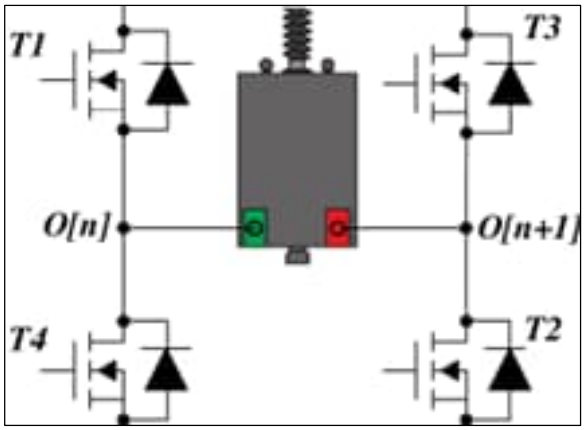
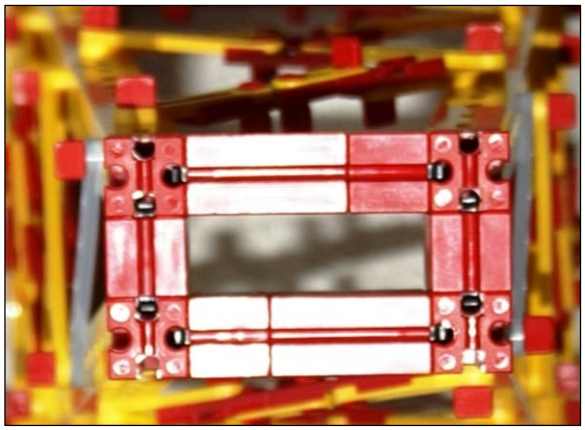
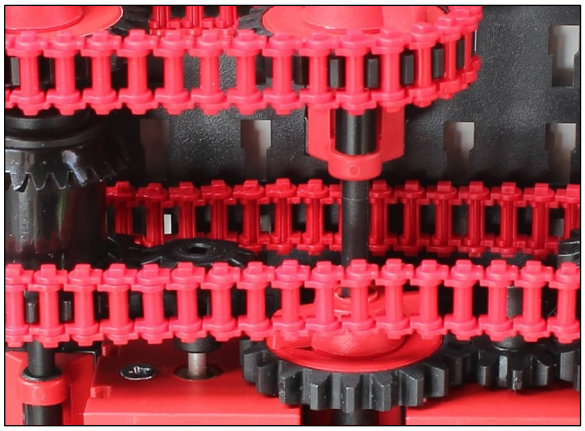
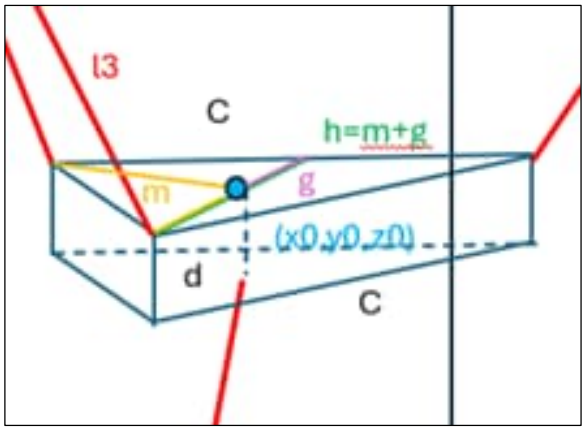
ft:pedia

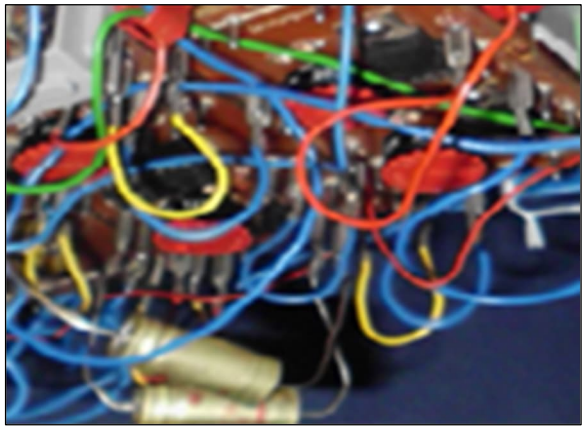
Heft 1/2025

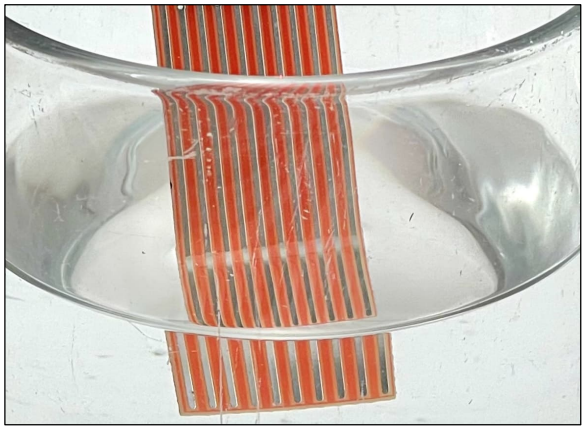












Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

Zum Geburtstag

Dirk Fox, Stefan Falk

Vor sechzig Jahren erblickte fischertechnik das Licht der Welt: Ein grauer Baustein mit fünf Nuten und einem Zapfen zog 1965 in die ersten Kinderzimmer ein. Ein revolutionäres Konzept, wie sich zeigte: In Publikationen jener Zeit überschlagen sich Pädagogen vor Begeisterung. An zahlreichen Hochschulen wurden fischertechnik-Modelle entwickelt, um Grundlagen der Mechanik, Statik und Elektronik zu vermitteln – für Schüler, Studenten und Berufstätige; nachzulesen z. B. in den legendären „Hobby"-Bänden aus den 70er Jahren. Wenig später war fischertechnik das erste Konstruktionssystem, dessen Modelle mit Home-Computern gesteuert wurden.

Artur Fischer erfand mit fischertechnik das erste MINT-Spielzeug. Und wohl auch das erfolgreichste – nicht wirtschaftlich, aber ganz sicher gesellschaftlich. Denn mit fischertechnik wurde Technikwissen „kinderzimmertauglich". Wer Ingenieure der Generation, die Deutschland zum Exportweltmeister gemacht hat (und sich gerade in den Ruhestand verabschiedet), nach ihren ersten Technik-Erfahrungen fragt, bekommt überraschend häufig dieselbe Antwort: fischertechnik.

Nicht nur für Entwicklungsingenieure, sondern auch für Unternehmer und Konzernvorstände war Artur Fischers Erfindung ein „Game Changer", wie man heute formulieren würde. Vor 15 Jahren veröffentlichte kein geringerer als Professor Dr.-Ing. Dr. Ing. E. h. Dr. h. c. Ekkehard D. Schulz, einigen sicher noch als Vorstandsvorsitzender der ThyssenKrupp AG bekannt (1999-2011), ein Büchlein mit dem Titel „55 Gründe, Ingenieur zu werden – über den schönsten Beruf der Welt". Eine Liebeserklärung an einen Beruf, der nicht nur faszinierend, sondern für die Menschheit überlebensnotwendig sein dürfte, wie Schulz selbst in der Einleitung begründet: „Die Probleme der Menschheit wie Klimaschutz und Ressourcenknappheit werden am Ende nicht auf den Gipfeltreffen der Regierungschefs gelöst, sondern im Labor. Ingenieure sind es, die die Welt verändern."

Beim 38sten Grund („… weil Ingenieure Kinder glücklich machen") geht dann dem fischertechniker das Herz auf. Zitat:

„Als ideales Beispiel für die Verbindung von Spiel und Ingenieurswesen dient das System fischertechnik. Der berühmte schwäbische Erfinder Artur Fischer, der Vater des Dübels, entwickelte den Konstruktionsbaukasten 1964 zunächst als Weihnachtsgeschenk für Geschäftspartner. Es ermöglicht auf Basis des grauen Grundelements so gut wie jede Konstruktion, einschließlich motorengetriebener Maschinen. fischertechnik-Bausätze werden oft im Schulunterricht eingesetzt. Viele Kinder entdecken so zum ersten Mal ihr Talent als Ingenieur."

Dem ist auch anlässlich des 60. Geburtstags nichts hinzuzufügen. Bis auf die Bitte an fischertechnik, niemals Hand an dieses Vermächtnis zu legen.

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

Zum Geburtstag 2
Hitachi KH1000 im Maßstab 1:15,5 4
Mars-Rover: Modell und Steuerung 16
Seilroboter mit fischertechnik 24
Großprojekt Seilbahn (Teil 11): Die Zukunft 32
Getriebe für Mecanum-Räder 36
Sensor-Adaptermodul 41
BT Smart Controller – der Universelle 50
Modelle steuern mit Python – auf dem Raspberry Pi 56
Der ft-RPI-sa – ein moderner BASIC-Controller für fischertechnik (2) 60

Termine

Was?	Wann?	Wo?
Maker Faire Ruhr	29.-30.03.25	DASA Dortmund

Impressum

http://www.ftpedia.de

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

Autoren: Florian Bauer, Axel Chobe, Arnoud van Delden, Stefan Falk, Dirk Fox, Peter Habermehl, Werner Hasselberg, Peter Krijnen, Robert Lippmann, Tilo Rust.

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

Modell

Hitachi KH1000 im Maßstab 1:15,5

Peter Krijnen

Im April 2015 konnte ich mehrere Broschüren der Marke Hitachi kaufen [1]. Genauer gesagt: die über die Raupenkrane der HK-Serie. Nach so vielen Jahren war ich endlich im Besitz vieler Daten über diese Krane. Einige Monate später entdeckte ich, dass detaillierte Zeichnungen (DWG + DXF) auch von der Hitachi/Sumutomi-Website [2] heruntergeladen werden konnten.

Abb. 1: Basis des KH1000 im Maßstab 15:5

Ich konnte mir diese Zeichnungen dann in AutoCAD anschauen und sogar anpassen. Mit den zahlreichen Messmöglichkeiten in AutoCAD gelang es mir, die Abmessungen aller Teile zu ermitteln. Obwohl ich ursprünglich den KH500 bauen wollte, habe ich mich nach Anschauen und Messung für den KH1000 entschieden (Abb. 1). Im 15-mm-Raster von fischertechnik klappte das deutlich besser.

Wichtig für mich war, dass ich die Bauplatte 90×15 als Raupenplatte nutzen wollte. Bei einer Breite von 1270 mm würde ich den Maßstab auf 1:14,1 einschätzen. Allerdings war auch die Dimensionierung des Auslegers von Bedeutung. Der Querschnitt des Auslegers betrug letztlich 150×150mm.

