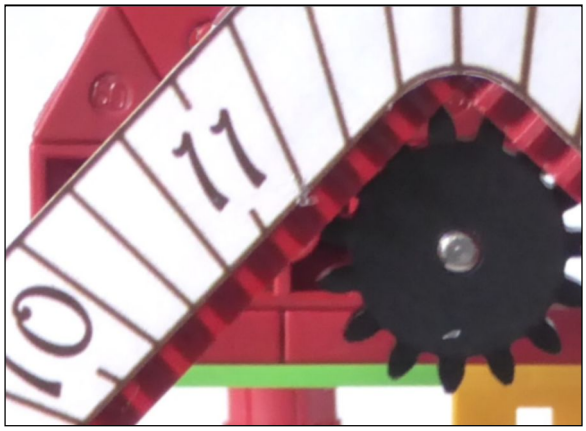
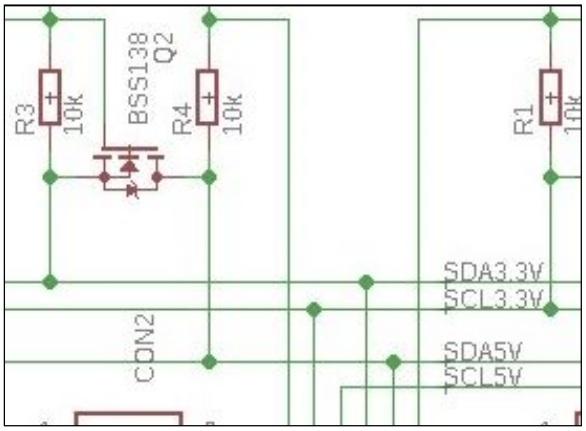
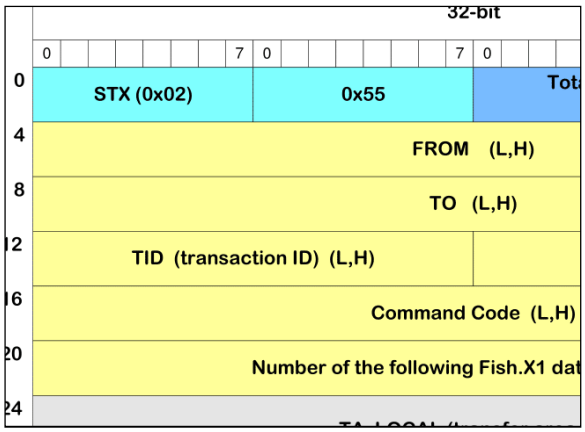
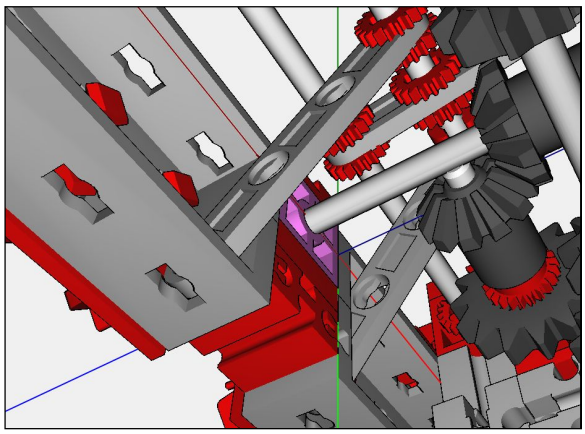
ft:pedia

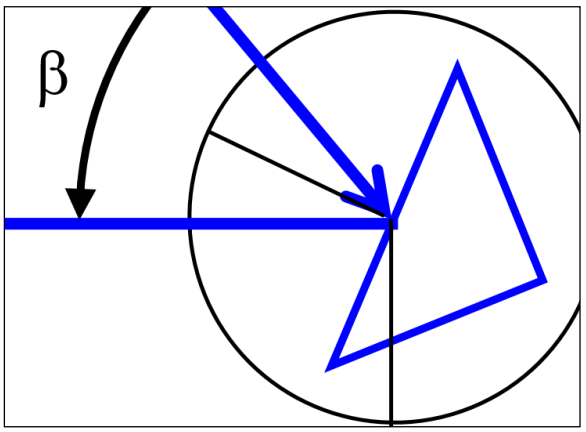
Heft 2/2018











Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

Nichts ist mächtiger als eine Idee, deren Zeit gekommen ist (Victor Hugo)

Dirk Fox, Stefan Falk

Wer sich (wie wir) in seiner ein paar Jahrzehnte zurückliegenden Jugend mit fischertechnik beschäftigt hat, wird sich zweifellos – wie der „Alte-Säcke-Thread“ der fischertechnik-Community immer wieder eindrucksvoll bestätigt – an dieselben Schlüsselerlebnisse erinnern:

- die unbeschreiblichen Flow-Zeiten, in denen wir Hunger, Durst und das Gefühl für die Zeit vollständig verloren,
- die ungezählten Aha-Erlebnisse, wenn wir auf einmal einen elementaren Zusammenhang (ein Getriebe, eine statische Versteifung, eine Schaltung) verstanden,
- die Glücksgefühle, wenn ein Modell nach tagelangen Konstruktionsarbeiten schließlich genau so funktionierte, wie wir uns das vorgestellt hatten.

Einige dieser Erlebnisse halten fraglos einen Ehrenplatz in der Liste unserer „Das-werde-ich-nie-vergessen“-Erinnerungen.

Welchen Grund könnte es geben, dass Kinder und Jugendliche diese Erlebnisse heute anders empfinden? Und dennoch existieren offenbar immer weniger Haushalte, in denen Technik-Talente dafür eine Chance bekommen. In Karlsruhe haben wir daher im Herbst 2013 an einem Gymnasium eine fischertechnik-AG initiiert – und ein ‚Kinderzimmer‘ mit fischertechnik ausgestattet.

Der Erfolg war so durchschlagend, dass wir im Jahr darauf begonnen haben, nach Sponsoren für weitere fischertechnik-AGs an anderen Schulen zu suchen. 2017 wurde die Initiative von Körber-Stiftung und Stifterverband als „MINT+“-Region ausgezeichnet; mit dem Preisgeld konnten wir eine befristete 50%-Stelle für einen MINT-Koordinator einrichten. Kürzlich hat die 38. Schule in der Region ihre fischertechnik-AG bekommen; acht weitere warten auf einen Sponsor. Fast 700 Schülerinnen und Schüler konstruieren und programmieren nun jede Woche mindestens zwei Stunden an ihren fischertechnik-Modellen – und besuchen Workshops (wie beim MINT-Feriencamp Ende Mai), gestalten den jährlichen fischertechnik-Tag der Karlsruher Grundschulen und messen sich mit ihren Robotern beim Karlsruher Schul-Robotik-Cup. Die ersten Grundschulen haben jetzt mit Programmier-AGs ab Klasse 3 begonnen, und gerade hat ein Sponsor der Initiative erhebliche Mittel in Aussicht gestellt, um nun auch die erforderliche Qualifizierung der Lehrerinnen und Lehrer in Technik und Informatik systematisch und in Kooperation mit den Karlsruher Hochschulen anzugehen.

In gerade einmal vier Jahren ist aus technikfreien Bildungsinseln eine vernetzte MINT-Bildungslandschaft geworden – und eine Blaupause für andere Regionen...

Beste Grüße,
Euer ft:pedia-Team

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

Inhalt

Termine

Was?	Wann?	Wo?
fischertechnik FanClubTag 2018	30.06.2018	Waldachtal
Süd-Convention	22.09.2018	Dreieich

Impressum

http://www.ftcommunity.de/ftpedia

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

Autoren: Christian Bergschneider, Stefan Falk, Dirk Fox, Stefan Fuss, Björn Gundermann, Till Harbaum, Werner Hasselberg, David Holtz, Raphael Jacob, Christian Lauff, Rüdiger Riedel, Rudenz Schulz, René Trapp.

Modell

Savonius-Rotor mit Magnetlager

Rüdiger Riedel

Die Maisonne verwöhnt uns, da möchte man doch auch etwas mit Solarzellen machen. Leider sind meine bisherigen Versuche gescheitert – sollte daraus noch etwas werden, berichte ich. Zum Glück weht ein ordentlicher Wind, da lässt sich auf die Schnelle was Passendes bauen.

Es gibt sie als Werbeträger und Kunstwerke: Windräder mit senkrechter Achse, manchmal nur aus zwei gebogenen Blechen bestehend, die sich langsam und bedächtig im Wind drehen. Bekannt sind auch die Anemometer, die Windmesser, die meist aus vier kugeligen Halbschalen bestehen (deshalb auch Schalenanemometer genannt) und die Windgeschwindigkeit anzeigen.

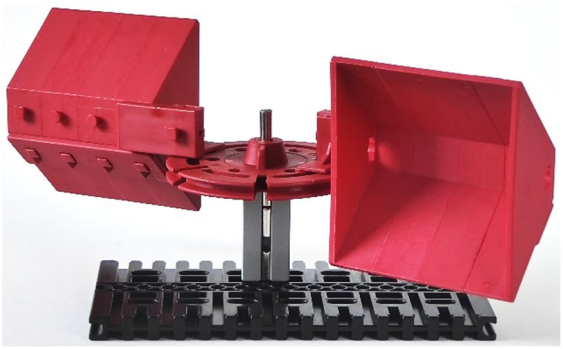


Abb. 1: Windrad mit Schalen

Das Windrad in Abb. 1 und 2 besteht aus den V-Schaufelteilen (37353, 37354 und 37355) und S-Aufnahmestreifen 60 (35771). Es lässt sich auch mit einfachen Bauplatten realisieren (Abb. 3).

Um den Reibungswiderstand zu verringern, habe ich unten an der Achse 30 einen Stabmagneten (Durchmesser 4 mm, Länge 10 mm) angefügt. Ein gleichartiger befindet sich im Baustein 30 so, dass sie sich abstoßen (Abb. 4).

Abb. 2: Ein Widerstandsläufer

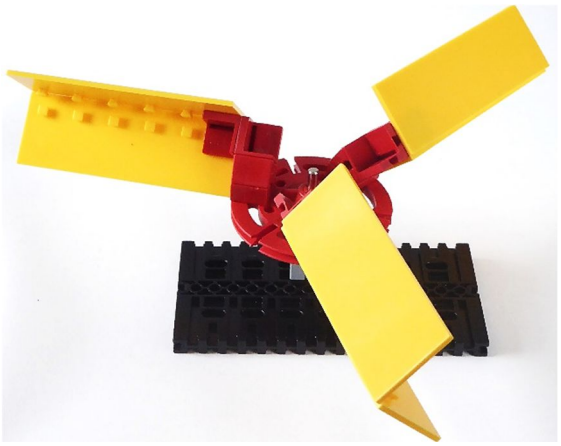


Abb. 3: Windrad mit Bauplatten

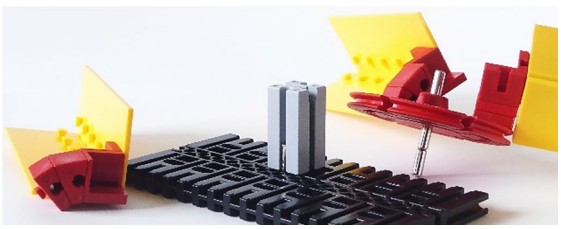


Abb. 4: Aufbau mit Entlastungsmagneten

Diese Windräder sind Widerstandsläufer, ihr Funktionsprinzip beruht auf dem unterschiedlichen Strömungswiderstand der Halbschalen. Im Gegensatz dazu sind die weit verbreiteten Windkraftanlagen mit ihren waagerechten Achsen sogenannte Auftriebsläufer.

Der Baggerlöffel 45 (156104) funktioniert bestimmt sehr gut, leider besitze ich nur einen. Mit den Kotflügeln (35050) hatte ich keinen Erfolg.

Ein wenig Aerodynamik

Jeder kennt den Luftwiderstand, den Fahrtwind, vom Fahrradfahren: Je schneller man fährt, desto größer ist der Widerstand. Wenn man sich aber duckt wie die Radrennfahrer, kommt man schneller voran oder braucht weniger Kraft.

Der Luftwiderstand F_w hängt unter anderem von der Querschnittsfläche des Körpers und vom Quadrat der Geschwindigkeit ab und lässt sich wie folgt berechnen:

$$F_w = c_w \rho A \frac{v^2}{2}$$

Dabei gilt:

- A Querschnittsfläche des Körpers zur Strömungsrichtung
- c_w Widerstandsbeiwert
- ρ Dichte des Fluids (Gas bzw. Flüssigkeit)
- v Geschwindigkeit des Fluids

Für uns interessant ist der Widerstandsbeiwert, der von der Geometrie des Körpers abhängt. Umgangssprachlich ausgedrückt ist der c_w-Wert ein Maß für die „Windschlüpfrigkeit“ eines Körpers.

In Abb. 5 sieht man im Querschnitt eine offene Halbkugel (z. B. ein halber Tischtennisball) mit dem Rücken zum Luftstrom, eine ebene Platte und eine offene Halbkugel mit der Öffnung zum Luftstrom. Die unterschiedlichen Widerstandsbeiwerte werden für die Windräder mit Widerstandsläufer ausgenutzt.

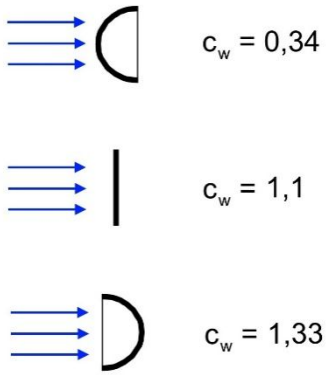


Abb. 5: Drei Körper im Luftstrom

Bei Autos soll der Widerstandsbeiwert möglichst klein sein, um Energie zu sparen, und liegt unter 0,5. Überraschend ist der recht hohe c_w-Wert von Formel-1-Rennwagen, der laut Internet bei 0,7 bis 1,2 liegen soll. Begründung: Der Anpressdruck auf die Straße ist wichtiger und Leistung zur Überwindung des Luftwiderstandes ist reichlich vorhanden.

Der Savonius-Rotor

In den sechziger Jahren des letzten Jahrhunderts sah man oft Rotoren auf dem Dach von Transportfahrzeugen, Omnibussen und Eisenbahnwaggons: Vom Wind bzw. Fahrtwind angetrieben sorgten sie für die Entlüftung (Abb. 6).

Abb. 6: Savonius-Lüfter auf einem alten Eisenbahn-Waggon

Das Windrad drehte sich im Fahrtwind, darunter befestigt war ein Lüfterrad, das die Luft aus dem Fahrzeug herauszog. Das Antriebsprinzip beruht auf dem *Savonius-*

Rotor. Sigurd Johannes Savonius (1884-1931) war ein finnischer Architekt und Erfinder. Savonius hatte den *Flettner-Rotor* (Anton Flettner, 1885-1961, deutscher Lehrer, Ingenieur und Erfinder) im Blick, der auf dem *Magnus-Effekt* beruht. Jeder Fußball- oder Tennis-Spieler kennt den Effekt: Ein sich drehender (angeschnittener) Ball fliegt eine veränderte Flugbahn (mit „Effet eine Bananenflanke“).

Der Flettner-Rotor war ein einfacher, um die lange Achse rotierender Zylinder, der bei Wind eine dazu senkrechte Kraft entwickelte. Als Ersatz für die damaligen Segelschiffe konnte er sich nicht durchsetzen. Die Idee von Savonius bestand im Halbieren des Zylinders und der Versetzung der Halbschalen um einen Durchmesser (Abb. 7). Das ergab eine analoge Form zum Halbschalen-Anemometer.

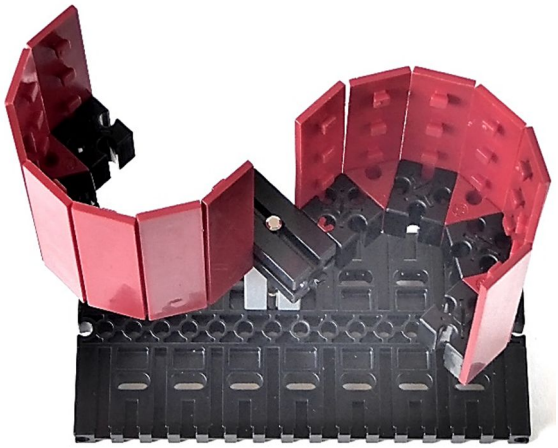


Abb. 7: Die ursprüngliche Version

Die wichtigste Verbesserung von Savonius bestand dann in der Verringerung des Abstandes der Halbschalen.

Jetzt wird die Luft nicht gezwungen rückwärts wieder hinaus zu strömen (bei Widerstandsläufern ist die Drehgeschwindigkeit des Rotors stets kleiner als die Windgeschwindigkeit), was zu Turbulenzen führt, die Energie kosten, sondern umgelenkt. Das Hindurchfließen wirkt auf die Innenseite der bremsenden Schale zusätzlich beschleunigend: Der Wirkungsgrad wird höher.

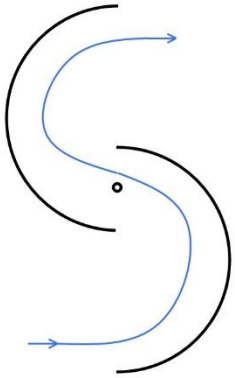


Abb. 8: Luftströmung

In Abb. 8 wird die Luft von der unteren offenen Halbschale weitergeleitet in die andere Halbschale und verstärkt so das Drehmoment. Hier erfolgt eine Linksdrehung.

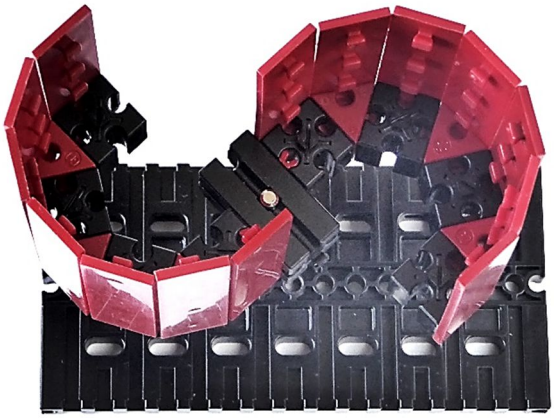


Abb. 9: Der fischertechnik-„Savonius“

Die Halbschalen in Abb. 9 sind mit Federnocken (31982) am BS30 mit Loch befestigt und können so verschoben werden.

Wie in Abb. 1 und 4 erhalten die Achse 30 und der senkrechte graue BS30 je einen Stabmagnet zur Entlastung des Lagers. Außerdem erhält die Achse einen Klemmring, der trägt den BS30 mit Loch. Alle vorgestellten Rotoren drehen sich munter schon bei mäßigem Wind.

Überraschenderweise gibt es diese Savonius-Lüfter heute noch zu kaufen, allerdings unter der Bezeichnung *Flettner-Ventilator*.

Modell

Ferngesteuerter Raupenbagger

Werner Hasselberg

Bagger sind des Bastlers liebstes Kind: Sie vereinen viele spannende, technische Raffinessen vom Kettenantrieb über Schwenkmechanismus und Baggerarm bis hin zur Schaufel. Alles Dinge, die gerade mit fischertechnik viel Spaß bei der Umsetzung machen, weil sie abseits der normalen Hydraulik wunderbar flexible Modelle ermöglicht. Flexibel deshalb, weil ein Hydraulikzylinder immer auf seine vorgegebene Größe beschränkt ist. fischertechnik dagegen ist grenzenlos, da anstelle der Zylinder mechanische Getriebe zum Einsatz kommen, die beliebig an die Modellgröße angepasst werden können. Der hier vorgestellte Bagger ist über 70cm lang und damit ein echtes Großmodell wie aus den guten alten Tagen des Schwerlastkrans oder der Containerbrücke. Insbesondere ist das Modell mit seinen sechs Motoren voll fernsteuerbar – mit nur einem Empfangsmodul. Das funktioniert ganz ohne technische Sonderteile oder spezielles Elektronikwissen mit der originalen fischertechnik-Elektronik bzw. Elektromechanik.

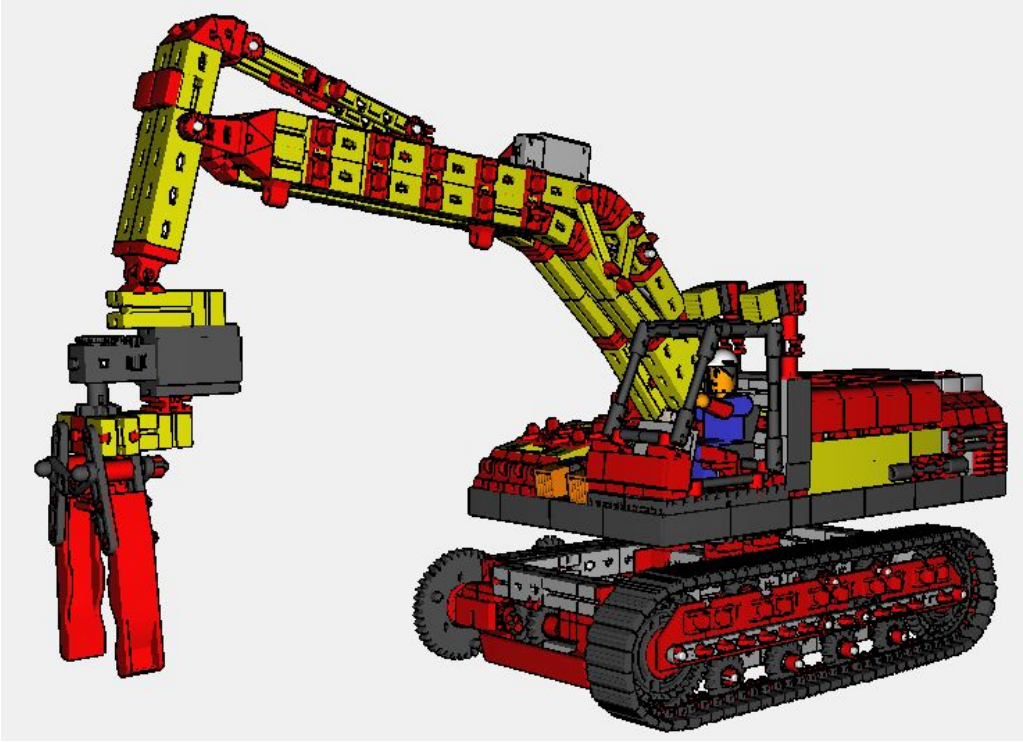


Abb. 1: Das fertig im fischertechnik-Designer entwickelte Modell (noch ohne rechte Kette)

Das Chassis

Es muss stabil aufgebaut und kräftig im Antrieb sein, weshalb zwei XM-Motoren zum Einsatz kommen. Jeder Motor wird doppelt durch ein Z20 und ein Z10 untersetzt und am Ende nochmal mittels einem Z15 und Z10. Das gibt enorme Kraft, ohne dass die Geschwindigkeit vernachlässigt wird. Der Bagger rollt auf diese Weise recht flott durch die Gegend.

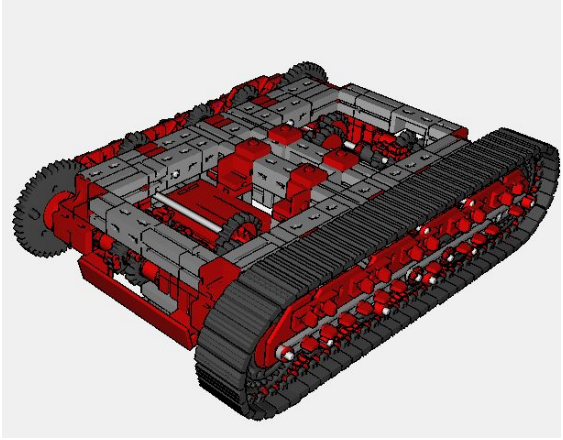


Abb. 2: Bagger-Chassis

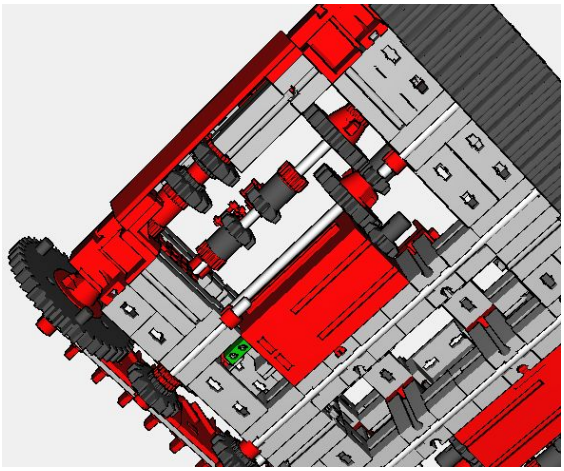


Abb. 3: Motoreinbau und Kabelanschluss

Beim Anbringen des jeweiligen Motors links und rechts ist zu beachten, dass die Statik-Winkelträger 30 unmittelbar hinter dem Motor so angebracht werden, dass die Seite mit der Nut auf der Seite der Stecker liegt (Abb. 3). Denn dies ist die schmalere Seite des Winkelträgers 30, und so kann der Stecker exakt an ihr entlang zum Motor geführt werden.

Die Kabelanschlüsse werden am Motor vorbei unter dem mittleren Aufbau nach oben zum Bagger geführt. Er enthält den Akku und die Steuerung. Somit kann er nicht beliebig oft um seine eigene Achse schwenken, weil sich sonst die Kabel zu sehr verdrehen. Das ist aber ein zu vernachlässigender Aspekt.

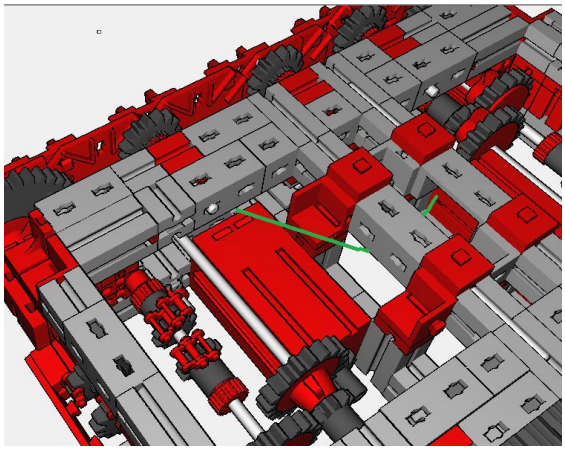


Abb. 4: Kabel nach oben zum Akku führen

Der Antrieb geht direkt an die Hauptachsen der großen Zahnräder an Front- und Rückseite, was eine bessere Lösung als ein Antrieb über eines der Seitenräder darstellt, weil die Ketten schnell mal über sie springen können. Die Frontachsen dürfen selbstverständlich nicht durchgehend sein, weil der Bagger ja gelenkt werden muss, die Ketten also unterschiedlich schnell oder entgegengesetzt laufen müssen.

Beispielsweise hat man sich das im Lego-Technik-Seilbagger gespart, sodass dieser leider nur vor- oder rückwärts fahren kann. Ein Bagger sollte aber auch gelenkt werden können. Die folgende Abb. 5 zeigt, wie an beiden Enden jeweils zwei Achsen zum Einsatz kommen.

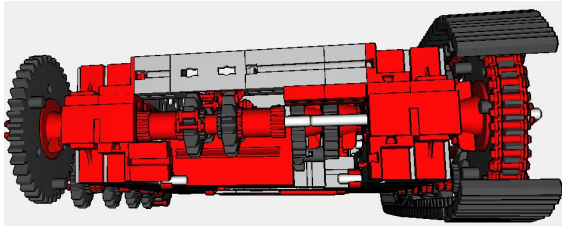


Abb. 5: Achsen der Haupträder

Das rechte Rad ist also nur lose eingeschoben und nicht gegenfixiert, was aber nichts ausmacht, da es von der Kette gut gehalten wird, sogar bei wilderen Lenkmanövern. Problemtisch ist eher die linke Stange, da sie einen Gegenhalt auch auf der rechten Seite benötigt, denn die Ketten übertragen viel Kraft und Zug auf die großen Antriebsräder. Das ist auch der

Grund, weshalb zwei Z15 verwendet werden. Beide sind fest auf einer Stange fixiert und treiben so das linke große Zahnrad an.

Dasselbe Konstrukt befindet sich auf der Bagger-Rückseite, nur mit dem Unterschied, dass sich das Antriebsrad auf der anderen Seite befinden muss – logisch. Abb. 6 zeigt, wie die lange Stange des linken Rades und zusätzlich auch das rechte Rad fixiert werden.

Dazu werden an die Achsen die roten BS15 mit Bohrung geschoben und am BS5 montiert.

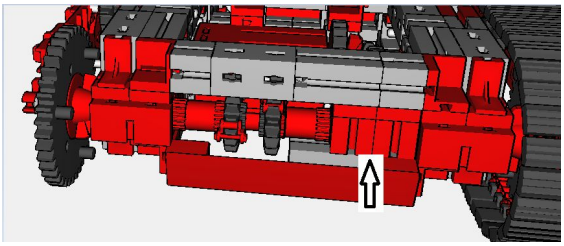


Abb. 6: Gegenhalt der Achsen

Das Chassis ist mit zwei Achsen 200 an der Unterseite stabilisiert und kann sich so seitlich nicht durchbiegen. An den Enden dieser Achsen befinden sich auf jeder Seite Z15. Die übrigen Z15 für die Kettenführung werden mit Achsen 60 angebracht.

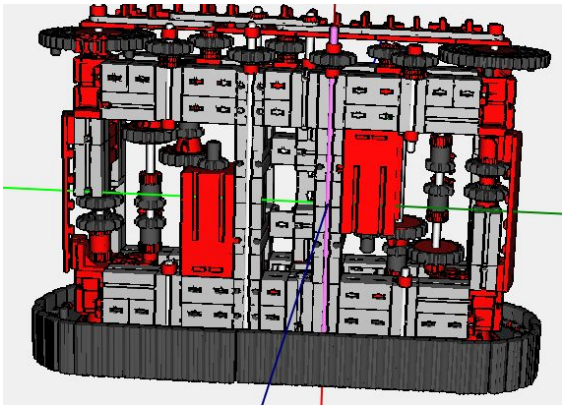


Abb. 7: Achsen 200 und 60

Unten führen auf jeder Seite vier Z15 die Kette. Oben genügen auf jeder Seite drei. Um das Gewicht so gering wie möglich zu halten, sind im Chassis viele Statik-Winkelträger anstelle der BS30 und BS15 verbaut.

Außerdem werden die oberen Z15 an Achsen 50 befestigt (Abb. 8).

Abb. 8: Achsen 50 mit den oberen Zahnrädern

Zur Abdeckung der Seiten werden Doppelknoten- und Eckknotenplatten mit Statikstreben sowie Laschen verbunden und an die beiden Achsen 200 mittig sowie an die Achsen der großen Zahnräder mit Klemmbuchsen befestigt. Fertig ist das Chassis.

Abb. 9: Seitenschutz

Der Bagger

Das grundsätzliche, aber lösbare Problem ist die Mechanik des Baggerarms, besonders wenn man alles frei gestalten will und deshalb die Standard-Pneumatik-Zylinder nicht verwenden kann, weil deren Größe nicht passt. Das ist aber kein Problem, denn fischertechnik bietet Alternativen, die mal wieder zeigen, wie flexibel es ist.

Somit werden uns die klemmbaren Schnecken in Kombination mit einem starken Getriebe (wie das mot. 2) zeigen, was in ihnen steckt.

