeln; die Auflösung erreicht damit gut die Hälfte der Encodermotoren.

Beim Zusammenbau des Laser-Impulsmessmoduls ist darauf zu achten, dass der Laserstrahl die Evolventen des Zahnrads möglichst genau in der Mitte trifft, damit die Intervalle gleich lang sind.

Das für den Encodermotor vorgesehene ROBO Pro-Kommando ‚Position erreicht' zählt allerdings leider nur Impulse, daher können wir den Motorsteuerungsbefehl ‚Abstand' nicht nutzen. Die Alternative ist die Verwendung des Flanken-Zähl-Befehls (Abb. 13), dem wir eine feste Impulszahl vorgeben können. Dabei stoppt der Motor allerdings nicht ganz so präzise wie bei den Encodermotor-Befehlen: Je nach Übersetzung und Geschwindigkeit des Motors kann es passieren, dass er ein oder zwei Impulse nachläuft.

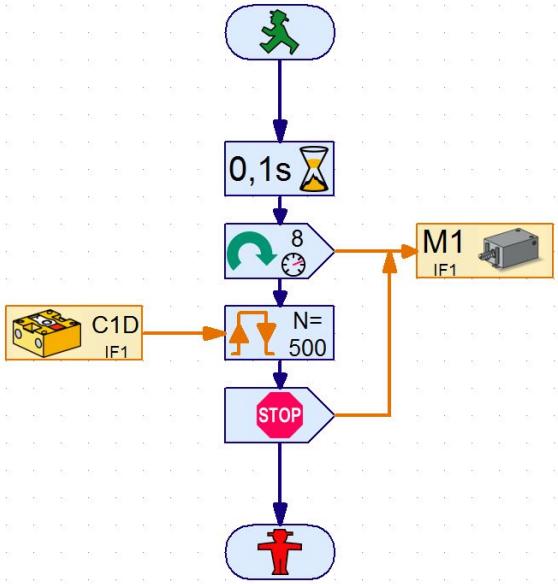


Abb. 13: Alternative Motorsteuerung: Impulszähler an C1 wertet steigende und fallende Flanken des Lasermoduls aus

Wenn wir mehr als wenige 100 Impulsflanken pro Sekunde messen möchten, kommt der TXT Controller nicht mehr mit. Dafür müssten wir das Lasermodul z. B. mit einem Arduino auswerten – der mit über 10 kHz deutlich höhere Impulsfrequenzen bewältigen kann. Das aber wäre Stoff für einen eigenen Beitrag …

Referenzen

[1] Artur Fischer: *fischertechnik hobby. Experimente + Modelle*. Hobby 4 Band 1, Fischer-Werke, 1978.

[2] Andras Tacke: *ft-Spezialteile made by TST (Teil 5): Hubgetriebe mit Impulsrad*. ft:pedia 3/2013, S. 36-37.

[3] Andreas Tacke: *ft-Spezialteile made by TST (Teil 10): Impulsrad*. ft:pedia 4/2014, S. 10-11.

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

[5] Andreas und Joachim Gail: *Laser-Anwendungen (1): Bewegungsmessung*. ft:pedia 1/2015, S. 64-67.

Computing

I²C mit dem TX(T) – Teil 16: Servo-Driver

Dirk Fox

Die fischertechnik-Controller TX und TXT können keine Servo-Motoren ansteuern. Das ist ein Ärgernis, ist doch der Einsatz von Servos nicht nur in Fahrzeugmodellen ein Muss: Immer dann, wenn eine schnelle und Winkel-genaue Motorbewegung (mit geringer Last) benötigt wird, ist ein Servo einem „ausgewachsenen" Elektromotor vorzuziehen – zumal sich der schlanke fischertechnik-Servo auch sehr elegant verbauen lässt. Den TXT kann man sogar mit der IR-Fernsteuerung ansprechen – wegen der fehlenden Servo-Unterstützung kann der TXT das Empfängermodul jedoch nicht ersetzen. Aber auch hier ist, wie so oft bei fischertechnik, Abhilfe möglich.

Servos

Servo-Motoren sind 5V-Elektromotoren mit einer Steuerelektronik, die über ein pulsweitenmoduliertes (PWM-) Signal gesteuert wird. Das Signal stellt den Winkel der Ausgangswelle ein – üblicherweise zwischen 0° und 180°. Diese Position wird vom Servo dann selbstständig gehalten.

Die Steuerelektronik des Servos vergleicht den Ist-Stand der Ausgangswelle über ein Potentiometer mit dem Soll-Wert, der über das PWM-Signal übermittelt wird, und regelt erforderlichenfalls nach.

Eingesetzt werden Servos überwiegend im Modellbau. Sie nehmen einen vorgegebenen Winkel extrem schnell und genau ein – vertragen jedoch meist keine großen Lasten. Bei Überlast (z. B. durch Verklemmen) hauchen sie daher schnell ihr Leben aus; das gilt auch für die fischertechnik-Servos.

Der gewünschte Winkel wird über die Pulslänge eines (in der Regel) 50-Hz-Signals übermittelt, also mit einer Periodenlänge von 20 ms. Die Pulslänge liegt üblicherweise zwischen 0,5 und 2,5 Millisekunden, bei manchen Servos auch zwischen einer und zwei Millisekunden.

Die Werte für das Minimum und das Maximum sollte man für den verwendeten Servo-Typ experimentell bestimmen, um ein Übersteuern (und damit eine Überlast) des Servos zu verhindern.

Die Signal-Frequenz von 50 Hz ist keine strikte Vorgabe; einige Servos akzeptieren bis zu 400 Hz (2,5 ms Periodenlänge). Auch der fischertechnik-Servo arbeitet mit unterschiedlichen Frequenzen; ich empfehle, ihn mit den üblichen 50 Hz anzusteuern.

Servos sind als Lenkmotoren für Modellfahrzeuge ideal. Daher ist es besonders ärgerlich, dass sich die fischertechnik-Servo-Motoren ausschließlich an die Empfänger der (alten) IR- und der (neuen) Bluetooth-Fernsteuerung anschließen lassen. Eine Ansteuerung über den TX(T), den LT-Controller oder den neuen BT Smart Controller ist nicht möglich, was zahlreiche interessante Anwendungen ausschließt.

Aber – wie so oft in der ‚fischertechnik-Welt' – gibt es auch für dieses Problem zumindest für TX und TXT dank des von beiden Controllern unterstützten I²C-Protokolls eine elegante Lösung.

Der 16-Kanal-Servo-Driver

In [1] haben Christian Bergschneider und Stefan Fuss vorgestellt, wie sich das *Arduino Motor Shield* von Adafruit via I²C-Protokoll am TX(T) betreiben und sich dieser damit um zusätzliche Motor-Anschlüsse erweitern lässt. Von Adafruit gibt es ein weiteres interessantes Arduino-Shield, das ausschließlich für die Ansteuerung von (bis zu 16!) Servo-Motoren gedacht ist und ebenfalls über das I²C-Protokoll angesprochen wird – das *16-Channel 12-bit PWM/Servo Shield*. Erfreulicherweise gibt es davon eine in Anschlüssen und Maßen nicht auf den Arduino zugeschnittene Variante mit sehr kleiner Baugröße: den *16-Channel Servo Driver* (Abb. 1). Dieser Servo Driver ist mit ca. 16 € nicht nur günstiger als das Motor Shield, sondern mit 62,5 x 25,4 x 3 mm auch deutlich kleiner, wenn auch ein paar Millimeter zu lang für das fischertechnik-Batteriegehäuse (32263).

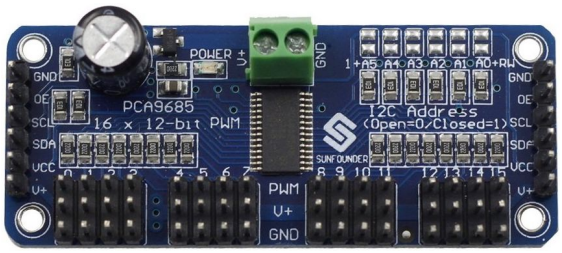


Abb. 1: 16-Kanal Servo Driver (Adafruit)

Anschluss und Stromversorgung

Der Servo Driver benötigt am V+ und GND-Anschluss (Abb. 1, oben Mitte) eine Versorgungsspannung von 5V. Da einzelne Servos unter Last durchaus 1 A und mehr ziehen können, empfiehlt Adafruit für die Ansteuerung mit einer Stromquelle für bis zu 2 A – und beim Einsatz von *n* Servos die Verwendung eines Kondensators an C2 (Abb. 1, links oben) mit einer Kapazität von $100 \cdot n \cdot 100\mu F$ (nicht im Lieferumfang des Servo Driver enthalten).

Eine geeignete 5V-Spannungsversorgung können wir mit einem Spannungswandler – bspw. dem D24V10F5 von Pololu (Abb. 2), erhältlich z. B. bei Exp-Tech für ca. 8 € – aus der 9V-Ausgangsspannung eines TX(T) gewinnen.

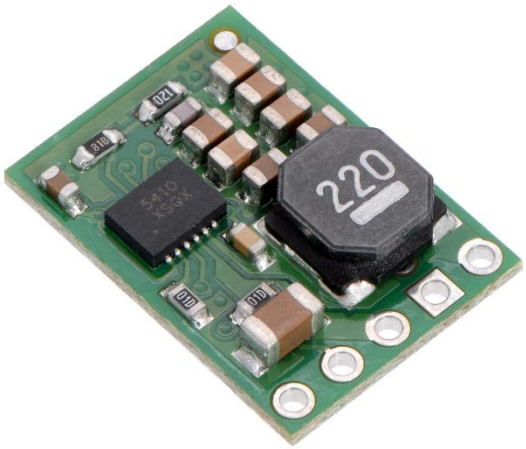


Abb. 2: 5V-1A-Spannungswandler (pololu)

Die I²C-Pins (VCC, SCA, SDL, GND) des Servo Driver können wir beim TX mit dem I²C-Universal-Adapter aus [2] oder mit vier Female-Jumpern direkt mit dem EXT2-Anschluss verbinden.

Beim TXT ist es nicht ganz so einfach, da dessen I²C-Bus mit 3,3 V arbeitet. Die Logik des Servo Driver ist jedoch für 3-5V ausgelegt, daher ist grundsätzlich auch ein direkter Anschluss an die EXT-Schnittstelle des TXT möglich. Allerdings wird von fischertechnik nicht empfohlen, die VCC-Versorgung des EXT-Anschluss zu nutzen, da sie nur eine Last von wenigen mA verträgt. Da das ‚Herz' des Servo Driver, ein PCA9685, maximal 10 mA Strom zieht, sollte der direkte Anschluss an die EXT-Schnittstelle eigentlich unproblematisch sein. Wer auf Nummer sicher gehen will, der beschafft einen weiteren Spannungswandler auf 3,3V – wie den D24V6F3 von Pololu, erhältlich z. B. bei Exp-Tech für ca. 5,50 € – für den VCC- und GND-Anschluss des Servo Driver.

Eine Alternative zu diesem zweiten Spannungswandler ist die Verwendung des I²C-Adapters, den Stefan Fuss und Christian Bergschneider kürzlich in der ft:pedia vorgestellt haben [3].

Die I²C-Pins SCL und SDA können wir über zwei Female-Jumper direkt mit den entsprechenden Pins des EXT-Anschlusses vom TXT verbinden.

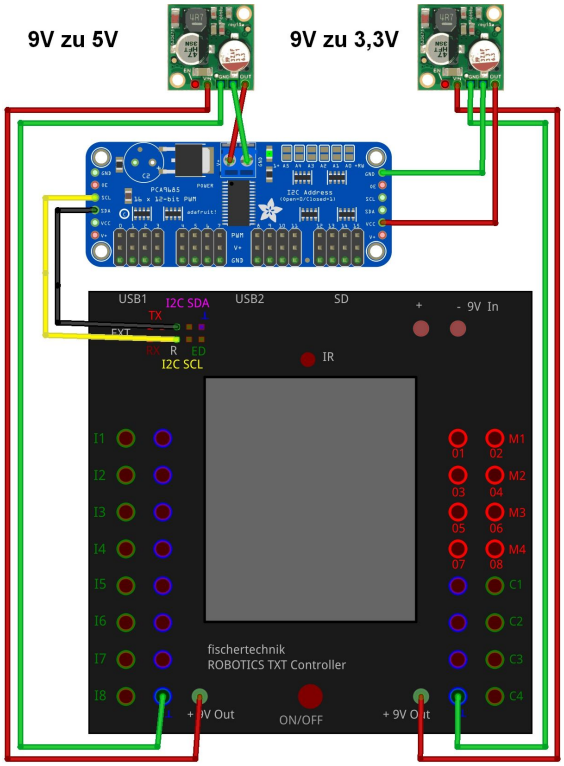


Abb. 3: Anschluss Servo Driver am TXT

Technische Daten

Die technischen Daten des Servo Driver sind denen des Motor Shield sehr ähnlich, da das Herzstück ebenfalls ein PCA9685 [4] ist:

- *Betriebsspannung: 3-5V*
- *Stromaufnahme (ohne Servos): ≤ 10 mA*
- *I²C-Adresse: 0x40 (0x41-0x7F)*
- *I²C-Geschwindigkeit: bis 1 Mbit/s (Fast Mode Plus)*
- *PWM-Auflösung: 12 bit (0-4095)*

Die I²C-Adresse kann hardwareseitig durch das Setzen von Lötpunkten auf einen Wert von 0x41 bis 0x7F geändert werden.

Außerdem reagiert der Servo Driver auf die *All Call Address* 0x70, die alle 16 Servo-Ausgänge gleichzeitig anspricht, und die *General Call Address* 0x00, hier als *Software Reset Address* genutzt. Die drei voreingestellten *Sub Call Addresses* 0x71, 0x72 und 0x74 sind beim *Power Up* deaktiviert.

Befehle und Register

Der PCA9685 verfügt über insgesamt 255 Register, von denen allerdings nicht alle genutzt werden. Tabelle 1 zeigt die nutzbaren Register und ihre Inhalte.

Register	Bedeutung
0x00	*Mode 1 Register*
0x01	*Mode 2 Register*
0x02-04	*Sub Address Register*
0x05	*All Call Address Register*
0x06-45	*LED/Servo Register*
0xFA-FB	*All LED On Register*
0xFC-FD	*All LED Off Register*
0xFE	*Prescale Register*

Tab. 1: Register des PCA9685 [4]

Die im Folgenden angegebenen Default-Einstellungen der jeweiligen Register werden nach jedem *Power Up* (Einschalten der Stromversorgung) und einem *Software Reset* aktiviert.

Mode 1 Register

Im *Mode 1 Register* (0x00) werden die folgenden Eigenschaften und Zustände des PCA9685 (vor)eingestellt:

- **Bit 0**: Aktivierung der *All Call Address* (default: 1)
- **Bit 1-3**: Aktivierung der drei *Sub Call Addresses* (default: 0)
- **Bit 4**: Aktivierung des *Sleep Mode* (default: 1)
- **Bit 5**: *Auto Increment*, d. h. das automatische Hochzählen der Adresse beim Lesen oder Schreiben (default: 0)
- **Bit 6**: Externer Takt (default: 0)

- **Bit 7**: *Restart enable* (default: 0)

Für den Betrieb des Servo Driver muss der *Sleep Mode* bei der Initialisierung deaktiviert werden (0).

Mode 2 Register

Das *Mode 2 Register* enthält die Einstellungen für spezielle Sonderfälle (wie das Invertieren der Ausgänge oder die Einstellung von *Open Drain*), die unsere Servo-Anwendung nicht betreffen – daher verzichte ich hier auf eine nähere Darstellung und auf eine Implementierung im ROBO Pro-Treiber; für Details siehe [4].

Sub und All Call Address Register

In den drei *Sub Address Registern* (0x02-0x04) können I²C-Adressen eingetragen werden, auf die der PCA9685 reagieren soll. Damit lässt sich z. B. eine Auswahl mehrerer am selben I²C-Bus angeschlossenen PCA9685 gleichzeitig adressieren. Voreingestellt sind die Adressen 0x71, 0x72 und 0x74; beim *Power Up* ist deren Nutzung deaktiviert (zu aktivieren über das *Mode 1 Register*).

Das *All Call Address Register* (0x05) erlaubt die Einstellung einer I²C-Adresse, unter der alle über einen I²C-Bus erreichbaren PCA9685 angesprochen werden können. Voreingestellt ist die Adresse 0x70; sie ist nach einem *Power Up* aktiviert.

LED und PWM Control Register

Die 16 PWM-Ausgänge werden mit einer Auflösung von 12 Bit gesteuert – sie lassen sich also in Stufen von 0-4095 (0x000-0xFFF) regeln. Dabei werden der Zeitpunkt innerhalb eines Signal-Zyklus, zu dem ein Ausgang ein- (LED_ON), und der, zu dem er wieder ausgeschaltet (LED_OFF) wird, separat festgelegt. Für die meisten Anwendungen (wie LED- oder Servo-Motor-Steuerung) können wir den LED_ON-Wert auf 0 belassen und über den LED_OFF-Wert die Länge des PWM-Signals festlegen.