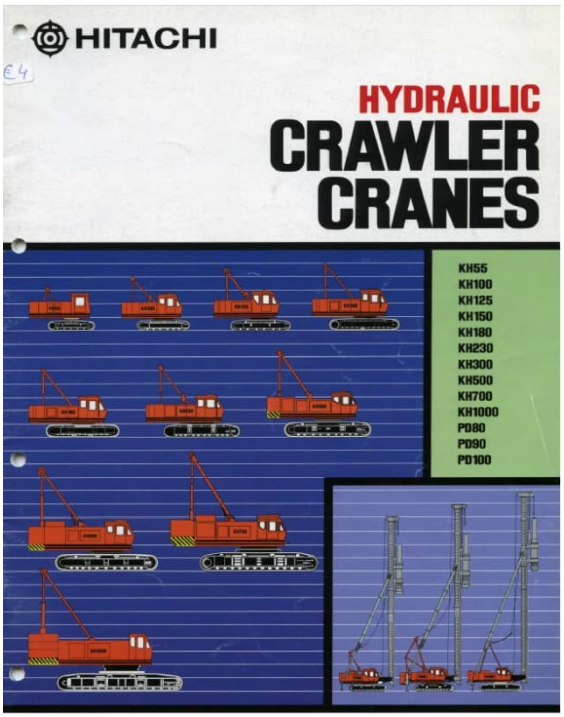


Abb. 2: Broschüre der HITACHI Raupenkräne der KH-Serie, die ich während der Modelshow Europe gefunden habe

Nach der Umrechnung aller wichtigen Maße bin ich auf den Maßstab 1:15,5 gekommen. Je nachdem, welche Maße mit den fischertechnik-Bauteilen erreichbar waren, musste ich hier und da etwas davon abweichen. (Kann man das sehen?)

Wie aus dem Datenblatt hervorgeht, sollen zwei Serien gebaut worden sein: Serie 1 für 130 t und Serie 2 für 180 t. Bei meiner Suche im WWW bin ich auch auf mehrere

KH1000-3 gestoßen, die maximal 200 t bewältigen können.

Abb. 3: Das KH1000-Datenblatt aus dem WWW heruntergeladen [2]

Der KH1000 wurde wahrscheinlich irgendwann zwischen 1987 und 2005 produziert.

Das Modell

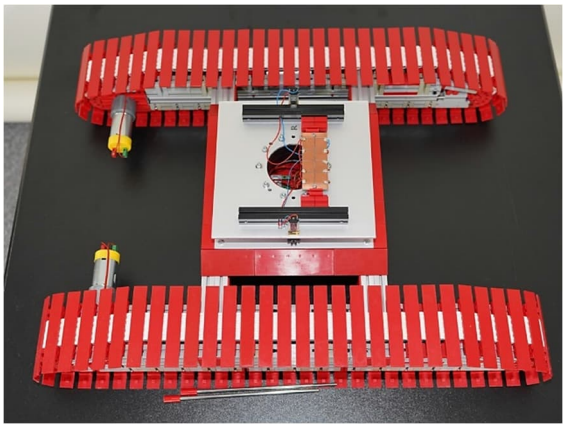


Abb. 4: Der komplette Unterbau: Grundrahmen und die beiden Raupenschiffe.

Die Raupen sind im gespannten Zustand 600 mm lang, 90 mm breit und 95 mm hoch. Die Gesamtbreite habe ich bei 450 mm belassen. Im Original lässt sich dieser allerdings zwischen 7070 und 5900 mm einstellen. Um es mir nicht zu schwer zu machen, habe ich darauf verzichtet.

Angetrieben wird der Raupe von einem 1:312-Igarachi-Getriebemotor, den ich bei Conrad (244040) gekauft habe. Das Spannen der Raupen erfolgt mittels Schnecke m1 und Schneckenmutter, welche von einem 1:125-Motor angetrieben wird (Conrad, 1711490).

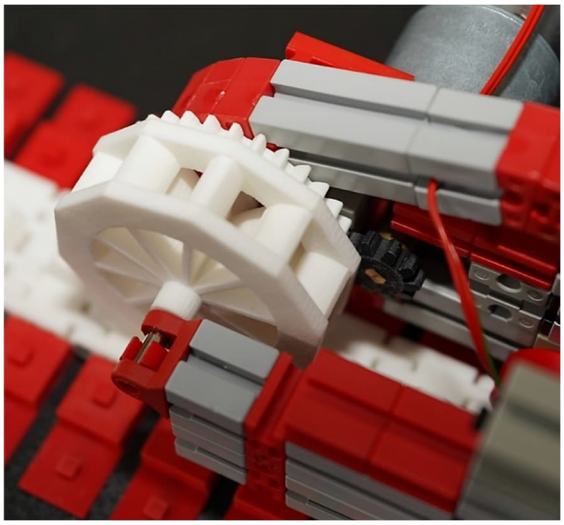


Abb. 5: Antrieb des Raupenantriebsrades: 1:312-Motor und ein modifiziertes Rastzahnrad Z10 auf dem Z25 des Rades

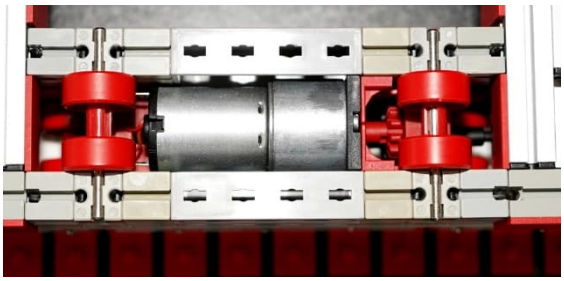


Abb. 6: Antrieb des Spannmechanismus: 1:125er Motor, der... (siehe Abb. 7)

Die Drehbühne ist auf Abb. 4 zu sehen. Sie besteht aus 2×3 mm dicken Aluminiumplatten 180×180 mm, POM-Ringen und zwei Wälzlagern Typ AXK100135 (Abb. 8).

Der Zahnkranz besteht aus 96 Kettengliedern (36263), die mit 4 Förderkettengliedern (37192) am äußeren POM-Ring befes-

tigt werden und somit einen Z100-Zahnkranz bilden.

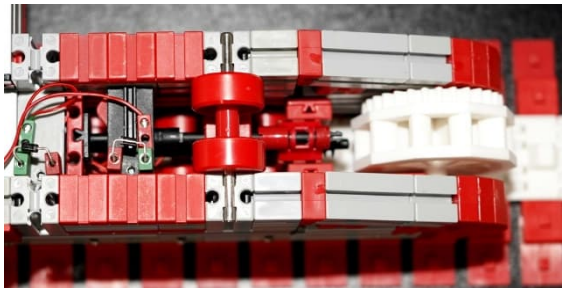


Abb. 7: ...das Spannrad mittels m1-Schnecke bewegt. Auf der linken Seite sind die beiden Endschalter in Reihe mit dem Motor geschaltet; die Betätigung erfolgt über eine Rastachse mit Platte (130593) von der Schneckenmutter

Die Drehung wird durch zwei 1:1000-Getriebemotoren vom Typ *Sol Expert G1000-12V* (Conrad 1289387) erreicht. Auf der Drehbank habe ich zwei Achsverlängerungen gefertigt, auf welche ich je ein Rastritzel 10 geschoben habe.

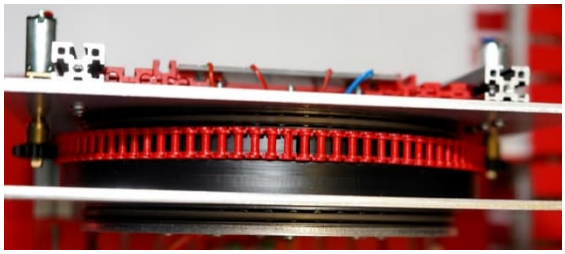


Abb. 8: Nicht wirklich deutlich zu erkennen, aber hier sind auf jeden Fall zwei 135-mm-Rollenlager verbaut. 100 Kettenglieder ergeben ein Z100.

Da ich den Oberbau und den Unterbau getrennt halten wollte, musste ich eine praktische Möglichkeit finden, beides miteinander zu verbinden. Dazu habe ich zwei 120 mm lange Makerbeam-Profile auf die Oberplatte des Drehtellers geschraubt (Abb. 4). Dahinein habe ich zwei Löcher von 4,1 mm gebohrt. Mit zwei Achsen kann ich nun beide Teile miteinander verbinden.

Da sich die RC-Anlage im Kranhaus selbst befindet, musste ich noch ein Kontaktsystem herstellen, um die elektrische Verbindung mit der Unterkonstruktion zu realisieren. Dies geschieht mittels acht Federkontakten (31306) unter dem Kranhaus, die den Kontakt zur Platine herstellen (Abb. 9), welche wiederum auf dem Drehteller befestigt ist (Abb. 4).

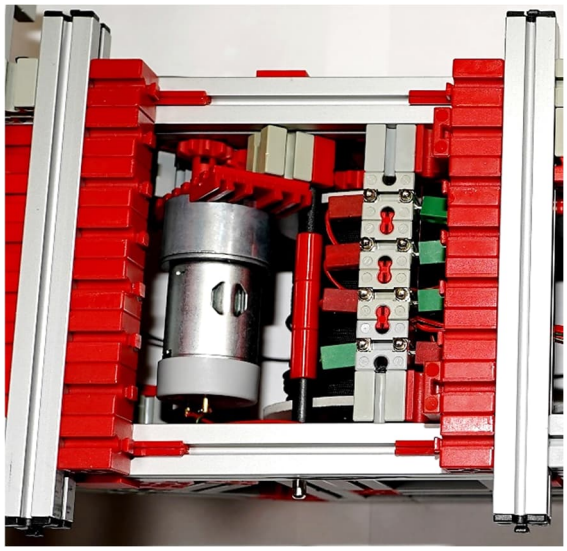


Abb. 9: Unterseite des Krans: acht Federkontakte (31306) für die Motoren, die im Fahrgestell montiert sind; der 1:20-Power-Motor für Winde 1 ist ebenfalls zu sehen

Kranhaus – Oberbau

Ich habe meinen Kran modular aufgebaut: Grundrahmen, Gegengewichtsgrundplatte, zwei Gegengewichtsblöcke und die beiden Maschinenräume, wobei auf der rechten Seite zusätzlich die Kabine aufgebaut ist. Die Gegengewichte sind hinten leicht gebogen.

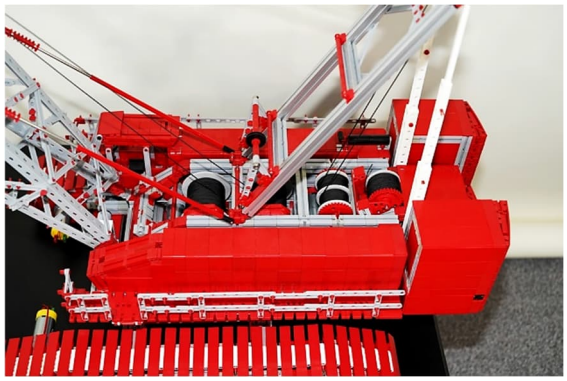


Abb. 10: Der Kran von oben gesehen

Im aufgebauten Zustand betragen die Länge 610 mm, die Breite 330 mm und der Wendekreis 420 mm.