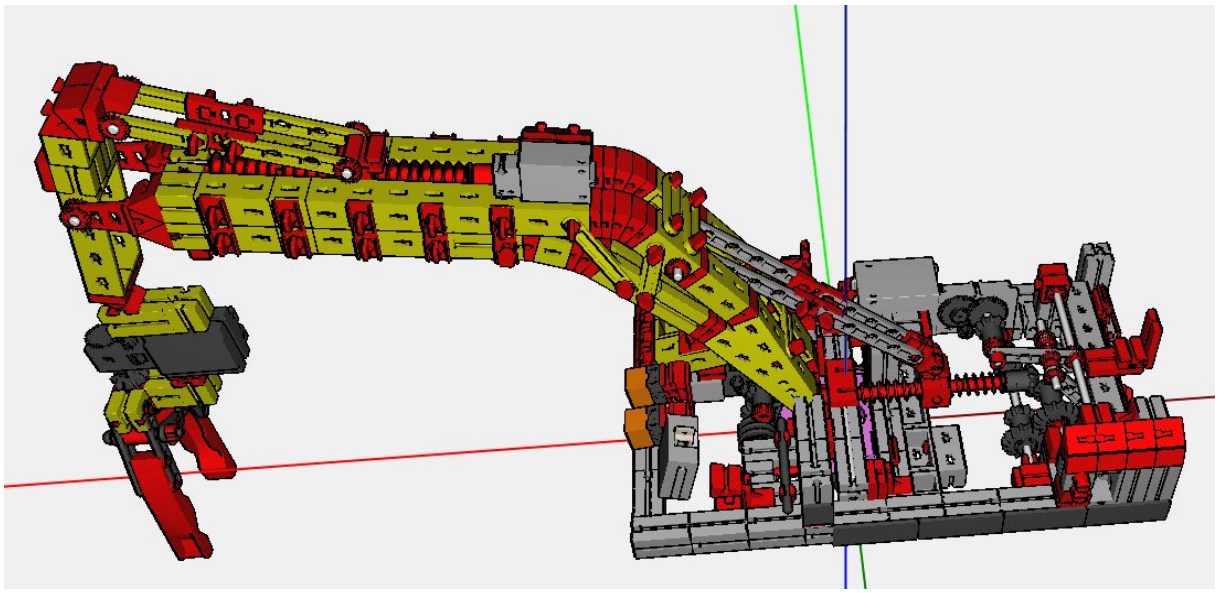


Abb. 10: Arm-Mechanik

Das mot. 2 dreht in möglichst großer Untersetzung über eine Kette und Kegelzahnräder, womit der Querantrieb in einen Längsantrieb umgelenkt wird, die Schnecken entlang einer Metallachse. Das sorgt für eine unglaubliche Kraftübertragung, wobei das Schneckengewinde das Gewicht des Armes auffängt und er damit nicht ständig an den Zahnrädern zerrt. Das schließt auch aus, dass er, wenn er länger mit gehobenen Arm steht, langsam infolge seines Gewichtes nachgibt. Die folgende Abb. 11 zeigt die Getriebemechanik noch etwas detaillierter.

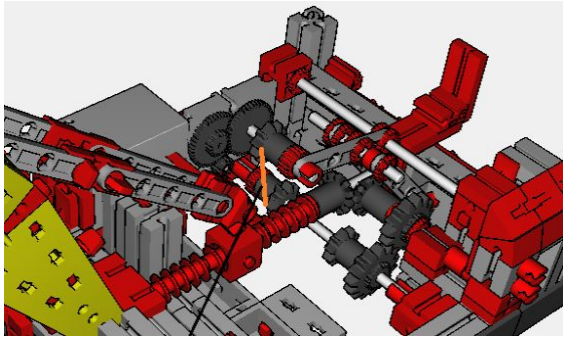


Abb. 11: Arm-Mechanik

Die Statik-Streben des Baggerarms sind dort noch nicht mit der Schneckenmutter verbunden. Das liegt aber nur daran, dass es im Designer schwierig ist, den Arm als Ganzes zu bewegen, sodass er direkt passt.

Beim echten Bau kann der Arm ja beliebig manuell auf und ab bewegt und damit an der Schneckenmutter befestigt werden.

Die Metallachse mit den Schnecken muss stabil verbaut werden, damit sie das ordentliche Gewicht des Armes auch halten kann.

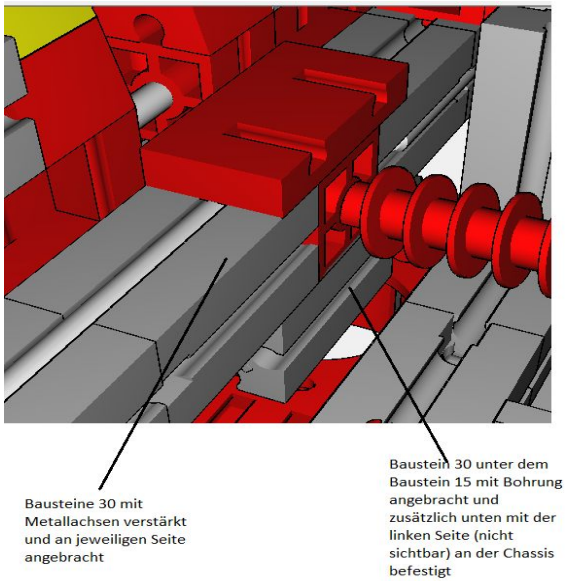


Abb. 12: Schnecken-Achse vorne befestigt.

Hinten halten doppelte Statik-Winkelträger, mit roten Bauplatten „verschraubt“, das Armgewicht ebenso locker und stabil.

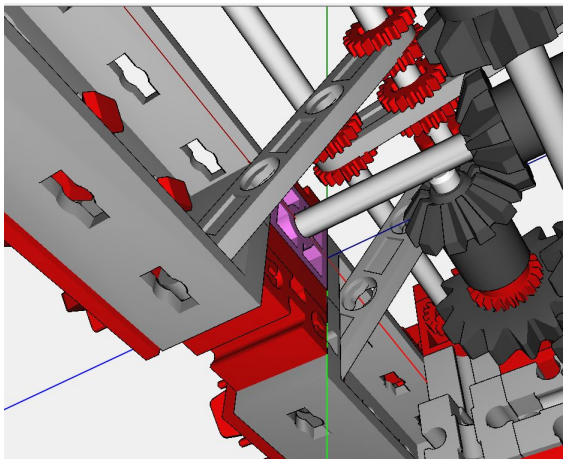


Abb. 13: Schnecken-Achse hinten befestigt.

Weitere Statik-Streben, leicht schräg in die Winkelträger geschoben, spannen die Kette. Damit kann sie bei Belastung nicht über das Zahnrad springen. Das Oberteil sieht dann so aus:

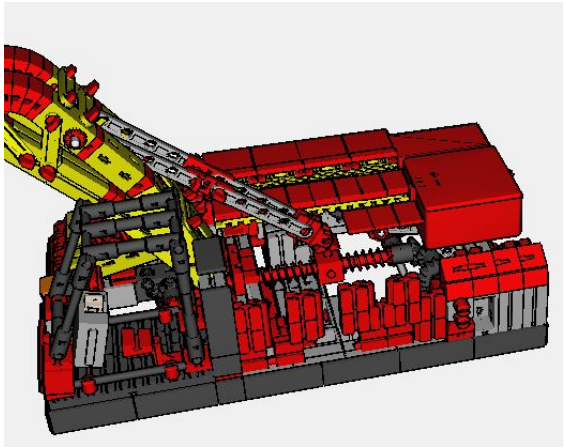


Abb. 14: Gehäuse

Die roten BS7,5 an der Seite halten die Seitenplatte. Die Lücken sparen ein paar der kostbaren Steine und bieten ausreichend Halt. Auf der anderen Seite, die in Abb. 14 bereits komplett aufgebaut ist, wird genauso vorgegangen.

Die Bagger-Drehung

Die folgende Abb. 15 zeigt die Drehmechanik. Sie wird unten am Baggeroberteil angebracht und dreht sich gemeinsam mit dem Bagger. Die in der Abbildung gezeigte Schnecke entspricht jedoch nicht der tatsächlich im Modell verbauten – dort befindet sich stattdessen die ursprünglich zu diesem Drehkranz für das mini-mot-Getriebe angefertigte rote Schnecke mit Metallstange und dazugehörigem Zahnrad.

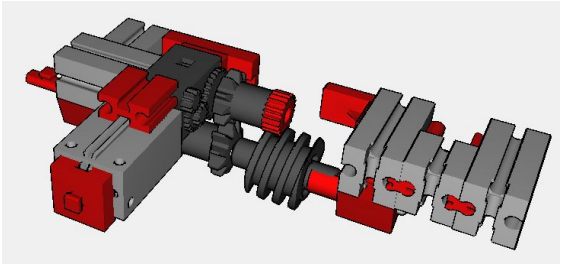


Abb. 15: Drehkranzmechanik

Leider ist sie in der derzeitigen Designer-Version (Stand 2018) nicht als Bauteil enthalten, weshalb die dargestellt Lösung an deren Stelle tritt. Einfach beim Nachbau die rote anstelle der schwarzen einbauen. Sie passt exakt in die dafür vorgefertigte Umgebung.

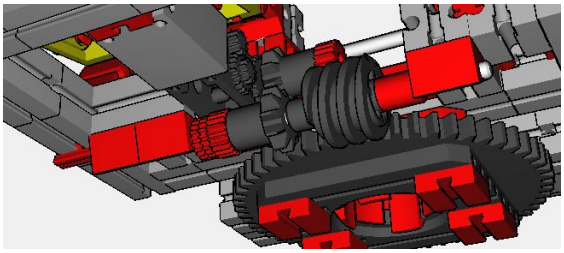


Abb. 16: Drehkranzmechanik eingebaut

Die Fernsteuerung

Kommen wir schließlich zum wichtigsten Teil, der versprochen vollständig fernsteuerbaren Funktionalität des Baggers. Es gibt nur eine Winzigkeit, die davon abweicht: Das ist der Polwendeschalter in der Kabine. Mit ihm werden alle Lampen oberhalb des Führerhauses und im Chassis vorne eingeschaltet. Die mechanische Steuerung erfolgt aber komplett über die Fernbedienung.

Erneut zeigt sich die Vielseitigkeit von fischertechnik. Es gibt Taster und eine flexibel nutzbare Servolenkung. Was also tun? Wir nehmen drei Minitaster und montieren sie so, dass die Servolenkung sie gleichzeitig ein- und ausschalten kann. An den Anschluss in der Mitte des Tasters

kommt jeweils eine Plus-Leitung vom Empfänger der Fernbedienung. Damit sind alle drei verfügbaren Motorausgänge schon mal vergeben. Jeder Taster verfügt aber noch über zwei weitere Steckmöglichkeiten. Jede davon erhält die Plusleitung von je einem Motor, zusammen also sechs. Somit haben wir hier eine Zwei-Motoren-Steuerung pro Taster. Drei Stück insgesamt, wenn die Taster gedrückt, weitere drei, wenn sie gelöst sind. Die Minusleitungen gehen direkt an die Minusbuchsen des Empfängers. Immer zwei Minusleitungen an einem Minusanschluss. Das wiederholen wir für alle sechs Stück, wie die folgende Abb. 17 zeigt.

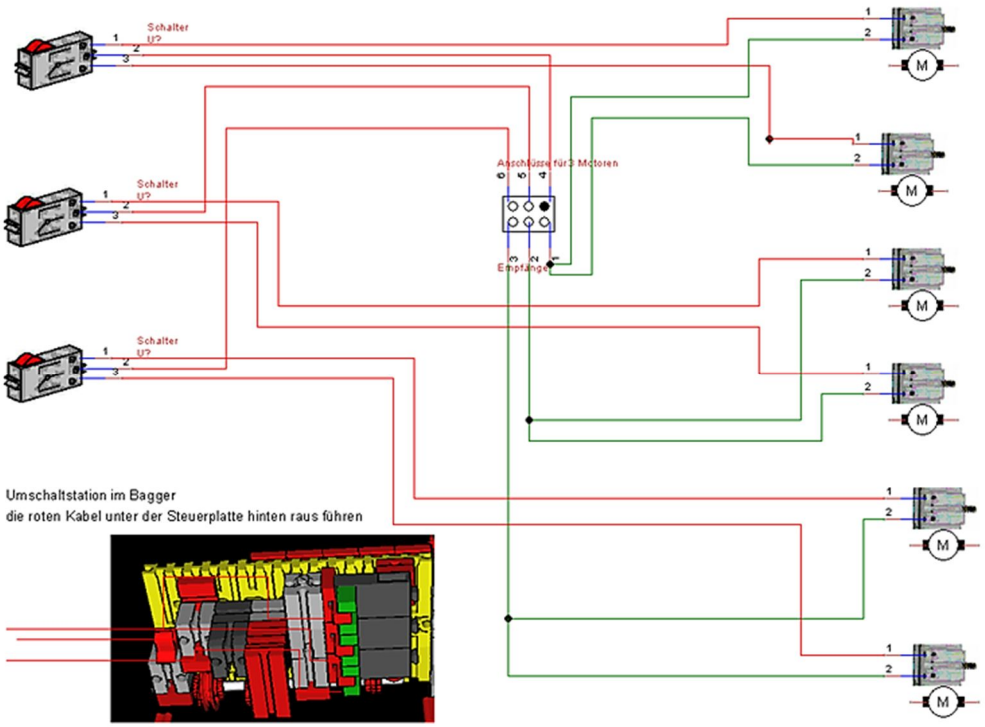


Abb. 17: Schaltplan Motor-Steuerung

Jetzt müssen die Motoren nur noch so angeschlossen werden, dass die Fernbedienung bequem benutzt werden kann. Folgendes bietet sich an.

- Raupenantrieb und Schwenken bei z. B. „alle Taster aus“.
- Baggerarm bei „alle Taster ein“.

Die Taster-Schaltung

Der Servo mit seinem Schwenkhebel wird anstelle der beiden schwarzen BS30 angebracht. Leider enthält der Designer auch den Servo nicht, weshalb dieser Behelf nötig ist. Abb. 19 zeigt aber ein Foto mit eingebautem Servo. Der Hebel liegt im Ruhezustand zwischen den beiden Seilrollen. Wird er in die eine Richtung mit Maximalausschlag bewegt, schiebt er die Stange mit den drei Riegelscheiben am Ende über drei Taster, die dann alle gleichzeitig gedrückt werden, und wählt damit die Gruppe der Motoren für die Steuerung aus.

Beim Zurückschwenken des Hebels in die Mittelposition (Standard, wenn nicht gelenkt wird) wird die Stange nicht zurückgeschoben, sondern hält die Taster weiterhin gedrückt. Erst wenn in die andere Richtung maximal „gelenkt“ wird, schiebt der Hebel die Stange wieder zurück und löst damit die Taster. Analog zu vorher bleiben sie gelöst, auch wenn der Servohebel wieder in seine Ausgangsstellung zurückgeht.

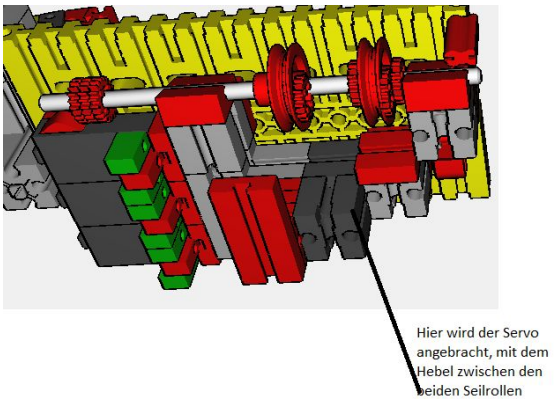


Abb. 18: Tasterschaltung

So kann schnell und einfach zwischen je drei Motoren gewechselt und damit sechs Stück mit nur einer Fernbedienung und einem Empfangsteil bequem gesteuert werden. Mit einem zweiten Empfangsteil wären sogar 12 möglich. Das soll ein anderes System fischertechnik erst einmal nachmachen… Abb. 19 zeigt die Schaltung mit montiertem Servo, Abb. 20 zeigt schließlich das fertige Modell in natura und mit einer klassischen Baggerschaufel.

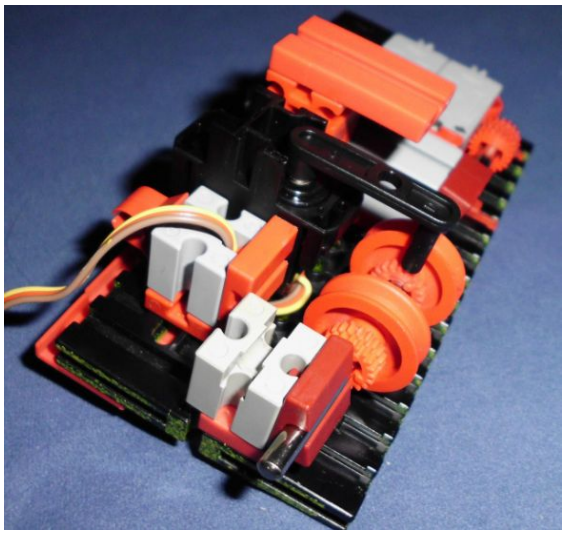


Abb. 19: Tasterschaltung mit Servo

Aus Platzgründen konnte ich hier nicht alle Details ausführen, ich kann bei Interesse aber gerne eine ausführliche Bauanleitung zusenden. Schreibt dazu eine E-Mail an Werner.Hasselberg@web.de. Ansonsten bleibt mir nur noch, euch viel Spaß beim Nachbauen zu wünschen.

Abb. 20: Das fertige Modell

Modell

Optisches Entfernungsmessgerät

René Trapp

Technikgeschichte trifft auf fischertechnik – so könnte man dieses kleine Projekt beschreiben. Geometrie und Strahlenoptik ermöglichen eine berührungslose Entfernungsmessung. Die Besucher der Süd²-Convention 2018 in Karlsruhe konnten sich von der einwandfreien Funktion des Gerätes überzeugen, wenngleich die Handhabung etwas umständlich war. Die hier vorgestellte Version kommt mit weniger optischen Bauteilen aus und ist einfacher zu bedienen.

Zugegeben, Entfernungen kann man auch mit anderen Messgeräten ermitteln. Die meisten gängigen Methoden erfordern jedoch das Überwinden der Strecke zwischen den Messpunkten, um das Messmittel anzulegen. Man denke nur mal an den Holzgliedermaßstab („Zollstock"), das Bandmaß und ähnliche Dinge. Dabei gibt es aber manchmal unüberwindbare Hindernisse. Die begrenzte Länge eines Bandmaßes ist wohl der offensichtlichste Grund. Meist handelt es sich bei diesen Hindernissen jedoch um unüberbrückbare Zwischenräume oder einen Einsatz unter Lebensgefahr. Wer bei Wikipedia unter dem Stichwort „Triangulation" nachstöbert, bemerkt sicher die historische Darstellung eines Schlachtfeldes aus 1607 [1, 2]. Das Verfahren selbst wurde bereits in der Antike angewendet [3].

Geometrie-Dreieck

Die Entfernungsmessung per Triangulation geht auf eine Winkelbestimmung im rechtwinkligen Dreieck zurück. Dabei muss eine Seitenlänge bekannt sein. Zweckmäßigerweise wird diese bekannte Seitenlänge auf der Seite des Beobachters liegen.

Ein entferntes Zielobjekt (hier das fischertechnik-Männchen) wird vom Beobachter zweimal angepeilt. Die rote Sichtlinie stellt eine Seite des Dreiecks dar und ist gleichzeitig die *unbekannte Distanz D.* Rechtwinklig zu dieser Sichtlinie liegt in einer genau bestimmten Entfernung, der *Basisbreite B,* ein zweiter Punkt,von dem aus das Zielobjekt unter einem bestimmten *Winkel* β zu sehen ist. Diese Sichtlinie ist blau dargestellt (Abb. 1).

Die Präzision des Ergebnisses steht und fällt hierbei mit der zweimalig gleichen exakten Ausrichtung des Nullpunktes bei der Winkelmessung.

Nimmt man einen drehbar gelagerten Spiegel, so gelingt es, die blau dargestellte Sichtlinie zu einem einzigen Beobachter an einer einzigen Position umzulenken. Der Beobachter bekommt dieses Bild über einen zweiten, fest auf 45° eingestellten Spiegel ebenfalls zu sehen. Beide Bilder, das der direkten Sicht und das über die beiden Spiegel, werden überlagert, und wenn sie deckungsgleich sind, kann auf der Skala des Drehspiegels der Sichtwinkel abgelesen werden. Sind die Bilder deckungsgleich überlagert, so nennt man das „koinzidente Bilder". Es handelt sich bei dieser Bauart also um einen Koinzidenzentfernungsmesser (KEM).

Eine sehr gute und lesenswerte Dokumentation eines „Telemeters" ist im Netz zu finden [4], wenngleich die Profi-Geräte etwas anders funktionieren.

Bei der Winkelbestimmung ist noch zu beachten, dass bei einem Spiegel „Einfallswinkel = Ausfallswinkel“ gilt. Wird der Spiegel um 5° gedreht, so überstreicht die Sichtlinie einen Bereich von 10°. Der mechanische Drehwinkel des Spiegels ist also nur halb so groß wie der optische Sichtwinkel.

Hat man nun den optischen Sichtwinkel ermittelt und kennt die Basislänge *B*, so lässt sich gemäß Tangenssatz („Tangens = Gegenkathete zu Ankathete“) die gesuchte Seitenlänge *D* errechnen. Diese Umrechnung kann man auch bereits vornehmen, wenn die Skala gezeichnet wird. So ergibt sich die Ablesung direkt in einer gängigen Maßeinheit anstelle in Winkelgraden.

Gemäß dem Tangenssatz ist die Skalenteilung erheblich nichtlinear. Je spitzer der Winkel α wird, desto extremer wird das Verhältnis aus Distanz und Basisbreite. Der KEM kann daher bis maximal 100 m brauchbare Ablesungen liefern.

Von der Theorie zur Praxis

Was auf dem Papier so einfach erscheint, hat in der fischertechnik-Praxis durchaus einige Tücken. Es beginnt mit der Idee, zwei Bilder an einer gemeinsamen Position zu überlagern. Halbdurchlässige Spiegel gibt es zwar zu kaufen, aber sie sind leider nicht Bestandteil des aktuellen Profi-Optics Kastens (520399).

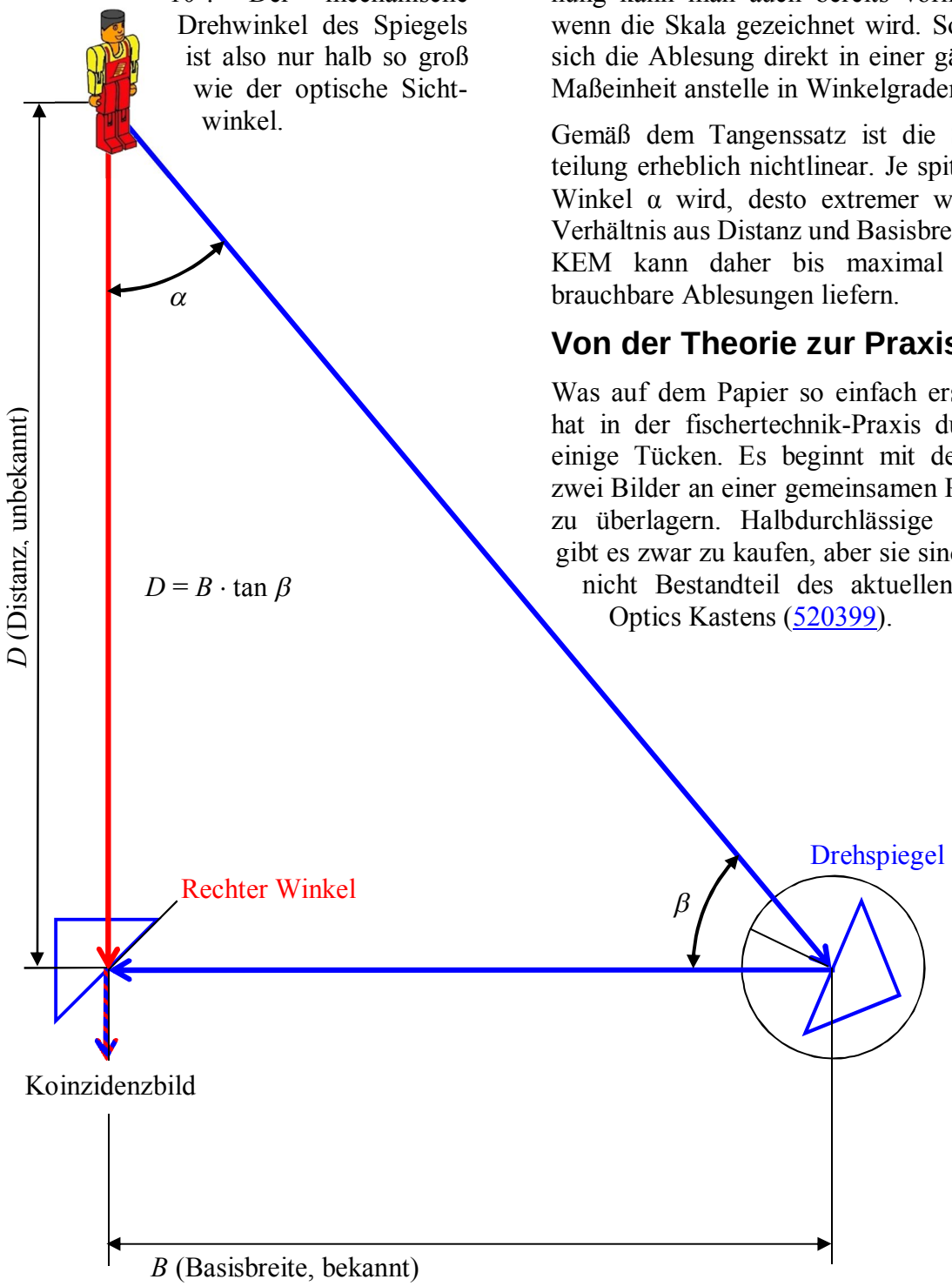


Abb. 1: Dreieck aus Sichtlinien

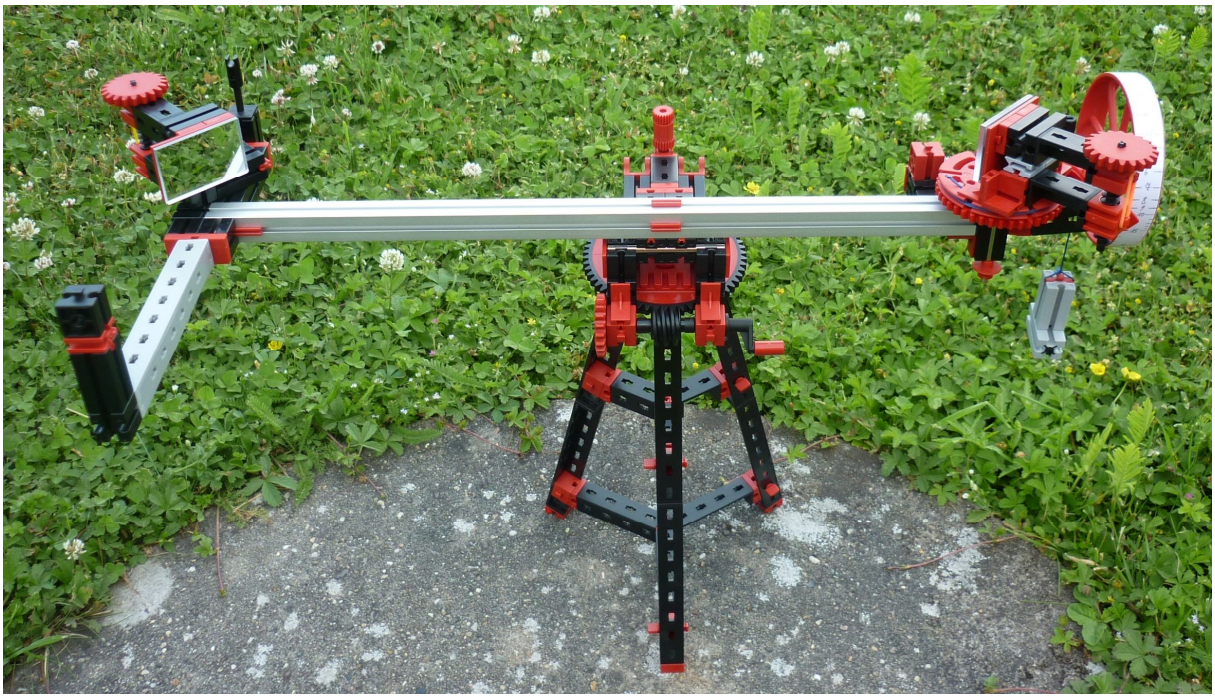


Abb. 2: Koinzidenzentfernungsmesser (KEM)

Ein nächstes Problem stellt die präzise Einstellung sehr kleiner Drehwinkel dar. Der Drehspiegel wird zu diesem Zweck über ein Schneckengetriebe 40:1 bewegt.

Dabei entspricht die Drehung von 1° am Skalenrad einer Spiegeldrehung von 1,5′ (eins-komma-fünf Bogenminuten). Der optische Winkel ändert sich dadurch um 3′. Das setzt eine präzise Fertigung der Schnecke und des Z40 voraus – und ein absolut spielfreies Getriebe. Das Getriebespiel lässt sich durch eine entsprechende sanfte Vorspannung in den Griff bekommen. Für die präzise Fertigung der Schnecke liegt das Vertrauen zur Gänze auf fischertechnik.

Schließlich ist die Anordnung aus Visier, Spiegeln und Drehmechanismus auf einer stabilen Grundlage zu befestigen. Je länger die Basisbreite *B* gewählt ist, umso größer wird die größte messbare Distanz *D*. Mit wachsender Länge wächst die Anforderung an die Stabilität der Anordnung. Auch hier findet sich eine Lösung: Ein Aluminiumprofil.

Vergleichsweise entspannt ist die Montage des feststehenden Spiegels unter exakt 45°. In diesem Fall sind es eher Beschaffungsprobleme, die dem fischertechniker das Leben schwermachen können.

Damit bleibt noch das Problem mit der gewünschten Überlagerung der Bilder: Zur Süd²-Convention in Karlsruhe konnte man einen Blick durch den Sextanten von Thomas Püttmann werfen [5]. Auch beim Sextanten wird ein eingespiegeltes Bild mit einem direkt sichtbaren Bild auf ein Auge des Betrachters gegeben. Der Sextant verwendet dafür einen der randlosen Spiegel aus dem Profi-Optics und ein Visier. Der Sextant ist übrigens auch ein Winkelmessgerät, allerdings für einen anderen Verwendungszweck optimiert.

Nach dem erhellenden Blick durch den Sextanten und der dort gezeigten überzeugenden Lösung des Koinzidenzproblems entstand innerhalb weniger Tage aus dem vom Autor in Karlsruhe gezeigten ersten Prototypen [6] die hier vorgestellte verbesserte Version des Koinzidenzentfernungsmessers (Abb. 2).

Die fischertechnik-Umsetzung

Sozusagen das Rückgrat des Entfernungsmessers bildet ein 390-mm-Aluminiumprofil aus dem Teilebestand.