Zusätzlich stellen die *PWM Control Register* für jeden der 16 Ausgänge zwei Bits zur Verfügung, über die die Ausgänge *full on* und *full off* gesetzt werden können. Die 16 PWM-Ausgänge werden daher über je vier Byte lange *PWM Control Register* (0x06-0x45) gesteuert, die wie folgt belegt sind:

- **Byte 1, Bit 0-7**: acht niederstwertige Bit (LSB) des 12-Bit-PWM-Wertes für LED_ON
- **Byte 2, Bit 0-3**: vier höchstwertige Bit (MSB) des 12-Bit-PWM-Wertes für LED_OFF
- **Byte 2, Bit 4**: *full on* Bit; wenn gesetzt, werden die LED_OFF-Werte ignoriert
- **Byte 3, Bit 0-7**: acht niederstwertige Bit (LSB) des 12-Bit-PWM-Wertes für LED_OFF
- **Byte 4, Bit 0-3**: vier höchstwertige Bit (MSB) des 12-Bit-PWM-Wertes für LED_OFF
- **Byte 4, Bit 4**: *full off* Bit; wenn gesetzt, werden die LED_ON-Werte ignoriert

Per default (also nach einem *Power Up* oder *Reset*) sind alle *full off*-Bits auf 1, alle weiteren Bits auf 0 gesetzt.

All LED ON/OFF Register

Die Register *All LED on* (0xFA, 0xFB) und *All LED off* (0xFC, 0xFD) belegen jeweils alle entsprechenden 32 PWM Control Bytes mit demselben Wert:

- **0xFA, Bit 0-7**: acht niederstwertige Bit (LSB) des einheitlichen 12-Bit-PWM-Wertes für LED_ON
- **0xFB, Bit 0-3**: vier höchstwertige Bit (MSB) des einheitlichen 12-Bit-PWM-Wertes für LED_ON
- **0xFB, Bit 4**: *full on* Bit
- **0xFC, Bit 0-7**: acht niederstwertige Bit (LSB) des einheitlichen 12-Bit-PWM-Wertes für LED_OFF

- **0xFD, Bit 0-3**: vier höchstwertige Bit (MSB) des einheitlichen 12-Bit-PWM-Wertes für LED_OFF
- **0xFD, Bit 4**: *full off* Bit

Damit kann man über alle 16 Ausgänge zugleich dasselbe PWM-Signal ausgeben.

Prescale Register

Über das *Prescale Register* (0xFE) wird die PWM-Frequenz eingestellt. Möglich sind Werte von 24 Hz (0xFF) bis 1.526 Hz (0x03). Für einen Servo-Motor sind allerdings nur Frequenzen bis 400 Hz sinnvoll, da sonst die Periodenlänge kürzer als 2,5 ms wird (siehe nächster Abschnitt).

Die Berechnung des Prescale-Werts aus der gewünschten Periodenlänge erfolgt nach der folgenden Formel:

$$Prescale = \left(\frac{Taktfrequenz}{4096 \cdot Frequenz}\right) - 1$$
$$= \left(\frac{25\,MHz}{4096 \cdot Frequenz}\right) - 1$$

Der voreingestellte Default-Wert 0x1E entspricht einer Frequenz von 200 Hz. Die für Servo-Motoren typische Frequenz von 50 Hz erhalten wir bei einem Prescale-Wert von 0x79. Der Prescale-Wert kann nur im *Sleep Mode* geändert werden.

ROBO Pro-Treiber

Der ROBO Pro-Treiber für den Adafruit Servo Driver umfasst die im Folgenden vorgestellten Kommandos und kann im Downloadbereich der ft:c oder von meiner ROBO Pro-I²C-Treibersammlung heruntergeladen werden:

- `PCA9685_Init`: Durchführung eines Software-Reset (0,5 ms), Aufwecken aus dem *Sleep Mode*.
- `PCA9685_SetPWM`: Setzt den „LED_ON“ und den „LED_OFF“-Wert für einen der Kanäle (PWM-Ausgänge) 0-15.
- `PCA9685_SetPWMFreq`: Berechnet aus einem Frequenzwert von 24 bis 1.526 Hz den zugehörigen Prescale-Wert und trägt ihn ein (kurzzeitiger Wechsel in den *Sleep Mode*).
- `PCA9685_SetS1Address`: Setzt eine neue Sub-Adresse S1, S2 oder S3 (Default: 0x71, 0x72, 074) und aktiviert sie im *Mode 1 Register*.
- `PCA9685_SetACAddress`: Setzt eine neue *All Call Address* (Default: 0x70) und aktiviert sie im *Mode 1 Register*.
- `PCA9685_Reset`: Auslösen eines Software-Resets über die Adresse 0x00.

Servo-Ansteuerung

Die Werte zur Winkelansteuerung eines 180°-Servos hängen von der über den Prescaler eingestellten Frequenz und die Auflösung des PWM-Werts ab: Bei typischen 50 Hz und einer 12-Bit-PWM-Auflösung entsprechen 0,5 ms (≈ 0°) einem PWM-Wert von etwa 100 und 2,5 ms (≈ 180°) einem Wert von 512. Allgemein lassen sich das Minimum und das Maximum für die Servo-Ansteuerung wie folgt berechnen:

$$Servo_{min} = 0{,}5\text{ ms} \cdot \frac{Frequenz}{1000} \cdot 4095$$

$$Servo_{max} = 2{,}5\text{ ms} \cdot \frac{Frequenz}{1000} \cdot 4095$$

Damit $Servo_{max} \leq 4095$ dürfen maximal 400 Hz als Frequenz eingestellt werden. Bei manchen Servos liegen die Pulslängen in einem kleineren Intervall, daher sollten $Servo_{min}$ und $Servo_{max}$ experimentell angepasst werden, um den Servo im Betrieb nicht zu übersteuern.

Das Anbaugehäuse des fischertechnik-Servo hat außerdem einen mechanischen „Anschlag“ bei etwa 30° bzw. 150°. Bei einer Frequenz von 50 Hz konnte ich mit dem untenstehenden einfachen ROBO Pro-Programm (Abb. 3) die folgenden PWM-Werte für $Servo_{min}$ und $Servo_{max}$ ermitteln:

- 30°: ca. 187 ($Servo_{min}$ mit Gehäuse)
- 90°: ca. 318

- 150°: ca. 446 (*Servo$_{max}$* mit Gehäuse)

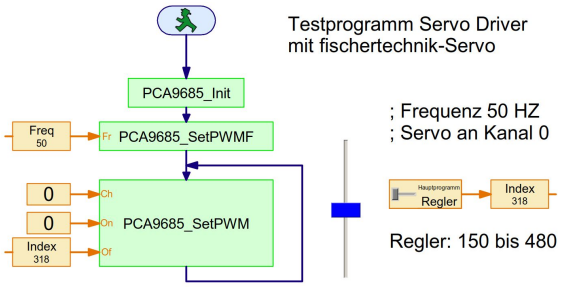


Abb. 3: Testprogramm zur Bestimmung von Servo$_{min}$ und Servo$_{max}$

Nach jedem *Power Up* des Servo Driver wird der Servo automatisch auf die Mittelstellung (90°) eingestellt.

IR-Fernsteuerung

Eine nahe liegende Anwendung für den Servo-Treiber ist natürlich die Steuerung der Achsschenkellenkung eines Fahrzeugs mit einem Lenk-Servo. Da der TXT über eine IR-Empfängerdiode verfügt, kann er die Signale der fischertechnik-Fernsteuerung empfangen und auswerten. Damit können wir ein Fahrzeug mit umfangreichen Steuerungsmöglichkeiten konstruieren, denn der TXT kann auch die DIP-Schalter-Stellungen auswerten [5] und darüber die Steuerhebel mit weiteren Funktionen belegen – oder auch unterschiedliche IR-Fernsteuerungen unterscheiden.

Das Beispielprogramm in Abb. 5 zeigt, wie sich der Lenk-Servo in ROBO Pro über die IR-Fernbedienung ansteuern lässt. Der linke Steuerhebel bestimmt (in Y-Richtung, also vor und zurück) die Geschwindigkeit des Antriebsmotors M1, der zuvor auf eine Auflösung von 512 eingestellt werden muss. Negative Joystick-Werte kehren die Antriebsrichtung um.

Der rechte Steuerhebel ist (wie beim IR-Empfängermodul) für die Lenkung zuständig: Der vom Joystick gelieferte Wert aus {-15, …, 15} wird zunächst auf den Gültigkeitsbereich {*Servo$_{min}$*, …, *Servo$_{max}$*} des Servos abgebildet und dann an den Servo Driver übergeben. Der Servo wird an den Anschluss von Kanal 0 angesteckt.

So, nun wünsche ich Euch viel Spaß mit Servo-Motoren am TX(T) – und bin gespannt auf die zahlreichen Modelle, die diese neue Fähigkeit des Controllers nutzen werden.

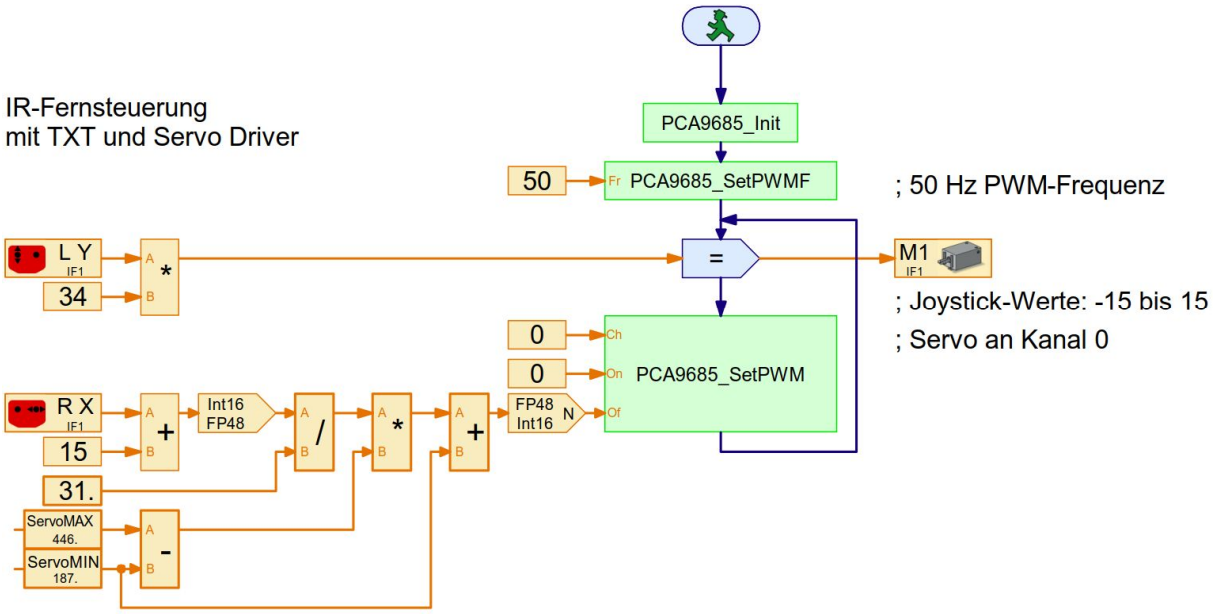


Abb. 4: Beispielprogramm: Servo-Steuerung mit IR-Fernbedienung

Referenzen

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

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

[3] Christian Bergschneider, Stefan Fuss: *Ein universeller I²C-Adapter für den TX(T)*. ft:pedia 4/2016, S. 72-79.

[4] NXP Semiconductors: *PCA 9685 16-channel 12-bit PWM Fm+ I²C-bus LED controller*. Product Data Sheet, Rev. 4, 16.04.2015.

[5] Dirk Wölffel: *Große Modelle mit nur einer fischertechnik-IR-Fernsteuerung ansteuern*. ft:pedia 3/2016, S. 33-34.

Computing

Codes der fischertechnik-Infrarot-Fernsteuerungen (2)

Dirk Uffmann

Da ich noch wesentliche, neue Erkenntnisse zu den beiden fischertechnik-Infrarotsendern gewinnen konnte, möchte ich diese hier als Nachtrag zu meinem Artikel in ft:pedia 3/2016 [1] darstellen. Mit der Kenntnis dieser Codes können die fischertechnik-Fernbedienungs-Sender zur Steuerung von Mikro-Controllern bzw. selbst gebauten Fernbedienungs-Empfängern genutzt werden. Gleiches gilt für den Bau eigener Fernbedienungssender zur Steuerung der fischertechnik-Fernbedienungs-Empfänger.

Abb. 1: Fernbedienungssender aus fischertechnik 30344 IR Control Set

Teil 1: fischertechnik 30344 IR Control Set

Nachdem ich nun auch in den Besitz eines Senders aus dem fischertechnik 30344 IR Control Set gekommen bin, habe ich die noch fehlenden Codes für den blauen Zusatzempfänger ermittelt. Das Ergebnis ist in Tabelle 1 dargestellt. Der schwarze Empfänger aus fischertechnik 30344 wird mit folgender Systemadresse angesprochen:

0x1B (hex) bzw. **27** (dec)

Der blaue Zusatz-Empfänger fischertechnik 30183 wird dagegen mit folgender Systemadresse angesprochen:

0x1E (hex) bzw. **30** (dec)

Die Umschaltung zwischen den beiden Systemadressen erfolgt über die Tasten 1 und 2. In Tabelle 1 sind die Command-Codes der Tasten dargestellt. Diese sind genau entsprechend der Nummerierung der Tasten in der Bedienungsanleitung codiert. „M1 Leistung“ bedeutet, bei Motor 1 die Leistungsstufe umzuschalten, d. h. zwischen langsam und schnell hin und her zu schalten. Mit diesen Codes kann man den Empfänger nun z. B. mit einem Arduino-

Board oder anderen Mikrocontrollern steuern.

Taste	Systemadresse (schwarzer / blauer Empfänger)	Code [hex]
M1 rückwärts	0x1B / 0x1E	0x07
M1 vorwärts	0x1B / 0x1E	0x08
M1 Leistung	0x1B / 0x1E	0x03
M2 links	0x1B / 0x1E	0x09
M2 rechts	0x1B / 0x1E	0x0A
M2 Leistung	0x1B / 0x1E	0x04
M3 Linkslauf	0x1B / 0x1E	0x01
M3 Rechtslauf	0x1B / 0x1E	0x02
M3 Leistung	0x1B / 0x1E	0x05
1 (Umschalten auf schwarzen Emfänger)	0x1B	0x0B
2 (Umschalten auf blauen Emfänger)	0x1E	0x06

Tab. 1: RC5 Fernbedienungscodes für den schwarzen Empfänger aus 30344 und den blauen Zusatz-Empfänger 30183

Teil 2: fischertechnik 500881 Control Set

Bei meinen Versuchen hatte sich herausgestellt, dass es immer wieder zu Fehlern beim Empfang der Daten kommt. Ich habe dann mit verschiedenen IR-Empfänger-ICs gearbeitet und hatte folgende Fehlerraten:

- SFH5110-38 (Osram): ca. 5-7% Parity-Fehler
- TSSP4P38: ca. 20% Parity-Fehler sowie unerkannte Doppelfehler, die ein korrektes Parity-Bit haben.

Dieses letztere IC ist aber eigentlich für Entfernungsmessungen gedacht und nicht als Empfänger für Fernbedienungen, da es einen speziellen logarithmischen AGC (*Automatic Gain Control*) hat.

Ähnlich schlecht lief es auch mit einem SFH5110-36 (36 kHz-Empfänger, der ist ja auch nicht wirklich geeignet für 38 kHz-Modulation).

Ich nehme an, dass am Robo-Interface ebenfalls ein 36 kHz-Empfänger verbaut ist, da der ja für die alte Fernbedienung aus 30344 mit RC5-Code gedacht war. Deshalb wundern mich die niedrigen Fehlerraten von Ad [6].

Meine Recherchen haben ergeben, dass die von mir verwendete Empfänger-ICs nicht gut geeignet waren für das von fischertechnik beim 500881 Control Set verwendete Modulationsverfahren mit kurzen Infrarot-Bursts. Die Ursache für die Fehler liegt in dem für das Timing der gesendeten Pulse unpassenden *Automatic Gain Control* (AGC) dieser Infrarotempfänger.

fischertechnik verwendet Bursts mit 16 Perioden der 38-KHz-Infrarot-Pulse, also 420µs für einen Burst. Das ist um einiges kürzer als die Bursts, die typischerweise in TV-Fernbedienungen verwendet werden, z. B. 550µs beim NEC-Protokoll.

Zudem sind bei fischertechnik die Unterschiede in den Pausenlängen zwischen den Bursts zum Unterscheiden der Bits mit jeweils 100µs extrem knapp ('00' = 380µs Pausenlänge, '01' = 480µs, '10' = 580µs, '11' = 680µs). Wenn da der Empfänger nicht optimal passt, sind Fehler zu erwarten.