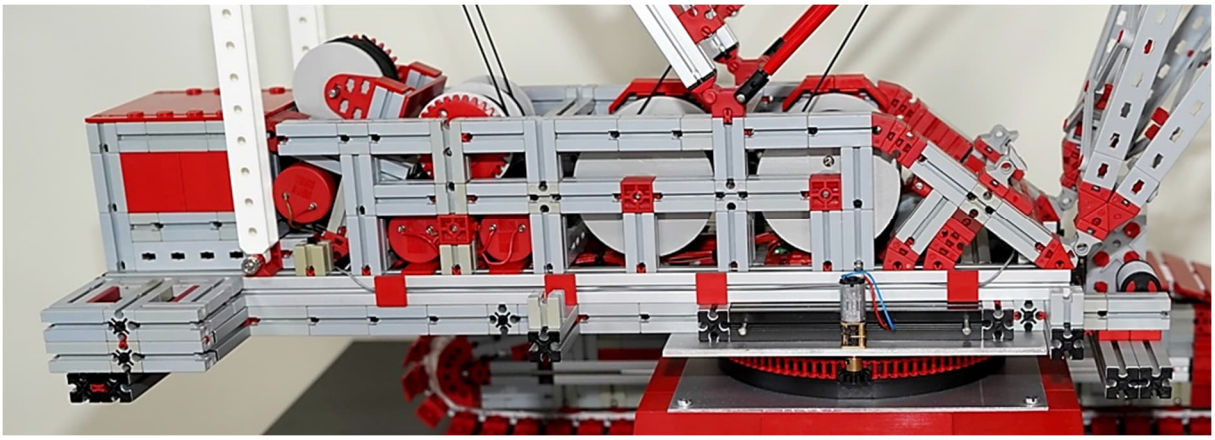


Abb. 11: Grundrahmen ohne die übrigen Aufbauten

Wenn man dem Link [3] folgt, gelangt man zur „Avrora parts“. Die verfügen über zahlreiche Übersichtszeichnungen aller Teile des KH1000. Auf der Zeichnung des „FRAMES“ ist gut zu erkennen, dass dieser hinter der zweiten Windentrommel schräg nach unten verläuft.

Ich habe versucht, dies auch in mein Modell zu integrieren – zweimal. Das hat allerdings nicht gut geklappt. Der 20°-Winkel ließ sich zwar noch aufbauen, allerdings fehlte dann der Platz, um die beiden Motoren und die Winde für den Aufrichte-Bock einzubauen. Für Winde 3 stimmte der Einbauwinkel daher auch nicht mehr. Allerdings war auf allen Fotos, die ich im WWW gefunden habe, dieser Punkt auf der Oberseite des Krans nicht klar zu erkennen. Zudem wurden die Maschinenräume beidseitig angebaut, so dass auch der Blick auf das „FRAME“ verdeckt wird. Deshalb habe ich es einfach bis zum Ende durchlaufen lassen.

Seilwinde

Auf den Bildern ist zu erkennen, dass ich für die Seilwinden auch andere Materialien verwendet habe. Der Kern besteht aus einem Rundstab, 20 oder 30 mm POM, die Seiten aus 3 mm dickem PVC. Die Zahnräder sind mit kleinen Schrauben angeschraubt. Bei den Seilwinden 1 und 2 habe ich einen 1:20-Power-Motor und bei den anderen beiden Seilwinden 1:50-Power-Motoren verbaut. Aufgrund des zu erwartenden hohen Gewichts des langen Auslegers habe ich für den Aufrichte-Bock zwei Motoren verwendet.

GANTRY

Wenn man dem Link in [4] folgt sieht man eine Zeichnung des GANTRY. Dieser Aufrichte-Bock besteht aus einem festen Teil und drei ineinander schiebbaren Teilen. Dieses dreiteilige „LEG“ ermöglicht es, den Aufrichte-Bock in verschiedenen Winkeln zu platzieren.

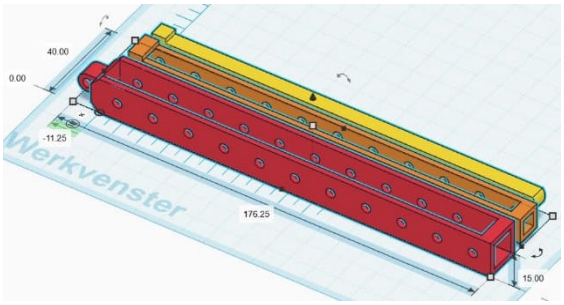


Abb. 12: 3D-Konstruktion des dreiteiligen „LEG“

Bei einem kurzen Ausleger hat der Ausleger einen größeren Winkel (Abb. 1). Bei einem langen Ausleger ist es in Kombination mit dem „MAST“ notwendig, den Winkel zu verringern (Abb. 13 und 19). Für den Transport liegt der Aufrichte-Bock komplett horizontal. Dies ist auch auf den Bildern im WWW zu erkennen. Das habe ich zunächst ganz mit I-Streben aufgebaut. Da dies einen hohen Auf- und Umbauaufwand am Kran

bedeutete, habe ich hierfür eine 3D-Version angefertigt [5].

Seile und Seilrollen

Ich habe vor Jahren ein paar Hundert Seilrollen anfertigen lassen. Beim KH1000 verwende ich die 25,5 mm großen für den Aufrichte-Bock und die 39,5 mm großen für die Haken. Das Seil ist 1,4 mm dick.

Abb. 13: Der „MAST“ ist eigentlich ein größerer, längerer Aufrichte-Bock

Bei allen Kränen, die ich bis vor Kurzem gebaut habe, liegen die für den Aufrichte-Bock benötigten Scheiben nebeneinander, also vertikal. Beim KH1000 sind sie jedoch horizontal und in drei Gruppen angeordnet. Vor dem Aufrichte-Bock liegen sieben Scheiben. Der Ausleger wird dann mit einem „BRIDLE“ mit acht Scheiben ausgestattet. Obwohl ich im WWW Versionen gesehen habe, in denen es acht auf dem Aufrichte-Bock und neun im „BRIDLE“ gibt.

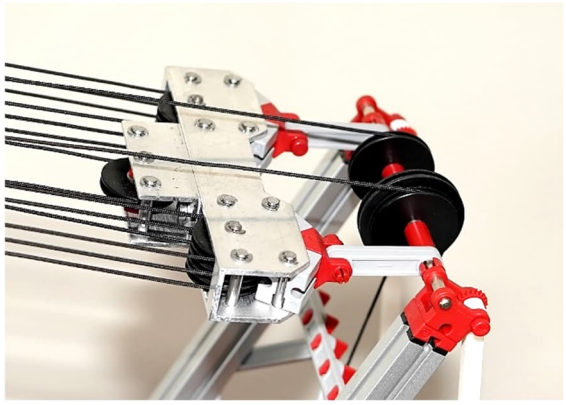


Abb. 14: ALU, Stahl und POM: etwas zu schwer geworden

Wichtig war mir dabei, dass die Seilrollen ein möglichst kompaktes Gehäuse erhalten. Am Ende habe ich es geschafft, allerdings nicht mit fischertechnik sondern mit Alu-Profilen (Abb. 13, 14).

Die Seilrollen sind auf Stahlwellen montiert, die mit Klemmringen gesichert sind. Um ein Herausrutschen des Seils aus den Seilrollen zu verhindern, habe ich zusätzlich Achsen zwischen den Seilrollen montiert.

Dass das alles zu schwer geworden ist, erkennt man an Abb. 14. Eine 3D-Version aus Kunststoff könnte möglicherweise leichter sein.

Auch bei den Haken gibt es Unterschiede. Da ich zuerst die CAD-Zeichnungen hatte, habe ich die Haken anhand der CAD-Zeichnungen gebaut (Abb. 15, 16). Dass ich die Haken fast komplett mit fischertechnik bauen konnte, war schon überraschend. Ich habe versucht, sie möglichst dem Original ähnlich zu machen, aber das hat nicht besonders gut geklappt. Das liegt daran, dass die Haken zu leicht geworden wären, wenn ich nicht zusätzlich andere Materialien verwendet hätte. Sowohl beim 200-t- als auch beim 100-t-Haken habe ich deshalb jeweils ein Stück aus massivem Messing in den Maßen 40 und 30 mm verbaut.

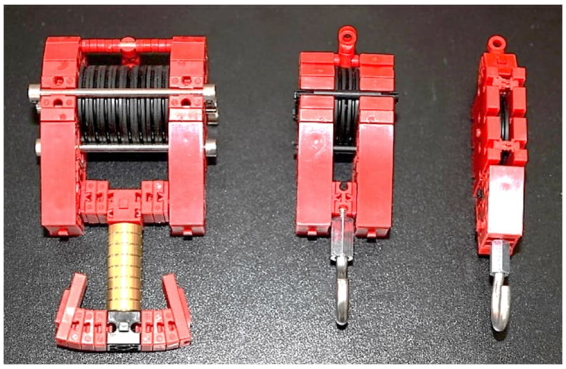


Abb. 15: Haken: links 200 t, in der Mitte 60 t, rechts 25 t

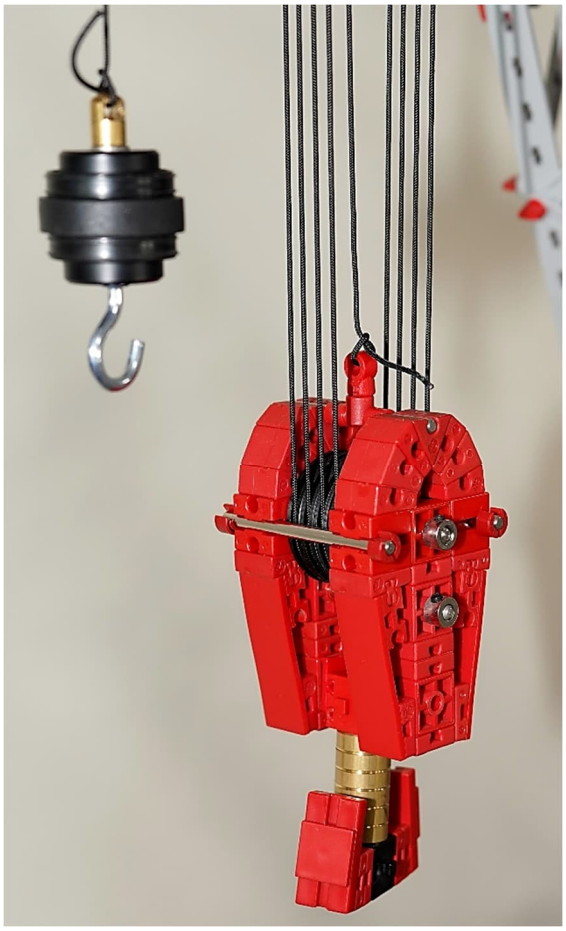


Abb. 16: Der kleinste Haken ist für max. 12,5 t, der größere für 100 t

Die 100-t-Version ist jetzt schwer genug. Allerdings ist der für 200 t immer noch zu leicht, es sei denn, ich benutze nur vier der acht Seilrollen. Für die kleineren habe ich im Baumarkt mehrere Haken gekauft, um sie schwerer zu machen.

Krumme Ausleger

Nachdem ich den Maßstab, in dem ich den KH1000 bauen wollte, auf 1:15,5 berechnet hatte, habe ich erstmal ein Auslegerteil von 9 m Länge gebaut. Der Durchmesser beträgt 2100 mm. Im Maßstab 1:15,5 würden das etwa 135,48 mm sein, was sich natürlich auf 135 mm begrenzt. Auch die Länge ist begrenzt. Das hat mit dem fischertechnik-Rastermaß von 15×15 mm zu tun: 9000 mm / 15,5 = 580 mm. Beim Vorbild gibt es acht Diagonalstreben: 580 mm / 8 = 72,5 mm. 72,5 mm sind aber nicht machbar, da sie außerhalb des fischertechnik-Rasters liegen. Deswegen wurden das 75 mm.

Die Länge der Ausleger wird angegeben über die Verbindungsaugen. Um die zu simulieren habe ich einfach Gelenkwürfel (31424/31425) angebaut. Die Gesamtlänge kommt damit auf 8×75 mm + 2×15 mm = 630 mm. Die I-Strebe wurde 60 mm + 15 mm + 60 mm = 135 mm und die X-Strebe 63,6 mm + 15 mm + 75 mm = 153,6 mm lang. Laut Pythagoras sollte damit

$$c = \sqrt{a^2 + b^2}$$
$$= \sqrt{(135\ \text{mm})^2 + (75\ \text{mm})^2}$$
$$\approx 154{,}4345\ \text{mm}$$

betragen. Die X-Strebe ist damit 0,8345 mm zu kurz, aber das ist nicht zu sehen.