Nun sind die fischertechnik-Aluminiumprofile, besonders die aus neuerer Produktion, üble Zapfenkiller. Das fachgerechte und saubere Entgraten der Nutenden ist bei der Produktion nicht immer gemacht worden und nachträglich kaum möglich. Eingeschobene Bauteilzapfen werden von den rasiermesserscharfen Aluminiumkanten abgeschält oder wenigstens zerdrückt. Derartig verwendete Bausteine gleiten anschließend sehr leichtgängig in Standardnuten anderer Bausteine. Generell, insbesondere jedoch für seltene Bausteine, ist eine derartige Schädigung inakzeptabel, daher werden bei diesem Modell ausschließlich billige und gut erhältliche Federnocken ins Aluminiumprofil eingeschoben:

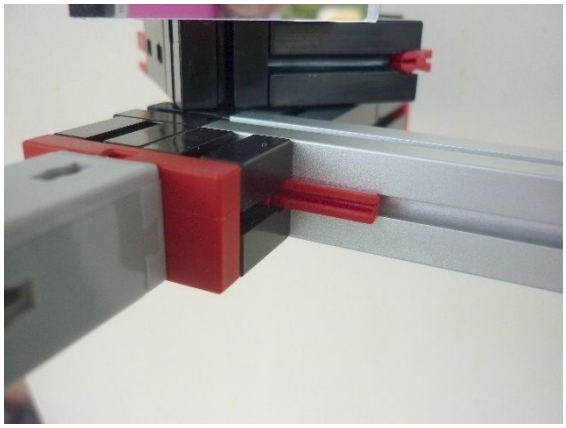


Abb. 3: Federnocken gegen Zapfenschäden

Obendrein werden dabei normalerweise die Federnocken nicht beschädigt; wer mag, kann sicherheitshalber diese Federnocken beim Zerlegen des Modells getrennt sortieren und zukünftig wieder für Aluminiumprofile verwenden.

Der Drehspiegel wird, wie schon weiter oben ausgeführt, über ein Schneckengetriebe 40:1 angetrieben. Dieses ist konventionell mit Klemmschnecke und Zahnrad 40 (Z40) aufgebaut. Das Z40 wird mit Muffenstopfen an der Drehscheibe befestigt. Auf dem Umfang der Drehscheibe ist ein Seil befestigt. Dieses führt über eine Umlenkrolle zum Spanngewicht, zwei Bausteinen 30. Die so erzeugte Umfangskraft drückt das Z40 immer gleichmäßig und in einer Richtung gegen die Flanke der Klemmschnecke. Die Feder auf der Schneckenwelle drückt die Klemmschnecke präzise gegen das Wellenlager. Das Getriebespiel wird auf diese Art eliminiert. Abb. 4 zeigt die Mechanik.

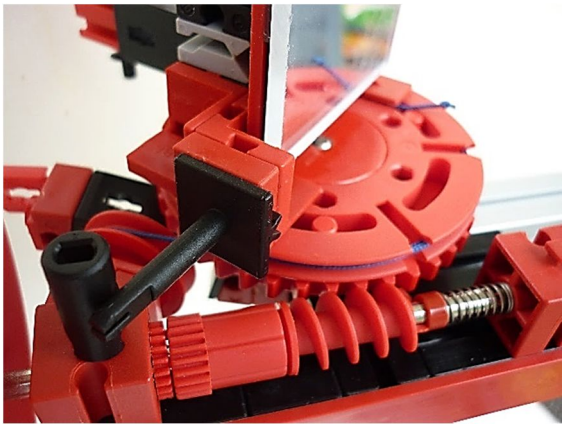


Abb. 4: Getriebe mit Vorspannung

Ein Anschlag begrenzt die Drehung etwas oberhalb der „Unendlich“-Stellung. Dies erleichtert den Zusammenbau, die präzise Einstellung und auch die Benutzung des Gerätes.

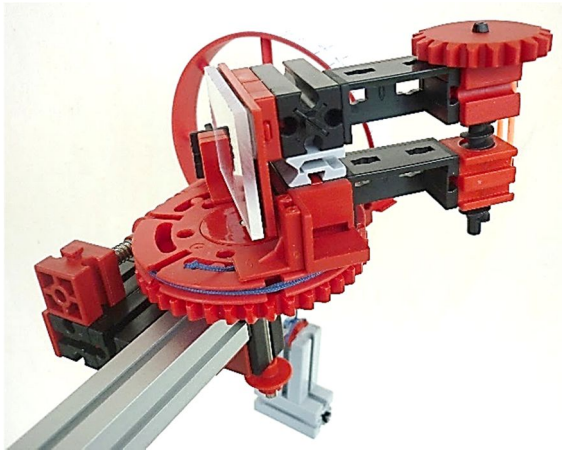


Abb. 5: Drehspiegel

Auf der Drehscheibe sitzt ein Spiegel möglichst genau über der Drehachse. Dieser

Spiegel ist per Rastschnecke und Gegenmutter in der Höhe minimal schwenkbar und in Statikscharnieren gelagert. Der Schwenk ist nur in engen Grenzen sinnvoll, deswegen wird bewusst auf die sonst üblichen weiteren Gelenke verzichtet. Ein Gummiring verhindert hier das Getriebespiel.

Auf der Welle der Klemmschnecke sitzt ein Speichenrad als Träger der Skala (Abb. 6). Je größer der Skalendurchmesser gewählt ist, umso so mehr Skalenlänge überstreicht ein bestimmter Drehwinkel dieses Rades.

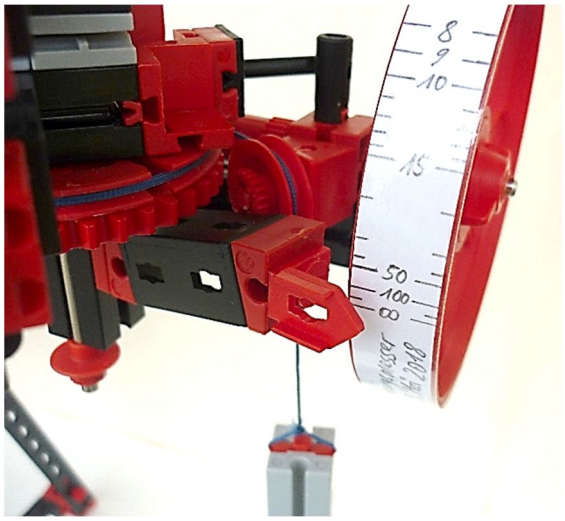


Abb. 6: Skalenrad

Das Spanngewicht ist mittels Schlaufe und Federnocken unverlierbar am Seil befestigt. Ein Kranhaken erwies sich beim Transport der Ursprungsversion als suboptimal.

Am anderen Ende des Aluminiumprofils sitzt der fixierte Umlenkspiegel unter 45°. Auch dieser Spiegel ist in der Höhe minimal schwenkbar gelagert. Der Mechanismus ist nahezu baugleich mit der Lagerung des Drehspiegels auf der Drehscheibe.

Besondere Beachtung verdient hier die feste Einstellung des 45°-Winkels, die durch zwei Bausteine 15 mit runden Zapfen realisiert wird. Diese beiden besonderen Bausteine sind über einen Winkel aus Grundbausteinen verbunden und in eine einzige lange Nut eingeschoben. Sie sitzen nicht im Aluminiumprofil.

Abb. 7: Fester Spiegel

Der in Abb. 1 dargestellte blaue Strahlengang ist nun Dank der beiden Spiegel schon vorhanden. Es fehlt allerdings noch die direkte Sichtlinie als Referenz, in Abb. 1 rot gezeichnet.

Für die direkte Sichtlinie ist eines schon einmal klar: Der feste Umlenkspiegel soll nur etwa halb im Bild zu sehen sein. Der Blick fällt, wie beim Sextant, streifend über den Rand hinweg. Aus diesem Grund eignet sich ein randloser Spiegel aus dem Profi-Optics-Kasten vorzüglich. Überhaupt hat der Sextant hier eine Schlüsselrolle gespielt. Die von Thomas Püttmann ausgesuchten Teile für die Visier-Einrichtung haben sich auch für den Einsatz am Koinzidenzentfernungsmesser als optimal erwiesen.

Der Blick des Bedieners fällt, in Abb. 8 von hinten kommend, durch einen Baustein 15 mit Loch und Ansenkung streifend über den Spiegelrand in Richtung Zielobjekt. Ein Rastadapter steht aus Beobachtersicht hinter, in Abb. 8 allerdings vor dem Spiegel und definiert zusammen mit dem Loch im Baustein die Blickachse. Die Blickachse muss senkrecht zur Längsachse des Aluminiumprofils verlaufen. Bei beiden Bauteilen ist auch die Höhe entscheidend für den Effekt.

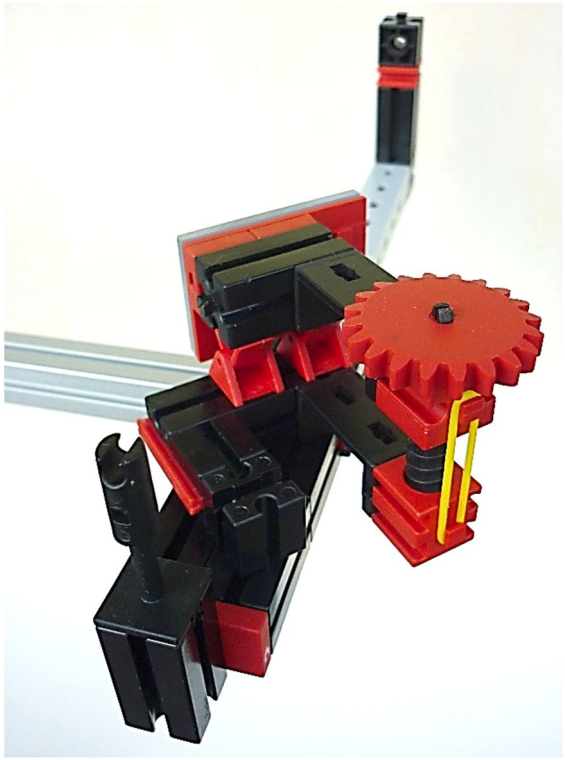


Abb. 8: Visiereinrichtung

Freihändig ist das Gerät durchaus schon gut einsetzbar. Ein Stativ mit Richt-Einrichtung veredelt die Konstruktion. Wer den Profi-Optics sein Eigen nennt, erkennt in Abb. 9 sicherlich die Grundkonstruktion des Dreibeins mit dem horizontalen Drehteller wieder.

Diese Konstruktion ist etwas modifiziert. Zum einen stimmt die in der Bauanleitung angegebene Position der oberen inneren Verstrebung nicht und führt zu Verspannungen, zum anderen war dem Autor die vertikale Schwenkeinrichtung zu groß und aufwändig. Da hier meist horizontale Abstände gemessen werden sollen, reicht ein erheblich kleinerer vertikaler Schwenkwinkel aus (Abb. 10). Außerdem sind die Beine verlängert, was die Torsionssteifigkeit insgesamt etwas beeinträchtigt.

Abb. 9: Dreibeinstativ

Abb. 10: Stativkopf

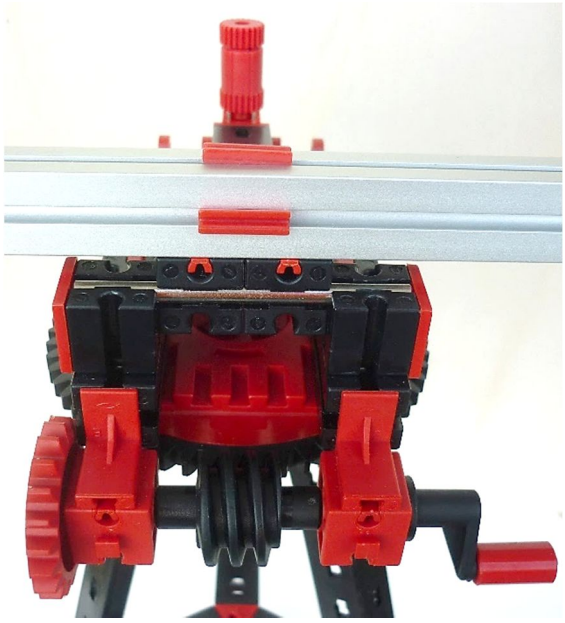


Abb. 11: Schnellmontagevorrichtung

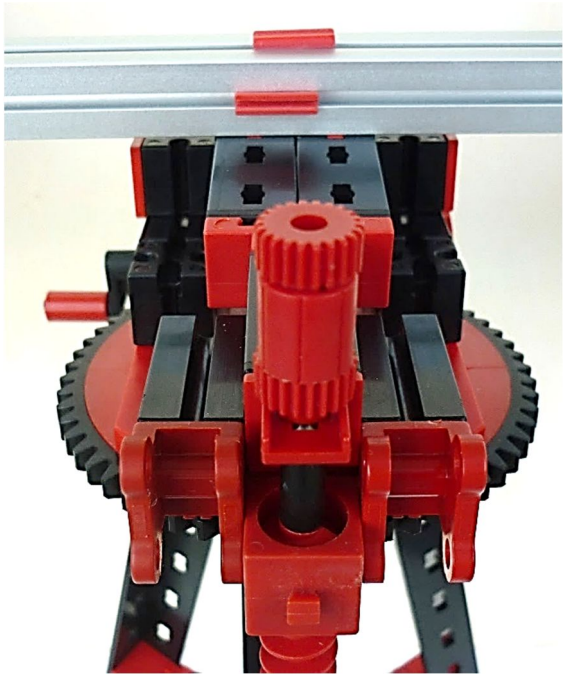


Abb. 12: Schnellmontagevorrichtung

Mit zwei möglichst genau unter dem Schwerpunkt des Entfernungsmessers platzierten Federnocken wird das Aluminiumprofil in den Stativkopf eingeschoben (Abb. 11 und 12).

„Malen nach Zahlen“

Das große Speichenrad wird als Träger der Skala verwendet. Für dieses Rad ist die Skala in Abb. 13 als Vorlage geeignet. Sie ist 15 mm breit und 160 mm lang.

Am einfachsten dürfte es sein, die ft:pedia-Seite auszudrucken. Der Ausdruck sollte allerdings vor dem Zerschneiden noch auf Maßhaltigkeit kontrolliert werden: Die „2,5 m“-Linie und die „∞“-Linie sind 140 mm voneinander entfernt. Stimmt das nicht, so ist der Druckmaßstab entsprechend anzupassen.

Die Skala wird zuerst an den beiden kurzen Seiten ausgeschnitten. Danach wird sie mit Klebefilm überklebt und die langen Seiten werden ausgeschnitten. Der Klebefilm ist nun genauso breit wie die Skala, steht aber an den Enden über. Die vorbereitete Skala kann so passgenau und straff auf dem Umfang des Speichenrades befestigt werden.

Wer die Skala selbst anfertigen möchte, kann dies natürlich tun. Der „∞“-Punkt wird als Bezugspunkt bei 0 mm genommen und die anderen Strecken davon abgetragen. Die Berechnung dieses Abstandes s erfolgt gemäß folgender Gleichung, das Ergebnis wird kaufmännisch auf 0,1 mm gerundet.

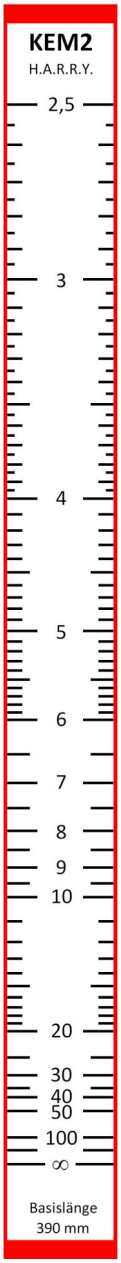


Abb. 13: Skala für 390 mm Basisbreite

$$s = \left(90° - \arctan\frac{D}{B}\right) \cdot \frac{n \cdot \pi \cdot d_{Skala}}{720°}$$

Mit:

- D = Distanz in Meter
- B = Basisbreite (hier 0,39 m)
- n = Übersetzung des Schneckengetriebes (hier 40)
- d_{Skala} = Durchmesser des Skalenrades (hier 90 mm)

Das Ergebnis s erhält man in Millimetern. Wer lieber in Zentimetern denkt, nimmt 9 cm für den Durchmesser der Skala.

Die erste Inbetriebnahme

Die präzise Einstellung der beiden Spiegel und des Visiers ist für die Funktion des Entfernungsmessers elementar.

Zuerst wird der feststehende Spiegel so an der Höhenverstellung eingestellt, dass vom Drehspiegel etwa 90% der Fläche zu sehen sind, wenn man durch den Baustein 15 des Visiers geradeaus blickt. Das Rast-Z20 dient als griffiger Drehknopf.

Nun wird die Visiereinrichtung mit dem feststehenden Spiegel eingerichtet. Dabei ist ein Tischlerwinkel oder notfalls ein Geometrie-Dreieck hilfreich. Durch den Baustein 15 blickt man auf den feststehenden Spiegel. Das Aluminiumprofil ist darin zu sehen und muss genau unter 90° zum realen Profil verlaufen. Mittels Winkel wird diese Einstellung kontrolliert. Minimale Korrekturen erfolgen durch seitliches Verschieben der in Abb. 8 vorderen beiden Bausteine 30 unter dem Spiegelträger.

Der Rastadapter steht in der Verlängerung des gespiegelten Aluminiumträgers. Seine genaue Höhe wird nach Geschmack am tragenden Baustein 30 eingestellt.

Abb. 14 dient hier zur Orientierung, wie das aussehen soll. Dieses Bild ist allerdings mit dem Kameraobjektiv an ungefähr der Position des „Gucklochs“ gemacht und stimmt deswegen nicht ganz exakt.

Richtet man das Gerät nun auf ein Objekt aus und dreht am Skalenrad, so sollte das Objekt irgendwann auch im Drehspiegel erscheinen. Genau muss das jetzt nicht passen. Die Höheneinstellung des Drehspiegels wird nun so verstellt, dass das Objektbild etwas unterhalb des Direktbildes liegt; Abb. 14 hilft auch hier nochmal weiter.

Das Gerät wird nun per Visier auf ein möglichst viele Kilometer entferntes, gut sichtbares und möglichst großes Objekt ausgerichtet. Der Drehspiegel wird per Skalenrad verstellt, bis beide Bilder exakt übereinanderliegen (Abb. 15). Jetzt beginnt die Fummelei: Das Skalenrad wird mitsamt der (nicht allzu fest angezogenen) Nabe gegenüber der Schneckenwelle soweit verdreht, bis die Unendlich-Markierung (die liegende Acht) unter dem Zeiger steht. Die exakte Ausrichtung auf das Objekt und auch die Einstellung des Drehspiegels wird dabei verloren gehen und man wiederholt diese letzten Schritte so lange, bis die Bildkoinzidenz und die Skala exakt aufeinander abgestimmt sind. Jetzt erst wird die Nabe vom Skalenrad fest zugedreht. Dabei rutscht das Skalenrad gerne nochmal ein wenig in der Nabe. Das lässt sich durch Festhalten der Nabe und Zurückdrehen des Skalenrades mit etwas Kraft wieder korrigieren.

Die Bedienung

Egal ob frei aus der Hand oder auf dem Stativ, die Visiereinrichtung gestattet es, das Zielobjekt direkt und genau anzupeilen. Dabei schneidet der Rand des 45°-Spiegels den sichtbaren Bereich in zwei Teile und das Zielobjekt ist zunächst nur über dem Spiegel sichtbar (Abb. 14). Das vordere Visierteil (die Spitze des Rastadapters) ist hierbei unscharf, aber über dem Spiegel noch zu erkennen.

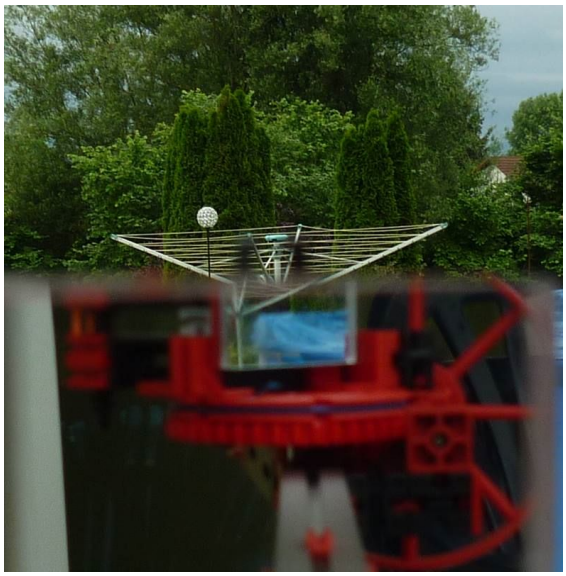


Abb. 14: Anvisieren des Zielobjektes

Im zweiten Schritt wird mittels des Skalenrades der Drehspiegel so ausgerichtet, dass das etwas kleinere Bild des Zielobjekts exakt unter dem Originalbild liegt. In Abb. 15 ist noch ein leichter Versatz der Bilder zu erkennen. Der ist meiner einfachen Fotoausrüstung beziehungsweise meiner fehlenden dritten Hand geschuldet.

Abb. 15: Koinzidenz der Bilder

Die nun angepeilte Entfernung wird auf dem Skalenrad abgelesen (Abb. 16). Das sieht nach rund 18 m aus und stimmt trotz des leichten Versatzes (Abb. 15) mit der Anzeige vom Maßband (17,5 m) ganz gut überein. Die handgezeichnete Skala ist etwas unvollkommen; der erste kurze Strich unter der „15“ stellt die 20 m-Marke dar.

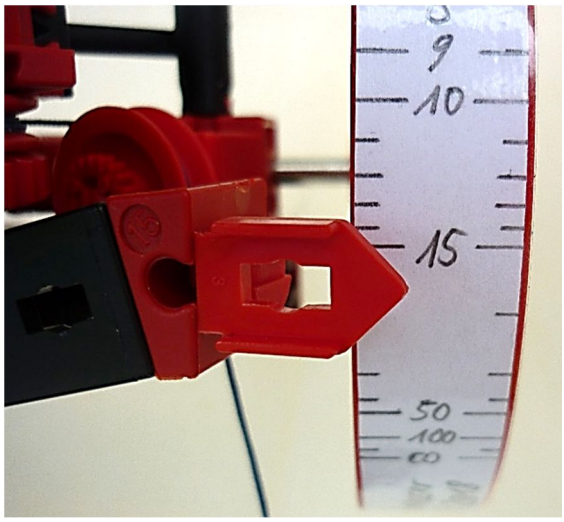


Abb. 16: Entfernung laut Skala

Antiquarisches

Bei diesem Modell kommen nicht nur Geometriekenntnisse aus der Antike, sondern auch ein paar antiquierte fischertechnik-Bauteile zum Einsatz. Wer sie (noch) nicht in der Sammlung hat, muss aber dennoch nicht auf den Nachbau des Gerätes verzichten:

- 2 x Statikscharnier (36329): Die grauen Filmscharniere gibt es im seriösen Gebrauchtteilehandel [7, 8]. Ob mit oder ohne Steg ist für diese Anwendung unerheblich.
- 2 x Baustein 15 mit rundem Zapfen (31059: grau oder 103448: schwarz): Auch die gibt es im Netz [7, 8], allerdings sind sie etwas seltener anzutreffen. Als Notbehelf ginge eventuell auch eine Schneckenmutter (37925), die über zwei runde Zapfen verfügt. Der Halter für den feststehenden Spiegel muss dann allerdings anders aufgebaut werden.
- 2 x Gelenkstein (38459): Derzeit nur bei fischerfriendsman [7] gelistet, aber immerhin nicht selten. Als etwas wackligerer Ersatz kann auch je ein Gelenk-

würfel (31423 oder 31426 + 31436) herhalten. Dazu kommen noch zwei Federnocken.

- 1 x Aluminiumprofil (31230): Das ist im Handel momentan seltener als rar [7, 8]. Alternativ sollte sich auch ein sogenanntes Open-Beam-Aluminiumprofil verwenden lassen.

Die Ersatzlösungen sind bislang nur Ideen. Der Autor hat sie mangels Bedarf nicht ausprobiert.

Quellen

[1] Wikipedia: *Triangulation (Messtechnik)*

[2] Wikipedia: *Optische Entfernungsmessung anno 1607*

[3] Wikipedia: *Entfernungsmessung – Triangulation*

[4] fernglasmuseum.at: *Wild Heerbrugg Koinzidenzentfernungsmesser TM2*

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

[6] Bilderpool der ftc: *KEM1*

[7] fischerfriendsman

[8] santjohanser

Modell

Die Zahnstangen-Uhr

Rüdiger Riedel

Kinetisches Objekt oder Teil einer Uhr? Eine ungewöhnliche Uhr, ein Weihnachtsgeschenk, weckte in mir den Ehrgeiz, etwas Ähnliches mit fischertechnik zu bauen.

Abb. 1: Die Uhr

Abb. 2: Die Uhr von der Seite

Eine Uhr, bei der sich das Zifferblatt dreht und keine Zeiger, dazu mit einem fast quadratischen Zahnrad: Die hat mich gleich begeistert.

Die müsste sich doch in fischertechnik nachbauen lassen, mit noch eckigerem Zahnrad. Und hier ist sie, meine Zahnstangen-Uhr:

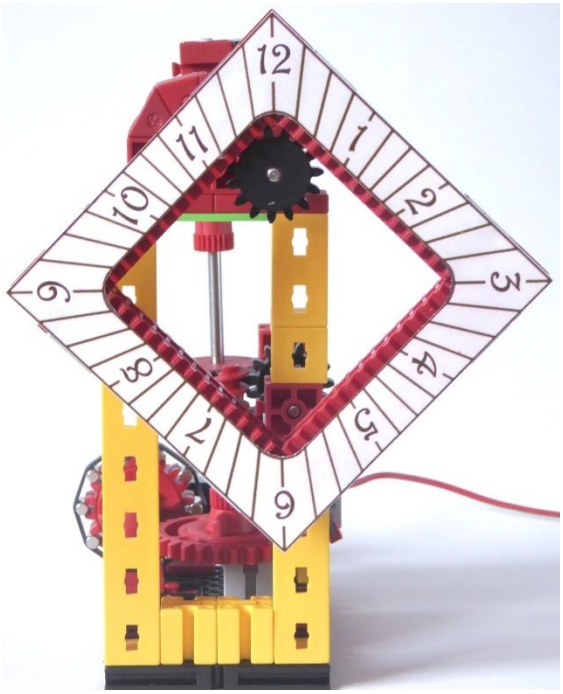


Abb. 3: Die Zahnstangen-Uhr

Im Gegensatz zum Vorbild habe ich das Zifferblatt entsprechend der üblichen Anordnung gestaltet, es dreht sich also links herum.

Die eckigen Zahnräder

Verwendung finden die Zahnstangen Z13 (Zahnstange 61 Z13 m1,5 Nr. 31053) und Z7 (Zahnstange 32 Z7 m1,5 Nr. 31054). Und wenn wir schon mal dabei sind, können wir auch andere Zahnstangen-Gebilde ausprobieren.

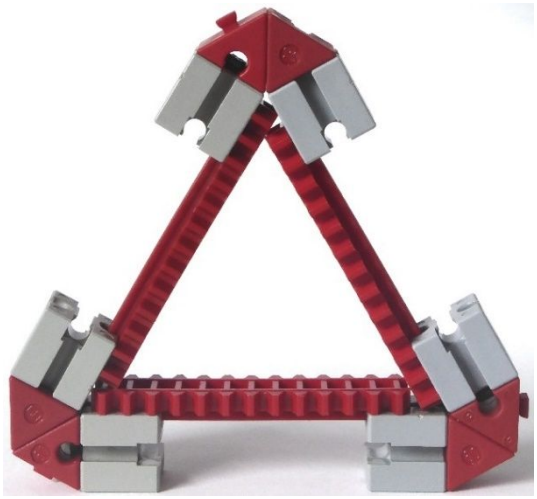


Abb. 4: Das dreieckige Zahnrad mit Z13

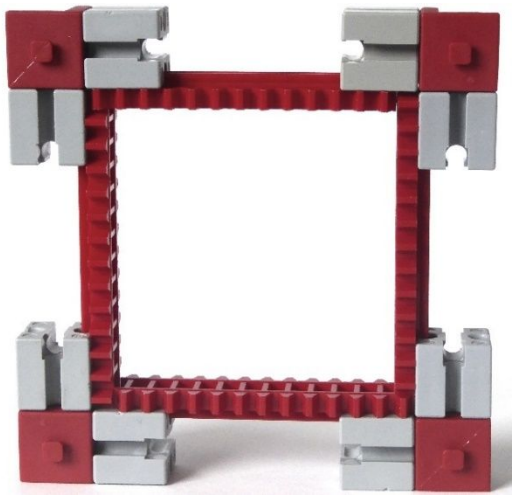


Abb. 5: Das viereckige Zahnrad mit Z13

Die Antriebszahnräder können nicht alle Zähne bzw. Zahnlücken der Zahnstangen in den Ecken des Dreiecks oder des Quadrats erfassen. Dreht man das Zahnstangen-Gebilde mit einer Handkurbel und zählt die Antriebs- und die Abtriebs-Drehungen, kommt man auf die effektiven Zähnezahlen, die hier kleiner als die realen Zähnezahlen sind.

Abb. 6: Handkurbel-Teststand

Ein und dasselbe Zahnrad kann mit verschiedenen Antriebszahnrädern auch unterschiedliche Zähnezahlen zeigen, wir haben also wandelbare

Chamäleon-Zahnräder

Antrieb	Übersetzung	Resultat
Z10	9:4	Z27
Z15	6:5	Z18

Tab. 1: Dreieck aus 3 x Z13

Die *wirksame (effektive) Zähnezahl* des eckigen Zahnrades hängt also ab von der Größe des antreibenden Zahnrades.

Antrieb	Übersetzung	Resultat
Z10	24:5	Z48
Z15	44:15	Z44
Z20	11:5	Z44
Z30	4:3	Z40

Tab. 2: Quadrat aus 4 x Z13

Statt des Z13 kann auch das Z7 eingesetzt werden:

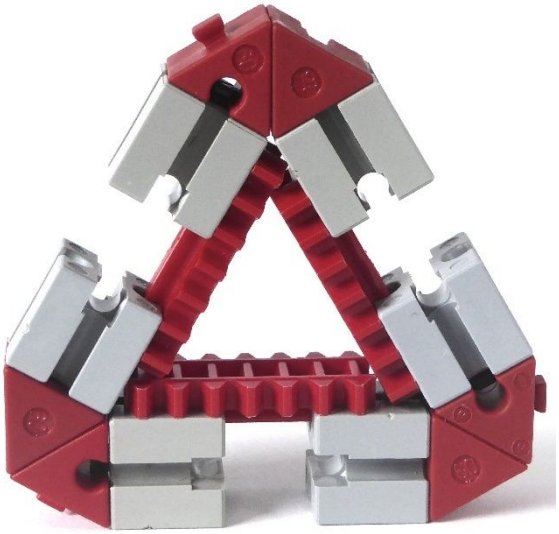


Abb. 7: Das dreieckige Zahnrad mit Z7

Antrieb	Übersetzung	Resultat
Z10	6:5	Z12

Tab. 3: Dreieck aus 3 x Z7

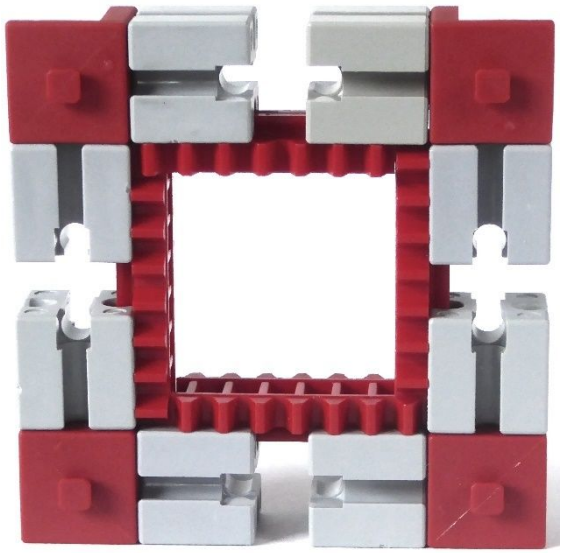


Abb. 8: Das viereckige Zahnrad mit Z7

Antrieb	Übersetzung	Resultat
Z10	12:5	Z24
Z15	4:3	Z20

Tab. 4: Viereck aus 4 x Z7

Einsatz der Federnocken

Durch Einsetzen je eines Federnockens in den Ecken verändert sich die effektive Zähnezahl.