Für kurze Bursts werden Infrarot-Empfänger-IC mit anderer Zeitkonstante für den AGC empfohlen, z. B. TSOP34338 mit AGC3, die allerdings schwerer zu beschaffen sind als TSOP4838, TSOP4438 und TSOP34438 mit AGC4, die für Standard-TV-Fernbedienungen geeignet sind. Die TSOP34338 bekommt man leider nur bei den Distributoren, also Digi-Key, Farnell, Rutronik etc.

Es ist mir gelungen, fünf Samples vom TSOP34538 mit AGC5 zu beschaffen, der dem TSOP34338 mit AGC3 ähnelt, nur mit stärkerer Störunterdrückung für Energiesparlampen. Die Fehlerhäufigkeit liegt damit unter 1%.

Alternativ könnte man noch einen TSOP4138 mit AGC1 nehmen (gleiche Zeitkonstante wie bei AGC3), die besitzen allerdings keine Unterdrückung von Störsignalen durch Fluoreszenzlampen. Von mir empfohlene Empfänger-ICs für den

Infrarotsender aus fischertechnik 500881 sind solche mit AGC1, AGC3 oder AGC5, die jeweils die gleiche Zeitkonstante haben und sich nur in der Störunterdrückung von Fremdlichtquellen unterscheiden, also z. B.

- TSOP 34338 (AGC3)
- TSOP 34538 (AGC5)
- TSOP 4138 (AGC1)
- TSOP 4338 (AGC3)

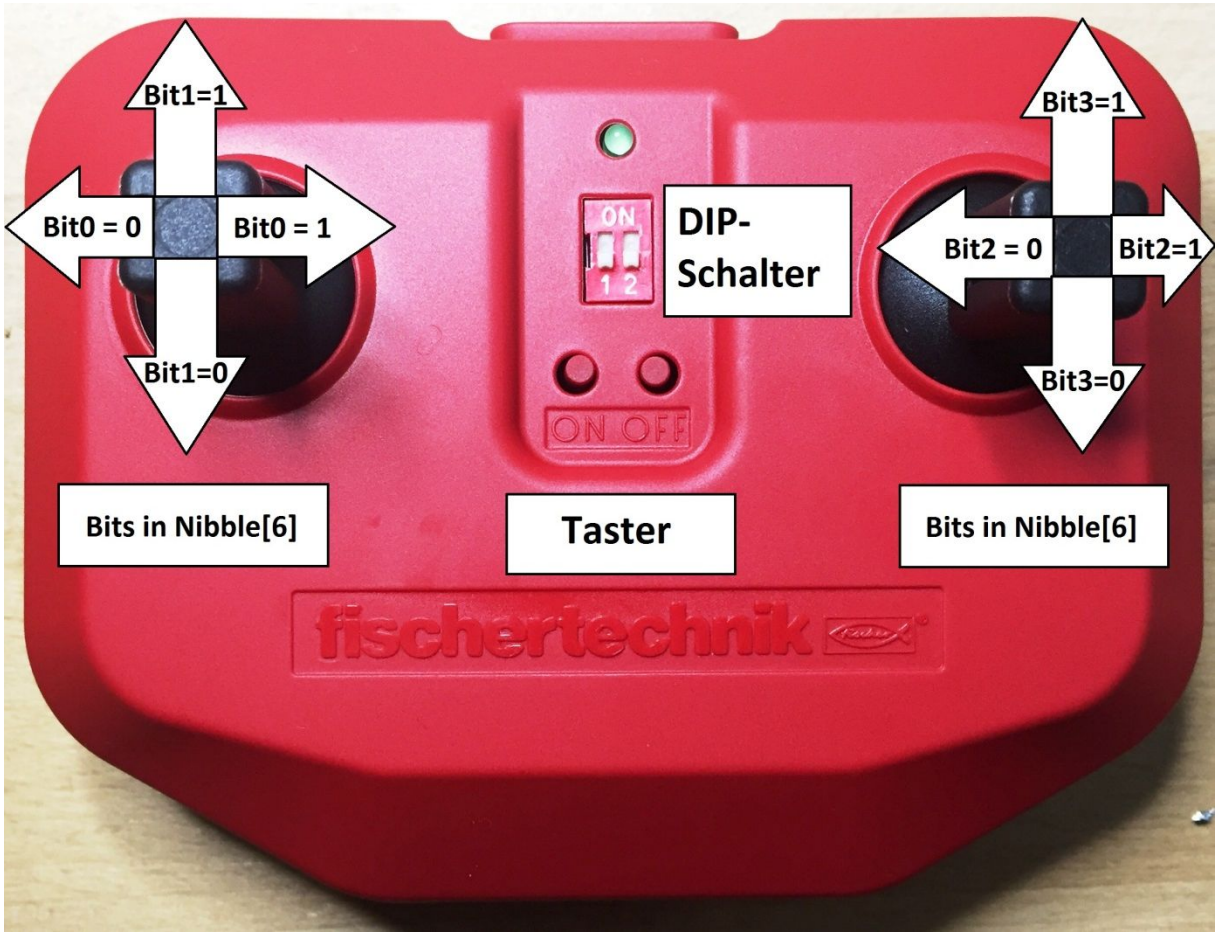


Abb. 2: IR-Fernbedienungssender aus 500881 mit der Zuordnung der Richtungs-Bits in Nibble [6].

Quellennachweis

[1] Uffmann, Dirk: *Codes der fischertechnik-Infrarot-Fernsteuerungen*. ft:pedia 3/2016, S. 35-42.

[2] fischertechnik: *30344 IR Control Set*, ft-Datenbank.

[3] opendcc: RC5 IR-Codes

[4] sprut: *IR-Fernbedienung – der RC-5 Code*.

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

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

Computing

Programmierung des TX in Java, C, C++, C# und Logo

Helmut Jawtusch

Zweifellos ist ROBO Pro eine leistungsfähige Programmiersprache nicht nur für Einsteiger – und geradezu prädestiniert für die Programmierung endlicher Automaten. Sie hat aber auch ihre Grenzen: Bei komplexen Programmen ist das Debugging mühsam, Fehlercodes sind nicht dokumentiert, die Sensorabfrage ist äußerst langsam und bei großen Programmen kommt es auch mal zu Abstürzen der IDE. Da wünscht man sich manchmal eine „vernünftige" Programmiersprache herbei – beim TX ist das keine große Sache.

In diesem Beitrag stelle ich vor, wie man den TX-Controller – ohne Verwendung von ROBO Pro – auf einer Windows-Plattform mit den Sprachen C++, C#, Java, Terrapin Logo und C programmieren kann.

Einleitung

Bis Januar 2012 hat sich Ulrich Müller um die Programmierung der fischertechnik-Interface gekümmert und auf seiner Webseite ftcomputing.de zahlreiche umfish-Treiber veröffentlicht. Inzwischen ist der letzte Stand seiner Webseite über das Archiv der fischertechnik-Community erreichbar. Ulrich Müllers letzter Treiber war umFish50.dll für den TX Controller (Firmware V1.18) mit der zugehörigen Dokumentation umFish50.pdf.

Am 13.11.2012 veröffentlichte fischertechnik die bis heute aktuelle Windows Library „ftMscLib.dll" im Download Programming ROBO TX Controller v1.5 inklusive Dokumentation (Windows_Library_ftMscLib). Diese Library unterstützt die Firmware V1.30 des TX-Controllers und damit auch den I^2C-Port (EXT2).

Ich habe umfish50.dll an die aktuelle Library ftMscLib.dll angepasst und um Befehle für das GPS-Modul Navigatron [1] sowie die Pixy-Kamera CMUcam5 [2] erweitert. Die neuste Treiberversion habe ich am 23.02.2017 über die Download-Seite der fischertechnik-Community zur Verfügung gestellt. Kopiere alle Treiber unter Windows 7 (32 bit) nach

```
..\windows\system32
```

und unter Windows 7 (64 bit) nach

```
..\windows\syswow64.
```

Im Folgenden behandele ich zuerst Programmbeispiele in den Sprachen C++, C#, Java und Terrapin Logo. Am Schluss gebe ich Hinweise, wie man mit dem C-Compiler Programmierpaket der MSC GmbH auf fischertechnik.de C-Programme schreiben und in den Speicher des TX laden kann.

Alle Programme habe ich unter Windows 7 (32 bit) und Visual Studio 2010 bzw. mit BlueJ 3.1.7 getestet und in dem Packet TX-Beispiel-Programme auf der Webseite der fischertechnik-Community zum Download bereitgestellt.

Der USB/Bluetooth-Treiber

Für alle Sprachen muss zuerst aus dem Packet *KeLib_V1_78a → Driver* mit `setup.exe` der Port-Treiber installiert werden. Bisweilen blockieren andere USB-

Treiber den TX-Treiber; dann muss KeLib deinstalliert und nach einem Neustart neu installiert werden.

Die COM-Nummer

Die emulierte COM-Nummer kann man bei USB über den Gerätemanager in der Systemsteuerung und bei Bluetooth über die Eigenschaften oder den Status des Bluetooth-Geräts ermitteln.

ROBO Pro überträgt die aktuelle Firmware V1.30 auf den TX-Controller, wenn man den Interface-Test aufruft.

Ein C++-Programm

Vorbereitung: *Datei → Neu → Projekt → CLR-Konsolenanwendung*

Füge nun dem Projekt das Element `tx.h` hinzu und oben im Quellkode `#include "tx.h"`. Unter *Projekt → ...Eigenschaften → Linker → Eingabe* muss bei Zusätzliche Abhängigkeiten noch `tx.lib` eingetragen werden. `tx.h` und `tx.lib` befinden sich im Paket TX-Beispiel-Programme und sollten in den Projektordner kopiert werden. Die aktuelle COM-Nummer muss bei

`txOpenController(..)`

eingetragen werden.

In `tx.h` findest du alle tx-Befehle mit den jeweils möglichen Parametern. Achtung: Die Nummerierung der Ein- und Ausgänge beginnt intern bei 0 und ist daher gegenüber der Beschriftung auf dem TX jeweils um eins keiner.

```
#include "stdafx.h"
#include <Windows.h>
#include "tx.h"
#include <conio.h>

using namespace System;

void main() {
  int res = txOpenController(95); // com_Nr
  if (res == 0)
    Console::WriteLine("TX-Controller an COM95 verbunden -\n");
  else {
    Console::WriteLine(L"OpenError");
    return;
  }
  Console::WriteLine("---Zum Blinken:Taster I3 drücken ---");
  while(txGetInput(0, 2) == 0)
    Sleep(12);  // Warten auf I3
  Console::WriteLine("--- Bei der Arbeit ---");
  txSetLamp(0, 4, 512);
  Sleep(200);   // Summer an O5
  txSetLamp(0, 4, 0);
  for (int i=0; i<10; i++) {   // Lampen an O7 und O8
    txSetLamp(0, 6, 512);
    txSetLamp(0, 7, 0);
    Sleep(200);
    txSetLamp(0, 6, 0);
    txSetLamp(0, 7, 512);
    Sleep(200);
  }
  txSetLamp(0, 7, 0);
  Console::WriteLine("Das war's");
  txCloseController();
  getch();
}
```

Listing 1: C++-Beispielprogramm

Ein C#-Programm

Vorbereitung: *Datei → Neues Projekt → Konsolenanwendung*

Alle benötigten TX-Befehle können mit `DllImport("TX.dll")` importiert werden. `tx.h` und `tx.lib` werden nicht benötigt, aber in `tx.h` findet man alle verfügbaren TX-Befehle.

Bei laufendem Programm kann man mit ESC bequem alle Lampen und Motoren abschalten. Dies ist nützlich, wenn laufende Motoren blockiert werden.

Statt

```
System.Threading.Thread.Sleep(..);
```

kann `txSleep(..)` verwendet werden.

```
using System;
using System.Runtime.InteropServices;
namespace ConsoleApplication1 {
  class Program {
    [DllImport("TX.dll")]
    private static extern int txOpenController(int comnr);
    [DllImport("TX.dll")]
    private static extern int txGetInput(int ctrl, int io);
    [DllImport("TX.dll")]
    private static extern int txSetLamp(int ctrl, int io, int power);
    [DllImport("TX.dll")]
    private static extern int txCloseController();
    [DllImport("TX.dll")]
    private static extern int txSleep(int ms);
    [DllImport("TX.dll")]
    private static extern int txEsc();
    static void Main(string[] args) {
      int res = txOpenController(95);
      if (res == 0)
        Console.WriteLine("TX-Controller an COM95 verbunden \n");
      else {
        Console.WriteLine("OpenError");
        return;
      }
      Console.WriteLine("--- Zum Blinken: Taster I3 drücken ---");
      while(txGetInput(0, 2) == 0)
        txSleep(12);   //Warten auf  I3
      Console.WriteLine("--- Bei der Arbeit ---");
      txSetLamp(0, 4, 512);
      txSleep(200);    //Summer
      txSetLamp(0, 4, 0);
      while (0 == txEsc()){         // Blinken bis ESC
        txSetLamp(0, 6, 512);
        txSetLamp(0, 7, 0);
        txSleep(200);
        txSetLamp(0, 6, 0);
        txSetLamp(0, 7, 512);
        txSleep(200);
      }
      txSetLamp(0, 7, 0);
      Console.WriteLine("Das war's");
      txSleep(1000);
      txCloseController();
    }
  }
}
```

Listing 2: C#-Beispielprogramm

Ein Java-Programm

Die Java-Entwicklungsumgebung BlueJ erleichtert die Einführung in die Objektorientierte Programmierung. Auch deshalb wird Java im Informatik-Unterricht der gymnasialen Oberstufe immer häufiger eingesetzt. Das java-Packet `ftcomputing.roboTX.jar` basiert auf den Treibern `ftMscLib.dll`, `javafish50.dll`, `TX.dll` und beinhaltet die Klasse FishFaceTX.

Vorbereitung: Unter *Werkzeuge* → *Einstellungen…* → *Bibliotheken* muss

`C:\Windows\System32\ftcomputing.roboTX.jar`

hinzugefügt werden (oder mit aktuellem Pfad). Die Klasse FishFaceTX wird durch `import ftcomputing.roboTX.*` mit den Methoden `openController`, `getInput`, `setLamp`, `finish()` usw. eingebunden. Eine vollständige Auflistung aller Methoden befindet sich in `FishFaceTX.doc`.

`finish()` wird mit der ESC-Taste auf `true` gesetzt. Die Klasse FishFaceTX wird im Tutorial `Eclipse34FishTX.pdf` ausführlich erläutert; dort finden sich viele nützlich Hinweise für die Programmierung mit Java.

```
import ftcomputing.roboTX.*;   /* Lampen an O8, O7 und O6 blinken. */

public class TXBasis {
  public FishFaceTX tx = new FishFaceTX();
  public static void main() {
    TXBasis tx = new TXBasis();
    try { tx.Action(); }
    catch(FishFaceException eft) {System.out.println(eft);}
    finally {System.exit(0);}
  }
  private void Action() {
    tx.openController("com95"); //usb oder bluetooth
    System.out.println("Action gestartet - Ende mit ESC");
    hupen(150);
    hupen(600);
    System.out.println("Drücke I3");
    while (!tx.getInput(Unv.I3)) {tx.pause(20);}
    do {
      tx.setLamp(Out.O8,500);
      tx.setLamp(Out.O7,0);
      tx.setLamp(Out.O6,0);
      tx.pause(200);
      tx.setLamp(Out.O8,0);
      tx.setLamp(Out.O7,500);
      tx.setLamp(Out.O6,500);
      tx.pause(200);
    }
    while (!tx.finish()); // Abbruch mit ESC
    tx.setLamp(Out.O8,0);
    tx.closeController();
    System.out.println("--- FINITO ---");
  }
  private void hupen(int ms) {
    tx.setLamp(Out.O5,500);
    tx.pause(ms);
    tx.setLamp(Out.O5,0);
    tx.pause(150);
  }
}
```

Listing 3: Java-Beispielprogramm

Ein Terrapin-Logo-Programm

In der Datei `TX_Erweiterung.lgo` werden alle TX-Befehle aus `TX.dll` importiert. Beispiel `txlamp`:

```
TO txlamp :Nr :Speed
IGNORE (.WINDOWS [|TX.dll|
|txSetLamp|] :ftiMain :NR - 1
:Speed)
END
```

Die Datei `init.lgo` enthält Spracherweiterungen, die automatisch beim Starten von `logo.exe` geladen werden. Füge deshalb den Inhalt der Datei `TX_Erweiterung.lgo` in die Datei `init.lgo` hinzu und speichere sie im Logo-Stammverzeichnis.

Terrapin Logo (V3.0a vom 03.11.2008) ist eine schöne Sprache, die sich schon ab Klasse 6 im Unterricht einsetzen lässt. Der Entwickler ist *Terrapin Software* (Cambridge, USA).