Da die fischertechnik-Streben leider nicht alle so lang sind, wie es der Aufdruck vermuten lässt, fällt diese Abweichung von 0,8345 mm unter Umständen nicht auf. Sollten alle Streben geringfügig länger sein, kann die Länge der Kombination größer sein, als sie laut Berechnung sein müsste und so die Abweichung reduzieren.

Wo die Streben zu kurz sind, werden die Winkelträger zusammengezogen. Wo die Streben zu lang sind, werden die Winkelträger auseinander gedrückt. Wie auf den Bildern zu sehen ist, kommen immer zwei Strebenpaare zusammen. Das bedeutet, dass

der Durchmesser nicht mehr quadratisch ist, sondern eher ein Parallelogramm. Zur Spitze hin werden die Streben immer kürzer. Eine willkürliche Reihenfolge der zu kurzen und zu langen Streben kann den Effekt nur verschlimmbessern (Abb. 17).

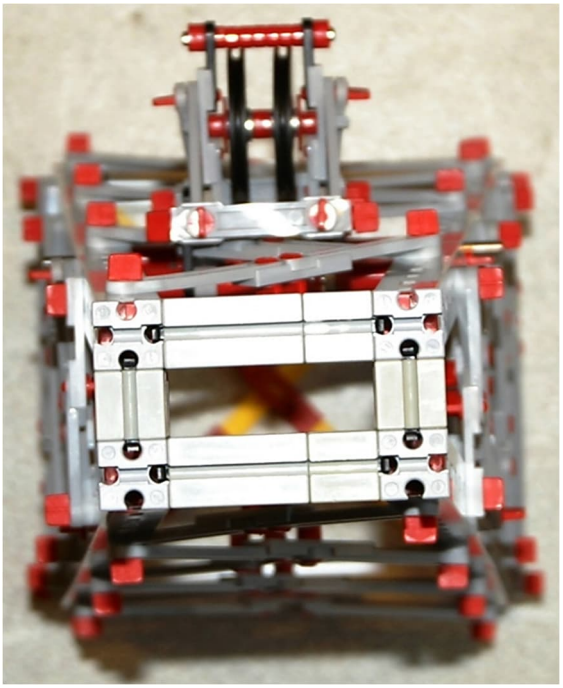


Abb. 17: Krumm oder verdreht: beliebig ausgewählte Streben

Also versuchte ich, die richtigen Längen mit anderen Kombinationen zu erwischen. Das ist einfacher gesagt als getan.

Doch wofür hat man einen Computer? Nicht nur, um einen Text für die ft:pedia zu schreiben. Excel ist mein bester Freund. Und mit ein wenig Programmieren habe ich einen einfachen Streben-Kalkulator erstellt.

Was dabei heraus kam, ist in Abb. 18 zu sehen. Der Effekt ist noch immer da, aber wesentlich kleiner. Die Statiklasche 15 (36326) in Kombination mit der Breite der Streben ergeben aber ein kleines Problem. Die Streben sind 8 mm breit (was außerhalb das fischertechnik-Rasters liegt). Deswegen ist es nicht möglich, das mittlere Loch der S-Lasche 15 zu nutzen, ohne die beiden Streben um 0,25 mm zu kürzen.

Die Verwendung der transparenten S-Prüfriegel 4 (36458) verschafft zusätzlich etwas mehr Freiheit zum Ausgleich.

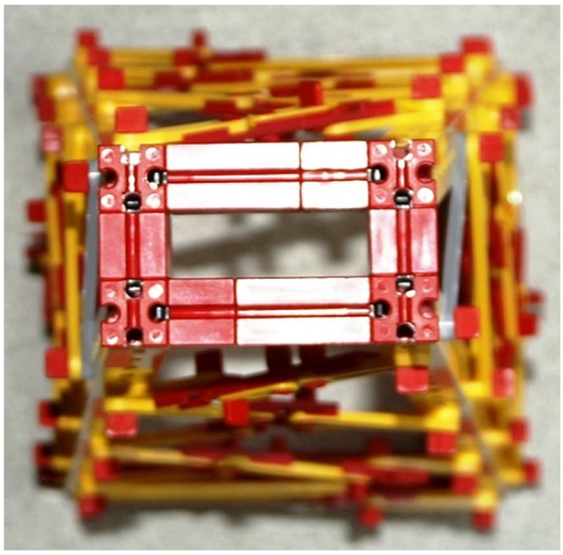


Abb. 18: Nach dem Ausprobieren mehrerer Kombinationen

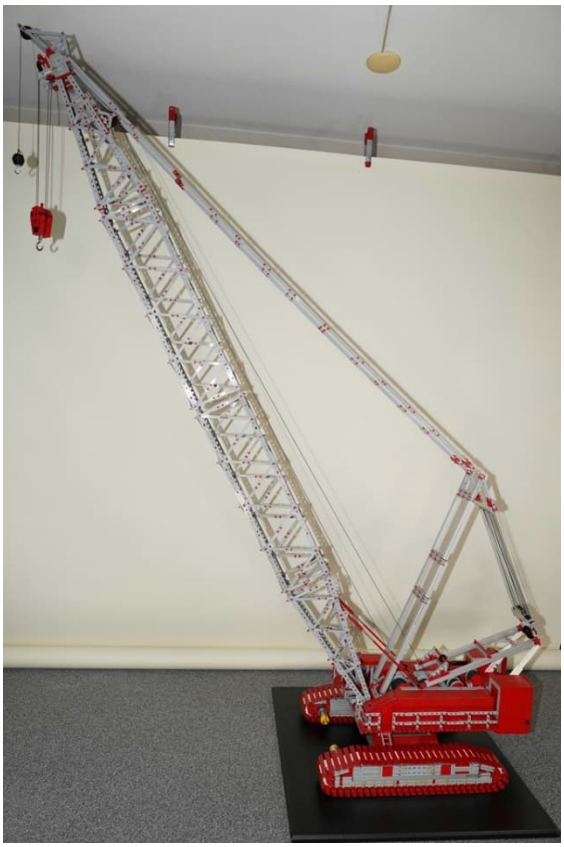


Abb. 19: Der KH1000 mit den beiden „9 m“-Auslegerteilen

RC-Anlage

Da ich noch einige Fahrregler von Conrad (190040, nml, Abb. 20) herumliegen hatte, wollte ich zunächst diese in den KH1000 einbauen.

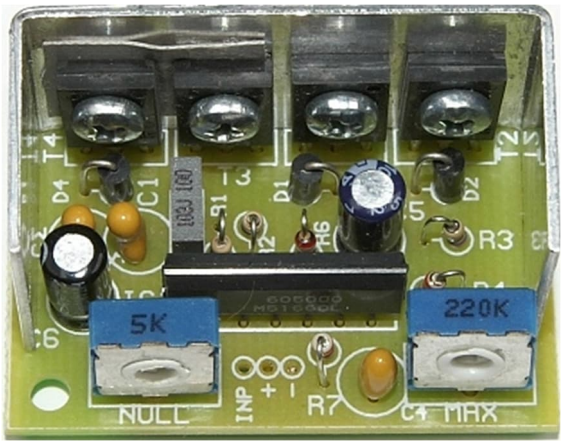


Abb. 20: Conrad 190040: 3-A-Fahrtregler. (NML)

Im Gegensatz zu meinem Demag CC 4800, bei dem ich die RC-Installation im Unterbau platziert hatte, war beim KH1000 hierfür viel zu wenig Platz. Ein Austausch der Batterie wäre dann auch nicht mehr möglich gewesen.

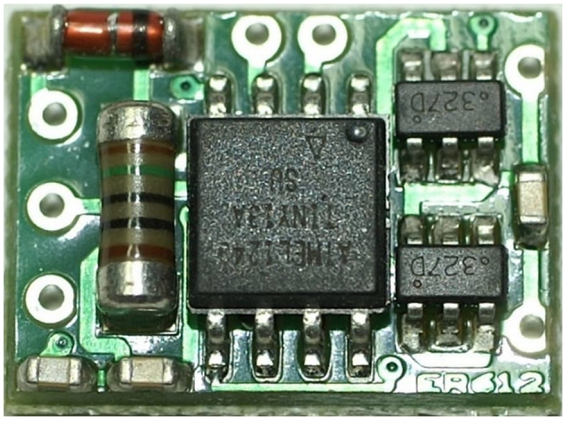


Abb. 21: Der ER125 aus der Sol-Expert-Group: 1,8-A-Fahrregler, geeignet für den Betrieb mit 12 V

Ich kaufte mir acht Stück Sol-Expert-Group ER125 Fahrtregler (Abb. 21). Solche hatte ich bereits einen in meinen Gottwald MK500/600-Kran installiert. Die sind schön klein und vertragen trotzdem 1,8 A. Der Nachteil besteht darin, dass sie keine Möglichkeit bieten, sie irgendwo anzuschrauben. Natürlich hätte ich sie mit etwas Knetgummi irgendwo festkleben können, aber ich glaubte nicht, dass das wirklich funktionieren würde. Aus diesem Grund habe ich eine neue Leiterplatte entworfen, die nicht nur die acht Controller, sondern auch einen 5-Volt-Stabilisator enthält. Als Leistungsteil wollte ich zusätzlich acht H-Brücken-ICs vom Typ TLE4202 verbauen (Abb. 22).

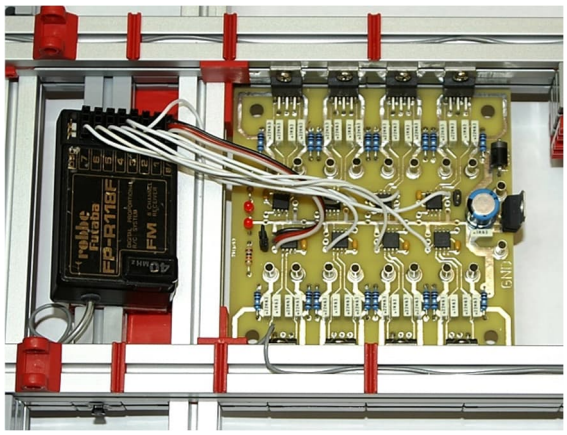


Abb. 22: Platine mit acht Reglern AT Tiny 13, TLE 4202 als Leistungsendstufe und 5-V-Spannungsstabilisator

Das Problem bestand nun darin, die kleinen AT-Tiny-13-Chips von ihrer kleinen Leiterplatte zu lösen. Glücklicherweise fand ich wenig später einen Artikel in einer alten Elektor-Zeitschrift, in dem dies beschrieben wurde.

Das Auslöten und Auflöten auf die neue Leiterplatte war nicht allzu schwierig. Leider stellte sich heraus, dass nur zwei der Chips die Transplantation überlebten, zwei teilweise und die anderen vier überhaupt nicht. Zweifellos war es ihnen zu heiß geworden.

Ich habe zunächst bei der Sol-Expert-Group nachgefragt, ob sie auch nur die Chips liefern könnten. Leider haben sie nie geantwortet.