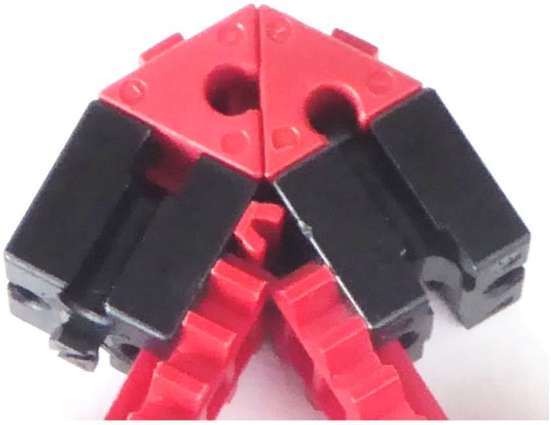


Abb. 9: Federnocken (31982)

Antrieb	Übersetzung	Resultat
Z10	33:10	Z33
Z15	klemmt	

Tab. 5: Dreieck aus 3 x Z13 mit Federnocken

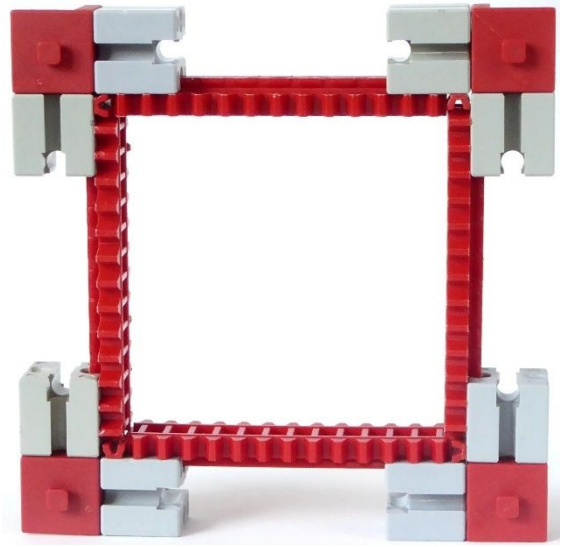


Abb. 10: Z13-Quadrat mit Federnocken

Dieses Quadrat wird für die Uhr nach Abb. 3 verwendet.

Antrieb	Übersetzung	Resultat
Z10	24:5	Z48
Z15	16:5	Z48
Z20	12:5	Z48
Z30	22:15	Z44

Tab. 6: Quadrat aus 4 x Z13 mit Federnocken

Antrieb	Übersetzung	Resultat
Z10	3:2	Z15

Tab. 7: Dreieck aus 3 x Z7 mit Federnocken

Antrieb	Übersetzung	Resultat
Z10	12:5	Z24
Z15	8:5	Z24

Tab. 8: Quadrat aus 4 x Z7 mit Federnocken

Die Vielecke

Beim Aufbau eines regelmäßigen Sechs-, Acht- oder Zwölfecks zeigt sich, dass die antreibenden Zahnräder jetzt alle Zähne der Zahnstangen erreichen können. Die wirksame (effektive) Zähnezahl ist also gleich der tatsächlichen. Das Gebilde in Abb. 11 entspricht somit einem Z42, unabhängig vom Antrieb mit Z10, Z15, Z20 oder Z30.

Abb. 11: Das Sechseck mit Z7, ein Z42

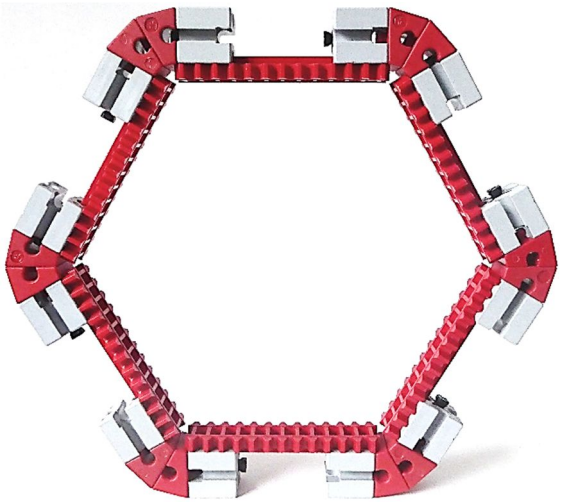


Abb. 12: Das eckige Z78

Kombiniert man die Zahnstangen Z13 und Z7, erhält man ein sechseckiges Z120.

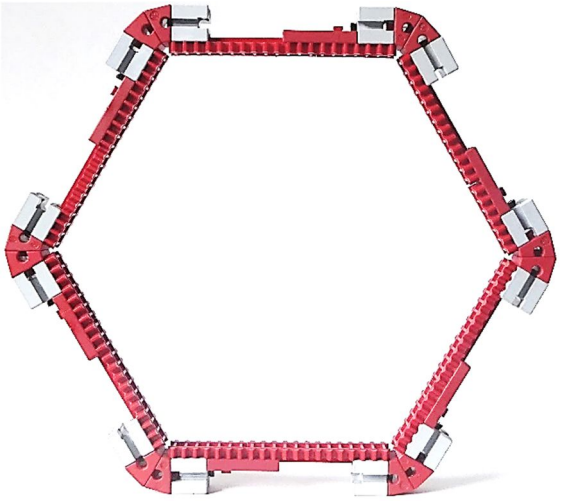


Abb. 13: Das eckige Z120

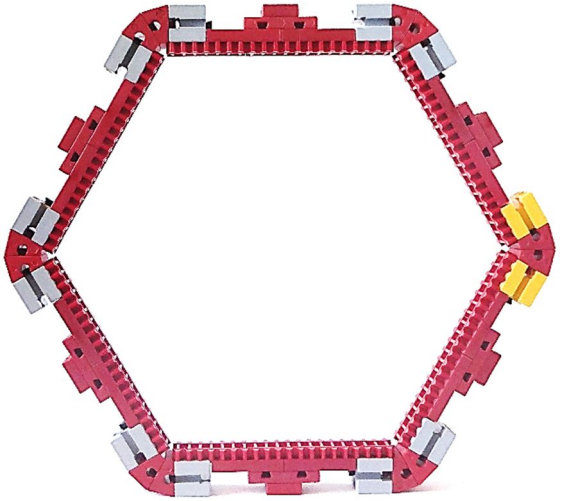


Abb. 14: Z120 mit Verzierungen

Die virtuellen Zähne

Treffen die Zahnstangen nicht aufeinander, sondern bleibt eine Lücke für einen oder zwei Zähne, dann erhöht sich die wirksame Zähnezahl der Zahnstangen.

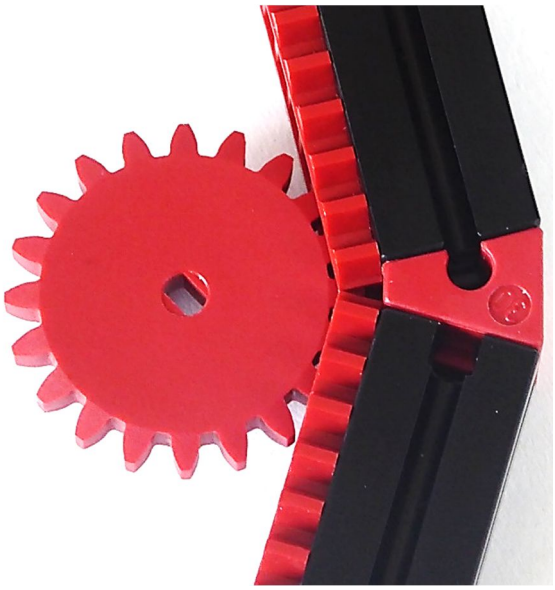


Abb. 15: Ein Knick von 30°

In Abb. 15 ist der Normalfall dargestellt, dass die Zahnstangen „auf Stoß“ angebracht sind. In diesen Stoß passt ein Zahn des antreibenden Zahnrades, hier das Z20, und damit sind die Zahnstangen mit ihrer vollen Zähnezahl wirksam.

Abb. 16: Ein Knick von 2 · 15° = 30°

In Abb. 16 ist der Abstand etwas größer; jetzt passen zwei Zähne des Z20 in die Lücke. Die wirksame Zähnezahl der Zahnstangen erhöht sich um 1, ebenso wie auf Abb. 17 und 18. Aus einem Z7 wird ein Z8 und aus einem Z13 ein Z14.

Abb. 17: Ein Knick von 15° + 7,5° = 22,5°

Abb. 18: Ein Knick von 30° - 7,5° = 22,5°

Abb. 19: Ein Knick von 2 · 7,5° + 15° = 30°

Aber selbst mit einer Zahnlücke von drei Zähnen wie in Abb. 19 bis 21 rollt das Z20 sauber über die Zahnstangen hinweg. Die wirksame Zähnezahl der Zahnstangen erhöht sich jeweils um 2.

Abb. 20: Ein Knick von 60° - 2 · 15° = 30°

Aus einem Z7 wird jetzt ein Z9 und aus einem Z13 ein Z15.

Abb. 21: Ein Knick von 4 · 15° = 60°

Mit diesen Erkenntnissen können wir nun einen Zahnstangen-Ring in der Größe eines 6teiligen Statik-Ringes bauen. Es ist ein 16-Eck, gebaut nach Abb. 18 und ergibt ein Z128.

Abb. 22: Ein Z128 mit Planetenrad

Darin bewegt sich leichtgängig und sauber der dreiarmige Steg mit den Planeten-Zahnrädern Z30, siehe Abb. 22.

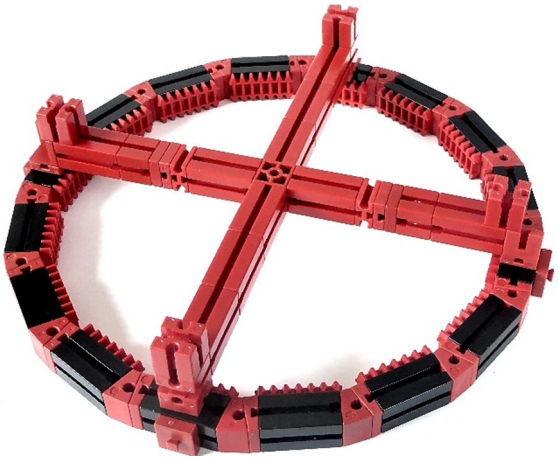


Abb. 23: Ständerseite

Mit der Bausteinkombination

- V-Baustein 15 Eck
- BS30
- BS15
- BS5 2Z
- BS5
- BS7,5
- BS30

je rechts und links vom BS15 mit Bohrung wird der Ring zentriert. Die dazu senkrechte Achse habe ich provisorisch aufgebaut ohne Verbindung zum Ring. Mit den vier BS5 außen am Ring lässt sich dieser am sechsteiligen Statik-Ring befestigen. Auf eine Darstellung verzichte ich hier; benötigt werden vier S-Riegelsteine (32850).

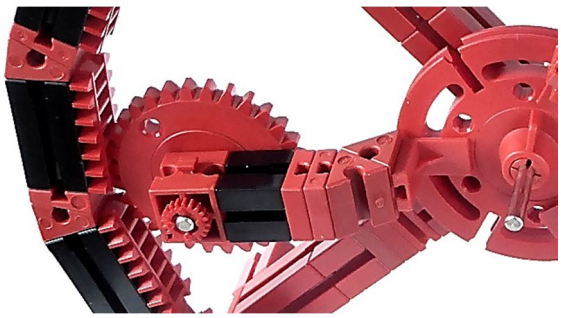


Abb. 24: Detailaufnahme

Die drei Arme für die Planetenzahnräder werden gemäß Abb. 24 aufgebaut. Die beiden WS60 sorgen dabei für richtige Höhe und Abstand der Zahnräder.

Ein Z96 und ein Z65

Aus je 12 BS30, WS30 sowie BS5 lässt sich das Innenzahnrad mit 12 Z7-Zahnstangen zu einem Z96 bauen.

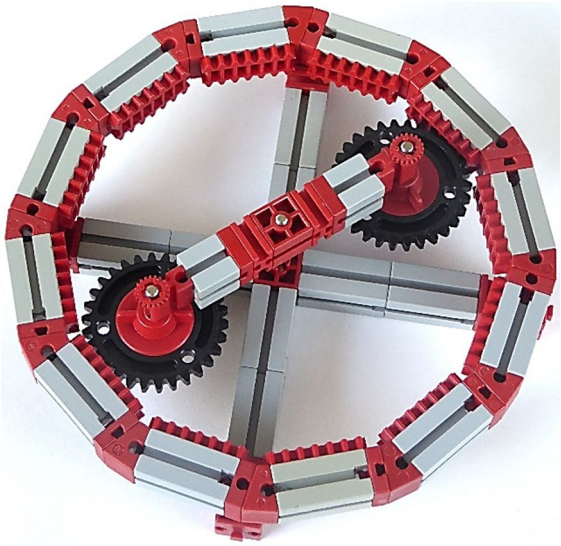


Abb. 25: Das Z96 mit Planeten-Zahnrädern

Um es zu zentrieren, werden vier der BS5 ausgetauscht gegen V-Radachsen (36586), die in die BS5 mit Bohrung des 4armigen Ständers eingreifen.

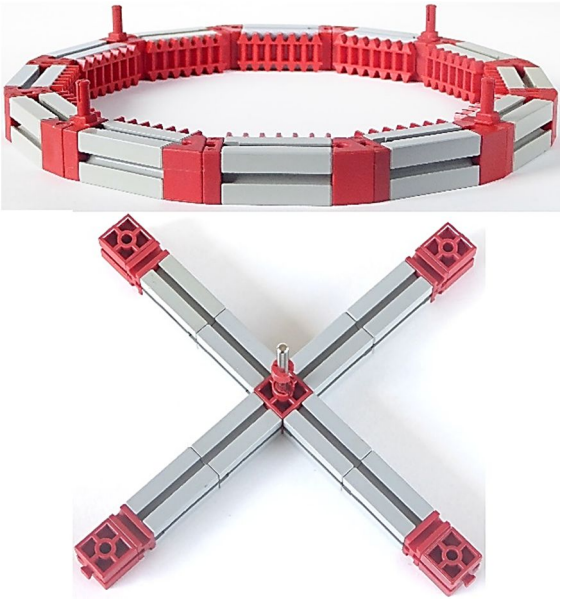


Abb. 26: Z96 und Ständer

Ein Z65 in Form eines Fünfecks lässt sich bauen, auch wenn die Winkelsumme der Ecken nicht 360° beträgt, sondern 375°.

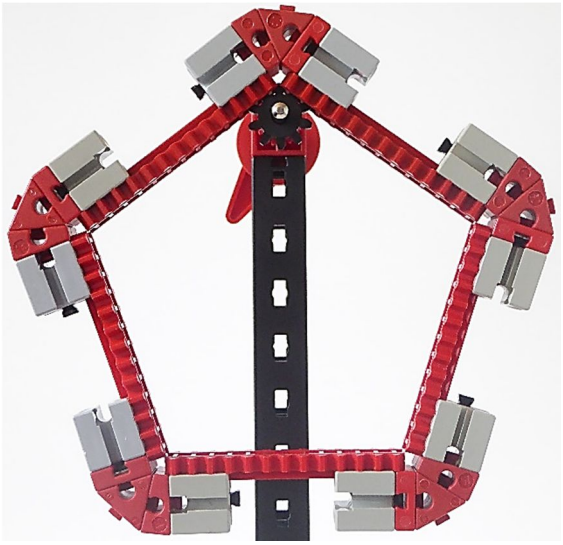


Abb. 27: Ein Fast-Fünfeck als Z65

Hinweis

Von der Zahnstange Z7 gibt es zwei Varianten mit unterschiedlicher Länge der Feder. In den meisten Fällen können wir nur die mit der kürzeren Feder verwenden, rechts in Abb. 28.

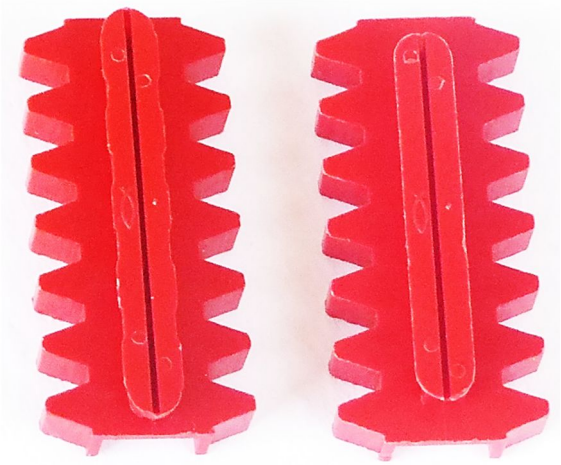


Abb. 28: Die Zahnstange Z7 in zwei Varianten

Vom eckigen Zahnrad zur Uhr

Die erste Uhr ist in Abb. 3 zu sehen. Das rotierende Zifferblatt besteht aus dem Quadrat mit Federnocken nach Abb. 6, ein passendes Zifferblatt zum Aufkleben zeigt Abb. 29.

Ausgehend von einem beliebigen Uhr-Taktgeber – ich bevorzuge Synchronmotoren – und der am Ende erforderlichen Drehzahl benötigen wir eine Übersetzung von

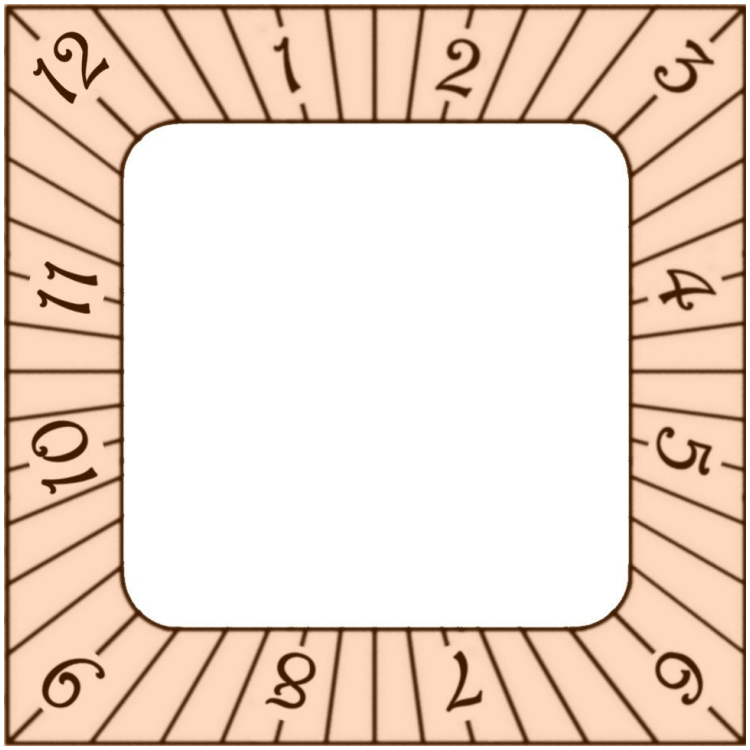


Abb. 29: Das Zifferblatt, Seitenlänge 9,8 cm

$$i = \frac{Z_2}{Z_1} = \frac{n_1}{n_2}$$

Antrieb mit Synchronmotor:

$$i_s = \frac{n_f}{p} \cdot \frac{12\text{h}}{\text{U}} \cdot \frac{Z_{Ende}}{Z_{Ziff}}$$

Bedeutung der Symbole:

- n_f = 3000 U/min)Drehzahl bei Netzfrequenz 50 Hz und Polpaarzahl = 1)
- p = Polpaarzahl [2]
- 12h/U = 12 Stunden für eine Umdrehung des Zifferblattes

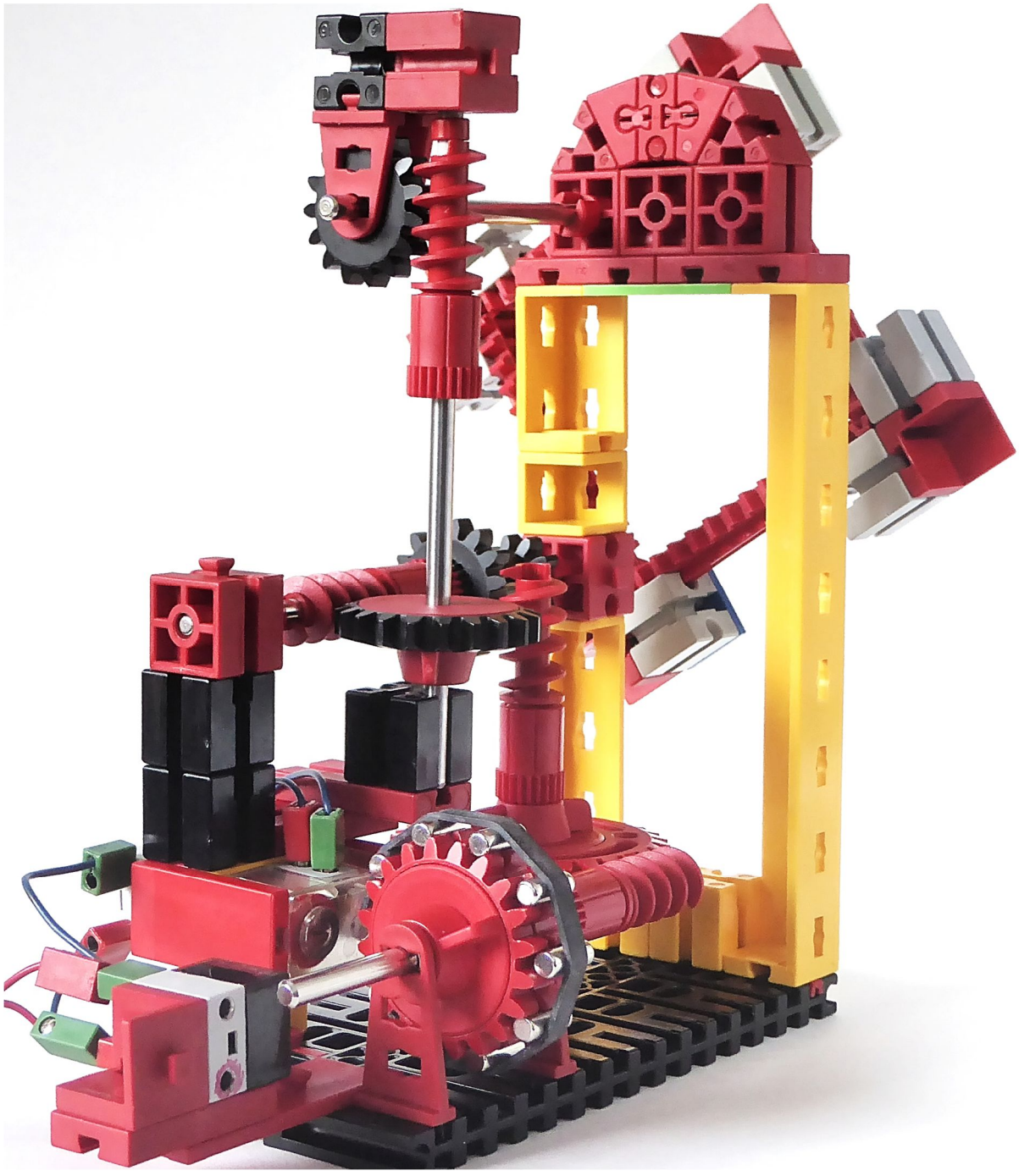


Abb. 30: Die Zahnstangen-Uhr von hinten

- Z_{Ziff} = Zähnezahl des Zifferblattes
- Z_{Ende} = Zähnezahl des Zifferblatt-Antriebs

Wir nehmen das Quadrat von Abb. 10, ein Z48, mit dem treibenden Zahnrad Klemm-Z15, und erhalten dann:

$$i_s = \frac{3000\ \mathrm{U}}{p \cdot \mathrm{min}} \cdot \frac{12 \cdot 60\ \mathrm{min}}{\mathrm{U}} \cdot \frac{15}{48}$$

$$= \frac{3000 \cdot 60 \cdot 15}{p \cdot 4}$$

Verwenden wir einen 10-poligen Synchronmotor (dessen Läufer wird in [6] dargestellt), also $p = 5$, dann vereinfacht sich die Übersetzung zu:

$$i_s = 30 \cdot 20 \cdot 15 \cdot 15$$

Das lässt sich umsetzen durch

- Schnecke auf Z30,
- Schnecke auf Z15,
- Schnecke auf Z20,
- Schnecke auf Z15

und zum Schluss ein Z15 auf das Zahnstangenquadrat.

Abb. 31: Uhr mit tanzendem ft-Mann

Links vor dem Läufer ist eine Stroboskop-LED angebracht, um den Anwurf des Synchronmotors zu erleichtern [1]. Es wird der ft-Magnet 31324 verwendet (immer noch erhältlich bei fi-tec-shop.de und fischertechnik-teile.de), der Typ 32363 geht ebenfalls und mit Anpassungen auch ein Topfmagnet [5].

Das Z30 ist lose im BS15 gelagert; durch die vorgegebene Drehrichtung neigt die Schnecke dazu, sich am Z15 hochzuziehen. Mit der in die Schnecke eingesteckten Achse und dem ft-Mann oben drauf (Abb. 31) wird das verhindert und zusätzlich entsteht ein flotter Tanz.

Die große Zahnstangen-Uhr

Abb. 32: Die große Zahnstangen-Uhr

Wir verwenden eine vergrößerte Version des Z120 von Abb. 13 mit Winkelbausteinen 30 in den 12 Ecken und den verbundenen Zahnstangen Z7 und Z13. Damit erhalten wir ein Z240. Die Verbindung zum Zifferblatt erfolgt mit einem Z20-Zahnrad. Als Antrieb verwende ich einen 30-poligen Synchronmotor ($p = 15$) [3], die nötige Getriebeübersetzung ist also:

$$i_s = \frac{3000\ \mathrm{U}}{p \cdot \mathrm{min}} \cdot \frac{12 \cdot 60\ \mathrm{min}}{\mathrm{U}} \cdot \frac{20}{240}$$
$$= \frac{3000 \cdot 60}{15}$$
$$= 30 \cdot 20 \cdot 20$$

Das lässt sich umsetzen mit:

- Schnecke auf Z30,
- Schnecke auf Z20,
- Schnecke auf Z20

und am Ende ein Z20, auf welches der Zahnstangen-Ring aufgelegt wird. Statt Ziffern bedeuten hier die gelben Bausteine 12 Uhr, die blauen Bauplatten 6 Uhr und die grauen Steine 3 bzw. 9 Uhr.

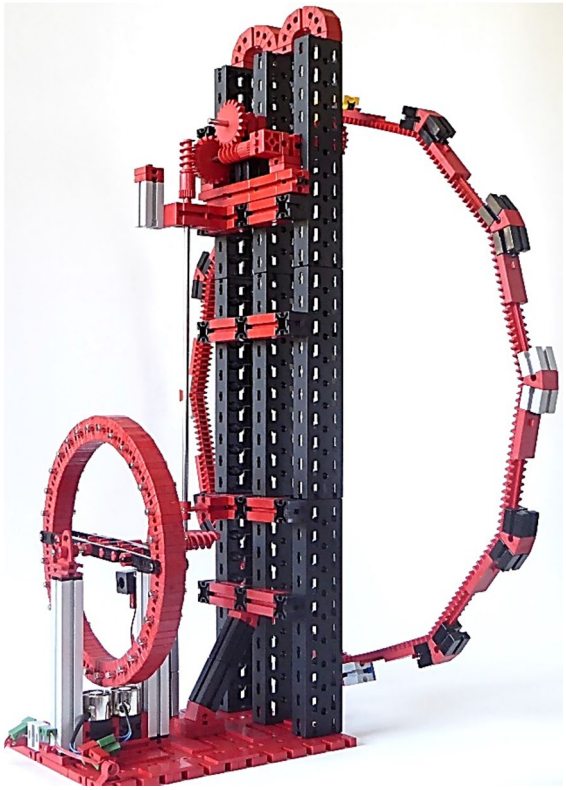


Abb. 33: Die Uhr schräg von hinten

Zum Betrieb des Synchronmotors setze ich hier zwei Topfmagnete ein (erhältlich z. B. bei fischerfriendsman.de mit der Bezeichnung *Chinamagnete*). Diese müssen antiparallel geschaltet werden, so dass sie sich bei Stromfluss gegenseitig anziehen.

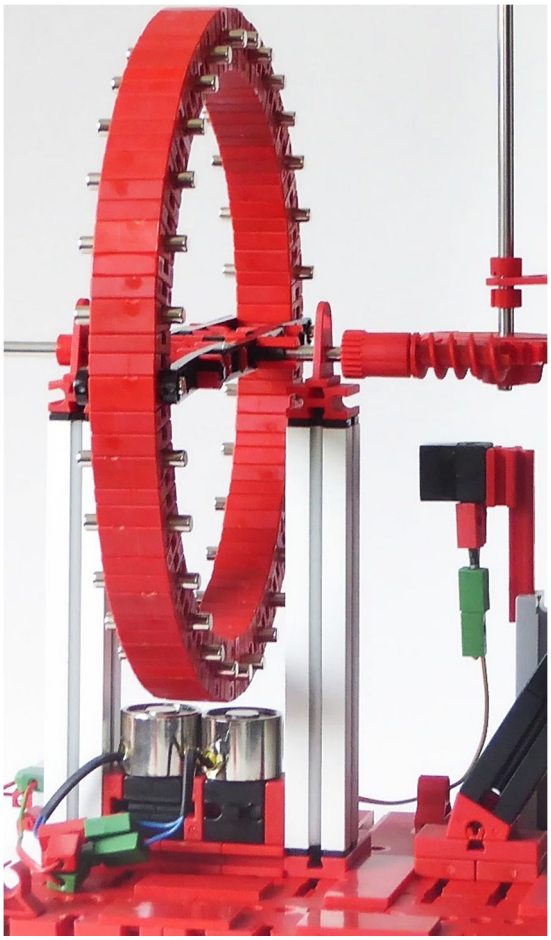


Abb. 34: Der 30-polige Synchronmotor, rechts die Stroboskoplampe (LED)

Die Uhren betreibe ich mit der 6,8V-Wechselspannung des alten Transformators 814 (ft-Nr. 30173, den gibt es immer noch bei fischerfriendsman.de und fi-tec-shop.de).