Durch die TX_Erweiterung können Schüler auf einfache Art Programme für TX-Modelle entwickeln. Besonders geeignet ist die Programmierung des Gabelstaplers aus dem Baukasten ROBO TX Training Lab, der gelegentlich bei ebay angeboten wird.

```
TO start
ct ts
txopen 95 ; Verbindung über USB
pr "|TX über USB (com95) verbunden| pr()
pr "|Drücke Taster 3 ...|
while [not txinput 3] [] ; Warte auf Taster 3
pr "|--- Blinken ---> Ende mit ESC|
txlamp 5 500 wait 100 txlamp 5 0
while [not txesc] [
  txlamp 7 512
  txlamp 8 0 wait 200
  txlamp 7 0
  txlamp 8 512 wait 200
]
txclose
pr "|...Verbindung wurde unterbrochen|
pr "|Ende|
END
```

Listing 4: Logo-Beispielprogramm

Programmierung mit C

Die Datenübertragung per Bluetooth zwischen dem TX und dem PC ist recht langsam. Um z. B. ein Datenpacket eines von der Pixycam [2] gesichteten Objektes zu übertragen benötigt man ca. 350 ms. Per USB dauert es ca. 80 ms. Verwendet man ein auf den TX geladenes C-Programm, dann entfallen diese Übertragungszeiten. Für zeitkritische Vorgänge empfiehlt sich die Programmierung in C.

In *ftMscCDemo_V1.3* fehlt noch der C-Compiler aus dem C-Compiler Programmierpaket FIRMWARE 1.30 aus dem Download-Bereich von fischertechnik.de. Dieses Paket enthält den GNU ARM C-Compiler und einige Batch-Programme zum Starten des Compilers sowie zum Kopieren des Binär-Kodes auf den TX.

Vorbereitung: Kopiere *ftMscCDemo_V1.3* in das Stammverzeichnis von `c:\` und dann alle Dateien aus dem Ordner GNU des fischertechnik-Pakets in den leeren Unterordner

`c:\ftMscCDemo_V1.3\Demo_C\Bin\GNU`.

Entpacke `pspad461en.zip` und verbinde alle `*.ppr`-Dateien mit dem Programmiereditor `PSPad.exe`.

Verbinde den TX mit einem USB-Port. Bearbeite die Datei `set_port.bat` im Ordner `Demo_C\Bin` und ändere in

```
set COM_PORT=COM95
```

die 95 in die aktuelle COM-Nummer.

Starte nun mit `PSPad TX_Basis.ppr` im Ordner

```
C:\ftMscCDemo_V1.3\Demo_C\Demo\TX_Basis
```

Unter *Projekt → Projekt einstellen* findest du unter *Externe Programme* sowie *Kompiler* Angaben zur Belegung der Tasten zum Starten externer Programme sowie des Kompilers. Ich habe die Kompilertaste [1010] so belegt, dass nach der Kompilierung sofort `load_flash.bat` ausgeführt wird, das den Binär-Kode in den Flash-Speicher des TX überträgt (Alternative: `load_ramdisk.bat`).

Danach kann das USB-Kabel abgezogen und das übertragene Programm mit der linken roten Taste am TX gestartet werden.

Um die C-Programme zu vereinfachen habe ich in `ROBO_TX_PRG.h` die folgenden TX-Befehle implementiert:

```
txlamp, txmotor, txstartmotorsync,
txare-motorsready, txstartmotor,
txmotorready, txclearcounter,
txgetcounter, txtrack, txinput,
txanalog, txvoltage, txdistance,
i2cread, i2cwrite.
```

```
// Demo program "TX_Basis"
//
// Can be run under control of the ROBO TX Controller
// firmware in download (local) mode.
// Switches lamps connected to the outputs O7 and O8.
// A buzzer is connected to O5 and a Switch to I3.
// Disclaimer - Exclusion of Liability

#include "ROBO_TX_PRG.h"

static unsigned long cur_time;
static unsigned long prev_time;
static enum {
  buzzer, pause_buzzer, taster, on, pause_on, off, pause_off
} stage;
static int wait;

/* Function Name: PrgInit
 * PrgInit is called once. */

void PrgInit (TA * p_ta_array,  // pointer to the array of transfer areas
  int ta_count) {               // number of transfer areas in array
  p_ta = &p_ta_array[TA_LOCAL];
  prev_time = 0;
  stage = buzzer;
  wait = 200;
}

/ * Function Name       : PrgTic
  * PrgTic is called every tic (1 ms) realtime. */

int PrgTic (TA * p_ta_array,  // pointer to the array of transfer areas
  int ta_count) {             // number of transfer areas in array
                              // return code influences firmware
  int rc = 0x7FFF;  // 0x7FFF - program will further called;
                    // 0    - program should be normally stopped;
                    // any other value is an error, program is stopped
  cur_time = gettime();  // get the value of the system time
  switch (stage) {
```

```
    case buzzer:
      txlamp(4,500);
      prev_time = cur_time;
      stage++;
      return rc;
    case pause_buzzer:
      if (cur_time - prev_time >= wait) {
        txlamp(4,0);
        stage++;
      }
      return rc;
    case taster:
      if (txinput(2)) { stage++; }
      return rc;
    case on:
      txlamp(6,0);
      txlamp(7,500);
      prev_time = cur_time;
      stage++;
      return rc;
    case pause_on:
      if (cur_time - prev_time >= wait) {
        prev_time = cur_time;
        stage++;
      }
      return rc;
    case off:
      txlamp(6,500);
      txlamp(7,0);
      prev_time = cur_time;
      stage++;
      return rc;
    case pause_off:
      if (cur_time - prev_time >= wait) {stage=on;}
      return rc;
    default:  return rc;
  }
  return rc;
}
```

Listing 5: C-Demoprogramm

Der Ordner `Demo` enthält noch vier weitere Demoprogramme. Für eigene Programme kann man weitere Ordner anlegen, alle Dateien aus `TX_Basis` dorthin kopieren und dann `TX_Basis.c` nach eigenen Vorstellungen verändern.

Achtung: Wenn der Programm-Name geändert wird, dann muss dieser neue Name in `…/param.mk` eingetragen werden.

Referenzen

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

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

Computing

Programmierung des TXT mit Python

Torsten Stuehn

Dass man Programme für den TXT – nicht zuletzt dank des enthaltenen Linux-Betriebssystems – prinzipiell mit praktisch jeder Programmiersprache entwickeln kann, ist offensichtlich. Aber wie startet man ganz konkret? Der Beitrag zeigt, wie leicht das mit Python und dem vom Autor für die Community entwickelten TXT-Treiber `ftrobopy` *gelingt.*

Die Programmiersprache Python hat sich mittlerweile in vielen fischertechnik-Projekten mit dem TXT als Quasi-Standard neben der grafischen Programmierung mit ROBO Pro etabliert. Dies gilt sowohl für die originale fischertechnik-Firmware als auch ganz besonders für die community-Firmware [1, 2], die sogar selbst zum Teil in Python programmiert ist.

Insbesondere den Benutzern von Linux- und MacOSX-Betriebssystemen, die ROBO Pro nur mit Hilfe eines Windows-Emulators (z. B. Wine [3]) oder einer virtuellen Windows-Maschine verwenden können, bietet Python eine einfache und komfortable Möglichkeit, alle Fähigkeiten des TXT auszureizen. Durch seine weite Verbreitung in Forschung und Wissenschaft bietet Python unzählige Module, die dem ambitionierten fischertechniker kaum noch Grenzen bei der Umsetzung auch aufwendigster Projekte setzen.

Als ausgewachsene Programmiersprache bietet Python auch fortschrittliche Konzepte für den Zugriff auf Dateien und Datenbanken. Selbst ein einfacher Webserver auf dem TXT ist in nur einer Zeile umsetzbar, und Zugriffe auf die microSD-Karte des TXT sind (nach dem *mounten* der Karte) natürlich auch möglich.

Da die meisten Linux- und MacOSX-Benutzer ihren TXT wahrscheinlich längst in Python programmieren, wende ich mich im folgenden Text hauptsächlich an Windows-User, die einen einfachen und schnellen Einstieg in diese Programmiersprache suchen oder die sich für eine nicht-grafische Alternative zu ROBO Pro interessieren.

Ähnlich wie mit ROBO Pro kann man den TXT auch mit Python sowohl im *Online*- als auch im *Offline*-Modus programmieren:

- *Online*-Modus*:* Das Python-Programm läuft vollständig auf einem PC, der per WLAN, USB-Kabel oder Bluetooth mit dem TXT verbunden ist. Um den TXT im *Online*-Modus mit Python anzusteuern ist kein Eingriff auf dem TXT notwendig. Auf dem PC muss dafür ein Python-Interpreter installiert werden. Linux und MacOSX bringen diesen bereits mit.
- *Offline*-Modus*:* Das Python-Programm läuft vollständig auf dem TXT ab. Ein PC wird nur zum Editieren des Python-Programmes und zur Übertragung des Programmes auf den TXT und (normalerweise) zum Starten benötigt. Für diesen Modus muss ein Python-Interpreter auf den TXT kopiert werden. Die community-Firmware hat diesen bereits standardmäßig installiert.

Online-Modus

Python-Installation auf dem PC

Der einfachste Weg für den Einstieg in die Python-Programmierung des TXT führt über den *Online*-Modus. Darüber kann selbst ein neuer, frisch ausgepackter TXT ohne weitere Modifikationen sofort in Python angesteuert werden. Dafür benötigt man zunächst einen Python-Interpreter auf dem PC. Wer diesen noch nicht hat, der kann ihn sich z. B. von der Seite python.org [4] kostenfrei herunterladen. Ich empfehle die Installation der Python-Version 3.x (aktuell 3.6.1). Bei der Installation sollte man die Option „Add Python 3.6 to Path“ (siehe Abb. 1) setzen, damit man Python auch von der Kommandozeile aus aufrufen kann.

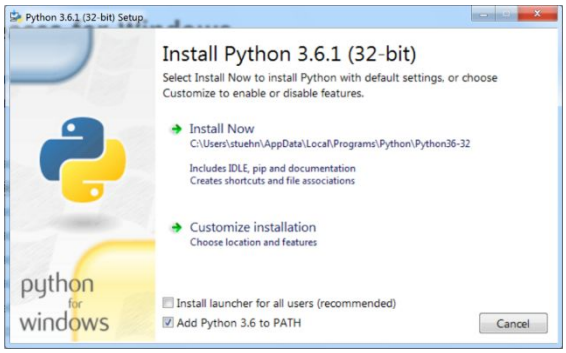


Abb. 1: Installationsdialog von Python

Nach einem Klick auf „Install Python 3.6.1“ steht ein einfacher Python-Interpreter auf der Windows-Kommandozeile zur Verfügung. Außerdem erreicht man über das Windows Start-Menü die Integrierte Python Entwicklungsumgebung (IDLE).

Um zu testen, ob Python richtig installiert wurde, kann man in der Windows-Kommandozeile („cmd.exe“) mit dem folgenden Befehl die Python-Version testen:

```
C:\Users\ts> python -version
```

Als Ausgabe sollte man dann das Folgende erhalten:

```
Python 3.6.1
```

Python ist damit erfolgreich auf dem PC installiert.

Installation von ftrobopy auf dem PC

Bevor der TXT in Python angesteuert werden kann, wird noch das Python-Modul *ftrobopy* benötigt. Dieses Modul kann mit dem folgenden Befehl über die Windows-Kommandozeile installiert werden:

```
C:\Users\ts> pip install ftrobopy
```

(Unter Linux mit „`sudo pip install ftrobopy`“)

Falls es hierbei zu Problemen kommt, kann *ftrobopy* auch über github [5] heruntergeladen oder geklont werden. Die Datei `ftrobopy.py` kann dann von Hand in das Verzeichnis kopiert werden, in dem die eigenen Python-Programme für den TXT abgespeichert werden sollen. Dieser Weg sollte auch grundsätzlich unter MacOSX eingeschlagen werden (für die Mac-Experten: „`pip install ftrobopy`“ kann z. B. mit Anaconda-Python auch auf dem Mac verwendet werden).

Um in Windows den TXT über das USB-Kabel ansteuern zu können, wird der USB-Netzwerktreiber benötigt. Dieser wird automatisch bei der Erstinstallation von ROBO Pro installiert. Linux und MacOSX bringen den Treiber bereits mit. Wer also den TXT unter Windows über USB-Kabel ansteuern möchte, sollte, falls er dies noch nicht getan hat, ROBO Pro auf dem Rechner installieren.

Das Modul *ftrobopy* funktioniert grundsätzlich bereits ab der Firmware-Version 4.1.5. Es ist allerdings zu empfehlen, den TXT auf die neueste Version (aktuell 4.2.4) upzudaten.

Erste Schritte mit IDLE

Nun kann über das Windows Start-Menü die Python-Entwicklungsumgebung IDLE aufgerufen werden. Es handelt sich hier um eine interaktive Python-Shell, die jeden eingegebenen Python-Befehl unmittelbar ausführt (Abb. 2).

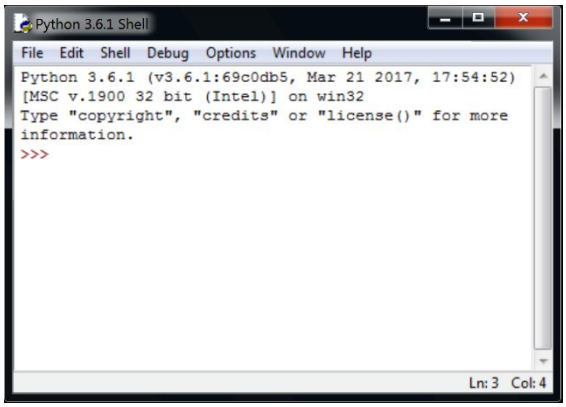


Abb. 2: Python-Entwicklungsumgebung IDLE

Die Zeichen „>>>“ in der Shell stellen das Command-Prompt von Python dar. Eine vereinfachte interaktive Python-Shell kann auch jederzeit in der Windows-Kommandozeile mit dem Befehl „python“ aufgerufen werden. In einem Linux- oder MacOSX-Terminal wird die interaktive Python-Shell auch einfach durch die Eingabe von „python“ gestartet.

Über das Menü Datei → Neue Datei (bzw. File → New File) kann auch ein Python-Editor aufgerufen werden, um komplexere Python-Programme zu schreiben und abzuspeichern. Diese werden nicht unmittelbar, sondern erst nach der Auswahl von „Run“ im Menü bzw. Drücken der Taste F5 ausgeführt.

Für den Einstieg und ein paar erste Tests, genügt die interaktive IDLE-Shell jedoch völlig. Zunächst muss dafür das *ftrobopy*-Modul importiert werden:

```
>>> import ftrobopy
```

Wenn dieses Modul nicht gefunden werden sollte, dann ist *ftrobopy* evtl. im falschen Verzeichnis installiert worden. Grundsätzlich muss die Datei `ftrobopy.py` entweder im aktuellen Verzeichnis oder im Python-System-Pfad vorhanden sein (z. B. unter „`C:\Users\ts\AppData\Local\Programs\Python\Python36-32\Lib\site-packages`“), damit das Modul importiert werden kann. Wenn keine Fehler aufgetreten sind, kann man mit

```
>>> txt=ftrobopy.ftrobopy('auto')
```

die Verbindung zum TXT herstellen. Der TXT muss dafür mit einer der Standard-Methoden (USB/WLAN/Bluetooth) angeschlossen sein. Falls der TXT als Client an einem WLAN-Accesspoint angeschlossen ist, muss anstelle von `'auto'` die Client IP-Adresse (z. B. `'192.168.2.114'`) angegeben werden. Ab diesem Zeitpunkt steht der TXT dann mit allen seinen Ein- und Ausgabe-Ports in Python zur Verfügung.

Wer mit der community-Firmware arbeitet, darf nicht vergessen, vorher die FT-GUI zu starten, um den „Online“-Modus zu ermöglichen.

Konfiguration und Ansteuerung der Ein- und Ausgänge des TXT

Zunächst können mit den folgenden Kommandos die Ein- und Ausgänge des TXT konfiguriert werden. Im Beispiel sind zwei Encoder-Motoren an den Ausgängen M1/C1 bzw. M2/C2 und ein Taster am Eingang I1 angeschlossen. Außerdem ist noch eine Lampe/LED an O5 angeschlossen:

```
>>> m1 = txt.motor(1)
>>> m2 = txt.motor(2)
>>> lampe  = txt.output(5)
>>> taster = txt.input(1)
>>> lampe.setLevel(512)
>>> m2.setSpeed(0)
>>> m2.setSpeed(0)
>>> m1.setDistance(0, syncto=m2)
>>> m2.setDistance(0, syncto=m1)
>>> def go(v):
...   m1.setSpeed(v)
...   m2.setSpeed(v)
...   while taster.state() != 1:
...     txt.updateWait()
...   m1.setSpeed(0)
...   m2.setSpeed(0)
>>> go(500)
```

Hier ist `go()` eine Funktion, die (auch interaktiv) aufgerufen werden kann, um zwei Motoren synchron mit einer Geschwindigkeit *v* laufen zu lassen, bis der Taster gedrückt wird. Falls zur Eingabe des kleinen Programmes nicht der interaktive Modus, sondern der Datei-Editor (Strg+N)

verwendet wird, sieht das komplette Programm folgendermaßen aus:

```
import ftrobopy
txt=ftrobopy.ftrobopy('auto')
m1 = txt.motor(1)
m2 = txt.motor(2)
lampe  = txt.output(5)
taster = txt.input(1)

txt.updateWait()

lampe.setLevel(512)
m1.setSpeed(0)
m2.setSpeed(0)
m1.setDistance(0, syncto=m2)
m2.setDistance(0, syncto=m1)
def go(v):
  m1.setSpeed(v)
  m2.setSpeed(v)
  while taster.state() != 1:
    txt.updateWait()
  m1.setSpeed(0)
  m2.setSpeed(0)

go(500)

txt.updateWait()
```

Es gibt hier zwei zusätzliche `txt.updateWait()`-Kommandos. Der Grund dafür ist der folgende: der TXT nimmt nur alle 0,01 Sekunden Kontakt zum PC auf, um die jeweils neuesten Befehle zu empfangen, die von einem *ftrobopy*-Hintergrundprozess bereitgehalten werden.