Was war nun zu tun? Erneut Leiterplatten entwerfen. Die Conrad-Fahrregler enthalten einen Chip vom Typ M51660L. Dieser

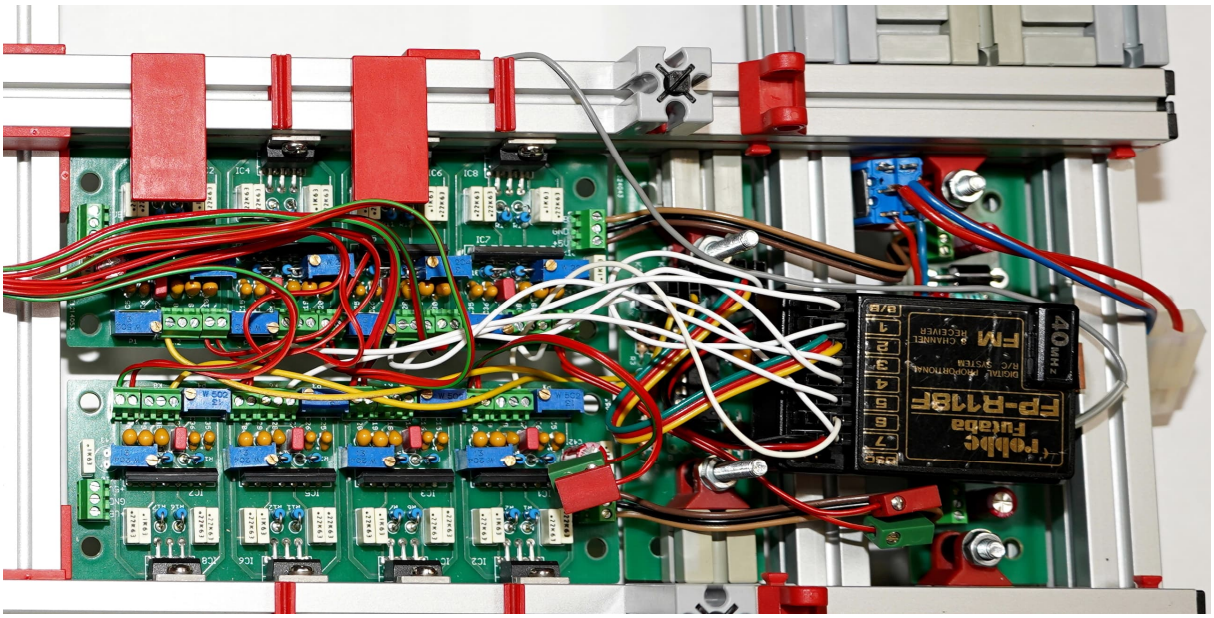


Abb. 23: Übersicht der RC-Steuerung: acht Fahrregler, 8-Kanal-Empfänger, (Delta-)V-Mischer zur Ansteuerung der Raupen und Spannungsstabilisator. Zur Kühlung der acht TLE4202 werden diese mit den Aluminiumprofilen verschraubt

wurde auch separat verkauft. Außerdem habe ich bei Lemo-Solar 20 Stück vom Typ NE544 bestellt. In einer niederländischen Modellbauzeitschrift hatte ich bereits Ende der 70er Jahre den Schaltplan eines Fahrreglers mit diesem Chip gefunden.

Für beide habe ich eine Platine entworfen, auf der vier Fahrregler untergebracht sind, auch wieder mit dem TLE4202 als Leistungsstufe (Abb. 23).

Dann fehlte nur noch eine Stabilisator-Platine. Auch diese habe ich selbst entwickelt (Abb. 24). Nachdem dies alles installiert war, habe ich den Kran fertiggestellt, ihn im Wohnzimmer aufgebaut und damit gespielt. Schnell wurden mir klar, dass das Fahren mit zwei Steuerhebeln, gesteuert über meinen Sender (Robbe/Futaba F14), nicht einwandfrei funktionierte. Trotz korrekter Einstellung wollte der Kran nicht geradeaus fahren. Mir schien, dass einer der Motoren zu schnell (oder zu langsam) lief. Der Austausch der Motoren stellte sich als keine Lösung heraus.

Delta-V Mixer

Auch hier brachte Conrad die Lösung: V-Mixer 225231 (Abb. 24) – leider auch nicht mehr lieferbar (nml). Der wurde auch von Carson angeboten (503010). Auch nml?

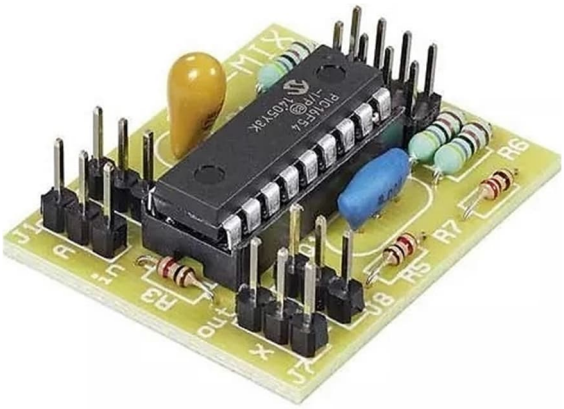


Abb. 24: V-Mixer Conrad 225231, auch von Carson angeboten (503010)

Ursprünglich ist diese Schaltung für die Steuerung von Flugzeugen mit Deltaflügeln vorgesehen, kann aber auch für Kettenfahrzeuge verwendet werden: Hebel nach vorne oder hinten bedeutet Geradeausfahren, Hebel nach links bedeutet Linksdrehen auf der Stelle, Hebel nach rechts bedeutet Rechtsdrehen. Wenn man den Hebel

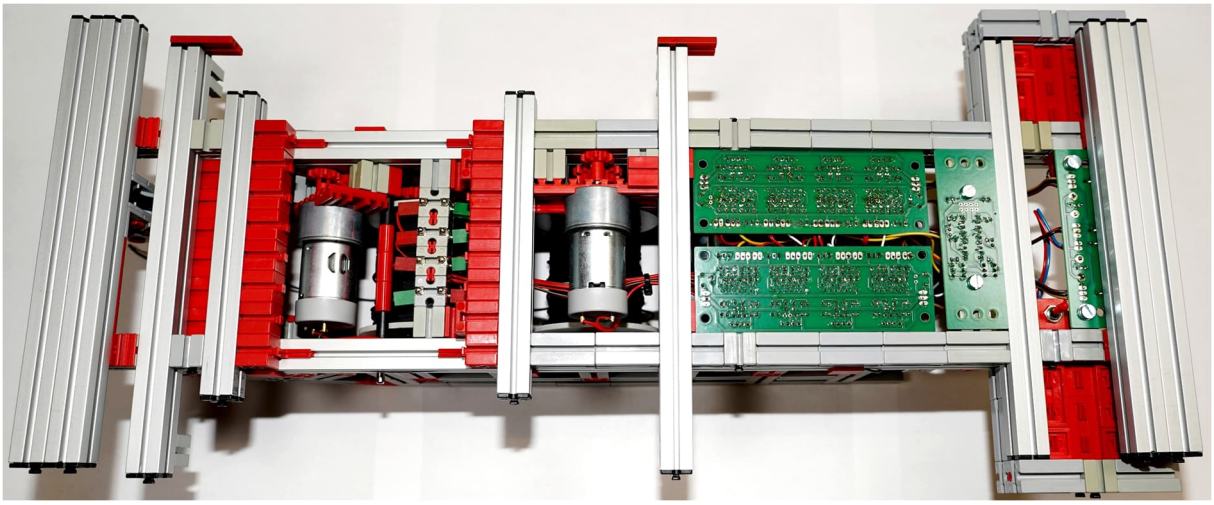


Abb. 25: Die Unterseite des Krans. Die acht Federkontakte 31306 für die Motoren (sichtbar zwischen den beiden Motoren) drücken auf… (siehe Abb. 26)

beispielsweise sowohl nach vorne als auch nach links bewegt, beschreibt der Kran einer Kurve nach links.

Und jetzt wiederhole ich mich: Es musste eine neue Leiterplatte hergestellt werden. Etwas, was ich tatsächlich gerne mache. Obwohl es für die Größe der Schaltung ausreichend gewesen wäre, eine 30 mm × 60 mm große Leiterplatte herzustellen, habe ich mich trotzdem für eine Leiterplatte in der Größe 30 mm × 90 mm entschieden. Durch zusätzliche Bohrungen ergeben sich weitere Montagemöglichkeiten (Abb. 28).

Nach dem Zusammenbau des Krans im Wohnzimmer stellte sich schnell heraus, dass der V-Mixer eine gute Geradeausfahrt des Krans ermöglichte.

Jetzt musste ich nur noch die restlichen Funktionen auf den Sender verteilen. Die beiden Seilwinden 1 und 2 können nun gemeinsam mit dem anderen Stick (Kanal 3 und 4) betätigt werden. Der Schieberegler von Kanal 5 dient zum Spannen der Raupen und Kanal 6 für den Aufrichte-Bock, mit Kanal 7 für Seilwinde 3 und Kanal 8 zum Wenden. Abb. 27 zeigt eine schematische Übersicht der RC-Anlage.

Das Gegengewicht besteht aus drei Teilen, die zusammen 1130 g wiegen (Abb. 29). Durch die 12-V-Batterie kommen noch einmal 300 g dazu. Für den kurzen Ausleger reicht das vollkommen aus.

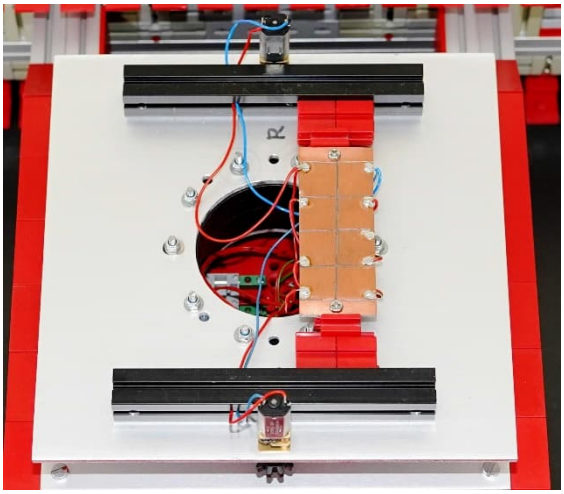


Abb. 26: … die Platine auf dem Fahrgestell

Da der lange Ausleger viel schwerer ist als der Kurze, ist es notwendig, zusätzliches Gewicht hinzuzufügen: 1140 g, bestehend aus vier Metallstangen mit je 285 g (Abb. 30).

Insgesamt beträgt das Gesamtgewicht meines Modells 17,6 kg. Nach dem Auf- und Abbau schmerzte mein Rücken dann auch einige Tage. Das musste ich akzeptieren.