Die Bewegung des Zahnrades Z240 ist ebenso wie die des kleinen Uhrzeigers normaler Uhren mit den Augen nicht wahrnehmbar. Mit einer 20-fachen Beschleunigung sieht man sie sehr gut, und das lässt sich mit einem einfachen Anbau realisieren. Dazu wird auf den grauen BS30 und die beiden BS5 (oben links in Abb. 33) ein BS15 mit Bohrung aufgesetzt. Die eingeschobene Achse trägt ein Z20, das von der Schnecke des Haupantriebs angetrieben wird. Das andere Ende der Achse sitzt auf einem einfachen Gestell und treibt mit einem weiteren Z20 das Zifferblatt gut sichtbar an.

Abb. 35: Die schnelle Drehung

Weitere Möglichkeiten

Bisher haben wir nur Innenzahnräder mit den Zahnstangen konstruiert. Nun will ich Außenzahnräder vorstellen, die sich auch zentrieren lassen.

Es ist dabei zu berücksichtigen, dass die Zahnstangen Z7 eine Länge von 32 mm haben. Deshalb stecken sie beim Z168 jeweils in einem Baustein 5 15·30 (35049) und diese in den BS7,5 des Ringes. Der Ring besteht aus acht Baugruppen, jeweils zusammengesetzt aus:

- BS7,5
- WS15
- BS5
- BS7,5
- BS5
- BS7,5
- WS15
- BS15
- BS7,5
- WS15
- BS15 sowie einigen Federnocken.

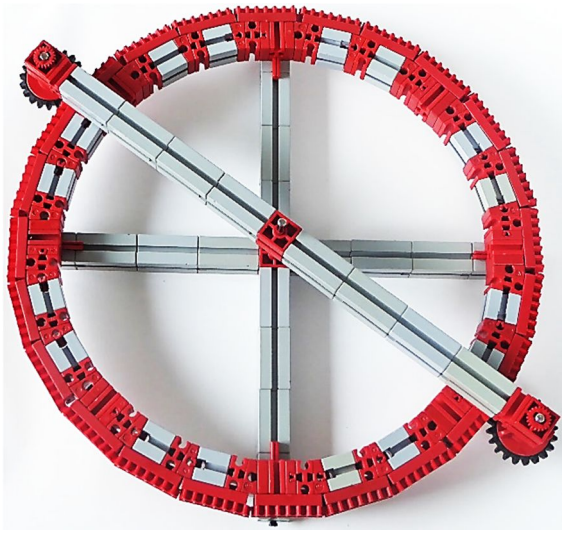


Abb. 36: Das Z168 mit zwei Z20

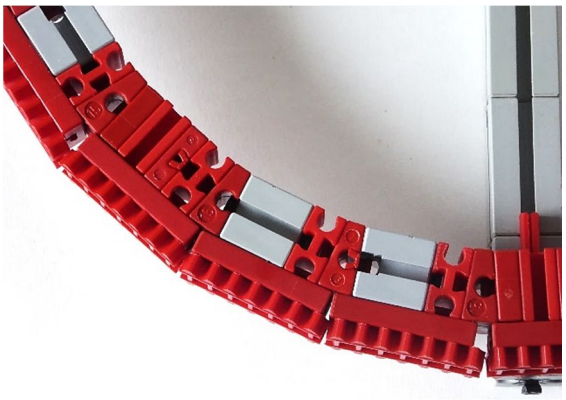


Abb. 37: Detailansicht des Z168

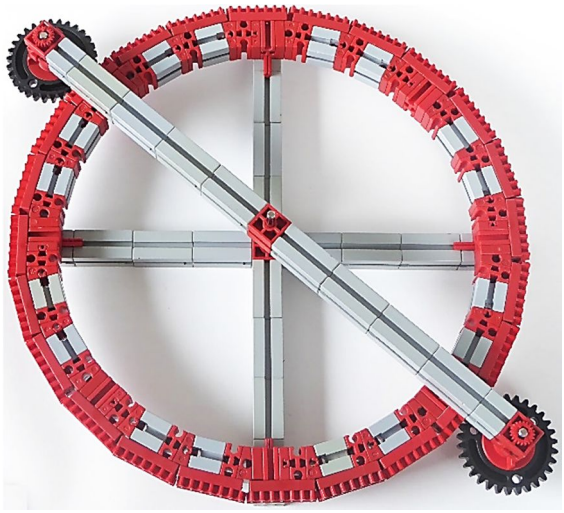


Abb. 38: Das Z168 mit zwei Z30

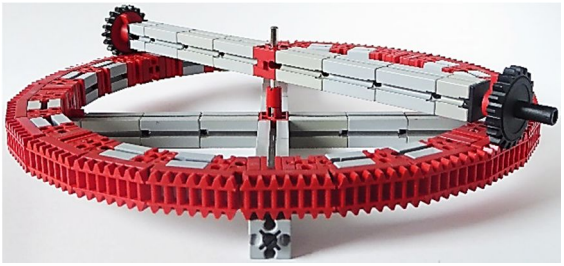


Abb. 39: Die Z20 auf dem Zahnstangen-Rand

In der Vergrößerung (Abb. 37) sieht man, dass die Positionen der Z7 in den Bausteinen 5 15·30 variiert werden müssen, damit sie überall aneinanderstoßen.

Und auch mit dem Zahnstangen-Rand geht etwas (Abb. 39 und 40)! Die beiden „Freilauf-Kettenzahnräder Z20“ (31779) in Abb. 39 sind in Rastaufnahmeachsen 22,5 (130593) gelagert. Mit einem Z40 in der Mitte, bei dem wir dessen Kronrad Z32 nutzen, und einigen Änderungen wird daraus ein Planetengetriebe (Abb. 40 und 41).

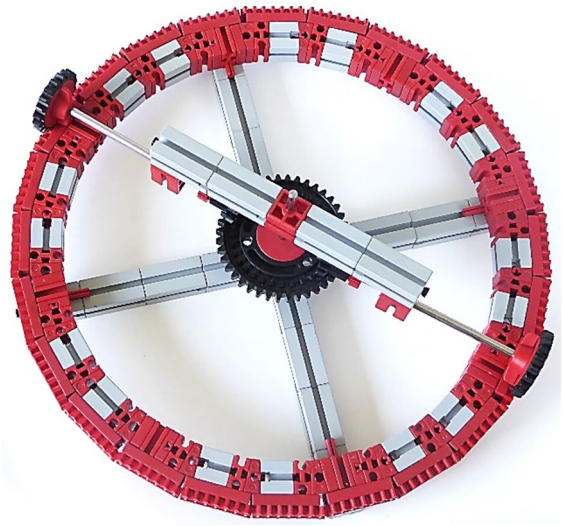


Abb. 40: Ein Planetengetriebe

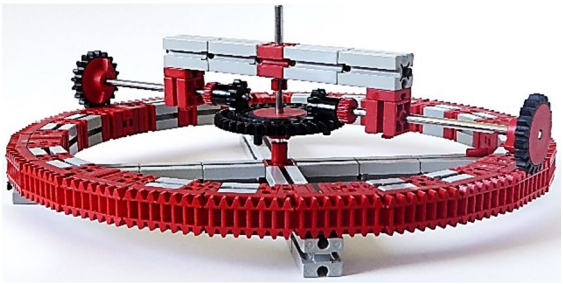


Abb. 41: Das Zahnstangen-Planetengetriebe

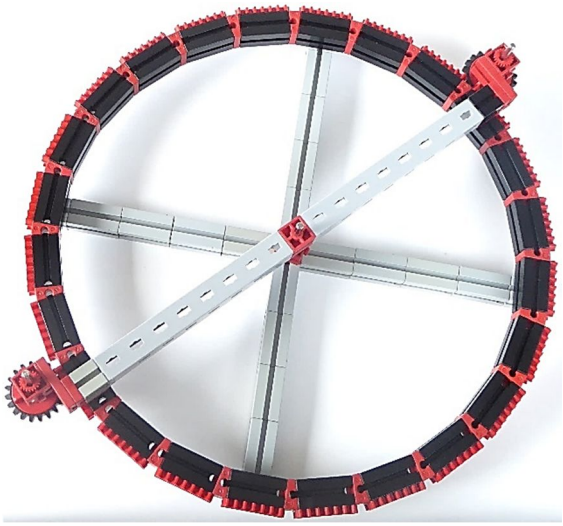


Abb. 42: Ein Z192

Beim Z192 von Abb. 42 sind die beiden Z20 um 15 mm aus der Führungsachse verschoben, um einen besseren Rundlauf zu erzielen.

Die Befestigung auf dem Ständer erfolgt mit Federnocken in vier der Ring-BS30.

Abb. 43: Seitenansicht des Z192

Ein Video der Uhren gibt es unter [7].

Ein wehmütiger Abschluss

Seit diesem Winter (beginnend mit der zweiten Januar-Woche) wissen wir, dass die Netzfrequenz nicht mehr stabil ist. Die Synchronuhren gingen monatelang um mehrere Minuten nach, am 1. März um 5 Minuten. Die maximale Abweichung wurde am 14. März mit 6 Minuten und 22 Sekunden erreicht. Erst Anfang April war diese Abweichung aufgeholt.

Angestrebt wurde früher und wird auch heute noch eine Begrenzung der Frequenzabweichungen dahin, dass Synchronuhren höchstens ± 20 Sekunden falsch gehen.

Am 15. und 16. April gingen die Synchronuhren 28 Sekunden vor, seitdem wird die angestrebte Bandbreite wieder eingehalten.

Das war früher anders (nicht alles war besser, aber das schon…): Als Student hatte ich in den siebziger Jahren nur einen Synchronwecker, der mich nie im Stich gelassen hat. Heute schaue ich auf die Funkuhr.

Unsere Synchronuhr-Modelle haben also keinen praktischen Zweck mehr, sie sind wirklich nur noch Modelle.

Quellen

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

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

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

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

[5] Rüdiger Riedel: *Ersatz für die Elektromagnete*. ft:pedia 3/2017, S. 19-22.

[6] Rüdiger Riedel, *10-poliger Synchronläufer* im Bilderpool der fischertechnik-Community.

[7] Rüdiger Riedel, *fischertechnik-Modelle, Zahnstangen Uhren*. Videos der Uhren auf youtube, 2018.

Modell

Ein Plotter für Polarkoordinaten

David Holtz

fischertechnik und Plotter verbindet eine lange gemeinsame Geschichte: Im Jahr 1985 wurde von fischertechnik ein Plotter-Bausatz eingeführt, der zur damaligen Zeit vermutlich so fortschrittlich war, wie es der 3D-Drucker Baukasten heute ist. Seither stoßen die Kurvenschreiber auf große Begeisterung bei Fischertechnikern, weshalb man im Bilderpool der fischertechnik-Community oder auf YouTube unterschiedlichste Weiter- und Eigenentwicklung finden kann. Die meisten dieser Maschinen verbindet jedoch eine Gemeinsamkeit: Sie verfügen über zwei senkrecht zueinanderstehende Linearachsen; diese Bauform ermöglicht eine Ansteuerung durch kartesische Koordinaten. Es ist an der Zeit, daran etwas zu verändern, denn auch das Polarkoordinatensystem eignet sich für zweidimensionale CNC-Anwendungen.

Einführung

Ich habe mir zum Ziel gesetzt, einen Plotter zu konstruieren, der bedingt durch seine Bauform mit Polarkoordinaten angesteuert werden kann. Verglichen mit herkömmlichen Plottern weist der Polarkoordinaten-Plotter daher ein grundlegendes Unterscheidungsmerkmal auf: Anstelle einer zweiten Linearachse verfügt er über eine Rotationsachse, die die runde Zeichenfläche dreht (Abb. 2).

Trotz dieser Veränderung werden an den Polarkoordinaten-Plotter dieselben Anforderungen wie an seine kartesischen Verwandten gestellt: Um eine hohe Wiederholgenauigkeit zu erzielen, bedarf es einer präzisen Konstruktion aller mechanischer Baugruppen. Damit der Plotter flexibel einsetzbar ist, sollte er außerdem eine verbreitete Maschinensprache (G-Code) beherrschen. Nicht zuletzt wäre es wünschenswert, wenn durch Optimierung der Motorsteuerung die Zeichengeschwindigkeit erhöht werden kann.

Der folgende Beitrag gibt einen Überblick, welche technischen Anpassungen auf Hardware- und Softwareseite notwendig sind, bis eine Zeichnung mit Hilfe von Polarkoordinaten angefertigt werden kann.

Mathematische Grundlagen: Das Polarkoordinatensystem

Das Polarkoordinatensystem ist ein zweidimensionales Koordinatensystem. Zur Orientierung dient ein festgelegter Punkt, der als *Pol* bezeichnet wird. Von diesem Punkt geht strahlenförmig die sogenannte *Polarachse* aus, die als Orientierungsrichtung dient.

Hierbei entspricht der Pol dem Ursprung eines kartesischen Koordinatensystems, die Polarachse repräsentiert den positiven Bereich der x-Achse [1].

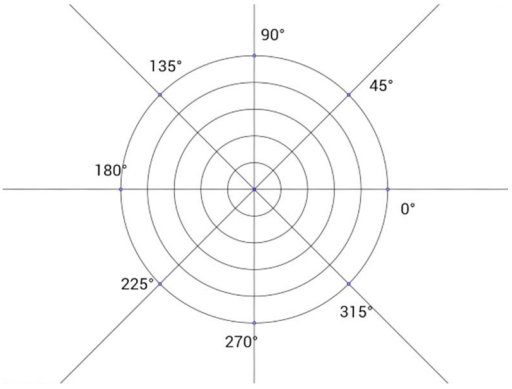


Abb. 1: Polarkoordinatensystem

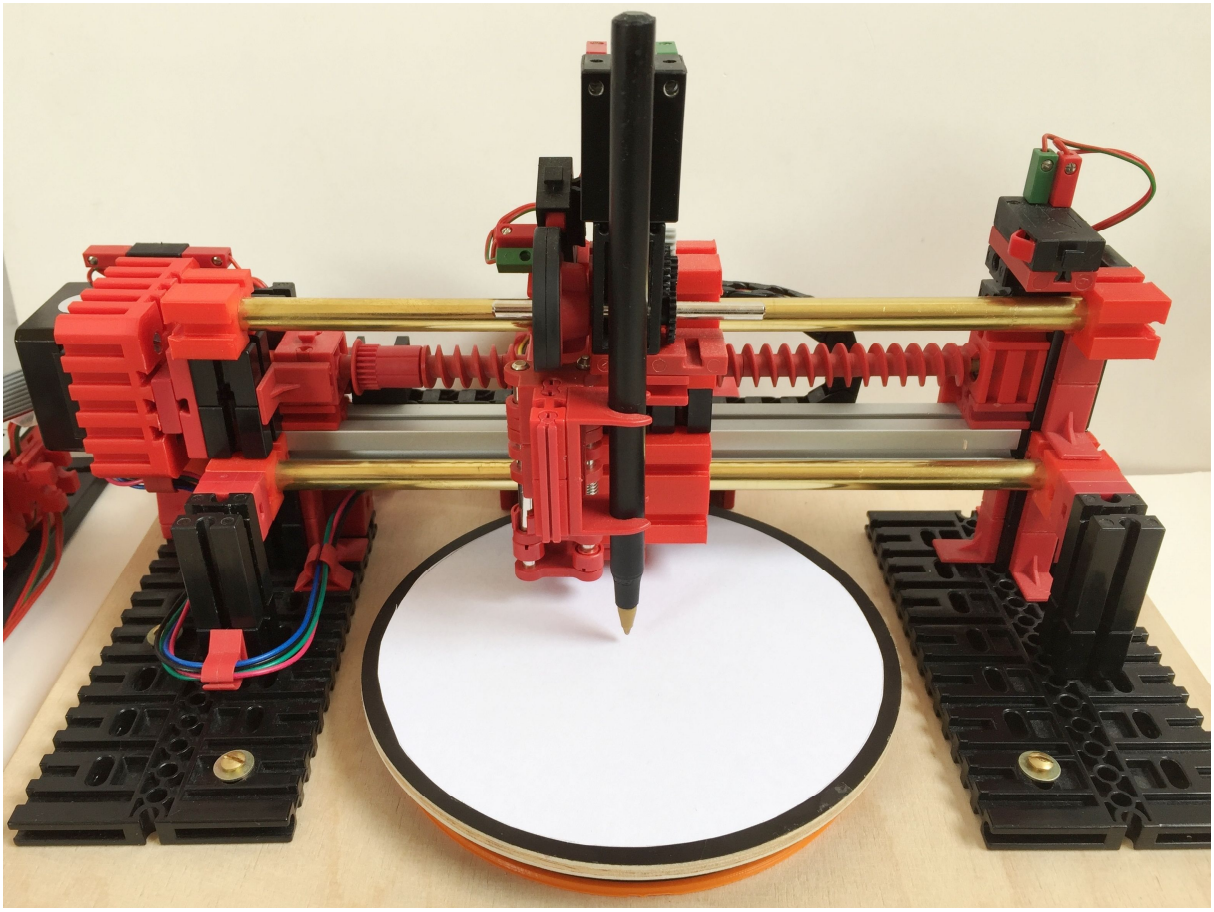


Abb. 2: Der Polarkoordinaten-Plotter

Polarkoordinaten

Jeder Punkt einer Ebene ist in der Polarkoordinatendarstellung durch den Winkel φ zu der Polarachse und durch den Abstand r (Radius) zum Pol definiert [1].

Prinzipiell ist es unerheblich, ob der Winkel im Grad- oder Bogenmaß angegeben wird. Tatsächlich verarbeitet die Steuersoftware des Plotters Winkelangaben im Bogenmaß. Zum besseren Verständnis beschränke ich mich in diesem Artikel jedoch auf die im Alltag gebräuchlichen Winkelangaben in Grad.

Mehrdeutigkeit

Grenzt man die Angabe der Winkelweiten nicht auf Werte zwischen 0° und 360° ein, so kann derselbe Punkt mit unterschiedlichen Koordinaten beschrieben werden: Ein Punkt Q sei definiert mit $\varphi_0 = 315°$ und $r = 5$. Der Punkt Q liegt an derselben Stelle wie Q', wenn Q' mit

- $\varphi_1 = \varphi_0 + 1 \cdot 360°$
- $\varphi_2 = \varphi_0 + 2 \cdot 360°$
- $\varphi_{-1} = \varphi_0 - 1 \cdot 360°$
- $\varphi_k = \varphi_0 + k \cdot 360°, k \in \mathbb{Z}$

und r beschrieben werden kann. Q ist somit nicht doppelpunktfrei.

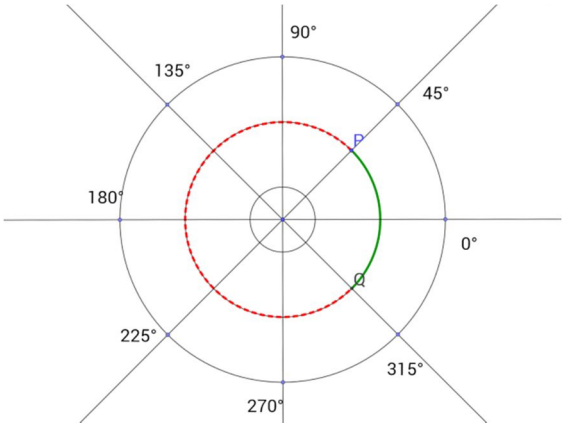


Abb. 3: Veranschaulichung der Mehrdeutigkeit

Diese Mehrdeutigkeit nutzt das Steuerprogramm des Plotters, denn sie ermöglicht es, kürzere Fahrwege zu wählen. Zur Veranschaulichung befinde sich der Stift am Punkt P(45°|5) und soll zum Punkt Q(315°|5) bewegt werden. Anstatt einer Drehung um 270° gegen den Uhrzeigersinn wählt das Steuerprogramm die kürzere Bewegung um 90° im Uhrzeigersinn zum Punkt Q'(-45°|5).

Umrechnung kartesischer Koordinaten in Polarkoordinaten

Um die Bewegungen des Plotters zu berechnen, müssen kartesische Koordinaten in Polarkoordinaten überführt werden. Ein allgemeiner Punkt P sei in kartesischen Koordinaten mit *x*- und *y*-Koordinate definiert. Gesucht sind die Polarkoordinaten φ und *r*, die P im Polarkoordinatensystem beschreiben.

Jeder Punkt P, der nicht im Ursprung liegt, bildet mit dem Ursprung des kartesischen Koordinatensystems und dessen x-Achse ein rechtwinkliges Dreieck. Hierbei reicht die Hypotenuse dieses Dreiecks vom Ursprung zum Punkt P. Die Länge der Hypotenuse entspricht dem gesuchten Abstand *r* zum Pol. Man erhält *r*, indem man den Satz des Pythagoras umformt zu:

$$r = \sqrt[2]{x^2 + y^2}$$

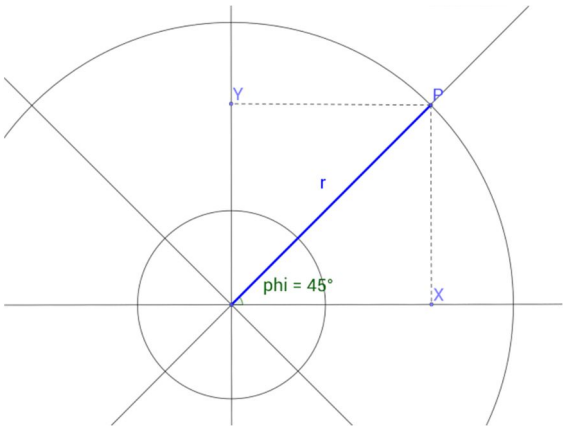


Abb. 4: Zur Umrechnung

Solange P im ersten Quadranten des kartesischen Koordinatensystems liegt, erhält man den Winkel φ, indem man Arkustangens aus dem Quotienten der *y*- und der *x*-Koordinate bildet ($\tan^{-1}$ steht für den arctan, nicht für einen Kehrwert):

$$\varphi = \tan^{-1}\frac{y}{x}$$

Da P in beliebigen Quadranten des kartesischen Koordinatensystems liegen kann, muss φ über eine Fallunterscheidung berechnet werden:

Fall	Berechnung von φ
$\boldsymbol{x > 0}$	$\varphi = \tan^{-1}\frac{y}{x}$
$\boldsymbol{x < 0}$	$\varphi = \tan^{-1}(\frac{y}{x}) + 180°$
$\boldsymbol{x = 0 \cap y > 0}$	$\varphi = 90°$
$\boldsymbol{x = 0 \cap y < 0}$	$\varphi = 270°$
$\boldsymbol{x = y = 0}$	φ mathematisch nicht definiert, P liegt im Ursprung bzw. Pol

Das Steuerprogramm sowie der G-Code-Postprozessor stützen sich auf diese mathematischen Vorüberlegungen.

Hardware: Die Mechanik

Im Wesentlichen besteht der Plotter aus drei Baugruppen:

- Drehteller als Rotationsachse
- Linearachse
- Stifthebemechanismus

Alle Baugruppen sind auf Präzision, Funktionalität und Robustheit ausgerichtet. Dank des minimalistischen, beinahe symmetrischen Designs des Plotters kann die Kinematik beim Zeichnen leicht erschlossen werden.

Drehteller

Die Baugruppe „Drehteller“ bildet das Charakteristikum des Plotters, da eine Rotationsachse bei gewöhnlichen Plottern üblicherweise nicht vorhanden ist. Zugleich macht der Drehteller eine Ansteuerung mit

Polarkoordinaten erst möglich. Um bei der Wiederholgenauigkeit mit Linearachsen mithalten zu können, erfordert die Konstruktion der Rotationsachse das höchste Maß an Präzision.

Bis auf einige Bauteile zur Befestigung des Schrittmotors kommt die Baugruppe „Drehteller“ ohne fischertechnik-Steine aus. Als Lager der Rotationsachse dient ein Kugellager, dessen Innenring fest mit der Grundplatte verbunden ist. Da das Kugellager teilweise in die Grundplatte eingelassen ist, misst die Höhe des Drehtellers weniger als 16 mm.

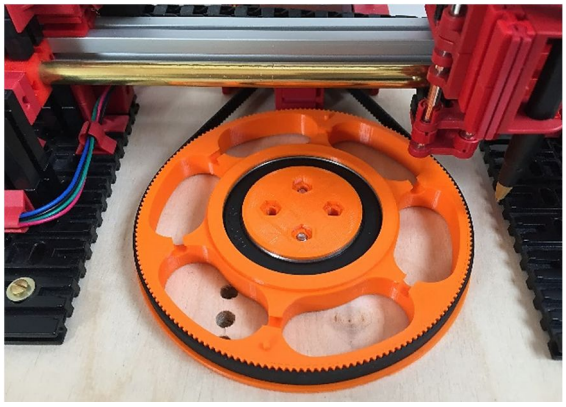


Abb. 5: Der Drehteller

Das Herzstück des Drehtellers, ein im 3D-Druckverfahren hergestelltes Zahnrad mit 180 Zähnen, wird auf den Außenring des Kugellagers gesteckt. Die schwarze Abdeckung des Drehtellers bildet zugleich die Zeichenfläche, auf die ein rundes Blatt Papier mit bis zu 13 cm Durchmesser geklebt werden kann. Der Drehteller wird von einem Schrittmotor angetrieben. Auf der Welle des Schrittmotors ist eine Zahnriemenscheibe mit 20 Zähnen befestigt. Über einen Zahnriemen wird so die Kraft auf das Zahnrad des Drehtellers übertragen. Daraus ergibt sich eine Untersetzung von 1:9.

Die Schrittweite des verwendeten Schrittmotors beträgt 1,8°. Aufgrund der 1:9-Untersetzung lässt sich der Drehteller somit bis auf 0,2° genau positionieren. Durch Abzählen der Schritte werden die Bewegungen des Drehtellers relativ zur Ausgangsstellung erfasst. Ein Endlagenschalter ist nicht notwendig, da die Maschine in jeder Stellung des Drehtellers mit dem Zeichenvorgang beginnen kann.

Abb. 6: Explosionsansicht des Drehtellers

Linearachse

Die Linearachse bewegt einen Schlitten, an dem die Stifthalterung befestigt ist. Das fischertechnik-System wird hierbei um eine präzise Linearführung, bestehend aus zwei Messingstangen mit 8 mm Durchmesser erweitert. Auf den Messingrohren gleiten Linearlager, sodass die Geradführung nahezu spielfrei ist.

Als Linearlager werden Rundbuchsen mit Kugelumlauf verwendet, die beispielsweise auch bei 3D-Druckern zum Einsatz kommen. Ein im 3D-Druckverfahren hergestellter Baustein verbindet die Linearlager mit dem fischertechnik-System.

Als Antrieb der Linearachse dient ein Schrittmotor, an dessen Welle eine Gewindeschnecke befestigt ist. Pro Motorumdrehung wird der Schlitten um rund 5,23 mm bewegt. Da die Schrittweite des Motors 1,8° beträgt, ließe sich der Schlitten theoretisch bis auf rund 0,026 mm genau positionieren. In der Praxis wird jedoch eher eine Auflösung von etwa 0,1 mm erreicht.

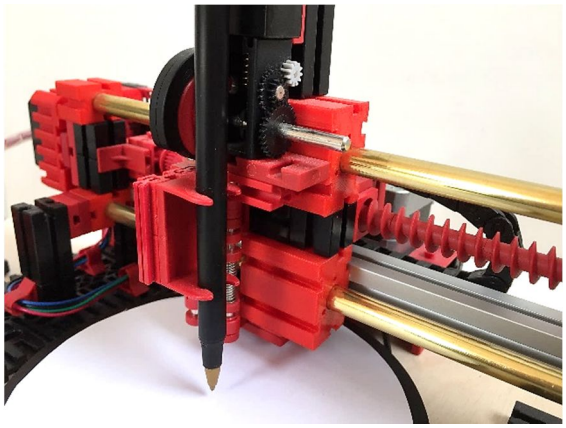


Abb. 7: Der Schlitten der Linearachse

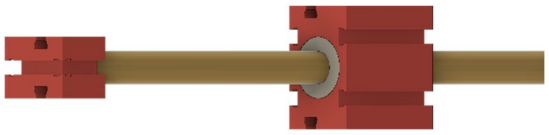


Abb. 8: Linearführung

Der maximale Vorschub des Schlittens ergibt sich aus der maximalen Drehzahl des Motors:

$$v = 600\frac{\mathrm{U}}{\mathrm{min}} \cdot \frac{5{,}23\mathrm{mm}}{\mathrm{U}} = \frac{52{,}3\mathrm{mm}}{\mathrm{s}}$$

Über einen Endlagenschalter kann die Linearachse zudem kalibriert werden. Bei gedrücktem Endlagenschalter befindet sich der Schlitten am Referenzpunkt. Alle weiteren Punkte werden durch das Abzählen von Motorschritten relativ zum Referenzpunkt bestimmt.

Die Schrittmotorhalterung ist aus wärmebeständigem ABS gedruckt und erlaubt, dass ein Schrittmotor der Baugröße NEMA 14 in das fischertechnik-System integriert werden kann.

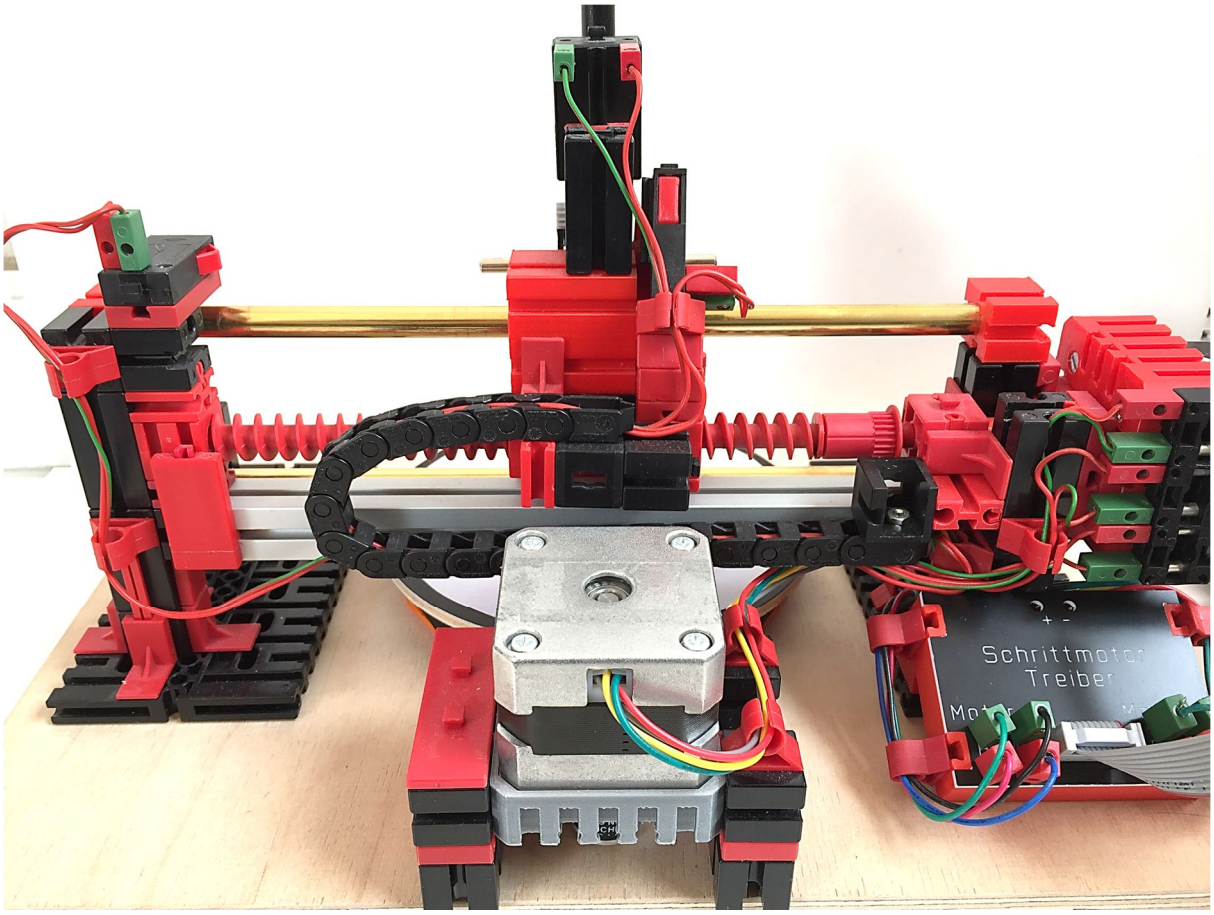


Abb. 9: Die Linearachse

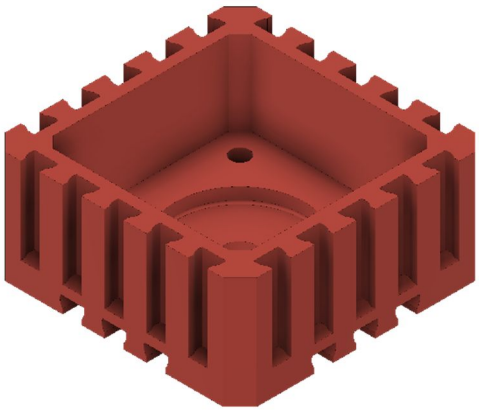


Abb. 10: CAD-Modell der Schrittmotorhalterung

Stifthebemechanismus

Dank des Stifthebemechanismus kann der Stift in Bruchteilen einer Sekunde angehoben und präzise abgesetzt werden. In die Stifthalterung wird hierfür ein klecksfreier Kugelschreiber geklemmt.