Das Kommando `txt.updateWait()` hält das Python-Programm so lange an, bis der nächste komplette Datenaustausch zwischen PC und TXT stattgefunden hat. Es könnte sonst passieren, dass das Python-Programm bis zum Ende einfach durchläuft, ohne dass auch nur ein einziger Befehl zum TXT gesendet wurde.

Der `txt.updateWait()`-Befehl am Ende des Python-Programms sorgt dafür, dass die Kommandos zum Stoppen der Motoren noch vollständig vom TXT empfangen werden, bevor das Python-Programm und damit auch der ftrobopy-Hintergrundprozess beendet werden.

Dies ist nur ein kleiner Ausschnitt der zur Verfügung stehenden Befehle. Die komplette Referenz mit allen Befehlen findet ihr im Benutzerhandbuch von *ftrobopy* [6].

Tipp: Grundsätzlich sollte man beim Erstellen von eigenen Python-Programmen möglichst darauf achten, Python-2 und -3 kompatiblen Code zu schreiben. Dies ist meistens recht einfach möglich. Im Web gibt es viele hilfreiche Seiten dazu (siehe z. B. [7, 8]). Das Modul *ftrobopy* ist selbst so geschrieben, dass es sowohl mit Python-2 (>2.7) als auch mit Python-3 verwendet werden kann.

Offline-Modus

Python-Installation auf dem TXT

Wer die Python-Programme nicht auf dem PC, sondern direkt auf dem TXT im *Offline*-Modus ausführen möchte, der benötigt einen Python-Interpreter auf dem TXT. Diesen kann man von den ftcommunity-Seiten herunterladen [9, 10]. Es handelt sich hier um einen Python-2.7-Interpreter mit einem Speicherplatzbedarf von ca. 19 MB. Dadurch ist es möglich den Interpreter auch auf einem TXT ohne zusätzliche microSD-Karte zu installieren. Ein Python-3-Interpreter ist deutlich grösser und lässt sich auf dem TXT nur mit zusätzlicher SD-Karte installieren.

Eine ausführliche Anleitung zur Installation des Python-Interpreters findet man bereits in einem früheren ft:pedia-Artikel [11].

Die community-Firmware (cfw) [2] für den TXT ist für den Python *Offline*-Modus deutlich besser geeignet als die original fischertechnik-Firmware. Überdies bringt die community-Firmware bereits standardmäßig einen Python-3-Interpreter mit. Auf den Seiten der cfw [2] gibt es weitergehende Hinweise zur Python-Programmierung des TXT. Insbesondere erhält man hier auch eine Anleitung für die Programmierung von Python-Programmen mit eigener grafischer Oberfläche auf dem Display des TXT.

Referenzen

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

[2] *fischertechnik TXT Community firmware*.

[3] *Wine HQ*.

[4] *Python*.

[5] *ftrobopy*, github.

[6] Torsten Stuehn: *ftrobopy - Ansteuerung des fischertechnik TXT Controllers in Python*. Manual, 2017.

[7] Ed Schofield: *Cheat Sheet: Writing Python 2-3 compatible code*. python-future.org.

[8] *Bilingual Quick Reference*. Python.org.

[9] *Python 2.7 für den TXT*. Downloadbereich der ftcommunity.

[10] *Digitalkamera ftDigiCam v0.83*. Downloadbereich der ftcommunity.

[11] Torsten Stuehn: *Digitalkamera mit Autofokus und Live-Video-Vorschau*. ft:pedia 1/2016, S. 74-76.

Computing

V. I. P. – Ein I²C-nach-Computing-Interface-Umsetzer (Teil 1)

René Trapp

Wie man ein altehrwürdiges fischertechnik Computing Interface als Slave an den I²C-Bus anschließt.

fischertechnik hat mit der Nutzung des I²C-Interface beim TX- und beim TXT-Controller die wohl universellste Erweiterungsmöglichkeit geschaffen. Reichen die vorhandenen Anschlussmöglichkeiten des Controllers nicht aus, kann man einfach zusätzliche Sensoren per I²C anschließen. Natürlich sind nicht nur Sensoren für den Anschluss am I²C-Bus geeignet, sondern auch entsprechend ausgerüstete Aktoren.

In der ft:pedia gibt es schon zahlreiche Publikationen zum Thema I²C am TX oder TXT. Aus dieser Artikelreihe stechen die beiden Beiträge von Dirk Fox mit den Steckerbelegungen des TX [1] und TXT [2] besonders hervor.

Computing Interfaces an diversen Microcontrollern sind auch schon in der ft:pedia oder auf privaten Fanseiten vorgestellt worden ([3], [4], [5] und [6]).

Der Plan

Warum also nicht die beiden Themen miteinander verbinden und ein Computing Interface als IO-Erweiterung am I²C-Bus des TX(T) betreiben?

Das I²C-Protokoll ist bekannt, die Ansteuerung des Computing-Interface ist kein Hexenwerk und so könnte solch eine Hardwerkerei zur Umsetzung des I²C-Bus an ein Computing Interface an ein paar entspannten Wochenenden angefertigt werden. Benötigt wird dazu ein Microcontroller und ein bisschen Elektronik drum herum. Eine Platine veredelt die Schaltung und erlaubt es, den Adapter gleich ins Gehäuse des alten Interface einzubauen. Abgerundet mit einer entsprechenden Software setzt der Microcontroller als Slave-Device das I²C-Protokoll in die nötigen Steuersequenzen für das Interface um.

Für den Autor hätte die Geschichte damit eigentlich enden können. Etwas Bastelei, ein oder zwei Fotos davon in den Bilderpool und gut. Aber über die ganzen Recherchen zu den Interfaces stellte sich schnell heraus, dass da noch viel mehr Informationen verborgen liegen, die auch für die Fan-Gemeinde zugänglich gemacht werden sollen. Und eine anständige Dokumentation muss natürlich auch geschrieben werden.

Für eine universelle Verwendung wird zusätzlich eine automatische Pegelanpassung benötigt, da der I²C-Bus des TX mit 5 V betrieben wird, beim TXT sind es dagegen 3,3 V. Andere, also nicht-fischertechnik-Controller, können davon natürlich profitieren.

Und es gibt noch Hürden, die durch eine unvollständige oder fehlerhafte Implementierung einiger I²C-Busmaster aufgestellt werden. Das Stichwort heißt hier „clock stretching“. Das ist ein Mechanismus, mit dem sich ein Slave am Bus eine kurze Auszeit gönnen darf, um seine Aufgabe aus-

zuführen. Im Forum der ftc gibt es widersprüchliche Aussagen hierzu. Einige Quellen berichten von Problemen am TX und auch am TXT ([7], [8] und [9]). Eine ganz frische Quelle belegt allerdings für den TXT das Gegenteil [10]. Der TXT kann es also doch, das clock stretching!

Das was in den bisherigen zahlreichen Selbstbauprojekten an Know-How zu den Interfaces vorgestellt wurde, gilt leider nicht so ganz universell. Schließlich gibt es nicht „das“ Computing-Interface, sondern unsere Datenbank listet ganze 10 Stück davon.

Anhand der Schaltpläne aus dem Buch „Robotik mit dem Homecomputer“ [11] lassen sich schon marginale Unterschiede zwischen den Interfaces erkennen. Diese Pläne gelten aber nur für einen Teil der Interfaces, so wie auch ein gut versteckter Originalplan aus dem Hause fischer [12]. Zusätzlich findet sich im Downloadbereich der ftc eine Art Kombinationsschaltplan für die verschiedenen Interfaces [13]. Der ist allerdings nicht ganz vollständig.

Ein richtig universell einsetzbarer Adapter kann alle diese alten Interfaces bedienen, erlaubt den Betrieb auch an TX oder TXT und kommt dabei mit einer gemeinsamen Software und einer einzigen Platine aus. Alle Bauteile sind für den Hobbyisten problemlos erhältlich. Das ist das V. I. P. – das VINTAGE INTERFACE PROJECT.

Bevor nun aber die kleine Hardwerkerei vorgestellt wird, ist es zuerst nötig, alle alten Interfaces mit ihren Eigenschaften näher kennenzulernen. Als „Beifang“ kommt dabei nicht nur die Historie der Interfaces ans Licht; zu jedem Interface gibt es ab jetzt auch noch den zugehörigen Schaltplan.

Der Untergrund

Sehr hilfsbereite Freunde aus der ftc stellten dem Autor hochauflösende Fotos ihrer Interfaces für die Rekonstruktion der Schaltpläne zur Verfügung. An dieser Stelle ein dickes Dankeschön an C. Hehr, H. Howey und auch an K. Merkert. Von letzterem stammt eine Umbauanleitung für ein Apple-Interface 30563 zum Anschluss an einen PC [14]. Basierend auf dieser Umbauanleitung mit den enthaltenen Fotos von Bestückungs- und Layoutseite einer Platine gelang es, erste Teile der Originalschaltung zu rekonstruieren. Die hochauflösenden Fotos der ftc-Freunde halfen dann, die noch offenen Lücken zu schließen.

Bei der Analyse der Platinenfotos ließ sich außer detaillierten Schaltplänen nebenbei auch noch die Geschichte der Interfaces zurückgewinnen.

Der Hintergrund

Also tauchen wir zunächst etwas in die Evolution der Interfaces ab.

Während diese Zeilen entstehen, gibt unsere Datenbank bei der Recherche zu technischen Informationen nicht mehr her als die Erscheinungsjahre der einzelnen Interfaces. Gemäß Datenbank und [15] ergibt sich immerhin diese Liste in chronologischer Reihenfolge:

Jahr	Nr.	Bezeichnung
1985	30561	Interface Commodore Business Machines (CBM) 4xxx / 8xxx, [16]
1985	30562	Interface Commodore (VC20, C64), [17]
1985	30563	Interface Apple II, [18]
1985	30564	Interface Acorn B, [19]
1985	30565	Interface Schneider CPC, [20]
1986	30567	Interface IBM-PC, [21]
1987	30566	Interface Centronics Port, [22]
1989	39319	Interface Centronics Port (67319) CVK, [23]

Jahr	Nr.	Bezeichnung
1989	3639x	Interface IBM-PC (66843) CVK, [24]
1991	30520	Interface IBM universal, [25]

Bei der Historie mutet es etwas seltsam an, dass beide Versionen des CVK-Interface 1989 erschienen sein sollen. Leider ist die Informationslage extrem dürftig; es lässt sich nicht mit Sicherheit sagen, ob das 66843 nicht erst doch 1991 auf den Markt kam.

Zusätzlich ranken sich noch Legenden um die Kaskadierung der Interfaces. Das Erste Interface ist dabei an den Computer angeschlossen, ein weiteres Interface, der Slave, an einen eigens dafür vorhandenen Steckplatz. Auf diese Art konnte schon 1985 die Anzahl der Ein- und Ausgänge erhöht werden [26], [27].

Der Stand der Technik

Grundsätzlich verfügen alle vorgenannten Interfaces über 5 Buchsen zum Anschluss von maximal 2 Netzteilen.

Eine 20-polige Stiftleiste dient zum Anschluss der Modelle. Diese Stiftleiste ist bei allen aufgeführten Interfaces einheitlich belegt (Abb. 1) und liegt auf der Seite der Netzteilbuchsen.

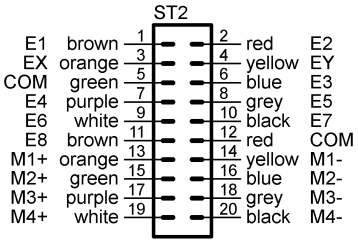


Abb. 1: Anschluss für das Modell

Die Ansteuerung der Motoren und das Auslesen der Eingänge erfolgt bei allen Interfaces einheitlich. Auch das Gehäuse der Interfaces ist über die Generationen mechanisch unverändert geblieben.

Die ersten Computing Interfaces stammen aus dem Jahr 1985.

Die Interfaces 30561 (Abb. 2), 30562 ([28]), 30563 und 30564 basieren auf einer gemeinsamen Platine mit einseitigem Layout. Durch Bestückungsoptionen und Lötbrücken wird der exakte Typ des Interface festgelegt. Diskrete (diskret: hier „aus einzelnen Transistoren und Widerständen bestehende“) Endstufen sind für den Anschluss der Motoren vorgesehen.

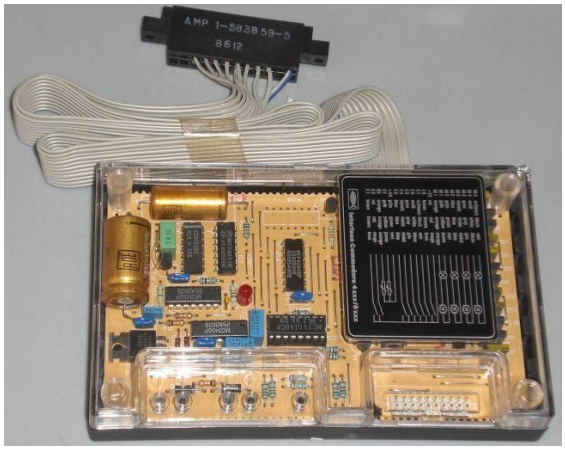


Abb. 2: Computing Interface 30561

Das Kabel zum Computer ist an einem von drei möglichen Plätzen fest angelötet. Es gibt einen 16-poligen Anschluss „DCH“, an den ein Interface für den Apple II (30563) als Slave angeschlossen werden kann. Ist der passende IC-Sockel nicht bestückt, kann bei Bedarf ein einfacher IC-Sockel (nicht die teuren mit den gedrehten Kontakten!) eingelötet werden.

Layouttechnisch schon weiter fortgeschritten offenbart sich die nächste Generation. Die Interfaces 30565 und 30567 ([29]) basieren ebenfalls auf einer gemeinsamen Platine mit einseitigem Layout. Diskrete Endstufen sind für den Anschluss der Motoren vorgesehen. Hier gibt es keine Bestückungsoptionen mehr. Für den Computeranschluss wird einheitlich eine 20-polige Verbindung benutzt. Je nach Computer ist das passende Kabel direkt mit der Platine verlötet. Es gibt einen 16-poligen Anschluss „DCH“, an den ein Interface für Apple II (30563) als Slave angeschlossen werden kann. Diese Platine kann als Vorläufer des Centronics-Interface

gelten. Sowohl IBM-PC als auch Schneider CPC verwenden bereits die Centronics-Schnittstelle als Anschluss für das Computing Interface.

Einen kleinen Evolutionssprung gibt es 1987.

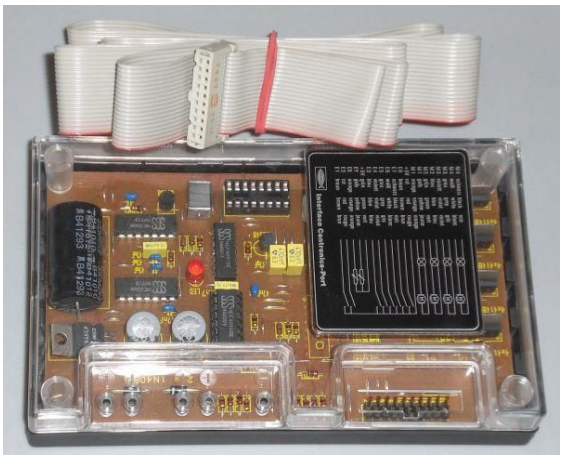


Abb. 3: Computing Interface 30566

Die Interfaces 30566 (Abb. 3) und 39319 (= 67319) basieren auf einer gemeinsamen Platine mit einseitigem Layout.