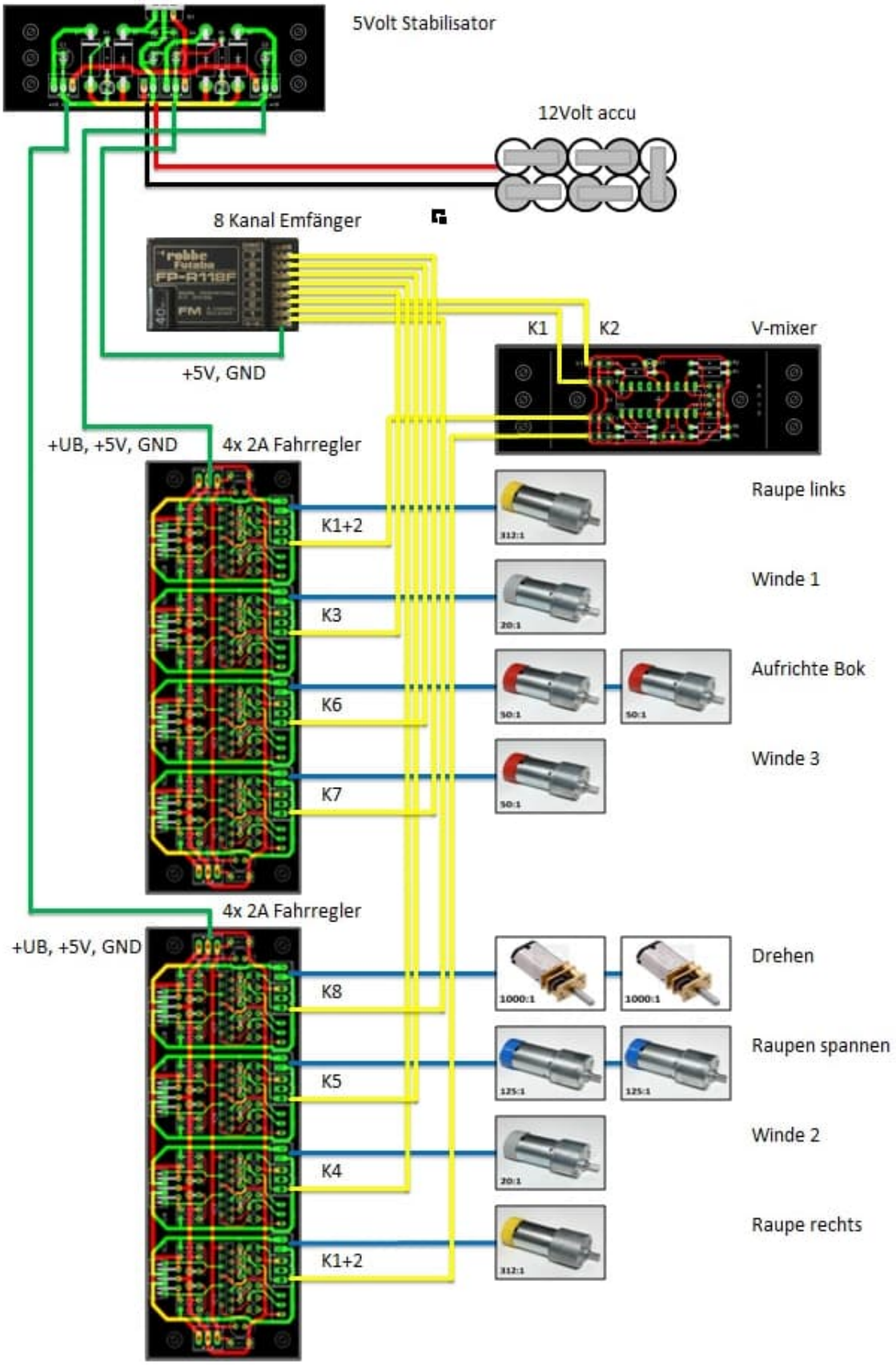


Abb. 27: Schema der gesamten RC-Anlage

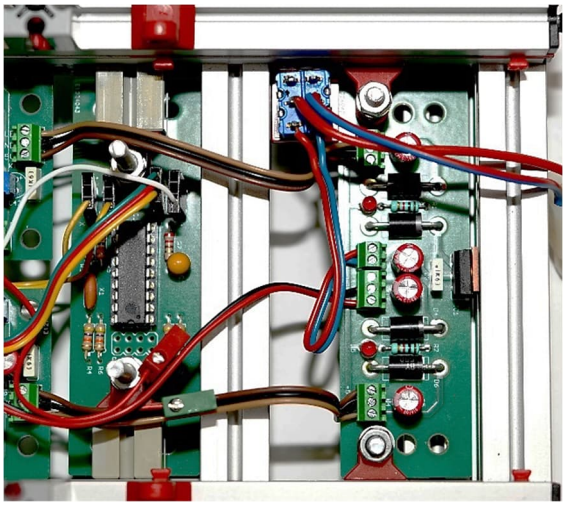


Abb. 28: Links der V-Mixer, rechts die Spannungsstabilisator-Platine

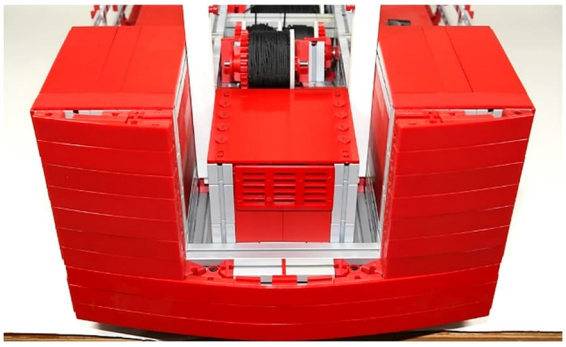


Abb. 29: Beim kurzen Ausleger ist es hier nicht notwendig, zusätzliches Gewicht anzubringen

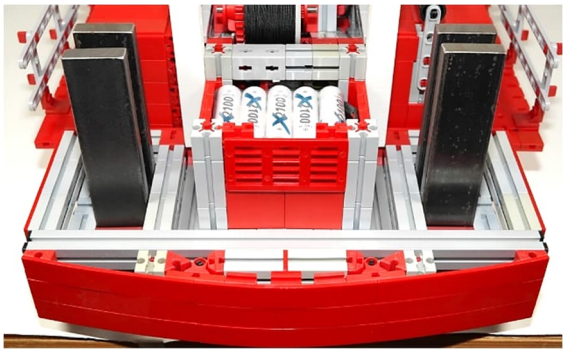


Abb. 30: Für den langen Ausleger ist es aber notwendig, zusätzliches Gewicht anzubringen: vier Metallstangen mit je 285 g

Fertig?

Ja, aber...

Es gibt jedoch immer etwas, was geändert werden kann (oder muss).

Ich möchte auch ein paar richtig scharfe Fotos machen. Ich habe mir eine neue Kameraausrüstung gekauft, weil meine alte nicht mehr richtig funktionierte. Ich muss aber noch herausfinden, wie sie funktioniert. Das wird also noch einige Zeit dauern. Letztlich ist es aber mein Ziel, dass alle Bilder im Bilderpool landen.

Quellen

[1] AutoCAD-Dateien (DXF) des KH1000, auf hsc-cranes.com.

[2] Hitachi: *KH1000 Counterbalancing Hydraulic Crawler Crane Specifications*. PDF-Download auf cranenetwork.com.

[3] Avrora LDT: *KH1000 FRAME Hitachi HOP online*. Auf k-part.777parts.org.

[4] Avrora LDT : *KH1000 GANTRY Hitachi HOP online*. Auf k-part.777parts.org.

[5] Peter Krijnen: *telescoop 165 voor KH1000*. 3D-Druckdaten auf thingiverse.com.

Modell

Mars-Rover: Modell und Steuerung

Werner Hasselberg

Spurensucher sind eine faszinierende Sache: Wie von Geisterhand gesteuert folgen sie ihrem vorgegebenen Weg und verirren sich „fast“ nie. Computergesteuert sind sie in der Lage, Hindernissen auszuweichen und verschiedenen Wegen zu folgen. Aber ist so etwas auch ohne Computer möglich? Dieser Beitrag liefert die Antwort.

Vorwort

Nachdem die Menschheit bereits den Mond betreten hat – nun schon vor über 50 Jahren –, schickt sie sich nun an, auch den Mars zu erobern. So landete vor gut 20 Jahren das erste Mal die *Pathfinder*-Sonde auf dem roten Planeten und lieferte gestochen scharfe Bilder von weiten Teilen einer fremden Welt. Seitdem gab es viele Missionen zum Mars. Manche davon, gar nicht mal so wenige, gingen jedoch auf die eine oder andere Weise gänzlich schief. Und erst vor ein paar Jahren ist Europas erster Versuch einer erfolgreichen Landung nach bangem Warten ebenso für gescheitert erklärt worden. Leider führte ein Softwarefehler während der Landung zum Absturz des ganzen Moduls.

Kein Grund für einen fischertechniker, Trübsal zu blasen. Wir bauen uns einfach selbst einen Mars-Rover und schicken ihn auf Mission. Und weil wir den Computern nicht immer vertrauen können, verzichten wir gleich ganz auf sie. Auch wenn der Mars, mangels geeigneter Transportmittel, vermutlich nicht das Endziel unseres Rovers sein wird und dessen riesige Gebirgszüge symbolisch von der Couchgarnitur im Wohnzimmer gemimt werden, ist es eine spannende Sache, mal ganz ohne PC-Hilfe ein brauchbares Vehikel zu konstruieren, das allem Unbill eines solchen „Geländes“ trotzen muss.

Grundlagen

Bevor wir den Rover auf seiner Reise begleiten und überwachen können, sind zuerst die Konstrukteure gefragt, ein geeignetes Vehikel mitsamt elektronischer Steuerung zu konstruieren. Doch weshalb eigentlich ohne Computersteuerung? Nun, damit verbringt man wieder die meiste Zeit vor dem Bildschirm für das Programmschreiben – und das ist genau das, was wir ohnehin alle schon viel zu oft tun. Freilich ist es sehr praktisch, es so zu erledigen, denn es erspart viel Kabelsalat und kann bequem an erforderlichen Stellen geändert oder ergänzt werden. Auch sind die Steuermöglichkeiten praktisch grenzenlos, was mit reiner Elektronik nicht so einfach bzw. unmöglich ist, da irgendwann sowohl der Platz als auch die Bausteine samt Kabel zur Neige gehen.

Das Schöne daran aber ist, nicht vor dem Bildschirm sitzen zu müssen, und dass die Motorik und das Feingefühl beim Konstruieren des Modells und der Schaltung gefördert werden. Schließlich wird nachhaltiges Grundwissen über Elektronik erworben, das weit weniger schnell veraltet als sämtliche Apps und Software-Tools. Dioden, Widerstände, Kondensatoren, Relais, Poti-Regler, Transistoren, beinahe so alt wie die Elektronik selbst, sind immer noch unabkömmlich und überall im Einsatz. Es ist erstaun-

lich, welche Wirkungen mit ihnen erzielt werden können.

Funktionsweise

Bei all meiner Reklame für Elektronik sind dennoch ein paar Abstriche aus Platzgründen unvermeidlich, wenn das Vehikel nicht die Ausmaße eines Monstrums annehmen und der Autor am Ende nicht von seinem eigenen Kabelsalat völlig überfordert werden soll. Trotzdem ist ein passables Modell machbar. Wir verzichten dabei aber auf komplizierte Manöver; stattdessen soll unser Pathfinder einfach einer vorgegebenen Spur folgen, wenden und gegebenenfalls anhalten können. Das reicht schon für einige Stunden Tüftelei, und das ist doch der eigentliche Spaß an der ganzen Sache, oder?

Überlegen wir uns zuerst die Technik des Rovers. Wir sehen ja z. B. bei der NASA, dass sie mit Rädern ausgestattet sind. Das wollen wir auch so machen, und die Lenkung soll möglichst einfach sein, ohne dass eine zusätzliche Konstruktion wie eine Achsschenkel-Lenkung dafür nötig ist. Das geht am besten wie bei einer Raupe und erfordert drei Räder mit Allradantrieb und je einen Motor auf jeder Seite. Bei Rückwärtsfahrt eines der Motoren dreht sich das Fahrzeug um seinen Schwerpunkt. Das ist sehr praktisch, denn so steuert es auf der Stelle und kann sich sehr einfach von einem Randstreifen entlang der Fahrbahn wegdrehen.