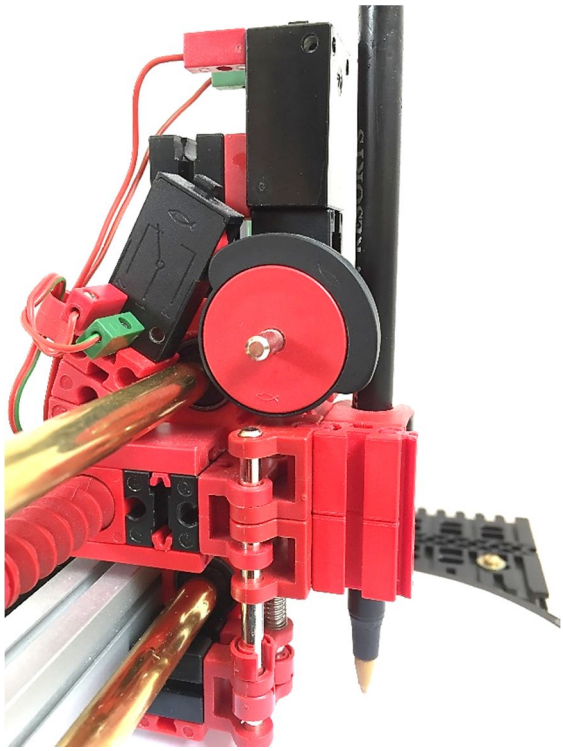


Abb. 11: Der Stifthebemechanismus

Ein Elektromotor bildet den Antrieb des Stifthebemechanismus, indem er eine Nockenscheibe dreht. Befindet sich die Nockenscheibe in der Ausgangsstellung, hält die Rückstellfeder die Stifthalterung in der oberen Position, sodass sich die Stiftmine etwa 3 mm oberhalb der Zeichenfläche befindet. Über den Endlagenschalter wird erfasst, ob sich die Nockenscheibe in der Ausgangsstellung befindet [4].

Eine Drehung der Nockenscheibe im Uhrzeigersinn um 90° bewirkt, dass die Stifthalterung nach unten gedrückt wird. Hierbei setzt der Stift mit konstantem, leichtem Druck auf der Zeichenfläche auf.

Steuerungstechnik

Das Herzstück der Steuerungstechnik ist ein Arduino Mega mit I/O-Shield. Der Mikrocontroller bildet die Schnittstelle zwischen der Software und der Hardware, indem er die G-Code-Maschinensprache in Steuerbefehle für Motoren übersetzt und diese ausführt. Außerdem ist er für die Regelkreise der einzelnen Achsen und das Auslesen der Endlageschalter verantwortlich.

Schrittmotorsteuerung

Ein zentrales Element der Steuerungstechnik des Plotters ist die Schrittmotorsteuerung. Schrittmotoren zählen zu den Synchronmotoren, da sich deren Rotor synchron zu einem schrittweise rotierenden Magnetfeld bewegt. Diese Eigenschaft ermöglicht, dass sich der Rotor ohne Verwendung eines zusätzlichen Sensors genau positionieren lässt. Verglichen mit Gleichstrommotoren erfordert sie jedoch eine kompliziertere Ansteuerung.

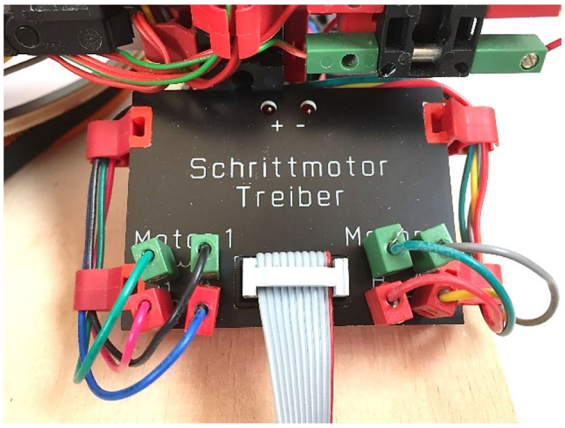


Abb. 12: Schrittmotorcontroller

Zur komfortablen Motorsteuerung werden die bipolaren Schrittmotoren über den Schrittmotorcontroller betrieben. Dessen Platine ist mit zwei Schrittmotortreibern vom Typ A4988 bestückt, die die Ansteuerung vereinfachen.

Bei jeder steigenden Flanke am STEP-Pin des Treibers wird der Rotor um einen Schritt weiterbewegt. Über den Pegel am DIR-Pin (*direction*) wird die Drehrichtung des Motors eingestellt. Ein Schrittmotor kann somit in Rotation versetzt werden, indem man den Signalzustand am STEP-Pin zyklisch verändert. Die Anzahl steigender Flanken pro Sekunde entspricht hierbei der Anzahl Schritte pro Sekunde.

Problematisch ist, wenn das Lastmoment das Motormoment übersteigt, sodass der Rotor dem sich drehenden Magnetfeld nicht mehr folgen kann. In der Folge werden einzelne Schritte übersprungen, dabei geht auch die Information über die Position des Rotors verloren. Ziel der Schrittmotorsteuerung ist es, Schrittverlust zu verhindern, um eine zuverlässige Positionierung der Achsen des Plotters zu gewährleisten.

Lineare Beschleunigungsrampen

Schrittverlust kann vermieden werden, indem die Motoren sanft beschleunigt und abgebremst werden. Der Ansatz besteht darin, dass der Mikrocontroller zur Laufzeit lineare Beschleunigungsrampen berechnet.

Abb. 13: Beschleunigungsrampen

Aus Überlegungen zur gleichmäßig beschleunigten Bewegung ergibt sich folgender Zusammenhang:

$$s(t) = \frac{1}{2} \cdot a \cdot t^2$$

$$t = \sqrt{\frac{2 \cdot s}{a}}$$

Hierbei ist die Beschleunigung a gegeben und konstant, s beschreibt im Sachzusammenanhang die Anzahl zurückgelegter Schritte, t beschreibt die Zeit, die während der beschleunigten Bewegung verstreicht.

Gesucht ist die Zeit, die zwischen zwei aufeinanderfolgenden Schritten verstreicht. Ausgedrückt als rekursive Folge mit diskreten Werten für n (Anzahl zurückgelegter Schritte) erhält man:

$$\Delta t = t_n - t_{n-1} = \sqrt{\frac{2 \cdot n}{a}} - t_{n-1}$$

Für den ersten Schritt gilt hierbei:

$$t_1 = \sqrt{\frac{2 \cdot 1}{a}}$$

Jede Berechnung erfordert eine Multiplikation, eine Division und anschließendes Wurzelziehen mit Gleitkommazahlen. Insbesondere dann, wenn mehrere Motoren gleichzeitig beschleunigt werden sollen oder wenn andere Aufgaben parallel ausgeführt werden, genügt die Rechenleistung des Arduino nicht für diese Berechnungen. Aus diesem Grund muss die Berechnung der Beschleunigungsrampen optimiert werden.

Optimierung

Basierend auf der Arbeit von David Austin [2] wird ein Algorithmus implementiert, der die Berechnung von Beschleunigungsrampen in Echtzeit auf rechenschwachen Prozessoren erlaubt.

Die exakte Folge (siehe oben) wird mit einer Taylorreihenentwicklung approximiert. Bereits das Polynom zweiten Grades bildet die ursprüngliche Folge sehr gut ab (vgl. Diagramm). Nach einigen Umformungen erhält man als Ergebnis eine vereinfachte rekursive Folge, deren Werte nur minimal von der exakten Folge abweichen. Dank des Algorithmus entfällt nun die Berechnung der Quadratwurzel, sodass der Mikrocontroller entlastet wird. Des Weiteren können die berechneten Werte als Festkommazahlen verarbeitet werden.

Das Diagramm in Abb. 14 zeigt die Zeit zwischen zwei aufeinanderfolgenden Schritten bei einer konstant beschleunigten Bewegung in Abhängigkeit des zurückgelegten Wegs. Die Annäherung ist dabei nicht mehr von der exakten Folge unterscheidbar.

Software: Von der Pixelgrafik zum fertigen Plot

Es sind einige Schritte notwendig, um aus einer Grafik Steuerbefehle für den Plotter zu berechnen. Liegt die Grafik als Pixelgrafik vor, sollte sie zunächst vektorisiert werden. Beim Vektorisieren wird die Pixelgrafik mit sogenannten grafischen Primitiven (Linien, Polygone, Kreise oder andere Kurven) nachgezeichnet. Die Vektorgrafik, die ausschließlich aus Primitiven anstelle von Pixeln zusammengesetzt ist, dient als Ausgangspunkt zur Erzeugung des Maschinencodes.

Erzeugung des Maschinencodes

Eine CAM-Anwendung (*computer-aided manufacturing*) erzeugt aus der Vektorgrafik den sogenannten G-Code. Dabei handelt es sich um die am weitesten verbreitete Programmiersprache zur rechnergestützten numerischen Steuerung (*computerized numerical control*, kurz: CNC) unterschiedlichster Maschinen [3].

Namensgebend ist die Eigenschaft, dass (fast) jede Zeile G-Code mit einem G beginnt, gefolgt von einer zweistelligen Zahl. Hierbei bestimmt die Zahl über die Bewegungsform der Maschine oder welche Einstellung vorgenommen werden soll. Dahinter können weitere Parameter, wie zum Beispiel Koordinaten oder Informationen zum Vorschub stehen:

```
G01: X4.5000 Y5.3600
```

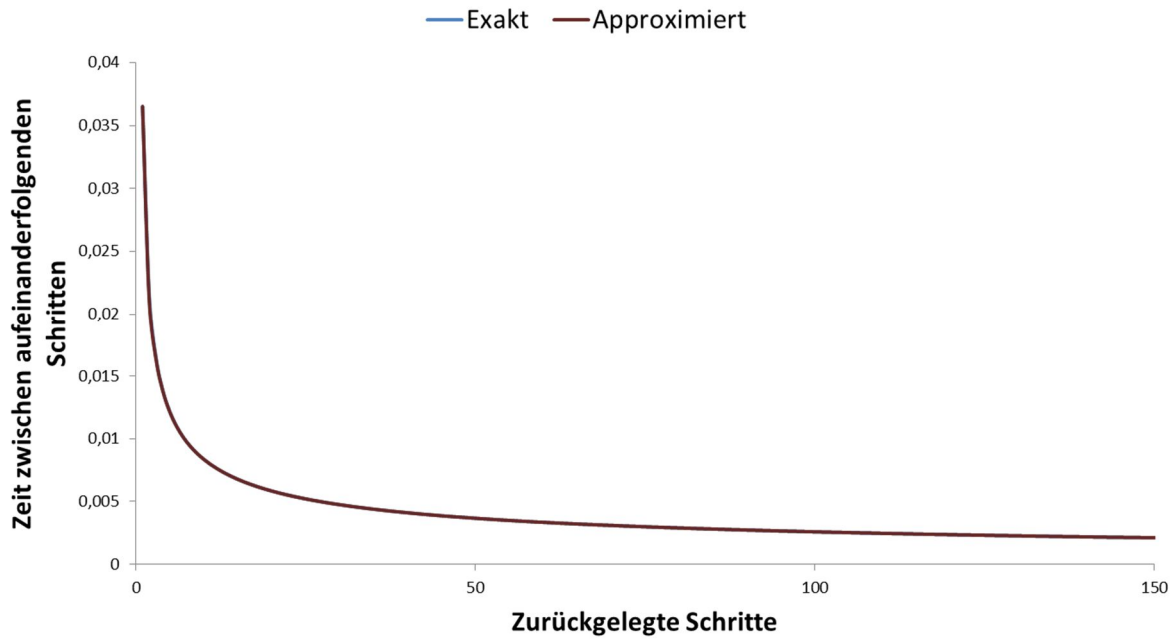


Abb. 14: Der optimierte Algorithmus entspricht sehr gut der exakten Beschleunigungsrampe

Das Beispiel beschreibt eine geradlinige Bewegung von der aktuellen, in diesem Kontext unbekannten Position zum Punkt (4,5|5,36). Für den Polarkoordinaten-Plotter sind nur vier G-Code Befehle relevant – siehe Abb. 15.

Plotter, die mit kartesischen Koordinaten gesteuert werden, können den erzeugten G-Code unmittelbar ausführen und die Grafik damit zu Papier bringen. Im Fall des Polarkoordinaten-Plotters muss der G-Code jedoch zuerst durch einen Postprozessor nachbearbeitet werden, damit die Maschine damit umgehen kann.

Nachbearbeitung des G-Codes

Der Postprozessor ist in C# implementiert. Aus Effizienzgründen führt ein Computer anstelle des Mikrocontrollers die Nachbearbeitung durch. Dazu wird der ursprüngliche G-Code eingelesen und die überarbeitete Polarkoordinatenversion als Kopie gespeichert. Die Nachbearbeitung umfasst mehrere Schritte:

- Umrechnung der kartesischen Koordinaten in Polarkoordinaten
- Auflösung der interpolierten Bewegungen
- Entfernen irrelevanter Informationen

Interpolierte Bewegungen (Linear- und Kreisinterpolationen) werden aufgelöst, indem viele Einzelpunkte berechnet werden, die auf der Bahnkurve des Endeffektors (Stifts) liegen. Dank des Bresenham-Algorithmus (vgl. [4]) können linear interpolierte Bewegungen effizient gerastert werden. Zur Auflösung von Kreisinterpolationen wird die Kreisvariante des Bresenham-Algorithmus verwendet. Auch sie vermeidet unter Zuhilfenahme einer Fehlervariablen rechenaufwändige trigonometrische Berechnungen [5].

Umgang mit interpolierten Bewegungen

Bei genauerer Betrachtung widerspricht die Auflösung interpolierter Bewegungen jedoch dem Grundkonzept eines Plotters: Plotter geben Vektorgrafiken unmittelbar aus, ohne dass im Vorfeld eine Rastergrafik berechnet wird [6]. Die Auflösung ist dennoch erforderlich, da bei der Umrechnung kartesischer Koordinaten in Polarkoordinaten Informationen über die Bahnkurve des Endeffektors verloren gehen.

Im Beispiel soll der Stift geradlinig von Punkt A(1|1) zu Punkt B(1|-1) bewegt werden (grüne Bahn). Alle Punkte, die zwischen A und B liegen, besitzen die gleiche *x*-Koordinate. Ein Plotter, der mit kartesischen Koordinaten gesteuert wird, muss daher lediglich seine *y*-Achse entsprechend bewegen. In Polarkoordinaten ist dies nicht ohne weiteres möglich: Zwar

	Bedeutung	Parameter
G00	Rapid positioning: schnelle Bewegung zum Zielpunkt ohne Beachtung der Bahnkurve des Endeffektors	*x*-*y* Koordinaten des Zielpunkts, *z*-Koordinate
G01	Linearinterpolation: geradlinige Bewegung zum Zielpunkt	*x*-*y* Koordinaten des Zielpunkts
G02	Kreisinterpolation im Uhrzeigersinn: Kreis(bogen)	*x*-*y* Koordinaten des Zielpunkts, Koordinaten des Kreismittelpunkts
G03	Kreisinterpolation gegen den Uhrzeigersinn: Kreis(bogen)	*x*-*y* Koordinaten des Zielpunkts, Koordinaten des Kreismittelpunkts

Abb. 15: Für den Plotter relevante G-Codes

unterscheiden sich die Punkte A(45°|√2) und B(-45°|√2) nur in ihrer Winkelkoordinate. Würde der Plotter jedoch ausschließlich seine Rotationsachse bewegen, entspräche die Bahnkurve einem Viertelkreisbogen (rote Bahn).

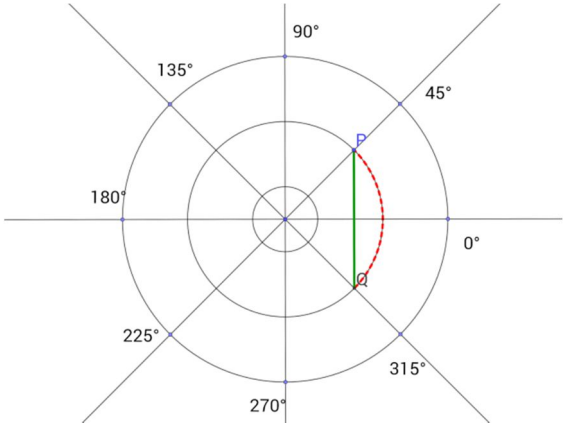


Abb. 16: Auflösung der Interpolation

Um eine geradlinige Bahnkurve zu erhalten, muss der Radius daher kontinuierlich verändert werden. Dies gelingt, indem der Postprozessor viele Punkte berechnet, die zwischen A und B liegen. In der Praxis ist das gewählte Raster allerdings derart klein, dass aus der Rasterung keine qualitativen Nachteile resultieren.

Mit dem Abschluss der Nachbearbeitung könnte eine Zeile des G-Codes wie folgt aussehen:

```
G00 P0.785 R42.4
```

P und R werden als Trennzeichen der Polarkoordinaten interpretiert, diese Interpretation weicht von der Norm des G-Codes ab. Hinter den Trennzeichen P und R steht der Winkel beziehungsweise der Abstand eines Punktes zur Kreismitte.

Windows Forms App (C#)

Grafisch aufbereitet ist der Postprozessor in der Windows-Forms-App „Polarkoordinaten Plotter Controller“ enthalten. Die App bildet zugleich die Benutzerschnittstelle, denn sie ermöglicht eine unkomplizierte Konfiguration des USB Ports sowie das Starten und Beenden von Zeichnungen (Plots).

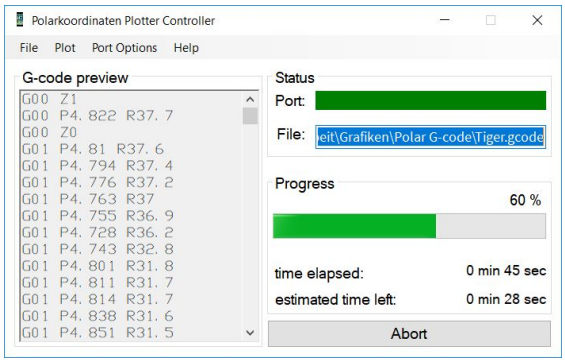


Abb. 17: Screenshot der Windows-App

Während der linke Teil des Fensters eine kurze Übersicht über den aktuell geladenen G-Code gibt, zeigt der rechte Teil den Status des Plotters an. Die App zeigt den Fortschritt beim Zeichnen an und berechnet einen Schätzwert für die verbleibende Restdauer des Plots. Ein separater Thread gewährleistet im Hintergrund die Datenübertragung zum Plotter.

Datenübertragung

Die Größe der überarbeiteten G-Code Dateien variiert je nach Komplexität der Zeichnung. Gemessen an der Speicherkapazität des Arduino Mega sind die teils mehrere hundert Kilobyte große Dateien „riesig“, weshalb sie portionsweise in den Speicher des Mikrocontrollers geladen werden müssen.

Mithilfe der Windows-Forms-App „Polarkoordinaten Plotter Controller“ wird der G-Code zeilenweise auf den Arduino übertragen. Während die Plotter-Firmware die aktuelle G-Code Zeile ausführt, wird parallel dazu die nächste Zeile in den Speicher geladen, sodass keine Wartezeiten aufgrund der Datenübertragung anfallen. Arduino und Computer kommunizieren hierbei über den seriellen Datenbus (USB), sodass der Plotter prinzipiell mit jedem (Windows) Computer betrieben werden kann.

Plotter-Firmware

Neben der Windows-App stellt die Plotter-Firmware den zweiten entscheidenden Teil der Softwarelösung dar. Im Wesentlichen handelt es sich hierbei um einen G-Code-

Interpreter, d. h. ein Programm, das Maschinensprache einliest, interpretiert und in Steuerbefehle umsetzt. Beispiel:

```
G00 P0.785 R42.4
```

Der G-Code-Interpreter entnimmt der obigen Zeile folgende Informationen:

- G00: gefordert ist eine schnelle Bewegung (rapid positioning)…
- zum Punkt (45°|42,4).

Abhängig von der aktuellen Position des Stifts berechnet die Firmware, um wie viele Schritte in welche Richtung die jeweiligen Achsen hierfür bewegt werden müssen und steuert die Motoren entsprechend an. Hierbei berechnet sie in Echtzeit die Beschleunigungsrampen für jede Bewegung.

Ergebnisse

In ersten Praxistests zeigte sich, dass die Hardware die zu Beginn genannten Anforderungen erfüllt. Besonders schön sind die Zeichnungen, die aus einfachen Spiralbögen bestehen. Sie profitieren von der Ansteuerung in Polarkoordinaten und lassen sich ganz ohne aufwändige G-Code Verarbeitung mit wenigen for-Schleifen erzeugen. Polarkoordinaten vereinfachen außerdem kreisförmige Anordnungen um den Pol, da Zeichenschritte nach einer kleinen Drehung der Zeichenfläche einfach wiederholt werden können.

Abb. 18: Spiralbogenzeichnung

Der G-Code Interpreter eröffnet weitere Möglichkeiten, sodass sich auch komplexere Bilder zeichnen lassen. Hierbei kann der Polarkoordinaten-Plotter genauso flexibel wie herkömmliche Plotter eingesetzt werden. Zur Demonstration hat der Plotter das Space Shuttle „Columbia“ gezeichnet. Dank optimierter Schrittmotorsteuerung beansprucht dies nur 3 Minuten und 57 Sekunden.

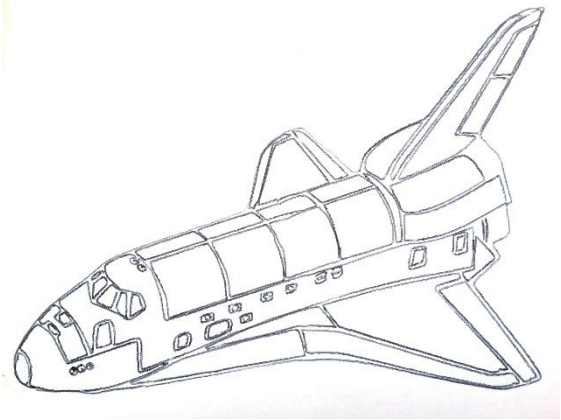


Abb. 19: Columbia

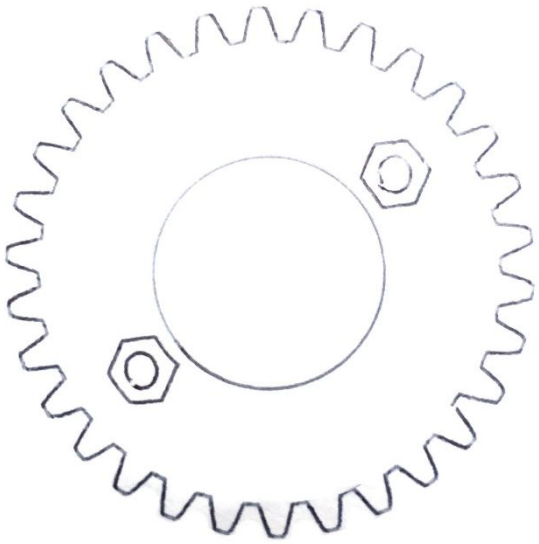


Abb. 20: Technische Zeichnung: Zahnrad

Ein klassischer Anwendungsfall von Plottern ist die Ausgabe technischer Zeichnungen. Über die Exportfunktion im CAD-Programm können Zeichnungen bereitgestellt werden. Der Polarkoordinaten-Plotter fertigt sowohl zweidimensionale

technische Zeichnungen als auch isometrische Projektionen eines dreidimensionalen Körpers an.

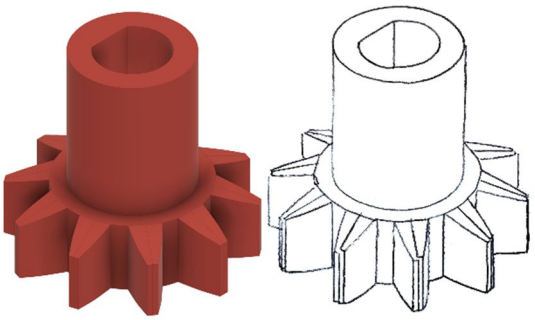


Abb. 21: Isometrie-Darstellung eines Zahnrads

Abb. 22: Text (normal, outline)

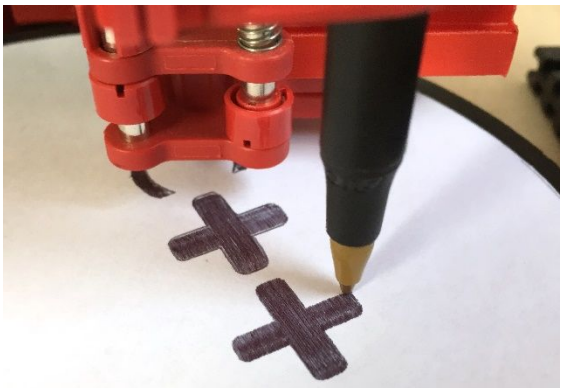


Abb. 23: Text (ausgefüllt, infill)

Nicht zuletzt lassen sich auch Buchstaben, Ziffern und Sonderzeichen zeichnen. Im Schnitt benötigt der Plotter knapp sieben Sekunden für einen Buchstaben. Deutlich zeitaufwändiger ist die Option, Buchstaben ausmalen zu lassen. Hierfür zeichnet der Plotter viele eng aneinander liegende, parallele Striche, sodass der Eindruck entsteht, die Fläche sei ausgemalt worden.

Ausblick

Im nächsten Schritt soll der Plotter noch um eine Smartphone-App ergänzt werden. Die App ermöglicht dem Nutzer, über den Touchscreen den Stift des Plotters zu bewegen und damit Zeichnung live anzufertigen.

Doch auch damit sind die Möglichkeiten der Zeichenmaschine noch nicht erschöpft: Denkbar wäre, den Plotter in einen 3D-Drucker umzufunktionieren. Dazu müsste der Stifthebemechanismus durch eine weitere Linearachse ersetzt werden, die orthogonal zur Zeichenfläche, respektive dem Druckbett, steht. Aus dieser Veränderung ergibt sich, dass ein solcher 3D-Drucker mit dreidimensionalen Polarkoordinaten, den sogenannten zylindrischen Koordinaten angesteuert werden müsste. Selbstverständlich müsste die Maschine um einen Druckkopf mit Extruder ergänzt werden.

Auch ohne diese Erweiterungen demonstriert das Projekt anschaulich, dass sich das Polarkoordinatensystem für zweidimensionale CNC-Anwendungen gut eignet. Allerdings sind einige Anpassungen insbesondere bei der G-Code-Verarbeitung notwendig, bis der Polarkoordinaten-Plotter eine flexibel einsetzbare Maschine ist. Hinsichtlich Wiederholgenauigkeit und Zeichengeschwindigkeit kann der Plotter durchaus mit seinen kartesischen Verwandten mithalten.

Quellen

[1] Wikipedia: *Polarkoordinaten*.

[2] David Austin: *Generate stepper-motor speed profiles in real time*.

[3] Wikipedia: G-Code.

[4] Dirk Fox: *HP-GL-Plotter (Teil 2)*. ft:pedia 1/2012, S. 4-12.

[5] Wikipedia: *Kreisvariante des Bresenham-Algorithmus*.

[6] Wikipedia: *Plotter*.

Computing

Schwarze, graue und sonstige Motoren am ftDuino

Till Harbaum

Motoren und andere Aktoren spielen in den meisten fischertechnik-Modellen eine große Rolle, sorgen sie doch für Bewegung und Action. Mit seinen zum TX- und TXT-Controller kompatiblen Ausgängen kann der ftDuino die gängigen Motoren problemlos ansteuern. Oft ist es aber nur eine Frage der passenden Software, auch exotische Motoren anzusteuern. Hier punktet der ftDuino mit seiner Open-Source-Philosophie und der damit einhergehenden völlig freien Programmierbarkeit.

Bei den eigenen Controllern tut fischertechnik gut daran, sich auf die aktuell lieferbaren Motoren zu konzentrieren und dafür zu sorgen, dass die aktuellen Baukästen und Controller miteinander kombinierbar sind. Der ftDuino entstand dagegen unabhängig von fischertechnik und ist nicht auf spezielle Baukästen festgelegt. Er wurde entworfen, um möglichst universell einsetzbar sein und auch längst aus dem Programm genommene Motoren ansteuern zu können.

Die aktuellen Motoren

Bei den aktuellen schwarzen Motoren gibt es bei der Verwendung am ftDuino nichts Spezielles zu beachten. Die XS-, S-, M- und XM-Motoren, der Encoder-Motor (153422) aus dem TXT-Discovery-Set sowie der schwarze „Traktormotor“ (1151178) werden allesamt mit 9 Volt betrieben und lassen sich alle direkt an den ftDuino anschließen. Der ftDuino kann dazu wahlweise mit dem Akku-Pack, dem 9 Volt-Batteriehalter oder dem Netzgerät (505287) betrieben werden. Auch der Kompressor sowie die Magnetventile aus dem Electro-Pneumatic-Kasten sind ohne Einschränkung kompatibel.