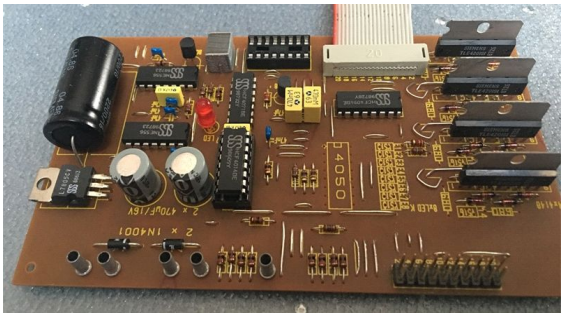


Abb. 4: Platine des 30566 [30]

Abb. 4 zeigt eine Platine ohne Gehäuse. Integrierte Endstufen sind hier für den Betrieb der Motoren zuständig. LEDs sind im Layout vorbereitet, aber nur bei der CVK-Version bestückt. Als Stecker zum Computer ist einheitlich ein 20-poliges Flachkabel verlötet. Je nach Computer wird eine Adapterplatine am anderen Kabelende aufgesteckt. Auf dieser Adapterplatine ist die passende Verschaltung zum Computer aufgebracht und das Kabel zum Computer wird dort angeschlossen. Dieser 20-polige Anschluss ist elektrisch identisch zum späteren 30520 belegt. Es gibt einen 16-poligen Anschluss „DCH“, an den ein Interface für Apple II (30563) als Slave angeschlossen werden kann.

Die letzte Generation der Interfaces ist mehr eine Art Facelift des 30566.

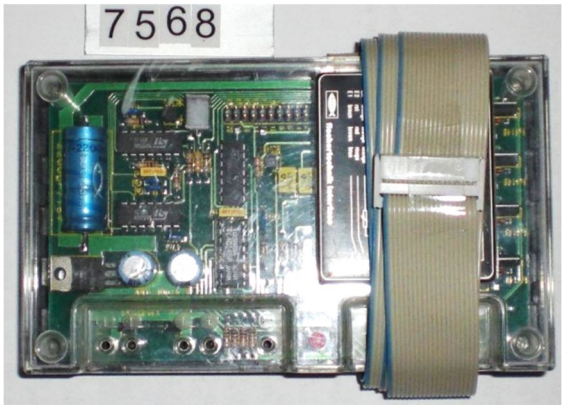


Abb. 5: Computing Interface 30520

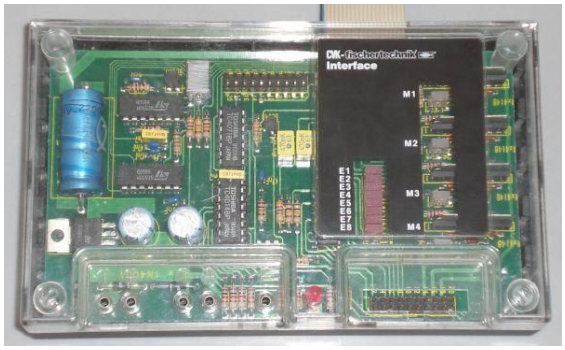


Abb. 6: Computing Interface 66843

Die Interfaces 30520 (Abb. 5) und 66843 (Abb. 6) basieren auf einer gemeinsamen Platine mit doppelseitigem Layout. Die integrierten Endstufen sind wieder für den Anschluss der Motoren vorgesehen. LEDs sind im Layout vorbereitet, aber nur bei der CVK-Version bestückt. Die Verbindung zum Computer übernimmt auch hier wieder das 20-polige Flachkabel; es ist in der Platine fest eingelötet. Auf der anderen Seite wird das Verbindungskabel zu einer je nach Computer angepassten Adapterplatine aufgesteckt. Die Vorgehensweise ist identisch zur Vorgängerversion 30566. Es gibt eine weitere 20-polige Stiftleiste „ST3“, an der ein weiteres 30520 (oder 66843) Interface als Slave angeschlossen werden kann. Ein 30566 oder 39319 ist hier als Slave ebenso geeignet.

Die Eckpfeiler

Zu jedem Interface gibt es selbstverständlich auch eine Anleitung. Die meisten dieser Dokumente sind bei unseren niederländischen Freunden versammelt [31]. Die umfangreichsten Informationen allerdings gibt es in einer der dort aufgeführten CVK-Anleitungen [32]. Ausführlich sind die Funktionen beschrieben, die jeweils benutzten Anschlüsse der Computer zusammengefasst und sogar Hinweise versteckt, welche LED bei welcher Motoransteuerung aufleuchten wird. Dieses Dokument erwies sich als unschätzbare Hilfe bei der Rekonstruktion der Interfaces 30566, 30520 und ihrer CVK-Brüder, ohne eine Platine in der Hand zu haben.

Die technischen Daten der Interfaces sind über alle Familienmitglieder einheitlich angegeben:

- 4 Motorausgänge M1…M4, je 1 A, Spitze 1,5 A
- Motorausgänge kurzschlusssicher
- 8 Schaltereingänge E1…E8
- 2 Widerstandsgeber EX, EY, 0…5 kΩ
- Watchdog
- Versorgung aus 6,8 V=

Für die damals in den Computing-Kästen enthaltenen 6 V-Motoren, Mini-Motor (31062) sowie S-Motor 6V (32240) ist das durchaus ausreichend. Diese entsprechen in ihrer Baugröße dem modernen XS-Motor (137096) und dem S-Motor 9V (32293).

Das Innenleben

Ein Interface stellt eine Schnittstelle über eine Grenze hinweg dar. Nichts Anderes macht ein fischertechnik Computing Interface. Es überbrückt die Grenze zwischen einem Computer und Peripherieeinheiten.

Um die bisherigen Grundlagen abzurunden, sei hier also noch kurz auf das Innenleben der Interfaces eingegangen (Abb. 7). Die nötigen Abläufe zur Kommunikation sind in jeder Anleitung enthalten und waren auch schon Gegenstand diverser Veröffentlichungen, beispielsweise [3], [4] oder [11].

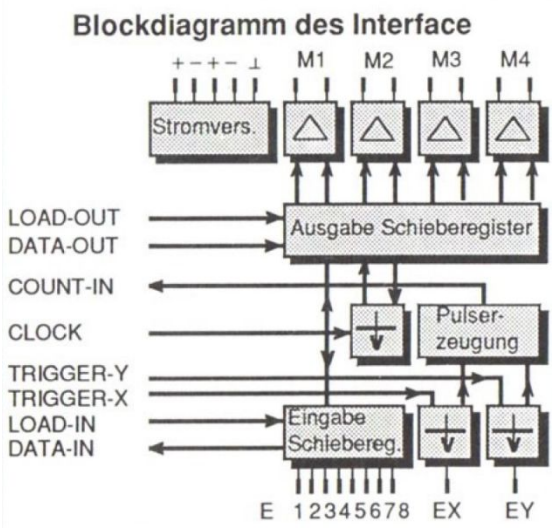


Abb. 7: Blockschaltbild [32]

Die vorhandenen Motorausgänge werden über lediglich 2 eigene Leitungen (DATA-OUT, LOAD-OUT) gesteuert. Eine dritte Leitung (CLOCK), wird gemeinsam mit der Eingangslogik genutzt.

Die vorhandenen Schaltereingänge werden ebenfalls über 2 eigene Leitungen (DATA-IN, LOAD-IN) abgefragt. Die Taktleitung CLOCK ist bei dieser Übertragung ebenfalls wieder in Benutzung.

Für die Analogeingänge sind die Steuerleitungen TRIGGER-X, TRIGGER-Y und COUNT-IN zuständig.

Sind mehrere Interfaces als Kette verschaltet, werden je Interface noch 8 Taktimpulse und 8 Datenbits in die Übertragung eingefügt.

Im referenzierten Artikel von Jens Lemkamp [3] und auch bei Dirk Uffmann [4] ist der Ablauf wunderschön erklärt.

Anders als in den alten Anleitungen dargestellt, kann die Ausgabe der Daten an die Motorendstufen zeitgleich mit dem Einlesen der Daten von den Schaltereingängen erfolgen [5]. Es sei noch angemerkt, dass

diese Art der seriellen Übertragung heutzutage auch als SPI (Serial Peripheral Interface) sehr bekannt ist und von modernen Mikrocontrollern per speziell eingebauter Hardware unterstützt wird.

Die Ansteuerung der Motorausgänge und die Abfrage der Schaltereingänge erfolgt bei allen besprochenen Interfaces identisch. Die Unterschiede sind in der Bedienung der Widerstandseingänge EX und EY begründet – und einen kleinen aber gemeinen Unterschied gibt es auch bei den Motortreibern selbst.

In den nächsten Teilen der Reihe wird bei den jeweiligen Schaltungsteilen noch genau auf die jeweilige Übertragung und alle weiteren Details eingegangen.

Die Brückenköpfe

Für den Anschluss des jeweiligen Interfaces an das zugehörige Computer-Modell ist ein entsprechend passendes Anschlusskabel vorgesehen. Bei allen Interfaces im Originalzustand ist das Kabel fest angelötet.

30561 für CBM4xxx / CBM8xxx / VC20

Bei diesem Interface für die Computer aus dem Hause Commodore ist ein 10-poliges Flachkabel direkt in der Platine verlötet. Die 10 Lötaugen sind mit „CO“ beschriftet und Ader 1 ist markiert. Am Commodore-Computer wird der Userport benutzt, der sich auf der Rückseite des Computers befindet. Die Zuordnung der Signale zeigt Abb. 8, rekonstruiert anhand [32] und [33].

Die Leitungen LOAD-OUT, LOAD-IN, DATA-OUT, DATA-IN und CLOCK bedienen die Schieberegister für Motoraus- und Schaltereingänge. DATA-IN ist nicht invertiert. Über die Leitungen TRIGGER-X und TRIGGER-Y wird die Abfrage der beiden Potentiometer an EX und EY gestartet. Als Ergebnis der jeweiligen Messung wird eine Anzahl Impulse auf der Leitung COUNT-IN vom Interface zurückgeliefert. Das Computerprogramm muss diese Impulse also zählen, um das Ergebnis zu erhalten.

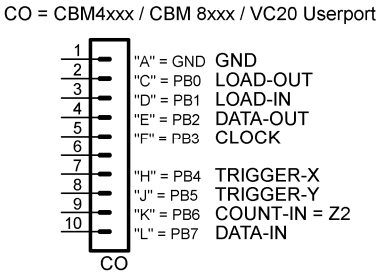


Abb. 8: Anschluss CBM / VC20

30562 für C64

Bei diesem Interface für den C64 aus dem Hause Commodore ist ein 10-poliges Flachkabel direkt in der Platine verlötet. Die 10 Lötaugen sind mit „CO“ beschriftet und Ader 1 ist markiert. Am C64 wird der Userport benutzt, der sich auf der Rückseite des Computers befindet. Die Zuordnung der Signale zeigt Abb. 9, rekonstruiert anhand [32] und [33].

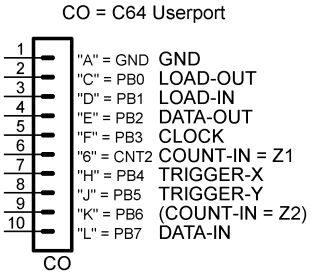


Abb. 9: Anschluss C64

Die Leitungen LOAD-OUT, LOAD-IN, DATA-OUT, DATA-IN und CLOCK bedienen die Schieberegister für Motoraus- und Schaltereingänge. DATA-IN ist nicht invertiert. Über die Leitungen TRIGGER-X und TRIGGER-Y wird die Abfrage der beiden Potentiometer an EX und EY gestartet. Als Ergebnis der jeweiligen Messung wird eine Anzahl Impulse auf der Leitung COUNT-IN vom Interface zurückgeliefert. Der Computer muss diese Impulse also zählen, um das Ergebnis zu erhalten. Dabei gibt es am C64 einen extra Eingang für einen Hardware-Zähler, so dass sich das Computerprogramm nicht selbst um die eigentliche Zählung kümmern muss.

30563 für Apple II

Diesem Interface für Computer der Apple-II-Serie kommt eine Sonderstellung zu.

In den Apple-II-Computern gibt es eine eigene Schaltung, um den Widerstand externer Potentiometer zu erfassen. Diese wird auch benutzt, so dass auf dem Interface die entsprechenden Schaltungsteile nicht vorhanden sind. Stattdessen sind die Potentiometeranschlüsse direkt mit dem Apple II verbunden. Ein 16-poliges Flachkabel ist direkt in einem 20-poligen Anschlussfeld verlötet, dieses ist mit „AP“ beschriftet. Ader 1 sowie die Adern 8, 9 und 16 sind markiert. Am Apple wird ein interner Steckplatz „Game-IO“ benutzt. Die Zuordnung der Signale zeigt Abb. 10, rekonstruiert anhand [32] und [34]:

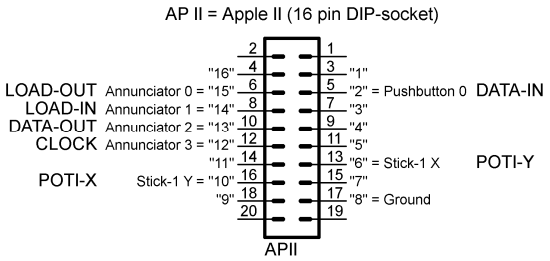


Abb. 10: Anschluss Apple II

Die Leitungen LOAD-OUT, LOAD-IN, DATA-OUT, DATA-IN und CLOCK bedienen die Schieberegister für Motoraus- und Schaltereingänge. DATA-IN ist nicht invertiert.

Außer zum Anschluss an einen Apple-II-Computer kann dieses spezielle Interface auch übergreifend als Slave zur Erweiterung der Ein- und Ausgänge eingesetzt werden.

30564 für Acorn B

Das Interface für den Acorn B ist ebenfalls mit einem festverlöteten Anschlusskabel ausgestattet. Die 5 Diagonalreihen aus je 4 Lötaugen sind mit „AC“ beschriftet, Ader 1 ist markiert. Am Acorn wird der Userport benutzt, der auf der Computerunterseite versteckt ist. Die Zuordnung der Signale zeigt Abb. 11, rekonstruiert anhand [35]:

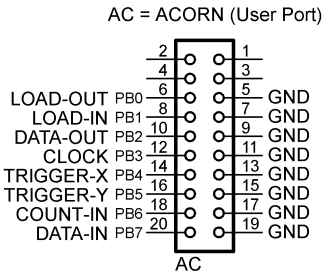


Abb. 11: Anschluss Acorn B

Die Leitungen LOAD-OUT, LOAD-IN, DATA-OUT, DATA-IN und CLOCK bedienen die Schieberegister für Motoraus- und Schaltereingänge. DATA-IN ist nicht invertiert. Über die Leitungen TRIGGER-X und TRIGGER-Y wird die Abfrage der beiden Potentiometer an EX und EY gestartet. Als Ergebnis der jeweiligen Messung wird eine Anzahl Impulse auf der Leitung COUNT-IN vom Interface zurückgeliefert. Das Computerprogramm muss diese Impulse also zählen, um das Ergebnis zu erhalten.

30565 für CPC464 / CPC664 / CPC6128

Eigentlich ist es ja schon ein Centronics-Interface, das 30565. Die damals populäre parallele Druckerschnittstelle basierte auf diesem noch recht jungen Standard. Allerdings wurde nur selten computerseitig der normgerechte Stecker verwendet.

Im Interface ist eine 20-polige Stiftleiste mit der Bezeichnung „ST1“ vorgesehen, auf die das Flachkabel zum Computer aufgesteckt ist. Ader 1 ist markiert. Am anderen Ende des Computerkabels ist ein Platinendirektstecker für den Centronics-Port der Amstrad/Schneider-Computer angepresst. Dabei ist bei einigen CPC-Modellen die Anschlusszählung am mechanisch unveränderten und elektrisch identisch belegten Stecker geändert. In Abb. 12 ist stellvertretend die Belegung am CPC464 wiedergegeben, basierend auf [37]:

Die Leitungen LOAD-OUT, LOAD-IN, DATA-OUT, DATA-IN und CLOCK bedienen die Schieberegister für Motoraus-

und Schaltereingänge. DATA-IN ist invertiert! Über die Leitungen TRIGGER-X und TRIGGER-Y wird die Abfrage der beiden Potentiometer an EX und EY gestartet. Während der jeweiligen Messung liegt die Leitung DATA-IN auf „0“ und ändert ihren Zustand zum Ende der Messung auf „1“. Das Computerprogramm muss diese Zeitdauer ermitteln, um das Ergebnis zu erhalten.

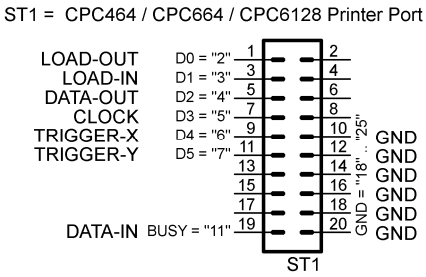


Abb. 12: Anschluss Amstrad / Schneider

Neben dem 20-poligen Stecker für das Anschlusskabel zum Computer gibt es auf der Platine auch wieder die 10 bekannten Lötaugen „CO“ für Commodore-Computer und ein 20-poliges Lötaugenfeld „AP“. Wegen einer falschen Anschlussbelegung kann hier allerding kein Apple II Computer angeschlossen werden! Ebenso fehlt für Commodore-Computer wegen der geänderten internen Schaltung der Impulsausgang COUNT-IN.