Dafür nehmen wir die alten Reifen 60 aus dem Ergänzungskasten 50/3. Wer davon nur vier hat kann es auch so probieren; das müsste ebenfalls funktionieren. Wichtig ist, sie ohne Gummiringe zu verwenden. Da die Reifen aus Hartplastik bestehen, können sie ohne die Ringe leichter auf Oberflächen rutschen bzw. sich auf der Stelle drehen. Deshalb sollten die Reifen keinen festen Gripp auf der Oberfläche haben. Im Prinzip ist nur wichtig, dass auf jeder Seite die drei Räder per Allradantrieb laufen. Die restliche Gestaltung kann nach Belieben erfolgen. Abb. 1 zeigt meine Variante.

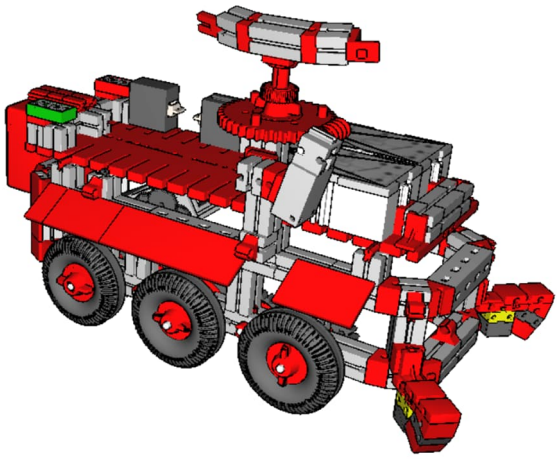


Abb. 1: Der Mars-Rover

Das Modell

Wie die Abbildung zeigt, sollte der Rover so gebaut werden, dass Elektronik und Akku genügend Platz haben. Je breiter er ausfällt, desto leichter kann er lenken. Ich habe allerdings versucht, ihn möglichst schmal zu machen, damit die Fahrbahn nicht zu breit sein muss und er nicht zu klobig wird. Und wie schon Howard Wollowitz in der Serie *Big Bang Theorie* während eines Baseballspiels witzig präsentierte, bewegt sich ein Mars Rover reichlich langsam, was aber auf der unwägbaren Oberfläche des Mars unerlässlich ist. Das gilt auch für unser Modell. So haben die benötigten Fotowiderstände Zeit genug zu reagieren, und das Vehikel rumpelt nicht unerwartet über die Markierungen hinaus, z. B. weil das Licht gerade ungünstig reflektiert wird und die Reaktion verzögert.

Für eine langsame und ruhige Fahrt eignen sich die alten mot.2-Getriebe ganz hervorragend. Mit ihnen kann man bequem die passende Übersetzung hinbekommen. Die folgenden Abbildungen zeigen, wie sie zusammen mit den Motoren in das Modell eingebaut werden. Für jede Seite sind ein mot.1-Motor und ein mot.2-Getriebe nötig.

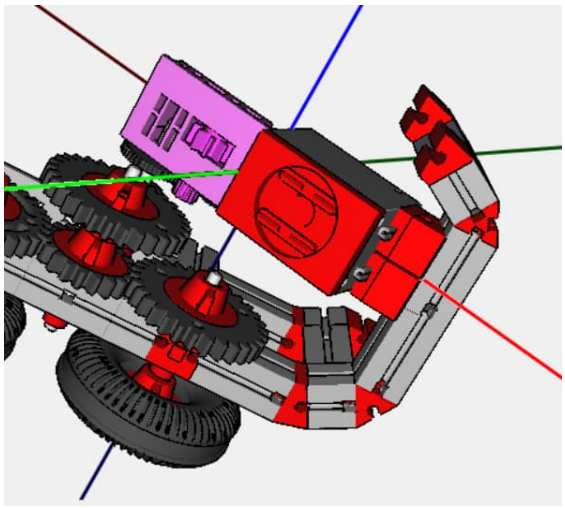


Abb. 2: Motorisierung des Modells hinten, von unten betrachtet

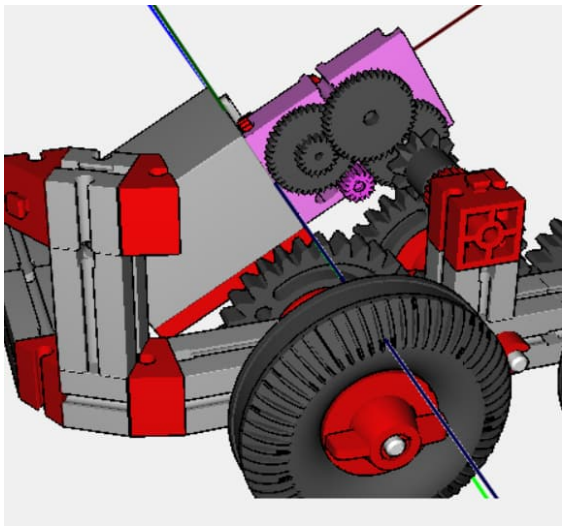


Abb. 3: Ansicht von der Seite

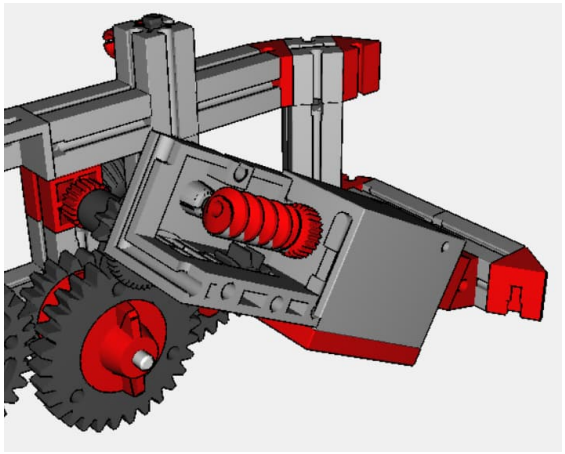


Abb. 4: Getriebeanschluss an die Räder

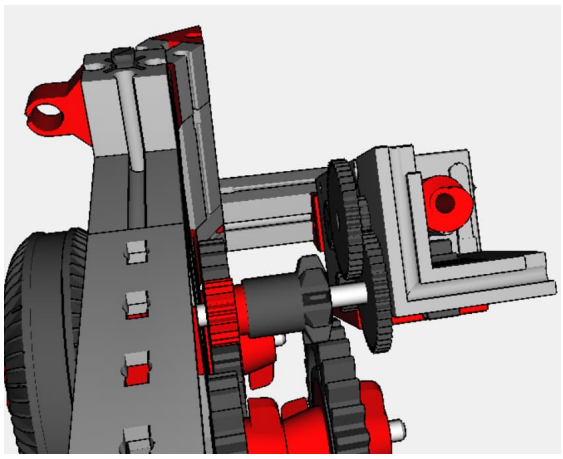


Abb. 5: Andere Ansicht des Getriebes

Über entsprechende Zahnräder wird der Allradantrieb auf jeder Seite umgesetzt. Vorwärts-, Rückwärts-, Links- und Rechtsfahrt sind so also kein Problem. Abb. 6 zeigt das gesamte Getriebe im Modell auf einer Seite. Der Motor mot.1 und das mot.2-Getriebe werden auf den beiden Winkelsteinen angebracht, wie in den vorherigen Abbildungen bereits dargestellt.

Damit erledigt die Technik schon sehr viel. Alles was jetzt noch an Elektronik nötig ist, sind zwei em3-Relais und einmal die IC-Elektronik-Baureihe. Sie steuert die Fotowiderstände.

Als Krönung des ganzen verfügt das Modell auch noch über ein motorisiertes Radar, das separat ein- oder ausgeschaltet werden kann.

Der zweite mot.1-Motor mit mot.2-Getriebe befindet sich auf der gegenüberliegenden linken Seite. Der Aufbau ist genauso wie auf der rechten.

Abb. 9 zeigt das komplette Modell mit Radar und Relais auf der Platte oben. Die Verkabelung kann man noch geschickter machen; sie fällt in der Realität allerdings nicht so stark auf wie auf dem Foto.