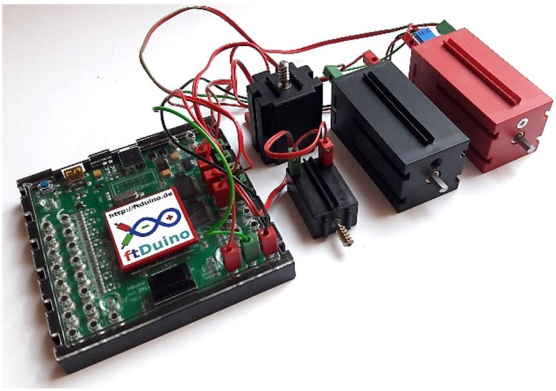


Abb. 1: ftDuino mit aktuellen Motoren

Nicht mehr ganz aktuell, aber oft noch zu kaufen ist der Encoder-Motor aus dem „ROBO TX Automation Robots“-Baukasten. Auch dieser lässt sich direkt am ftDuino betreiben, man muss lediglich die gegenüber dem TXT-Encoder-Motor etwas andere Impulszahl beachten. Während der aktuelle TXT-Motor 63 1/3 Impulse pro Umdrehung liefert sind es beim älteren TX-Motor 75 Impulse pro Umdrehung. Der namensgebende Encoder dieser Motoren wird analog zum TX oder TXT an einen der Zähler-Eingänge des ftDuino angeschlossen. Die Verkabelung erfolgt dabei exakt wie beim TXT.

Jüngere, aber nicht mehr erhältliche Motoren

Einige Motoren aus der aktuellen 9 Volt-Zeit sind bereits nicht mehr verfügbar. Die Power-Motoren zählen zum Beispiel dazu. Sie lassen sich wie jeder andere 9 Volt-Motor ebenfalls am ftDuino betreiben.

Als Encoder wurden in den dazugehörigen Baukästen einfache Taster verwendet, die natürlich ebenfalls problemlos am ftDuino zu nutzen sind.

Motoren aus grauer Vorzeit

Wenn es um die noch älteren grauen Motoren geht, stellt sich zunächst die Frage nach der Betriebsspannung. fischertechnik hat über die Zeit die spezifizierte Betriebsspannung seiner Modelle kontinuierlich erhöht: von zunächst 4,5 Volt aus dem Batteriestab (31041) über 6 Volt aus dem Batteriehalter (37261), den grauen 6,8 Volt-Trafo (812/814), den 8,4-Volt-Akkupack bis zu den heutigen 9 V-Netzteilen.

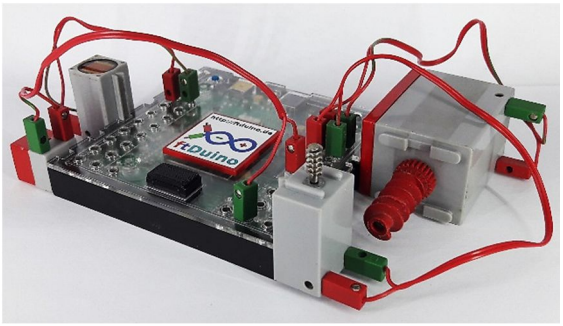


Abb. 2: Ältere Motoren und Sensoren am ftDuino

Was zunächst unübersichtlich und potenziell schädlich klingt, wird beim genauen Hinsehen schnell harmloser. Der bereits vor 35 Jahren auf den Markt gekommene Trafo 812 sollte laut Aufdruck 6,8 Volt liefern, stellte aber eine Leerlaufspannung von fast 12 Volt bereit. Selbst bei moderater Belastung liefert er in Kombination mit dem Gleichrichter-Silberling (36393) wie in Abb. 3 dargestellt eine fast 9 Volt erreichende Gleichspannung. Diese Kombination wurde häufig eingesetzt und belieferte auf diese Weise schon vor 35 Jahren die damaligen Motoren und Lampen mit Spannungen von 9 Volt und mehr. Motoren, die schon damals an dieser Kombination betrieben wurden, sollten also auch mit heutiger 9 Volt-Versorgung keine Probleme bekommen. Und auch andersrum gilt, dass der ftDuino an der Kombination aus Trafo und Gleichrichter-Silberling problemlos betrieben werden kann.

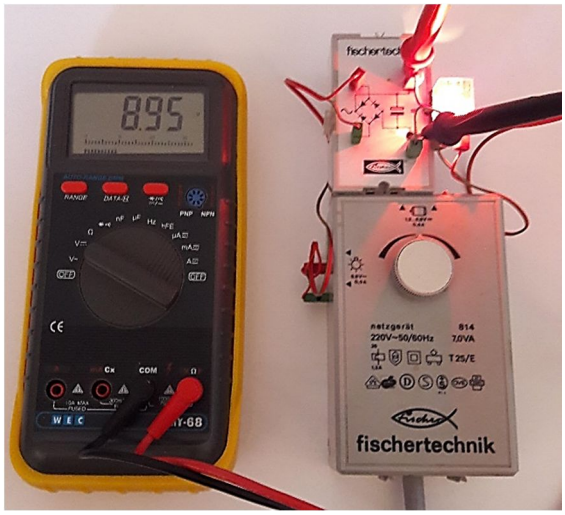


Abb. 3: Lampe am Trafo 812

Auf den Versuch, den ftDuino direkt aus dem stirnseitigen Gleichspannungsausgang des Trafos zu versorgen sollte man hingegen verzichten. Hier liegt keine geglättete Gleichspannung an, sondern stattdessen pulsiert diese Gleichspannung 100-mal pro Sekunde. Erst der große Kondensator des Gleichrichtersilberlings glättet die Spannung so weit, dass der ftDuino stabil genug versorgt wird.

Die alten grauen Mini-Motoren und natürlich die graue Variante dessen, was heute M-Motor heißt, lassen sich also mit dem aktuellen 9 Volt-Netzteil am ftDuino gefahrlos ansteuern oder aber stilecht mit Trafo und Gleichrichterbaustein. Der ftDuino eignet sich daher ausgezeichnet, die üblichen grauen Dachbodenfunde ins aktuelle Jahrtausend zu katapultieren.

Vorsicht ist bei einigen Sensoren aus den Computing-Anfängen geboten. So sollte

zum Beispiel die Gabellichtschranke (32357) aus dem „Computing Trainings-Roboter“ (30572) von 1985 nicht mit den 9 Volt des ftDuino versorgt werden, stattdessen können die nötigen 5 Volt z. B. vom I²C-Anschluss des ftDuino abgezweigt werden. Hier ist etwas Vorsicht angeraten. Der I²C-Anschluss des ftDuino wird nicht zufällig mit einer Schutzkappe ausgeliefert. Die hier anliegenden Signale sind empfindlich gegen Kurzschlüsse und Spannungen über 5 Volt.

Schrittmotoren

Die bisher beschriebenen Gleichstrom-motoren kommen in fast allen fischer-technik-Modellen zum Einsatz. Eine der wenigen Ausnahmen war der Plotter (30571), ebenfalls von 1985. In diesem Modell kamen zwei sogenannte Schritt-motoren (32311) zum Einsatz.

Diese sehr interessanten Motoren erlauben es, Positionen exakt und wiederholbar anzu-fahren. Der Nachteil besteht darin, dass pro Schrittmotor zwei Motorausgänge am da-maligen Intelligent-Interface belegt wur-den. Versorgt wurden der Plotter und seine Schrittmotoren damals aus dem Compu-ting-Netzgerät (32680). Auch dieses Netz-gerät ist mit 6,8 Volt spezifiziert, lieferte aber in der Praxis ebenfalls eine deutlich höhere Spannung, so dass davon auszu-gehen ist, dass die Motoren ebenfalls eine deutlich höhere Spannung vertrugen. Trotzdem ist beim Einsatz der historischen Schrittmotoren (32311) größte Vorsicht angebracht, um unnötige Schäden zu vermeiden. Wenn vorhanden, sollte der ftDuino beim Betrieb mit diesen Motoren zunächst an einem Labornetzteil bei 6,8 Volt betrieben werden.

Uns steht bisher keiner der Originalmotoren zur Verfügung, so dass wir die prinzipielle Machbarkeit mit einem NEMA17-HS26-Motor überprüft haben. Dieser Motor ist für 12 Volt ausgelegt, lässt sich aber an 9 Volt problemlos betreiben. Zusammen mit einem passenden 3D-Druck-Gehäuse von Thingiverse [1] und einem Wellenadapter von 5 mm auf fischertechnik-übliche 4 mm ergibt sich auch mit diesem Motor eine nahtlose fischertechnik-Integration, wie in Abb. 4 zu sehen ist. Ein passender Beispiel-Sketch wird mit der ftDuino-Installation bereits mitgeliefert und kann als Basis für das komplette Modell dienen.

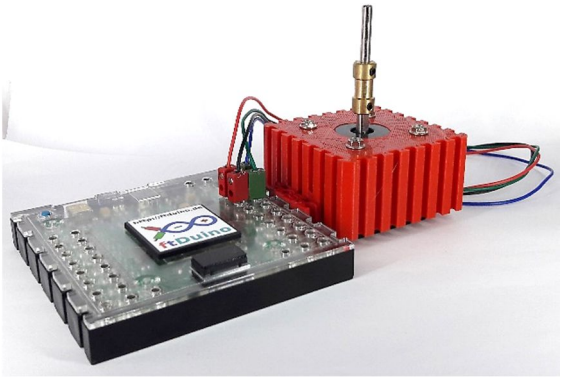


Abb. 4: Nicht-ft-Schrittmotor am ftDuino

Die Ansteuerung des Plotters wurde von fischertechnik damals trickreich realisiert, da eigentlich für jeden Motor zwei der ins-gesamt vier Motorausgänge des Interfaces nötig wären und damit kein Ausgang zum Heben und Senken des Plotterstifts mehr frei gewesen wäre. Da der ftDuino den gleichen Beschränkungen unterworfen ist, kann auch hier die gleiche Lösung helfen. Damals wurden einfach jeweils eine der beiden Spulen der Motoren an den gleichen Ausgang gesteckt. Pro Motor gab es daher eine unabhängig und frei steuerbare Spule und eine, die gemeinsam mit der Spule des anderen Motors angesteuert wurde. Das Resultat ist, dass bei Steuerung eines Motors der andere leicht „zuckt“. Da die gesamte Plottermechanik damals etwas Spiel hatte, ist dieses Zucken nie bis zum Stift vorgedrungen und hat entsprechend nicht gestört.

Angesteuert wurde der Plotter damals mit speziellen Programmen auf den zu der Zeit üblichen Homecomputern. Diese stehen heute in der Regel nicht mehr zur Ver-fügung. Der ftDuino kann diese Lücke aber

sehr elegant füllen, da er sich einem PC gegenüber per USB als nahezu beliebiges Gerät ausgeben kann. Mit einer entsprechenden Programmierung wird er so kompatibel zu heute noch verfügbaren Plottern und kann aus aktueller PC-Software direkt angesprochen werden. Bei der dafür nötigen Programmierung ist der ftDuino dem TX- oder TXT-Controller deutlich überlegen und seine Quell-offenen Treiber erlauben eine leichte Anpassung und die speziellen Bedürfnisse des alten Plotters.

Servos

Eine vor allem im Modellbau verbreitete Motorsorte sind die sogenannten Servos. fischertechnik hatte in den 80er Jahren bereits ein Servo unter der Artikelnummer 31495 im Programm. Auch heute findet sich als Teil der Infrarot- oder Bluetooth-Fernsteuersets ein Micro-Servo unter der Nummer 132292 im fischertechnik-Portfolio.

Im Inneren dieser Servos steckt ein einfacher Gleichstrommotor. Allerdings wird dieser von einer Elektronik sowie einem Winkelgeber begleitet, sodass der Servo einem externen Steuersignal folgend einen bestimmten Winkel anfahren kann. fischertechnik nutzt Servos daher in erster Linie zur Lenkung seiner Modelle. An den TX- und TXT-Controllern lassen sich Servos nicht betreiben, was sie im fischertechnik-Umfeld für den Einsatz in Robotics-Modellen disqualifiziert – was schade ist, da sie sich vielfältig einsetzen lassen [2].

Servos sind zum Betrieb an 6 Volt ausgelegt, und dieser Wert ist ausnahmsweise ernst zu nehmen. Oft lassen sich Servos auch an 5 Volt betreiben, was eine Versorgung aus dem I²C-Anschluss ermöglichen würde. Allerdings ist dieser nicht zur Motorversorgung ausgelegte Anschluss des ftDuino der oft recht üppigen Stromaufnahme der Servos nicht gewachsen. Die heute bei fischertechnik üblichen 9 Volt sind definitiv zu viel für Servos. Stattdessen empfiehlt sich der Einsatz eines billigen Spannungsreglers wie im entsprechenden Kapitel des ftDuino-Handbuchs beschrieben. Zusammen mit einem passenden Adapterkabel und ein paar fischertechnik-Steckern erhält man so eine einfache Möglichkeit, Servos am ftDuino zu betreiben wie in Abb. 5 dargestellt. Das eigentliche Steuersignal wird dabei aus zweckentfremdeten I²C-Signalen gebildet. Der ftDuino hat wie jeder Arduino flexibel konfigurierbare Ausgänge. Dadurch ist es unter anderem möglich, auf den Anschlüssen des I²C-Anschlusses völlig andere Signale zu erzeugen. Wir nutzen das, um das vom Servo benötigte PWM-Signal zu erzeugen. Ein Mischbetrieb von I²C und Servo ist allerdings nicht möglich. Die jeweils für den anderen Empfänger gedachten Signale könnten für Verwirrung sorgen und unerwünschte Reaktionen hervorrufen.

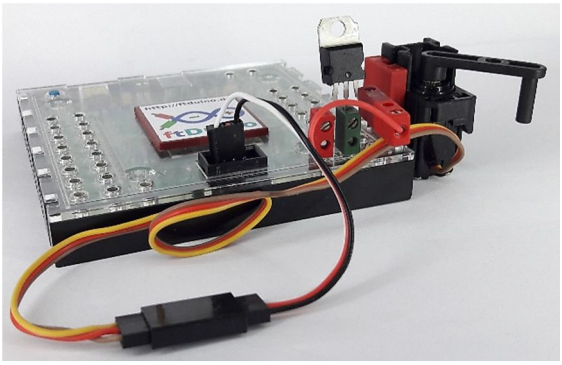


Abb. 5: Servo am ftDuino

Fazit

Der ftDuino ist ein Kommunikationswunder: Kaum ein Motor aus der langen fischertechnik-Historie lässt sich mit seiner Hilfe nicht zum Leben erwecken. Die Möglichkeit, sehr direkt auf die Hardware zuzugreifen, erlaubt es, auch exotische Motoren wie Schrittmotoren und Servos anzusteuern. Diese Möglichkeiten hätte fischertechnik selbst auch, allerdings beschränkt sich der offizielle Support der TX- und TXT-Controller auf Motoren aus dem aktuellen Baukasten-Programm und auch hier nur auf diejenigen Motoren, die

fischertechnik als Teil der Robotics-Linie betrachtet. Selbst aktuelle Servos bleiben daher außen vor.

Bisher konnte die Community dem nicht viel entgegensetzen. Mit dem ftDuino hat sie nun aber die volle Kontrolle über einen Controller und kann fischertechnik zeigen, was an verborgenen Fähigkeiten in ihrem System schlummert.

Ich freue mich schon auf den ersten Plotter von 1985, der 33 Jahre später einem aktuellen PC zur Seite steht als wäre es das Normalste der Welt. Dass so ein Setup nicht ganz billig wäre ist allein der Tatsache zuzuschreiben, dass der Plotter inzwischen ein seltenes Sammlerstück ist.

Mehr Informationen zu Preisen und Verfügbarkeit gibt es per E-Mail an info@ftduino.de.

Quellen

[1] Thomas Dragon, *fischertechnik nema 17 stepper motor holder* auf Thingiverse.

[2] Dirk Fox, Thomas Püttmann: *fischertechnik-Roboter mit Arduino*. dpunkt-Verlag, 2018.

Computing

Plug & Play am I²C-Bus mit dem ftExtender

Björn Gundermann, Christian Lauff, Rudenz Schulz,
Christian Bergschneider, Stefan Fuss

Der I²C-Bus am TX(T) ist universell: Es lassen sich Sensoren und Aktoren für alle Anwendungsfälle anschließen, sogar der ftDuino kann als preiswertes Erweiterungsboard mit dem TX(T) verbunden werden. Theoretisch, denn die immer wiederkehrenden Themen Versorgungsspannung, Logikspannung und der Anschluss mehrerer Devices verhindern den breiten Einsatz.

Lange vor dem Aufleben der Maker-Szene hat fischertechnik den TX- und den TXT-Controller mit dem I²C-Bus ausgestattet.[1] Arduino und Raspberry Pi setzen ebenfalls auf dem I²C-Bus, sodass es viele Sensoren mit I²C-Businterface fertig zu kaufen gibt. In der ft:pedia und im Internet finden sich für viele Sensor-Boards ROBOPro-Treiber, sodass die jeweilige Platine nur noch elektrisch angeschlossen werden muss.

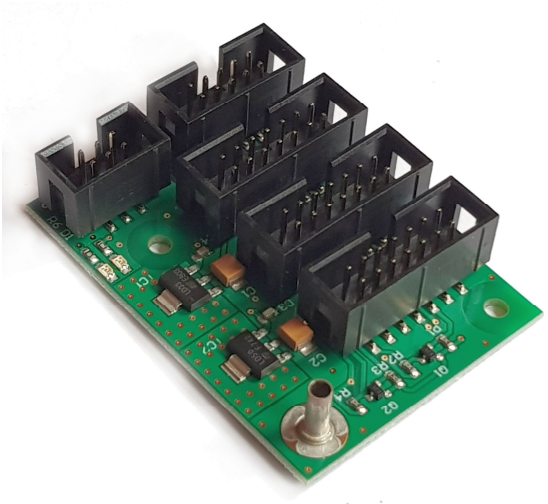


Abb. 1: Der ftExtender

Auch der ftDuino kann über den I²C-Bus als preiswertes Erweiterungsboard an TX und TXT angeschlossen und über ROBOPro programmiert werden [4]. Der Anschluss am TX ist einfach, der TXT benötigt eine zusätzliche Beschaltung.

Die Idee für den ftExtender wurde in der ft:pedia und der fischertechnik community schon mehrfach diskutiert und es wurden auch schon verschiedene Lösungen vorgestellt [1, 2, 3]. Allerdings lösen die Schaltungen immer nur einen Teil der Fragestellung, und bei [2, 3] ist der Nachbau einer Platine notwendig. Der hier vorgestellte ftExtender ist ein durch die fischertechnik community entwickeltes Fertiggerät. TX, TXT, ftDuino und Sensoren werden einfach per Flachbandkabel angeschlossen. Es kann über ft-extender@gundermann.org bezogen werden.

Mehrere I²C-Bus Devices

Am I²C-Bus lassen sich bis zu 100 Devices gleichzeitig anschließen.[2] Die Verkabelung kann als Bus, als Stern oder als Mischung von beiden ausgeführt werden. Es gibt eine Längenbeschränkung über die Kapazität des Kabels; diese liegt für fischertechnik-Anwendungen im Bereich von mehreren

[1] Der I²C-Bus selbst ist ein Industriebus und stammt aus den 1980er Jahren. Beim Design wurde auf einfache Technik und hohe Störsicherheit geachtet, so dass er sich hervorragend für Maker-Projekte eignet.

[2] Durch die Verwendung eines I²C-Adress-Multiplexers kann die Zahl der Devices nahezu beliebig vergrößert werden [8].

Metern. So große Modelle sind allerdings selten.

Allerdings steht am TX(T) nur ein EXT-Stecker zur Verfügung, an dem man seine Sensoren anschließen kann. Die Lösungen in [1, 2] sind einfache Mehrfachstecker, so dass man mehrere Sensoren an den TX(T) anschließen kann.

Am ftExtender lassen sich über einfache Flachbandkabel bis zu fünf I²C-Devices anschließen. Wem das nicht reicht, der kann mehrere ftExtender kaskadieren.

Logikspannung & Stromversorgung

Beim Anschluss eines I²C-Devices hat man immer wieder die gleichen Fragestellungen nach Stromversorgung und Logikspannung.

Die fischertechnik-typischen 9V können als Stromversorgung nicht direkt verwendet werden. Alle I²C-Devices benötigen entweder 3,3V oder 5V. Diese Spannungen lassen sich mit einem Linearregler und zwei Elkos einfach herstellen.

Das gleiche gilt für die Logikspannung: Auch hier gibt es I²C-Devices mit 5V und mit 3,3V. Ältere Sensoren, der TX und der ftDuino arbeiten mit 5V, der TXT sowie einige neuere Sensoren arbeiten mit 3,3V.

Mit so genannten Levelshiftern lassen sich Devices mit unterschiedlicher Logikspannung miteinander verbinden. Diese kann man mit zwei MOS-FETs und vier Widerständen selber bauen oder fertige kleine Platinen kaufen. Mit dem ersten I²C-Device macht das noch Spaß: Man sucht die passenden Beschaltungen für das Device heraus und adaptiert diese an seinen Sensor/Aktor. Man bekommt ein Plug & Play-Device mit einem 9V-Anschluss und wahlweise 6-poligem TX- oder 10-poligem TXT-Anschluss. Mit dem nächsten Device fängt man wieder von vorne an und baut die gleiche Beschaltung an den nächsten Baustein.

Der ftExtender hat einen Levelshifter und zwei Linearregler auf der Platine. Somit können problemlos 5 V- und 3,3 V-Devices ohne Zusatzbeschaltung über ein 6- oder 10-poliges Kabel angeschlossen werden. Als Stromversorgung stehen 250 mA für jedes angeschlossene Device zur Verfügung. Dies ist für den Betrieb von Sensoren völlig ausreichend. Beim Anschluss von Aktoren wie Motoren, Lampen und Servos sind 250 mA schnell erreicht, so dass diese eine eigene Stromversorgung bekommen sollten.

Anschlüsse

Der 6-polige Stecker (Abb. 2) ist kompatibel zum EXT-Anschluss des TX und des ftDuino, sodass die beiden Geräte direkt angeschlossen werden können. Für den Anschluss reicht ein 1:1-belegtes 6-poliges Kabel (Abb. 2).

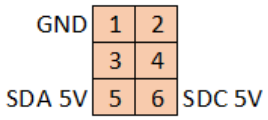


Abb. 2: Belegung des 6-poligen Steckers

Entsprechend ist der 10-polige Anschluss kompatibel zum TXT (Abb. 3). Hier können direkt der TXT oder vorhandene Sensoren mit 10-poligem Stecker angeschlossen werden. Für den Anschluss reicht ein 1:1-belegtes 10-poliges Kabel.

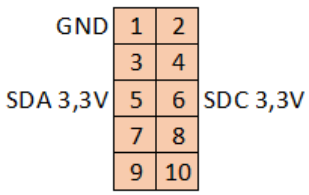


Abb. 3: Belegung des 10-poligen Steckers

Die drei 14-poligen Stecker sind für den kombinierten Einsatz von 3.3 V- und 5 V-Sensoren gedacht (Abb. 4). Dabei sind die Pins 1-6 in der Belegung mit dem TXT, die Pins 9-14 sind mit dem TX/ftDuino kompatibel. An Pin 7 und 8 stehen die Versorgungsspannungen 3,3 V und 5 V zur Verfügung.

GND	1	2	
	3	4	
SDA 3,3V	5	6	SDC 3,3V
3,3V	7	8	5V
GND	9	10	
	11	12	
SDA 5V	13	14	SDC 5V

Abb. 4: Belegung des 14-poligen Steckers

Der letzte Anschluss am ftExtender ist eine 9 V-Buchse. Hier wird die fischertechnik-Spannung angeschlossen, um die Versorgung der Levelshifter und Sensoren zu ermöglichen. Es ist nur die Buchse für +9V Gleichspannung vorhanden; den Minuspol gibt es bereits auf jedem Flachbandkabel.

Der ftExtender kann auch mit dem fischertechnik-Akkuset betrieben werden. Benötigen die I²C-Devices nur wenige mA, so ist das kein Problem. Bei höherem Stromverbrauch kann der eingebaute Spannungsregler jedoch schnell zur Spaßbremse werden.

Ein Linearregler setzt die Spannungsdifferenz zwischen Eingangs- und Ausgangsspannung in Wärme um. Schließt man an den 3,3V Linearregler einen Baustein mit 50mA Stromaufnahme an, so verbraucht das I²C-Device

$$50mA * 3{,}3V = 165mW$$

Am Linearregler muss von 8,6V Akkuspannung auf 3,3V heruntergeregelt werden:

$$50mA * (8{,}6V - 3{,}3V) = 265mW$$

Der Spannungsregler hat die 1,6fache Leistungsaufnahme des eingesetzten Sensors.

Wird der ftExtender nur als Mehrfachstecker eingesetzt, so wird die 9 V-Versorgung nicht benötigt. Ist die Bordspannung vorhanden, so zeigen zwei LEDs die Betriebsbereitschaft an.

Crimpen von Flachbandkabeln

Die Buchsen an den Flachbandkabeln lassen sich auch ohne Spezialwerkzeug crimpen.

An der Buchse ist Pin 1 mit einem Pfeil markiert. Für ein 1:1-Kabel wird hier die farblich markierte Ader des Flachbandkabels angelegt. Ob das Kabel links oder rechts aus der Buchse geführt wird ist völlig egal, die Pinbelegung am Kabel bleibt gleich. Wichtig ist, dass das Kabel möglichst senkrecht zur Buchse geführt wird, damit es zu keinem Kurzschluss kommt.

Danach wird entweder mit einer Wasserpumpenzange oder mit dem Schraubstock der Sicherungsclip auf die Buchse gedrückt. Dabei wird das Flachbandkabel automatisch gecrimpt. Bitte nicht mit zu viel Kraft, die Sicherungsclips brechen sonst gerne.

OLED Display am ftExtender

Als Beispiel haben wir ein OLED-Display an den ftExtender angeschlossen. Auf Basis des Treiberbausteins SSD1306 gibt es von Adafruit, Waveshare und vielen anderen Herstellern kleine 0,96“ OLED-Displays mit I²C Interface (Abb. 5, 6).

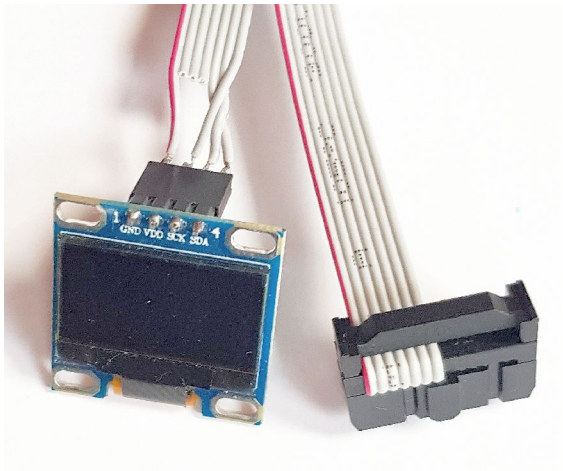


Abb. 5: OLED-Display am ftExtender

Alle Implementierungen haben einen 4-poligen Stecker mit GND, VCC/VDD, SDA und SCL. Sie funktionieren in der Regel sowohl mit 3,3V als auch mit 5V. Wer sich nicht sicher ist, startet erst einmal mit 3,3V.

Achtung, die Steckerbelegung des OLED-Displays kann je nach Hersteller abweichend sein! Einen ROBOPro-Treiber für das Display gibt es in [8] zum Download.

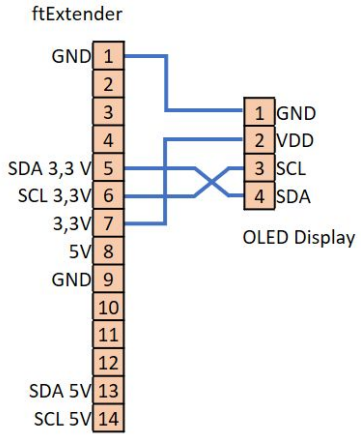


Abb. 6: Für den Anschluss eines OLED-Displays wird nur ein einfaches Kabel benötigt

Schaltung

Der ftExtender ist aus wenigen Standardschaltungen aufgebaut (Abb. 7). Der Schaltplan zeigt den geringen Aufwand und dokumentiert die Verschaltung der verschiedenen Stecker. Für die Anwendung des ftExtenders wird kein tiefes Verständnis der Funktion der Baugruppen benötigt.

Zwei Linearregler (IC1, IC2) erzeugen aus den eingespeisten 9 V die benötigten 3.3 V und 5 V für die Levelshifter und Versorgungsspannung der Sensoren.

Die beiden LEDs D1 und D2 zeigen die Betriebsbereitschaft der beiden Versorgungsspannungen an.

Die Levelshifter (bestehend aus den MOSFET Q1 und Q2) sind identisch mit dem Levelshifter aus [2]. Sie arbeiten zuverlässig mit den beiden I²C-Busfrequenzen 100 und 400kHz.

Produktion

Durch den Zugriff auf einen SMD-Bestückungsautomaten mit Reflow-Ofen haben wir uns für eine Bestückung in SMD-Technik entschieden. Dazu wurde das Platinenlayout für die Bestückung optimiert und ein Nutzen bestehend aus 20 ftExtender-Platinen erzeugt.

Auf alle Kontaktstellen, an die später ein Bauteil angelötet wird, wird zuerst mit einem Lotpastensieb Lotpaste aufgetragen. Die Lotpaste wird im letzten Produktionsschritt die eigentlichen Lötverbindungen herstellen.

Danach kommt der Bestückungsautomat PanteraXV von Essemtec [7] zum Einsatz. Aus dem Platinenlayout wird eine Pick & Place-Liste generiert, über die der Automat Bauteil für Bauteil mittels Unterdruck greift und an der richtigen Stelle auf den Nutzen klebt. Damit das Bauteil durch den Automaten leicht gegriffen werden kann, werden diese auf Gurten geliefert.

Für den eigentlichen Lötvorgang kommt der bestückte Nutzen in den Reflow-Ofen. Hier wird dieser aufgeheizt, so dass die Lötpaste flüssig wird und die Lötstellen sich von selbst herstellen.

Qualitätssicherung

Als Endtest wird an jedem 14-poligen Stecker ein 3,3V- und ein 5V-I²C-Gerät, an die 6-polige Buchse ebenfalls ein 5V-I²C-Gerät angeschlossen.

Die 10-polige Buchse wird mit einem TXT-Controller verbunden. Auf diesem läuft ein Testprogramm, das prüft, ob von allen angeschlossenen Geräten die ID ausgelesen werden kann.

Gehäuse

Der ftExtender ist fertig in einem 3D-gedruckten Gehäuse montiert (Abb. 8). Als Druckverfahren wird das selektive Laser-Sintern (SLS) genutzt, da dieses Verfahren sich sehr gut eignet, um Kleinserien zu produzieren. Wer einen 3D-Drucker sein eigen nennt kann das Gehäuse aber auch selber drucken. Die entsprechenden Dateien sind in [6] zu finden.