30567 für IBM-PC

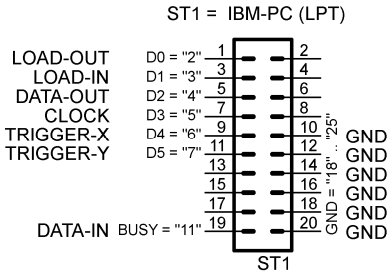


Abb. 13: Anschluss IBM-PC

Das Interface entspricht 1:1 dem vorgenannten 30565. Lediglich am anderen Ende des Computerkabels von „ST1“ ist das Gegenstück zur sattsam bekannten 25-poligen SUB-D-Buchse am PC angebracht.

Alle weiteren Details sind identisch mit dem 30565 vom Schneider. Insbesondere ist auch hier DATA-IN invertiert.

30566 für Centronics

Dieses Interface eignet sich für den Anschluss an jeden Computer mit der parallelen Centronics-Druckerschnittstelle – so impliziert es der Name. Commodore Amiga, Atari ST und weitere waren die Adressaten dieses Interfaces. Das 20-polige Flachkabel zum Computer ist an der Position „ST1“ verlötet, Ader 1 ist markiert. Die Belegung des Steckverbinders ist in Abb. 14 angegeben und basiert ausschließlich auf einer Layoutanalyse der Originalplatine.

Im Falle der Centronics-Schnittstelle wird über die Leitungen TRIGGER-X und TRIGGER-Y die Abfrage der beiden Potentiometer an EX und EY gestartet. Die Leitung POTI-X ist direkt mit RC-X verbunden, die Leitung POTI-Y direkt mit RC-Y. Während der jeweiligen Messung liegt die Leitung CENTRONICS-DATA-IN auf „0“ und ändert ihren Zustand zum Ende der Messung auf „1“. Das Computerprogramm muss diese Zeitdauer ermitteln, um das Ergebnis zu erhalten. Die Daten von den Schaltern werden ebenfalls invertiert zurückgeliefert.

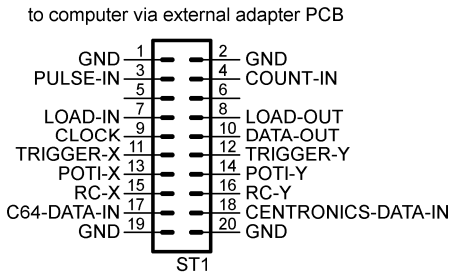


Abb. 14: Anschluss Centronics

Zusätzlich stehen die Impulse wie bei C64 und ähnlich an der Leitung COUNT-IN zur Zählung zur Verfügung, und es gibt die Pulslänge einzeln auch noch auf der Leitung PULSE-IN. Die getrennte Datenleitung C64-DATA-IN führt ausschließlich die

serielle Information von den Eingängen E1 bis E8.

Die Leitungen LOAD-OUT, LOAD-IN, DATA-OUT, DATA-IN und CLOCK bedienen die Schieberegister für Motoraus- und Schaltereingänge.

Soll das Interface an einen Apple II angeschlossen werden, so werden POTI-X und POTI-Y direkt zum Apple verbunden, die Verbindungen zu RC-X und RC-Y sind unterbrochen.

Es scheint kaum bekannt zu sein, dass diese Steckerbelegung mit dem späteren Universal-Interface 30520 völlig identisch ist. Auch sind alle nötigen Bauteile bestückt, um die volle Funktion des 30520 zu erfüllen.

Daher ist zu vermuten, dass für den Anschluss an den Computer eine weitere Platine zwischengeschaltet wurde, so wie vom 30520 bekannt. Und wegen dieser Steckerbelegung eignet sich dieses Interface auch als Slave für ein 30520 / 66843.

39319 = 67319 Centronics-CVK

Dieses Interface ist um LEDs zur Anzeige der Eingangszustände und der Motorausgänge ergänzt. Die Steckerbelegung ist identisch zum 30566, der Anschluss an den jeweiligen Computer ist entsprechend vorzunehmen, siehe Abb. 14.

30520 Universal

Offenbar hat das neue universelle Anschlusskonzept des 30566 überzeugt, und so wurde mit der Änderung des Slave-Steckers nicht nur eine neue Teilenummer vergeben, sondern auch der Name geändert.

Dieses Interface wurde zusammen mit einer individuell passenden zusätzlichen Adapterplatine offiziell für den Anschluss an alle damals bekannten Computer vermarktet. Es kann die komplette bisherige 3056x-Serie der Interfaces ersetzen.

Die Technik selbst ist aber gleichgeblieben und entspricht immer noch dem 30566 – siehe Abb. 14 und die Erklärung zum 30566.

Dieses Interface kann auch als Slave eines weiteren 30520 / 66843 genutzt werden.

66843 Universal CVK

Logischerweise gibt es auch hier die Ergänzung um LEDs, um das Interface für Ausbildungszwecke etwas interessanter zu machen. Am Stecker (Abb. 14) und dem Computeranschluss ist die Technik wiederum identisch zum 30520 und zum 30566 geblieben.

Dieses Interface kann ebenfalls als Slave eines weiteren 30520 / 66843 genutzt werden.

Sklaventreiberei

In diversen Internetquellen finden sich Aussagen, dass es bei einigen Interfaces keinen Stecker für einen Slave-Anschluss gäbe. Dies stimmt so nicht. Zumindest im Layout ist der Anschluss für einen Slave immer vorgesehen und das fehlende Bauteil kann bei Bedarf eingelötet werden.

Als Slave sind grundsätzlich nur die Interfaces 30563, 30566 oder 30520 sowie deren CVK-Verwandtschaft geeignet.

Bei den Interfaces 30561, 30562, 30563, 30564, 30565, 30566, 30567 und 39319 wird ein Apple-Interface (30563) als Slave am 16-poligen Steckplatz „DCH“ angeschlossen. Dessen Anschlussbelegung zeigt Abb. 15:

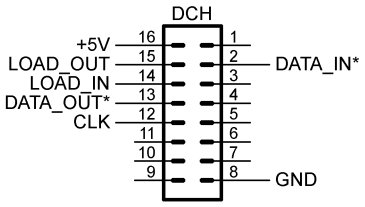


Abb. 15: Slave-Anschluss 16-polig

Bestückt ist eine einfache 16-polige IC-Fassung (DIL 16).

Bei den modernsten Interfaces 30520 und 66843 wird ein Slave-Interface am Stecker

„ST3“ angeschlossen. Dessen Anschlussbelegung zeigt Abb. 16.

Bestückt ist hier eine 20-polige Stiftleiste, zweireihig mit Rastermaß 2,54 mm.

Technisch ist es übrigens möglich, mehr als 2 Interfaces zu kaskadieren. Die maximale Anzahl der Interfaces in einer Slave-Kette wird durch die Leitungslängen zwischen den Interfaces begrenzt. Es gibt keinerlei Belege für eine jemals existierende Master-Slave-Kette aus mehr als zwei Interfaces.

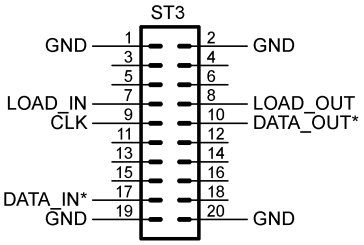


Abb. 16: Slave-Anschluss 20-polig

Und so geht es weiter

Im nächsten Teil der Artikelserie rücken die Stromversorgung und die verschiedenen Motorendstufen in den Fokus. Es gibt Informationen über die Eigenschaften der Schaltungen. Ebenso werfen wir einen Blick auf mehr oder weniger bekannte Probleme.

Quellen

[1] Dirk Fox: *I²C mit TX und Robo Pro – Teil 1: Grundlagen.* ft:pedia 3/2012, S. 32

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

[3] Jens Lemkamp: *Parallel-Interface durch Arduino gesteuert (1).* ft:pedia 1/2014, S. 24

[4] Dirk Uffmann: *Nutzung des Universal-Interfaces 30520 als Port-Erweiterung an einem Mikrocontroller.* ft:pedia 2/2014, S. 30

[5] Dirk Uffmann: *Altes FT Universal-Interface (Parallelschnittstelle) gesteuert durch AVR Mikrocontroller.*

[6] Thomas Kaiser: *C-64 Interface an RoboPro.*

[7] Forumsdiskussion: *I²C clock stretching am TXT.*

[8] Forumsanfrage an fischertechnik: *Unterstützung von i2c clock stretching beim TXT controller?.*

[9] Erneute Forumsanfrage an fischertechnik: *Unterstützung von i2c clock stretching beim TXT controller?.*

[10] Forumsdiskussion: *TXT und clock-stretching.*

[11] J. P. M. Steeman: *Robotik mit dem Homecomputer*. Aachen: Elektor, 1987

[12] Klaus Merkert: *Zwei Schaltpläne.*

[13] Holger Howey: *Kombi-Schaltplan.*

[14] Klaus Merkert: *Umbau Apple II-Interface auf PC.*

[15] Ulrich Müller: *Übersicht der Interfaces.*

[16] fischertechnik-Datenbank: *30561 (Interface CBM).*

[17] fischertechnik-Datenbank: *30562 (Interface Commodore).*

[18] fischertechnik-Datenbank: *30563 (Interface Apple).*

[19] fischertechnik-Datenbank: *30564 (Interface Acorn).*

[20] fischertechnik-Datenbank: *30565 (Interface Schneider).*

[21] fischertechnik-Datenbank: *30567 (Interface IBM).*

[22] fischertechnik-Datenbank: *30566 (Interface Centronics).*

[23] fischertechnik-Datenbank: *39319 (Centronics Schul-Interface CVK)*.

[24] fischertechnik-Datenbank: *66843 (Universal Schul-Interface CVK)*.

[25] fischertechnik-Datenbank: *30520 (Interface Universal)*.

[26] Fischertechnikclub Nederland: *Twee robotarmen om je fantasie te prikkelen*. Clubblad 2/1992, S. 6

[27] Gerhard Bader: *Fischer-Technik und Computer: Programme für Atari ST, Commodore 64/128, Schneider CPC, IBM PC u. Kompatible*. CHIP-SPECIAL 1987, Vogel Verlag, Würzburg, 1987

[28] CPC Wiki: *Foto eines 30562*.

[29] CPC Wiki: *Foto eines 30567*.

[30] C. Hehr: *Platine eines 30566*. fischertechnik community.

[31] Fischertechnikclub Nederland: *Computing Dozen*.

[32] fischertechnik: *CVK-fischertechnik Schul-Interface (67319)*.

[33] C64 Wiki: *C64 Userport*.

[34] A2wiki: *Apple II Game-IO*.

[35] Chris Whytehead: *Acorn Pinouts*.

[36] Larry Davis: *PC Parallel Port Pin-Out*.

[37] CPC Wiki: *CPC Printerport*.

Bildnachweis

Abb. 2 Stefan Roth: Foto eines 30561

Abb. 3 Stefan Roth: Foto eines 30566

Abb. 5 Stefan Roth: Foto eines 30520

Abb. 6 Stefan Roth: Foto eines 66843

Die Bilder sind mit der freundlichen Genehmigung des jeweiligen Rechteinhabers wiedergegeben.

DANKE AN ALLE HELFER:

Christian Hehr, Holger Howey, Ulrich Müller und Klaus Merkert, die durch Ihre Vorarbeiten und Unterstützung die Rekonstruktion der Schaltpläne von Ihren Originalplatinen ermöglichten.

Modell

fischertechnik-Flipper

Dirk Wölffel

Einfache fischertechnik-Flipper finden sich schon in ganz frühen fischertechnik-Anleitungen, so zum Beispiel in den Club-Nachrichten 2/1971 oder im Modell-Anhang des Büchleins „Kleine Erfinder – große Ideen“ aus dem Jahr 1972. Aber erst mit den Kugelbahnen (Metallkugeln und Flexschienen) und den Magnetventilen wurde im Jahr 2012 die Konstruktion von Flippern mit echtem „Spielhallen-Flair“ möglich. Dass auch bei fischertechnik-Flippern noch viel Luft nach oben ist, zeigt das hier vorgestellte Modell.

Der fischertechnik-Flipper ist ein Modell aus dem Baukasten ROBO TXT Electro-Pneumatic (516186, Abb. 1). Auf verschiedenen Ausstellungen habe ich festgestellt, dass dieser Flipper immer ein Publikumsmagnet für Kinder und deren Eltern ist.

Abb. 1: ROBO TXT ElectroPneumatic 516186

Dennoch hat das Original-Modell ein paar Schwächen. Vor ungefähr einem Jahr habe ich mir daher überlegt, wie ich den Flipper verbessern könnte:

- Optisch und vom Aufbau her sollte der fischertechnik-Flipper einem originalen ähneln.
- Funktionen wie zum Beispiel Lichteffekte, Sound und ein Mehrspielermodus sollten implementiert werden.

Mein Anspruch war, den Spaßfaktor beim Spielen zu erhöhen. Dabei kam schließlich der fischertechnik 3D Printer (536624) zum Einsatz, und ich verwendete Treiber aus der ft:pedia – aber dazu später mehr.

Das Ergebnis meiner Tüfteleien ist ein einem originalen Flipper nachempfundenes Modell: „Pirates of the Carribean“ (Abb. 2).

Abb. 2: Flipper „Pirates of the Caribbean“

Der Aufbau

Der Flipper hat die Maße 60 cm x 29 cm x 67 cm. Er besteht aus dem Grundkörper, dem Kopfaufbau und dem Gestell (Abb. 2).

Der Bau der Grundplatte machte am Anfang Schwierigkeiten, denn ich benötigte sowohl für die Aufbauten als auch auf der Unterseite für die Füße Nuten zum Befestigen. Die Lösung war, zwei Grundplatten 500 (32985) mit einem Klemmstift 15 x 4,1 mm (107356) zu verbinden (Abb. 3).

Abb. 3: 32985 Grundplatten 500 mit Klemmstift (107356)

Für das Grundgestell und die Beine habe ich MakerBeam XL-Profile (15 x 15 mm) verbaut. Statikbauteile erwiesen sich als zu instabil bei Bewegungen (Abb. 4).

Abb. 4: Grundgestell mit MakerBeam XL-Profilen

Jetzt trat allerdings ein neues Problem auf: Die verstellbaren Gelenkfüße sollten mit den Profilen verbunden werden. Meine Lösung war, einen Fußhalter zu konstruieren (Abb. 5) und diesen mit dem fischertechnik 3D-Printer auszudrucken (Abb. 6).

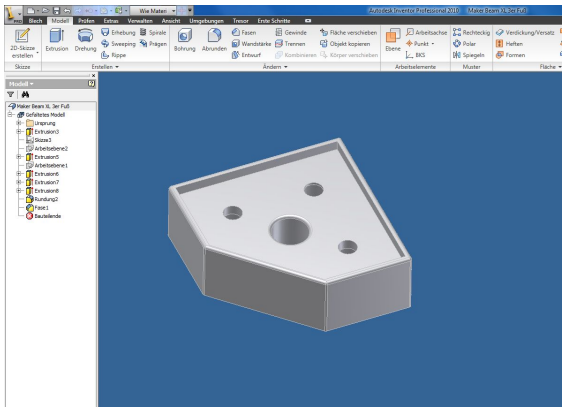


Abb. 5: AutoCAD Inventor 2010 Fußhalter mit Bohrungen

Abb. 6: Fußhalter mit Gelenkfuß und M3-Innensechskantschrauben

Als Lauffläche für die Kugeln habe ich eine Acrylglasscheibe verwendet. Alle Aufbauten des Flippers sind mit UHU 2-K Epoxidkleber auf der Acrylglasscheibe aufgeklebt (Abb. 7).

Abb. 7: Acrylglasscheibe

Die Mechanik

Da der Aufbau durch die Statikplatten 90 x 180 mm (36321) erhöht ist, war es eine weitere Herausforderung, die Kugel aus der

Flexschiene (grün) von unten nach oben in die Abschussrinne (blau) zu bekommen. Gelöst habe ich dieses Problem, indem ich die Kugel über einen Magnetkugelhalter (119850) mitnehmen lasse (Abb. 8).

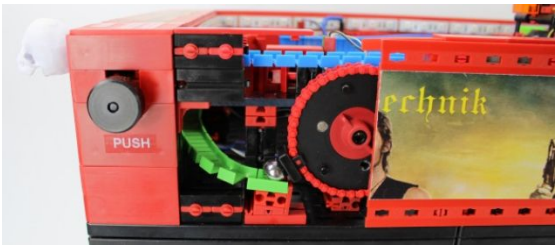


Abb. 8: Die Kugelförderung

Dieser ist über eine Rastkette auf einem Zahnrad Z40 (31022) montiert. Oben wird die Kugel über eine Schräge abgestreift (Abb. 9).