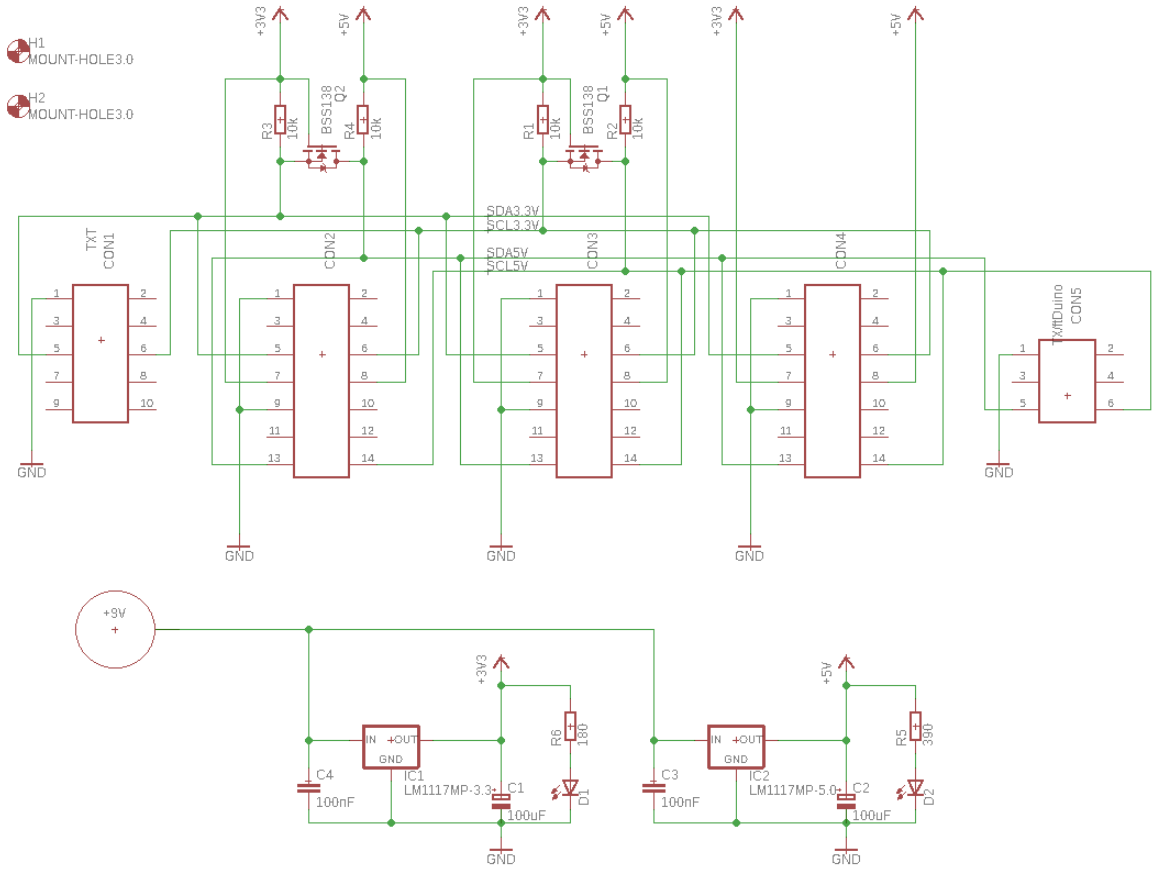


Abb. 7: Schaltplan des ftExtenders

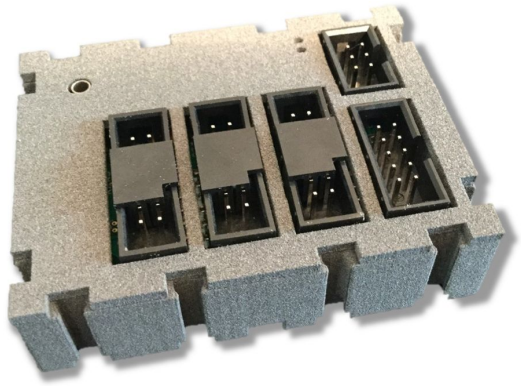


Abb. 8: ftExtender im 3D-Druck-Gehäuse

Quellen

[1] Dirk Fox: *I²C mit dem TX – Teil 2: LED-Display*, ftpedia 4/2012, S. 33.

[2] Christian Bergschneider, Stefan Fuss: *Ein universeller I²C-Adapter für den TX(T)*, ftpedia 4/2016, S. 72 ff.

[3] Till Harbaum: *ftDuino Bedienungs-anleitung*, github.org

[4] Christian Bergschneider, Stefan Fuss: *ftDuino_TXT*, github.org

[5] Georg Stiegler: *I²C mit dem TX – Teil 5: Multiplexer*. ft:pedia 2/2013, S. 50-52.

[6] Björn Gundermann, Rudenz Schulz, Christian Bergschneider, Stefan Fuss: *ftExtender Schaltplan, Layout und Gehäuse*, github.org

[7] Essemtec PanteraXV

[8] Christian Bergschneider, Stefan Fuss: *OLED Display mit ROBOPro*, github.org

Computing

Programmierung des TX-Controllers mit Python

Raphael Jacob

Im Sommer 2009 kam der TX-Controller auf den Markt. Unter Windows ist dieser mit ROBOPro sehr einfach anzusteuern, aber wenn ein Programm unter Linux oder gar einer anderen Architektur, wie zum Beispiel ARM (TXT-Controller) laufen soll, stößt man schnell an Grenzen. Das Problem lässt sich aber lösen – denn auch den TX-Controller kann man mit Python ansteuern.

Problemstellung

Alle Interfaces vor dem TX-Controller (im Folgenden kurz TX-C) und der TXT-Controller lassen sich auf allen Plattformen mit Python ansteuern, aber den TX-C kann man – zumindest „von Haus aus“ – nur unter Windows mit ROBOPro oder unter Verwendung einer bereits fertig kompilierten Bibliothek bspw. mit einem C-Programm ansteuern. Sollte es da nicht eine alternative Möglichkeit geben, auch den TX direkt mit Python zu steuern?

Idee

Daher begann ich mit Nachforschungen, welche Informationen über den TX von fischertechnik bereitgestellt werden oder von Fans entdeckt wurde. fischertechnik stellt auf seiner Homepage zu der oben genannten Bibliothek die Headerdateien und eine Anleitung sowie eine Beschreibung des Betriebssystems bereit [1].

Schnell stellte sich heraus, dass das Protokoll auf dem Erweiterungsbus (RS-485) identisch mit dem auf dem USB-Anschluss und einer Bluetooth-Verbindung ist. So boten sich drei weitere Quellen an:

- Christoph Nießen untersucht den Erweiterungsbus
- Thomas Kaiser analysiert noch etwas genauer die Kommandos des Protokolls
- Ad van der Weiden hat einen Konverter vom TX-C zur Robo-Interface-Extension gebaut

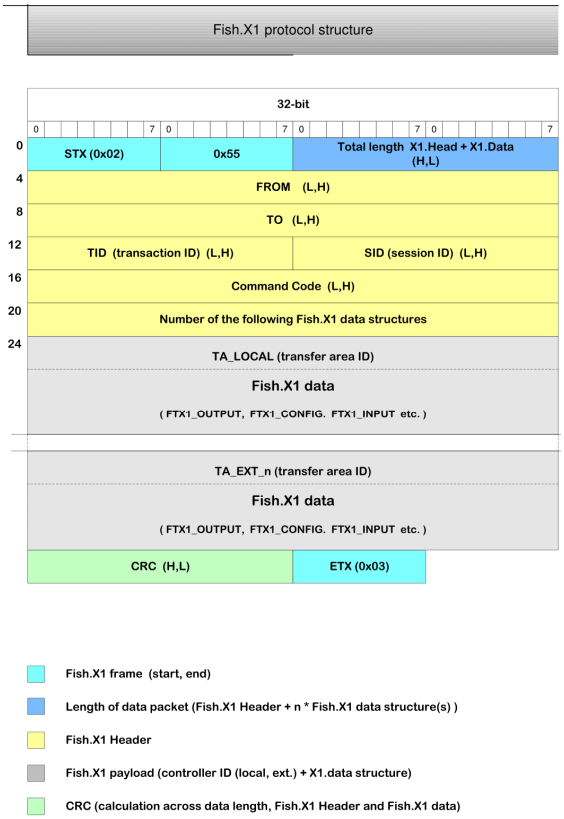


Abb. 1: Paketaufbau (Quelle: MSC, [1])

Analyse

Im zweiten Schritt habe ich den TX-C an einen Computer angeschlossen, ROBOPro geöffnet und den Interfacetest gestartet. Gleichzeitig habe ich im Hintergrund mit Wireshark die Kommunikation mitgeschnitten. Als Ergebnis hatte ich einen riesigen Berg von HEX-Zeichen. Zunächst habe ich das Ganze auf die Nutzdaten reduziert und dann mit meinen vier Quellen abgeglichen. MSC liefert für das X.1-Protokoll eine Grafik, in welcher der Paketaufbau sehr gut beschrieben ist (Abb. 1).

Offset	Length	Contents and meaning
0	1	STX (0x02) Marks the beginning of a Fish.X1 data packet with the following byte
1	1	0x55
2	2	Length (H,L) of a data packet Consists of the header length (20 bytes) and the payload length.
4	4	Bus address, 4 bytes (L,H), <FROM> (data packet transmitter) Possible values: BUS_ADR_WILDCARD=1, BUS_ADR_MASTER=2, BUS_ADR_SLAVE1=3 ... BUS_ADR_SLAVE8=10
8	4	Bus address, 4 bytes (L,H), <TO> (data packet receiver) Possible values, see above
12	2	Transaction ID (TID), 2 bytes (L,H) Each data packet is marked by the transmitter with a sequential number; the response to a request is returned by the message receiver with the same TID.
14	2	Session ID (SID), 2 bytes (L,H) The ROBO TX interface internally manages a session ID, which is reassigned every time there is a change in task from local to remote or remote to local. For example, the session ID is reassigned by the firmware if a timeout occurs during communication. The transmitter can then respond accordingly.
16	4	Fish.X1 command code (tele-code), 4 bytes (L,H) Defines a unique request or reply code. The command code thus describes the supplied data structure in the payload.
20	4	Number of subsequent Fish.X1 data structures, 4 bytes (L,H). (= n)
24	4	Transfer area ID, 4 bytes (L,H) The ID specifies for which ROBO TX Controller, and thus for which transfer area subsection, the subsequent data structure is addressed. The data structure specified via the command code is updated from the individual transfer area. The interfaces can be configured as master (always local) or as slaves (to a master). A corresponding ID is specified here via the transfer area ID: TA_LOCAL=0, TA_EXT_1=1, TA_EXT_2= 2, etc. The master (TA_LOCAL) manages the IO values of the individual slaves in its transfer area and can thereby assign the current IO data using the share memory ID.
28	m	Message payload, dependent upon the transmitted data structure. The payload, together with the share memory ID, can also have a length of 0, typically a request or a reply to a request. The structures defined in the header file ROBO_TX_FW.H are set as the payload. The command code identifies the supplied data structure.
24 + n*(4+m)	2	CRC, 2 bytes (H,L) The CRC is calculated from byte 2 (data length) to the end of the payload.
24 + 2 n*(4+m)	1	ETX (0x03) Marks the end of a Fish.X1 message

Abb. 2: Erläuterungen (Quelle: MSC)

Die Struktur der Pakete ist relativ einfach:

- Jedes Paket beginnt mit einem Präambel bestehend aus zwei Bytes (0x02 und 0x55)
- Jedes Paket endet mit 0x03
- Jedes Paket besteht aus Präambel, Länge, Header, Daten, einer Prüfsumme und dem Schluss
- Die Länge ist die Summe der Anzahl Bytes im Header (20) und der Anzahl Bytes in den Daten
- Der Header enthält Sender, Empfänger, die Transaction-ID, die Session-ID, den Command Code und die Anzahl übertragener Datenblöcke
- Die Sender- und Empfängeradresse sind bei der Übertragung via USB fest; bei Paketen vom PC zum TX-C ist der Sender 2 und der Empfänger 1
- Die Transaction-ID erhöht sich bei jedem gesendeten Paket
- Die Session-ID kann am Anfang auf 0 gesetzt werden, bis der TX-C eine neue Session-ID setzt
- Die Anzahl der übertragenen Datenblöcke gibt an, an wie viele TX-Cs Daten übermittelt werden
- Die Prüfsumme ist laut Spezifikation eine CRC-Prüfsumme, die aus allen Bytes von der Länge bis zum Ende des Datenpakets berechnet wird

Der Command Code war das größte Problem. Dieser beschreibt, was hier überhaupt übertragen wird und somit, wie die Struktur in den Daten aussieht. In den offiziellen Quellen wird dies nicht beschrieben, und in den Analysen zum Erweiterungsbus fehlen leider einige benötigte Codes. Beim Senden ist der Code eine Zahl (hier: 1, 2, 5, 6, 7) und in umgekehrter Richtung ist es diese Zahl plus 100.

Die Daten selbst bestehen außer bei I²C aus einem Block pro angesteuerter Extension. Innerhalb eines Blocks kommen erst die Transfer Area ID und dann die Nutzdaten. Die Transfer Area ID ist für den Master 0 und zählt dann hoch bis 8 für die achte Extension. Der Aufbau der Nutzdaten ist bei jedem Command Code unterschiedlich, darauf gehe ich gleich ein.

Schließlich konnte ich die Datenpakete genauer analysieren und für einige Pakete herausfinden, wie die Daten aufgebaut sind. Wie auch von Thomas Kaiser festgestellt, ist die Prüfsumme kein CRC, sondern eine

Subtraktion aller Bytes von der Länge bis zum Ende der Daten von der Zahl 0x00.

Leider konnte ich nicht alles identifizieren, weshalb ich fischertechnik um Mithilfe bat. fischertechnik stellte mir alle erforderlichen Daten zur Verfügung – ein großes Dankeschön an Herrn Knecht, ohne den diese Herausforderung nicht zu meistern gewesen wäre. Leider waren auch diese Daten nicht zu 100% aktuell, aber irgendwann konnte ich das Puzzle lösen. Jetzt wusste ich, wann welches Paket mit welchem Command Code gesendet werden muss und wie die Daten darin aufgebaut sein müssen.

Command Codes

Durch die Analyse der Daten, die zwischen ROBOPro und TX-C ausgetauscht werden, fand ich heraus, dass die Command Codes 1, 2, 5, 6, 7, 101, 102, 105, 106 und 107 genutzt werden. Folgende Übersicht beschreibt die Bedeutung dieser Codes:

Echo (1)

- Enthält keine Daten
- Wird beim Öffnen der Verbindung gesendet

Echo Reply (101)

- Antwort auf Echo
- Enthält keine Daten
- Wird ignoriert

Remote IO (2)

- Sendet Ausgangsdaten und Daten zur Steuerung der Zähler an die TX-Cs und dient der Abfrage von Eingangsdaten
- Sendet einen Datenblock an den Master und jede verbundene Extension
- Die Counter Reset ID wird bei jedem Zurücksetzten des Zählers auf 0 um eins erhöht
- In das Feld Motor Sync wird die ID des Motors geschrieben, mit dem synchronisiert wird (0: Synchronisierung deaktiviert)
- Die Motor Command ID muss bei jeder Änderung der Entfernung oder des Synchronisationspartners für einen Motor um eins erhöht werden; anschließend bewegt sich der Motor erneut um die vorgegebene Strecke

	4 Bytes	
0	Transfer Area ID	
4	Counter Reset ID 1	Counter Reset ID 2
8	Counter Reset ID 3	Counter Reset ID 4
12	Motor Sync 1 / Motor Sync 2	Motor Sync 3 / Motor Sync 4
16	Output Duty 1	Output Duty 2
20	Output Duty 3	Output Duty 4
24	Output Duty 5	Output Duty 6
28	Output Duty 7	Output Duty 8
32	Motor Distance 1	Motor Distance 2
36	Motor Distance 3	Motor Distance 4
40	Motor Command ID 1	Motor Command ID 2
44	Motor Command ID 3	Motor Command ID 4

Abb. 3: Byteschema Remote IO Send

Remote IO Reply (102)

- Antwort auf Remote IO
- Enthält die Eingangswerte, Informationen der Zähler und Informationen zur erweiterten Motorsteuerung
- Enthält einen Datenblock an den Master und jede verbundene Extension
- Die Felder Input Value enthalten den fertig berechneten Eingangswert. Es muss keine Umwandlung für die verschiedenen Modi durchgeführt werden
- In den Feldern Counter State ist gespeichert, ob der entsprechende Zählereingang geöffnet (0x01) oder geschlossen (0x00) ist
- Counter Count ist der Wert des Zählers
- Aus der Counter Reset ID des zuletzt fertiggestellten Zählerresets, ergibt sich, ob der letzte Reset bereits durchgeführt wurde
- Aus der Motor Command ID des zuletzt fertiggestellten Motorkommandos ergibt sich, ob das letzte Kommando bereits fertiggestellt wurde

	4 Bytes			
	Transfer Area ID			
0	Input Value 1		Input Value 2	
4	Input Value 3		Input Value 4	
8	Input Value 5		Input Value 6	
12	Input Value 7		Input Value 8	
16	Counter 1 State	Counter 2 State	Counter 3 State	Counter 4 State
20	Counter 1 Count		Counter 2 Count	
24	Counter 3 Count		Counter 4 Count	
28	Counter Reset ID 1		Counter Reset ID 2	
32	Counter Reset ID 3		Counter Reset ID 4	
36	Motor Command ID 1		Motor Command ID 2	
40	Motor Command ID 3		Motor Command ID 4	
44	Empty			

Abb. 4: Byteschema Remote IO Reply

Remote Config Write (5)

- Senden von der Eingangskonfiguration an die TX-Cs
- Sendet einen Datenblock an den Master und jede verbundene Extension
- In die Felder Input Config wird je nach Modus ein Byte geschrieben. Soll ein Analogwert gemessen werden, so muss der Wert mit 128 multipliziert werden:

Modus	Digital	Analog
Spannung	0x00	0x80
Widerstand 5k	0x01	0x81
Widerstand 15k	0x02	0x82
Ultraschall	0x03	0x83

	4 Bytes			
0	Transfer Area ID			
4	0x01	0x01	0x01	0x01
8	Input 1 Config	Input 2 Config	Input 3 Config	Input 4 Config
12	Input 5 Config	Input 6 Config	Input 7 Config	Input 8 Config
16	0x01	0x01	0x01	0x01
20	0x00	0x00	0x00	0x00
24	0x00	0x00	0x00	0x00
28	0x00	0x00	0x00	0x00
32	0x00	0x00	0x00	0x00

Abb. 5: Byteschema Remote Config Write

Remote Config Write Reply (105)

- Antwort auf Remote Config Write
- Enthält keine Daten
- Wird ignoriert

Info (6)

- Abfrage von Metadaten von Master und Extensionen
- Sendet einen leeren Datenblock an den Master und jede verbundene Extension

Info Reply (106)

- Antwort auf Info
- Enthält für den Master und jede angeschlossene Extension einen Datenblock
- Bei der Bluetooth MAC-Adresse sind die Doppelpunkte bereits in den Daten enthalten

	4 Bytes		
0	Transfer Area ID		
4	Name des TX-Controller		
8			
12			
16			
20	Name des TX-Controller	Bluetooth MAC-Adresse	
24	Bluetooth MAC-Adresse		
28			
32			
36	Bluetooth MAC-Adresse	Empty	
40	Empty		
44			
48			
52			
56	Empty	Version 1	Version 2
60	Empty		
64			

Abb. 6: Byteschema Info Write

State (7)

- Abfrage nach verbundener Extensionen
- Sendet einen leeren Datenblock an den Master

State Reply(107)

- Antwort auf State
- Enthält einen Datenblock vom Master
- Die 8 Bytes Extension State geben an, ob die entsprechende Extension verbunden ist. Ist das Byte 0, so ist diese getrennt; ist es 1, so ist diese Extension verbunden

	4 Bytes			
0	Transfer Area ID			
4	Extension 1 State	Extension 2 State	Extension 3 State	Extension 4 State
8	Extension 5 State	Extension 6 State	Extension 7 State	Extension 8 State
12	Empty			
16				
20				
24				

Abb. 7: Byteschema State Reply

Software

Die Entscheidung, die Software in Python zu entwickeln, war eigentlich schon vorher klar, da die Community Firmware von Anfang an auf Python gesetzt hat und ich die Kompatibilität dazu wahren wollte. Um die Ein- und Ausgänge des TXT-Controllers anzusteuern hat Torsten Stuehn die Bibliothek *ftrobopy* geschrieben. Um auch hier möglichst kompatibel zu bleiben hatte ich mir vorgenommen, auf der höchsten Abstraktionsebene die Funktionsaufrufe gleich zu gestalten. In Anlehnung an *ftrobopy* habe ich die Software *fttxpy* getauft.

Abstraktionsebenen

Um den Überblick nicht zu verlieren habe ich die Software in fünf Abstraktionsebenen unterteilt:

TXSerial: Hier werden die serielle Verbindung und das X.1-Protokoll abstrahiert. Nach dem Aufbau der Verbindung können hier Pakete gesendet und Empfangen werden. Die Informationen werden in einem einheitlichen Datenformat gespeichert, welches Sender, Empfänger, Command Code und die Datenblöcke für den Master und die Extensionen enthält.

ftTX: Hier werden die Daten aller Ein- und Ausgänge verwaltet. Ein Hintergrundprozess kümmert sich um das Aufrechterhalten der Verbindung, indem möglichst dauerhaft alle Ausgangsdaten an den TX-C gesendet und alle Eingangsdaten eingelesen werden. Falls eine Eingangskonfiguration geändert wurde, wird diese automatisch an den TX-C übertragen. Des Weiteren werden hier die verbundenen Extensionen verwaltet, um zum Beispiel die Firmwareversion und den Namen der Extension auslesen zu können.

Um die gespeicherten Daten leichter editieren zu können, werden hier Funktionen bereitgestellt, mit welchen beispielsweise die Motorgeschwindigkeit gesetzt oder ein Eingang konfiguriert werden kann. Diese Funktionen werden in den folgenden Ebenen genutzt und benutzerfreundlicher bereitgestellt.

fttxpy: Dies ist die erste Ebene, mit der der Benutzer in Berührung kommt; sie stellt den Verbund aus dem Master und den Extensionen dar. Um die Bibliothek zu benutzen ruft der Benutzer diese Klasse auf. Dabei wird automatisch der erste verbundene TX-C ausgewählt. Nach dem Aufruf der Klasse werden automatisch der Name und die Firmwareversionen des Masters sowie aller verbundener Extensionen ausgegeben.

robotx: Diese Klasse stellt einen einzelnen TX-C im Verbund dar. Hier stehen Klassen zur Abstraktion verschiedener Peripheriegeräte wie Motoren, Lampen, Tastern und anderen Sensoren bereit. Die Funktion *robotx* wird aufgerufen, um einen bestimmten TX-C aus dem Verbund anzusprechen. Ohne Argumente erhält man ein Objekt des Masters, mit einer Zahl als Argument erhält man die entsprechende Extension.

Die *Sensoren* und *Aktoren*: Die wohl wichtigsten Klassen sind für die verschiedenen Sensoren und Aktoren verantwortlich. Eine genaue Beschreibung zu diesen Klassen erfolgt weiter unten.

Programmierung

Zur Benutzung der Software braucht ihr einen Rechner mit Linux und Python 3 oder einen TXT-Controller mit der Community Firmware. Auf der CFW ist *fttxpy* bereits vorinstalliert, auf anderen Geräten muss man sich *fttxpy* herunterladen und dann wie jedes andere Python Modul installieren. Dazu gibt man ein:

```
python3 setup.py install
```

Für die ersten Versuche öffnet man am besten den interaktiven Python Interpreter. Um die Bibliothek zu importieren und die Verbindung aufzubauen gibt man folgendes ein:

```
from fttxpy import fttxpy
tx = fttxpy()
```

Wenn nur ein TX-C ohne Extensionen verbunden ist, so steht dort zum Beispiel:

```
Found 1 TX-Controller(s).
Automatically selecting the first
one!
Connected to ROBO TX-308 Version
1.30
```

Sind Extensionen verbunden, dann steht dort zusätzlich:

```
Found Extensions: Ext 1 (ROBO TX-
309, 1.30), Ext 2 (ROBO TX-309,
1.30)
```

Um nun auf den Master oder eine Extension zuzugreifen gibt man folgendes ein:

```
master = tx.robotx()
ext1 = tx.robotx(1)
ext2 = tx.robotx(2)
...
```

Die Objekte `master, ext1, …` bieten nun die Klassen zur Ansteuerung der Ein- und Ausgänge. Für jeden TX-C können mehrere Objekte erstellt werden, für die Ein- und Ausgänge jedoch nicht. Wenn man ein Ein- bzw. Ausgangsobjekt nicht mehr benötigt, so muss dieses mit `del(<Objekt>)` gelöscht werden. Danach kann ein neues Objekt erstellt werden.

Motor

Ein Motor kann viel mehr, als viele denken. Er kann nicht nur in der Richtung und Geschwindigkeit gesteuert werden, sondern auch definierte Distanzen fahren und zu einem anderen Motor synchronisiert werden.

Zunächst möchten wir einen Motor nur in Richtung und Geschwindigkeit variieren:

```
# Zunaechst erzeugen wir ein
Motorobjekt
m1 = master.motor(1)
# warten bis Nutzer Enter drueckt
input()
# Geschwindigkeit 128 rechts
m1.setSpeed(128)
input()
# Geschwindigkeit 512 rechts
m1.setSpeed(512)
input()
# Geschwindigkeit 128 links
m1.setSpeed(-128)
input()
# Geschwindigkeit 512 links
m1.setSpeed(-512)
input()
# Stop
m1.setSpeed(0)
# Oder
m1.stop()
```

Eine vorgegebene Distanz zu fahren und dann automatisch zu stoppen ist auch ganz einfach:

```
# Distanz auf 1000 setzen
m1.setDistance(1000)
# Geschwindigkeit auf 300
m1.setSpeed(300)
# Abfragen, ob Position erreicht
ist
while not m1.finished():
      pass
# Geschwindigkeit und Distanz auf
0 setzen
m1.stop()
print("Fertig")
```

Die Abfrage `finished()` gibt `True` oder `False` zurück. Wenn die Position erreicht ist, so ist der Wert `True`, andernfalls `False`.

Der Befehl `stop()` setzt Geschwindigkeit und Distanz auf 0. Will man den Motor, bevor er sein Ziel erreicht hat, anhalten und dann wieder weiterfahren lassen, ohne diese Distanzmessung zu unterbrechen, sollte man `setSpeed(0)` verwenden.

Die Kunst in der Robotik ist es, einen fahrenden Roboter exakt geradeaus fahren zu lassen. Dies ist auch mit der `setDistance`-Funktion möglich:

```
# Zunaechst erzeugen wir zwei
Motorobjekte
m1 = master.motor(1)
m2 = master.motor(2)
# Distanz für M1: 1000
m1.setDistance(1000)
# Distanz für M2: 1000
# M2 zu M1 synchronisieren
m2.setDistance(1000, m1)
# Geschwindigkeit beide 512
m1.setSpeed(512)
m2.setSpeed(512)
# Warten auf Position erreicht
while not m1.finished() and not
m2.finished():
      pass
print("Fertig")
m1.stop()
```

```
m2.stop()
```

Der Befehl `stop()` deaktiviert die Synchronisierung. Will man die Motoren erneut synchronisieren, so muss man mit `setDistance()` erneut den Synchronisationspartner übergeben. Wie auch in ROBOPro kann ein Motor nur zu einem anderen Motor am gleichen TX-C synchronisiert werden.

Ausgang

Um zum Beispiel Lampen, Magnetventile oder Summer anzusteuern genügt ein ein „O“-Ausgang. Dieser lässt sich nur in der Intensität, aber nicht in der Richtung steuern:

```
# Ausgangsobjekt erzeugen
o5 = master.output(5)
# Intensitaet 128
o5.setLevel(128)
input()
# Intensitaet 255
o5.setLevel(255)
input()
# Ausschalten
o5.setLevel(0)
input()
```

Taster

Der wohl wichtigste, aber auch simpelste Sensor ist der Taster. So wird er eingelesen:

```
from time import sleep
# Eingangsobjekt generieren
i1 = master.input(1)
while True:
      # Taster ein Mal pro Sekunde
einlesen und ausgeben
      print(i1.state())
      sleep(1)
```

Widerstände

Der Helligkeitssensor und der NTC-Temperatursensor sind Widerstände. Dieses können sehr einfach ausgewertet werden. Zudem kann die Temperatur des NTC-Temperatursensors in Grad Celsius umgewandelt werden:

```
# Eingangsobjekt generieren
i2 = master.resistor(2)
# Rohwert auslesen
print(i2.value())
# Temperatur in Grad Celsius
ausgeben
print(str(i2.ntcTemperature()),
"Grad Celsius")
```

Spannung

Will man nun einen Farbsensor verwenden oder die Akkuspannung prüfen geht man wie folgt vor:

```
# Eingangsobjekt generieren
i3 = master.colorsensor(3)
# Spannung in Millivolt ausgeben
print(str(i3.voltage()), "mV")
# Farbe ausgeben - experimentell!
print(i3.color())
```

Der Aufruf `color()` ist sehr experimentell. Ich habe die Werte nur aus *ftrobopy* kopiert. Ob die Werte auch bei euch funktionieren, weiß ich nicht. Aufgabe für die Profis: Schreibt euch eine Funktion um die Farbe unter euren Bedingungen auszugeben.

Spursensor

Beim RoboCup Junior muss ein Roboter in der Disziplin Rescue Line u. a. einer Linie folgen. fischertechnik bietet dazu den Spursensor an. Dieser liefert am Ausgang 0V oder 9V. Die Auswertung ist ganz einfach:

```
# Eingangsobjekt generieren
i4 = master.trailfollower(4)
# Aktuellen Zustand ausgeben
print(i4.state())
```

Um die Spur auszuwerten muss man natürlich zwei Eingangsobjekte für die beiden Eingänge erzeugen und mit ein wenig Logik auswerten.

Distanzsensor

Will man eine Distanz mit dem Ultraschall-Abstandssensor von fischertechnik messen, muss man wie folgt vorgehen:

```
# Eingangsobjekt generieren
i5 = master.ultrasonic(5)
# Aktuelle Distanz ausgeben
print(i5.distance())
```

Schlusswort

Ich weiß nicht, für welchen Anwendungszweck das X.1-Protokoll ursprünglich entwickelt wurde, aber für eine möglichst schnelle Datenübertragung zwischen einem PC und einem TX mit bis zu neun Erweiterungen ist es nicht geeignet. An vielen Stellen wurde sehr viel Platz verschwendet, angefangen bei der Sender- und Empfängeradresse, die mit vier Byte völlig überdimensioniert ist (vier Bit hätten genügt), über den Command Code mit vier Bytes (es gibt noch ein paar Codes mehr, als hier dargestellt, aber ein Byte hätte ausgereicht) und schließlich der Anzahl Datenstrukturen mit vier Bytes (ein Master und acht Erweiterungen, also maximal neun Datenstrukturen; dafür wären fünf Bit ausreichend gewesen) sieht man, wie ineffizient alleine der Paketheader ist, der bei jedem Paket übertragen werden muss. An vielen Stellen in den Paketen hätte man auch viele ungenutzte Bytes entfernen können. Das von fischertechnik versprochene 10 ms-Raster ist mit nur fünf Erweiterungen nicht mehr zu halten. Hier sind pro Buszyklus 14 ms erforderlich [2]; Für acht Erweiterungen benötigt das Protokoll 22,4 ms.

Bei der Analyse ist mir irgendwann aufgefallen, dass man die Transaction-ID frei wählen kann. Hier darf auch während der Verbindung frei „gesprungen“ werden, also hätte man diese zwei Bytes auch einsparen können.

Egal, ob man den Distanzsensor als digital oder analog abruft, Werte liefert er immer, und wenn man den Sensor während der Verbindung trennt, bleibt der alte Wert erhalten.

Quellen

[1] fischertechnik GmbH: *Zusammenstellung der wichtigsten Dateien*.

[2] Ad van der Weiden: *Logic analyser trace of the RS485 communication between a Master and 5 Slaves*.