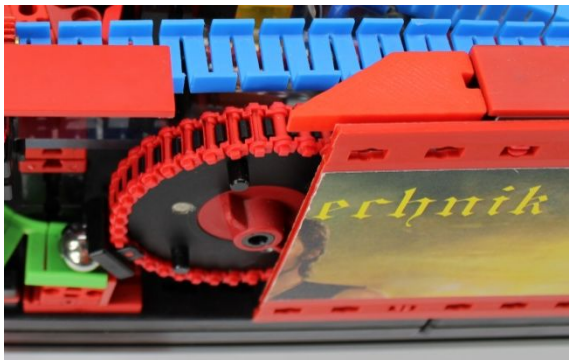


Abb. 9: Schräge zum Abstreifen

Mit Original-Bauteilen und verschiedenen 3D-Druck-Varianten habe ich mich an die Lösung für das Abstreifen der Kugel herangetastet (Abb. 10). Das Zahnrad Z40 bleibt unterhalb der grünen Flexschiene stehen, gestoppt von einem Reedkontakt.

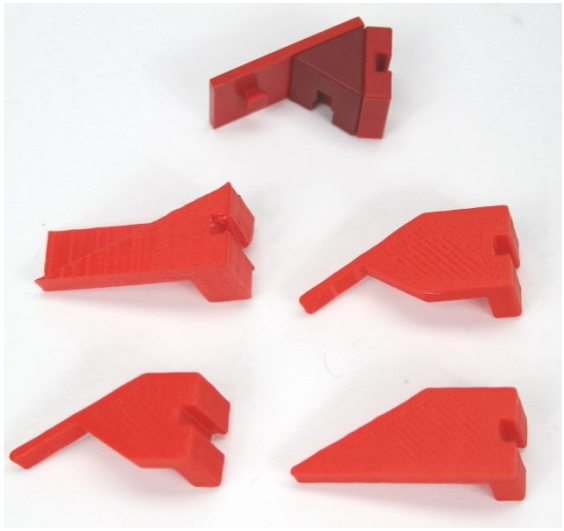


Abb. 10: Varianten des „Abstreifers“

Was sich im nachherein als sehr wartungsfreundlich erwiesen hat, sind die Statikplatten an den Außenseiten. Dadurch kommt man leichter an die verbauten Bauteile, Stecker und Kabel heran (Abb. 11).

Abb. 11: Rechte Seite des Flippers (geöffnet)

Der Kopfaufbau

Der Kopfaufbau (*Backbox*) am Ende des Flippers ist 26 cm x 7 cm x 27 cm groß (Abb. 12). Vorne zu sehen sind die vier Serial 7-Segment Display von Sparkfun für die Punkteanzeige. Die Bicolor 8 x 8 LED Matrix von Adafruit habe ich für die Animation einer Laufschrift verwendet.

Abb. 12: Der Kopfaufbau

Links und rechts davon sieht man die Ziergitter für die dahinter befindlichen Soundmodule. Auf dem Kopfteil habe ich ein Blaulicht befestigt, bestehend aus vier LEDs.

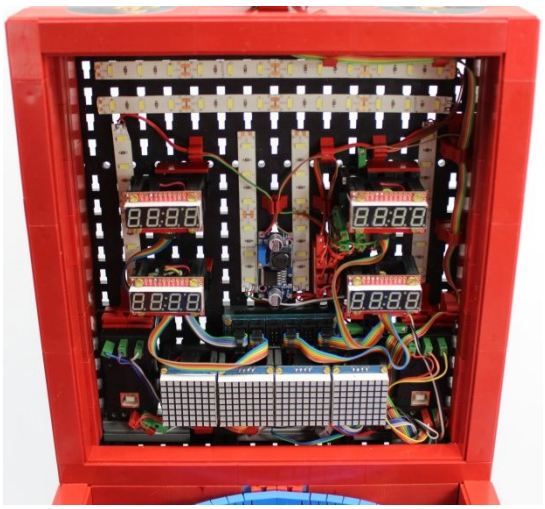


Abb. 13: Elektronik im Kopfaufbau

Die Elektronik

Nun ein Blick in das Innere des Kopfaufbaus (Abb. 13). Zu sehen sind die weißen LED-Stripes (12 Volt) für die Hintergrundbeleuchtung der Frontscheibe. Die Helligkeit wird über einen Spannungswandler (mittig) geregelt. Die Ansteuerung der Segment- und der Matrixanzeigen erfolgt über einen I^2C-Verteiler. Links und rechts sieht man die beiden Soundmodule (Abb. 14).

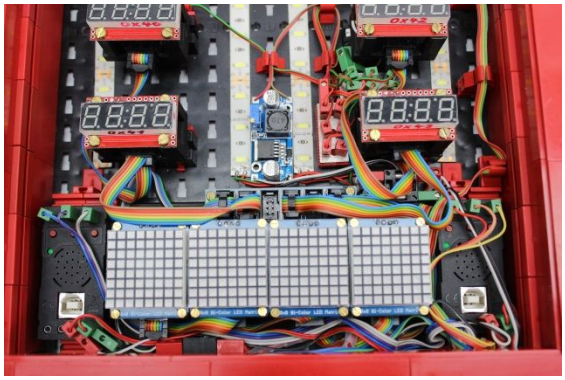


Abb. 14: Soundmodule links und rechts

Um die Segment- und Matrixanzeigen alle auf die gleiche Bauhöhe zu bringen, habe ich mir Halterungen konstruiert und mit dem 3D-Printer ausgedruckt (Abb. 15).

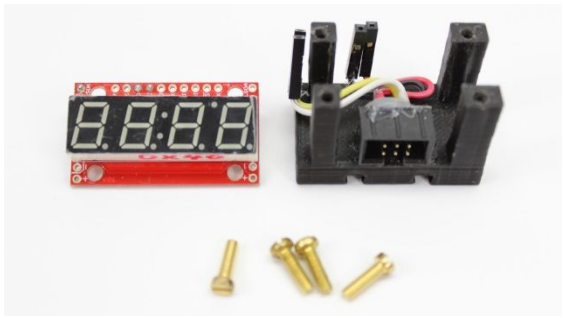


Abb. 15: Segmentanzeige mit Halter

Das Kopfteil kann schnell montiert bzw. demontiert werden, indem man es einfach von hinten aufschiebt. Die Elektronik-Komponenten sind über einen Wannenstecker (mittig rechts) miteinander verbunden (Abb. 16).

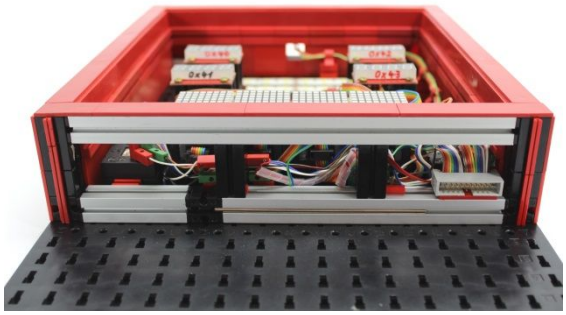


Abb. 16: Kopfaufbau von unten

Auf der Rückseite seht ihr die Ansteuerung des Flippers über zwei ROBO TX Controller (500995). Die Ausgänge der Controller haben nicht für die komplette Ansteuerung des Flippers ausgereicht, daher hatte ich mich für die Verwendung von I^2C-Komponenten entschieden (Abb. 17).

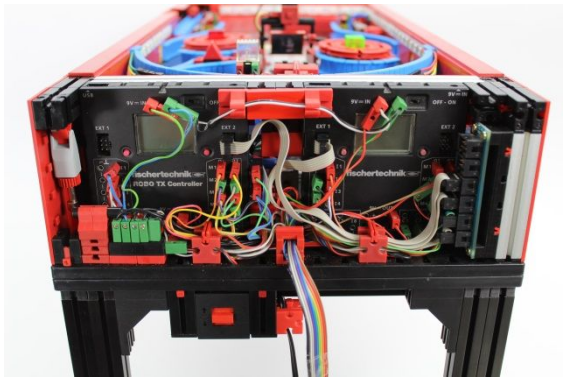


Abb. 17: Rückseite Elektronik

Damit der Flipper möglichst authentisch funktioniert, habe ich einen Tilt Reedkontakt mit einem Magnetpendel konstruiert (Abb. 18).

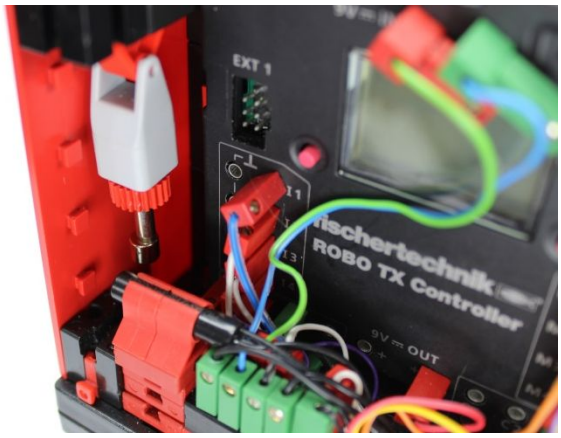


Abb. 18: Tilt Reedkontakt mit Magnet

Das Spielfeld

Das Spielfeld unterscheidet sich in der Größe nicht wesentlich von dem originalen fischertechnik-Flipper. Links und rechts sind jeweils zwei Fototransistoren und oben ist ein Farbsensor verbaut (Abb. 19).

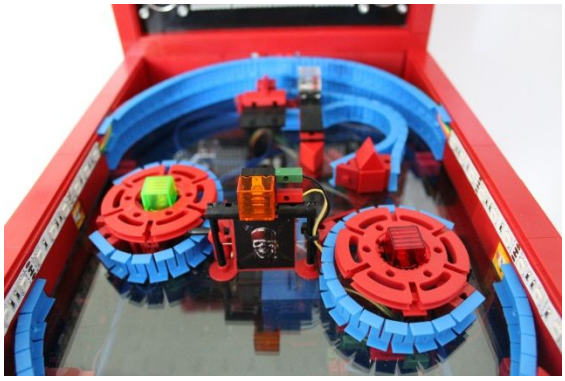


Abb. 19: Spielfeld

Mittig seht ihr ein Tor, welches aus einer Bauplatte 30 x 30 mm (38259) besteht und über zwei Stecknadeln drehend gelagert ist. Über einen kleinen Neodym-Magneten an der Bauplatte und einen Reedkontakt wird die Drehung in Punkte umgerechnet (Abb. 20).

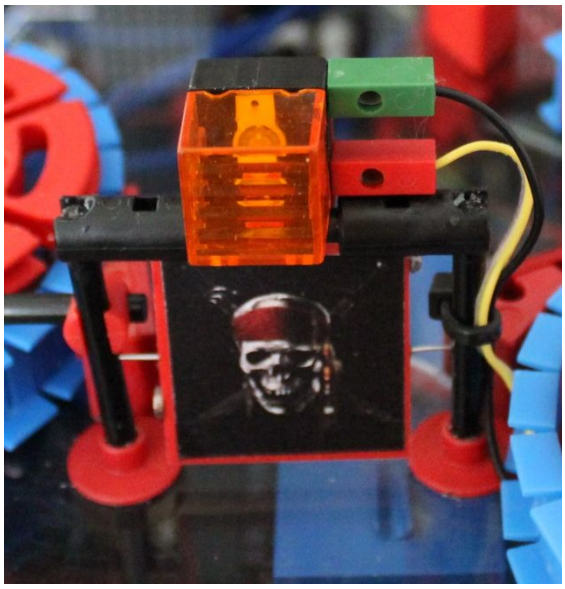


Abb. 20: Drehend gelagertes Tor

Um den Flipper mit mehr Lichteffekten auszustatten wurden farbige Leuchtmittel und innen ein *RGB-LED Stripe* (12 Volt)

verbaut (Abb. 21). Damit lassen sich viele verschiedene Farben erzeugen.

Abb. 21: RGB-LED (12 Volt)

Die Software

Das Ansteuern des Flippers erfolgt über die ROBO Pro-Software. Da viele I²C-Komponenten verbaut worden sind, war die Vergabe der Hex-Adressen und deren Ansteuerung eine große Herausforderung. Das ROBO Pro-Programm besitzt 92 (!) Unterprogramme – viel mehr als in meinen vorherigen Projekten.

Für das Ansteuern der vier *Serial 7-Segmet Displays* habe ich den LED-S7SD-Treiber v1.2 von Dirk Fox aus der ft:pedia 4/2016 eingesetzt.

Um dem Original näher zu kommen wurde ein Mehrspielermodus programmiert. Somit erfolgt das abwechselnde Zählen der Kugeln und Abspeichern der Punkte pro Spieler über Listenelemente.

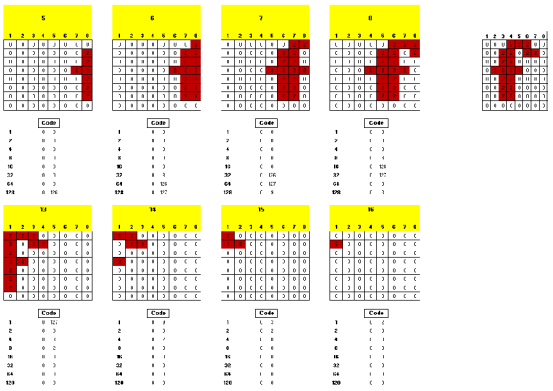


Abb. 22: Excel Hilfsprogramm

Für die *Bi-Color 8 x 8-Matrixanzeige* habe ich den ftpiled Treiber von Christian Bergschneider & Stefan Fuss aus der ft:pedia 4/2016 modifiziert. Damit kann ich nun mehrere Displays hintereinander ansteuern und auch Laufschriften ausgeben. Dazu habe ich mir ein Excel-Programm erstellt, mit dem ich ein Schriftzeichen über die *Bi-Color 8x8-Matrixanzeigen* wandern lasse (Abb. 22). Das Bitmuster speichere ich dann in einer *.csv-Datei ab.

Das Finish

Da der Flipper möglichst authentisch werden sollte, habe ich ihn mit ein paar Gimmicks versehen. Für die beleuchtete Frontscheibe habe ich opakes Acrylglas verwendet. Darauf wurde Backlight-Folie von www.wir-machen-druck.de geklebt. Die Frontscheibe wird von hinten beleuchtet, wie beim Original-Flipper (Abb. 23). Den Totenkopf habe ich bei www.thingiverse.com gefunden und ebenfalls mit dem 3D-Printer ausgedruckt (Abb. 24).

Abb. 23: Frontscheibe mit Backlight-Folie

Abb. 24: Totenkopf

Um noch mehr „Piratenfeeling“ zu bekommen wurde der Flipper komplett mit verschiedenen Wallpaper-Motiven aus dem Internet versehen. Diese habe ich mit *CorelDraw X5* ausgeschnitten, vermessen und auf selbstklebende Folie aufgedruckt (Abb. 25).

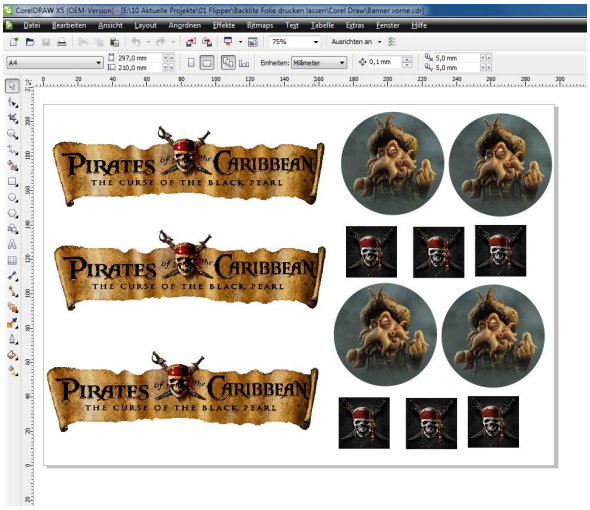


Abb. 25: Aufkleber in Corel-Draw X5

Fazit

Der Flipper soll euch zeigen, wie ihr mit verschiedenen Ideen an ein Modell herangehen könnt. Hierbei ist es mir wichtig zu zeigen, dass ein 3D-Drucker auch eine wertvolle Hilfe sein und eure Kreativität und euren Einfallsreichtum fördern kann. Die Halter zum Nachdrucken mit dem fischertechnik 3D Printer findet ihr im Downloadbereich der ft:c.

Ein Video des Flippers findet ihr auf YouTube.

Referenzen

[1] Christian Bergschneider, Stefan Fuss: *ftpiLED, LED-Backpack im Retrodesign*, ft:pedia 4/2016, S.80-83.

[2] Dirk Fox: *I²C mit dem TX(T) – Teil 14: LED-Display (2)*. ft:pedia 4/2016, S. 84-88.

[3] Sparkfun: *Serial 7 Segment Display Datasheet*. Github.

[4] Adafruit: *Bi-Color 8x8 Matrix*. learn.adafruit.com

Abb. 26: Flipper „Pirates of the Caribbean“