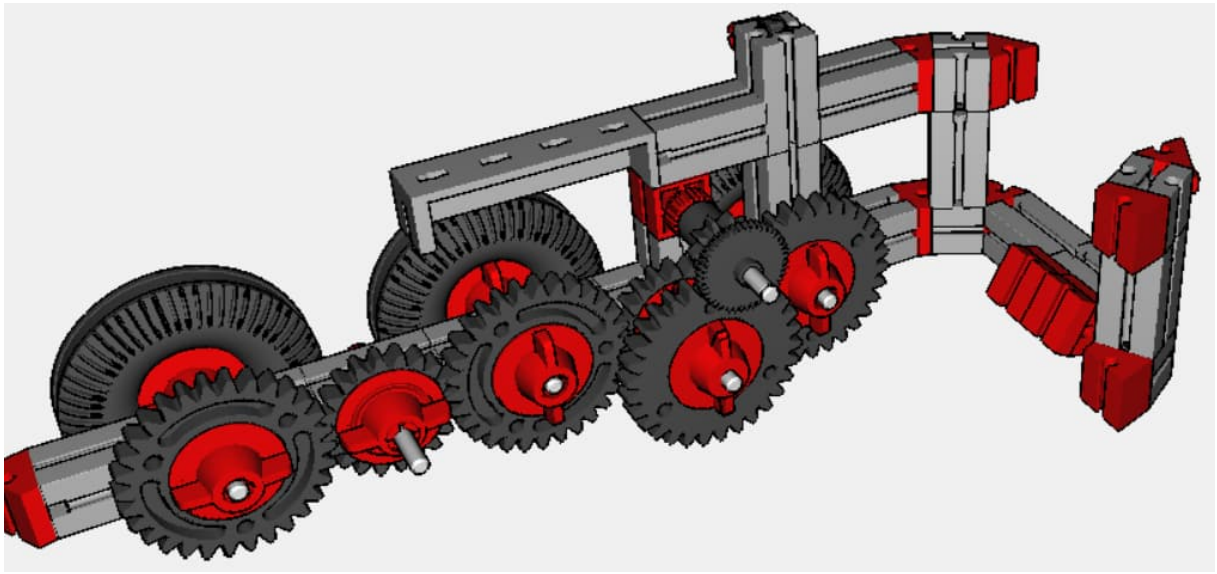


Abb. 6: Allradgetriebe

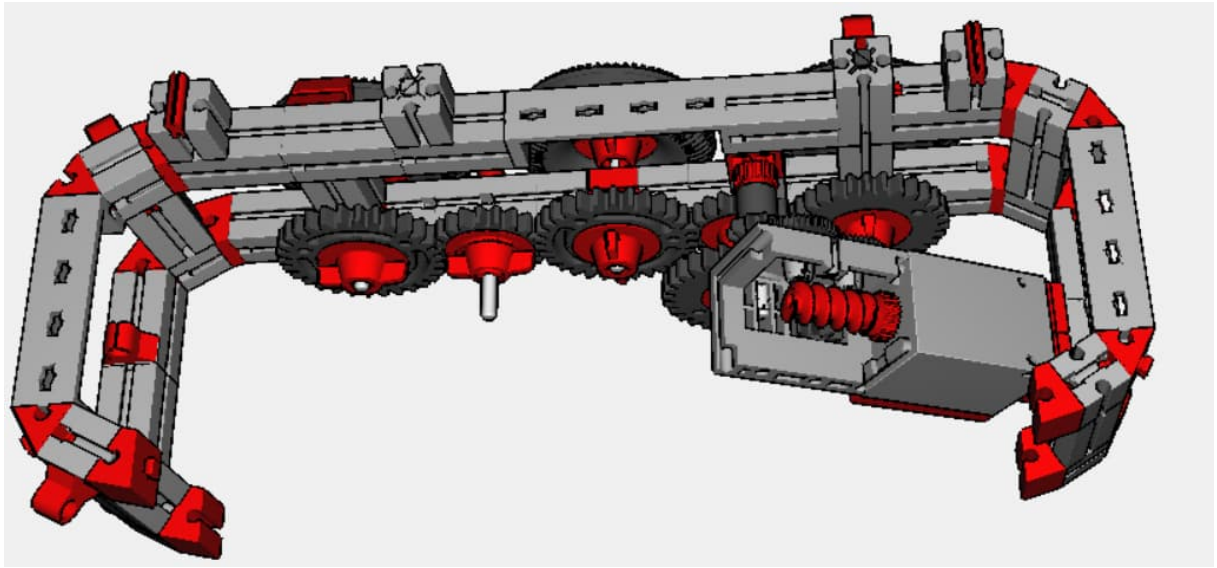


Abb. 7: Antrieb auf der rechten Seite; linke Seite analog dazu aufbauen

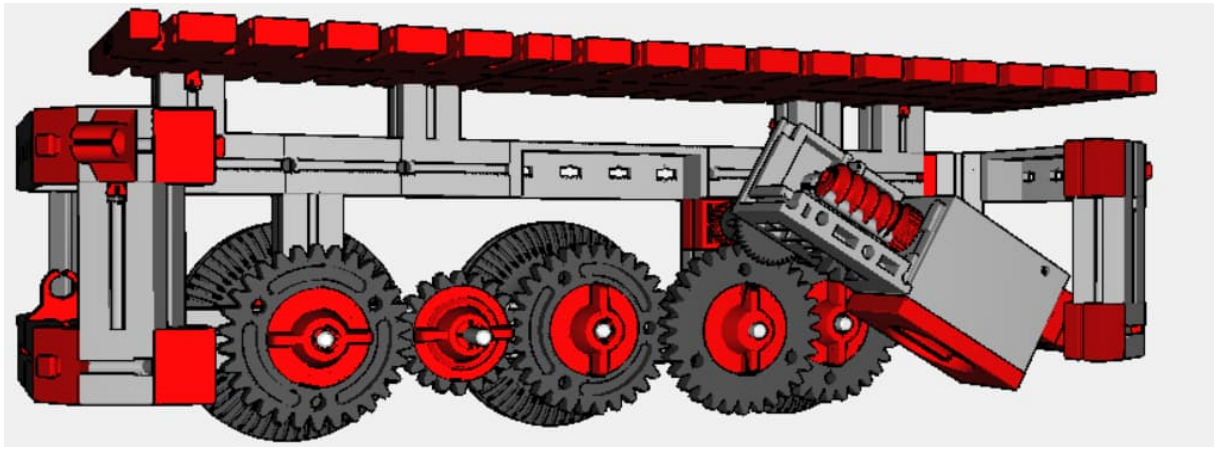


Abb. 8: Modell mit Deckplatte für die Elektronik

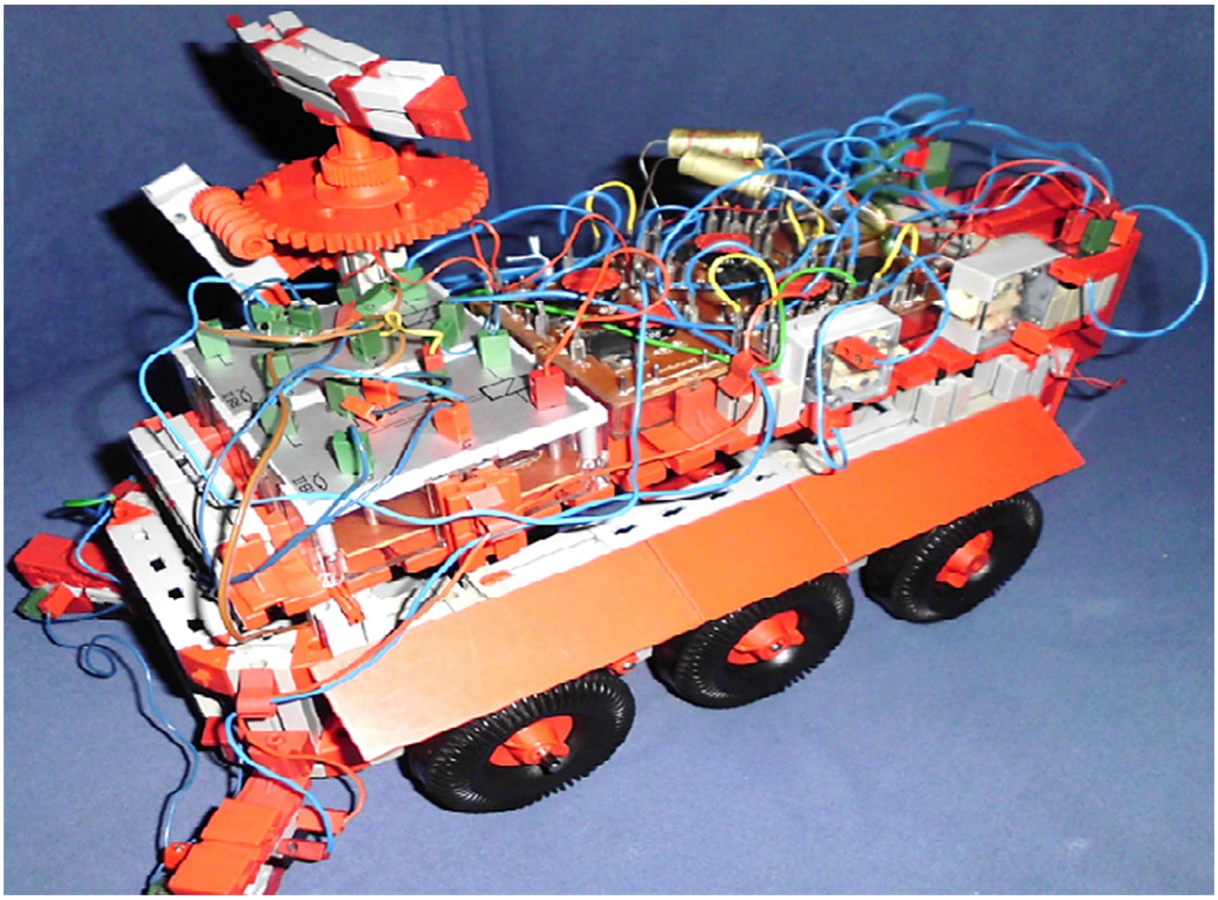


Abb. 9: Modell mit Radar und Elektronikaufbauten

Die Elektronik

Fotowiderstände übernehmen wie bereits erwähnt die Steuerung. Auf jeder Seite des Modells schräg vorne angebracht befindet sich einer und direkt hinter ihm eine auf einem Winkelstein montierte Linsenlampe, die in Richtung Boden unter dem Fotowiderstand einen Lichtkegel bildet. Er reflektiert nur wenig Licht, wenn der Untergrund schwarz ist, und sehr viel mehr, wenn er weiß ist. In diesem Fall schaltet ein Fotowiderstand (FW) über die Elektronik ein Relais ein, das den Motor der jeweiligen Seite auf Rückwärtsfahrt umpolt. Dadurch beginnt sich der Rover zu drehen. Je nachdem welcher FW aktiv wird, ist das nach rechts oder links. Wir brauchen also eine komplette IC-Schaltung mit den beiden Schwellenwertschaltern (SWS) und Leistungsstufen (LST). Jede ist für einen Motor verantwortlich und an jedem ist ein em3-Relais geschaltet, das ausgeschaltet für Vorwärts- und eingeschaltet für Rückwärtsfahrt sorgt.

Die Elektronik muss aber noch etwas mehr leisten. Der Rover fährt eine schwarze Fahrbahn mit weißen Randstreifen entlang. Da der Übergangsbereich zwischen Hell und Dunkel innerhalb von Millimetern stattfindet, steht für das Umpolen nur ein Augenblick Zeit zur Verfügung. Der Rover dreht für kurze Zeit und fährt dann geradeaus weiter. So gerät er schnell wieder über den Randstreifen und das ganze wiederholt sich. Es beginnt damit ein ruckeliges Hin und Her und er bleibt im schlimmsten Fall an einer Stelle hängen.

Es gibt aber einen einfachen Trick, das zu verhindern. Nötig ist eigentlich nur, dass der Rover, sobald er auf den Randstreifen gerät, sich etwas länger davon wegdreht als das reflektierte Licht den zuständigen SWS einschaltet. Hier kommt ein Zeitschalter,

ein nachtriggerbares Monoflop (MF) zu seiner Ehre. In diesem Augenblick beginnt die Einschaltzeit des MF, und die Einschaltzeit des dazugehörigen SWS geht dadurch über die Dauer des Lichtimpulses hinaus. Erhält ein FW währenddessen einen weiteren Lichtimpuls, wird die Einschaltzeit erneut verlängert (nachgetriggert).

Ermöglicht wird das Ganze durch zwei Elektrolytkondensatoren. Für jede Motorsteuerung brauchen wir einen. So kann der Rover auf verschiedene Begebenheiten flexibel reagieren und je nach Gelände längere oder kürzere Drehmanöver ausführen. Jeder FW wird über die (+)-Schiene vom SWS an *Ea* bzw. *Eb* angeschlossen. Sie schalten so bei jedem Lichtimpuls den zugehörigen SWS ein und laden gleichzeitig die Kondensatoren auf. Dazu befestigen wir je einen Elko 470 µF mit seiner (+)-Seite an *Ea* bzw. *Eb* und mit (-) an die (-)-Schiene. Obwohl die Ladezeit nur kurz und wegen des Widerstands der FW auch schwach ist, liefern sie genügend Strom, um den SWS noch etwas zu versorgen, sobald ein Lichtimpuls zu Ende ist. Dass dabei jeder Elko jeweils nur kurz Strom bekommt, ist sogar ein Vorteil, denn wir können seine Entladezeit nur wenig über die Poti steuern, denn diese müssen so eingestellt sein, dass sich der SWS sofort einschaltet, sobald das vom Randstreifen reflektierte Licht auf den FW fällt. Deshalb ist keine große Zeitspanne möglich, nur ca. 2-3 Sekunden. Aber das genügt völlig. Der Rover dreht sich weit genug vom Randstreifen weg, bevor der Motor erneut umschaltet, und er wieder geradeaus fährt. So findet er sicher seinen Weg.

Er kann sogar komplett wenden, wenn die Fahrbahn entsprechend gestaltet ist, etwa aus schwarzen DIN-A3-Bastelpapierbögen, auf denen man die Fahrbahn mit hellem Klebestreifen darstellt.

Betrachten wir nun die Schaltung genauer. Abb. 10 zeigt den Aufbau für die Steuerung der FW und der Motoren. Der Schwellenwert – da muss ein wenig herumprobiert werden – soll so eingestellt sein, dass der SWS gerade noch aus bleibt, solange der jeweilige FW über die schwarze Fahrbahn wandert. Als Störlichtklappen bitte die mit

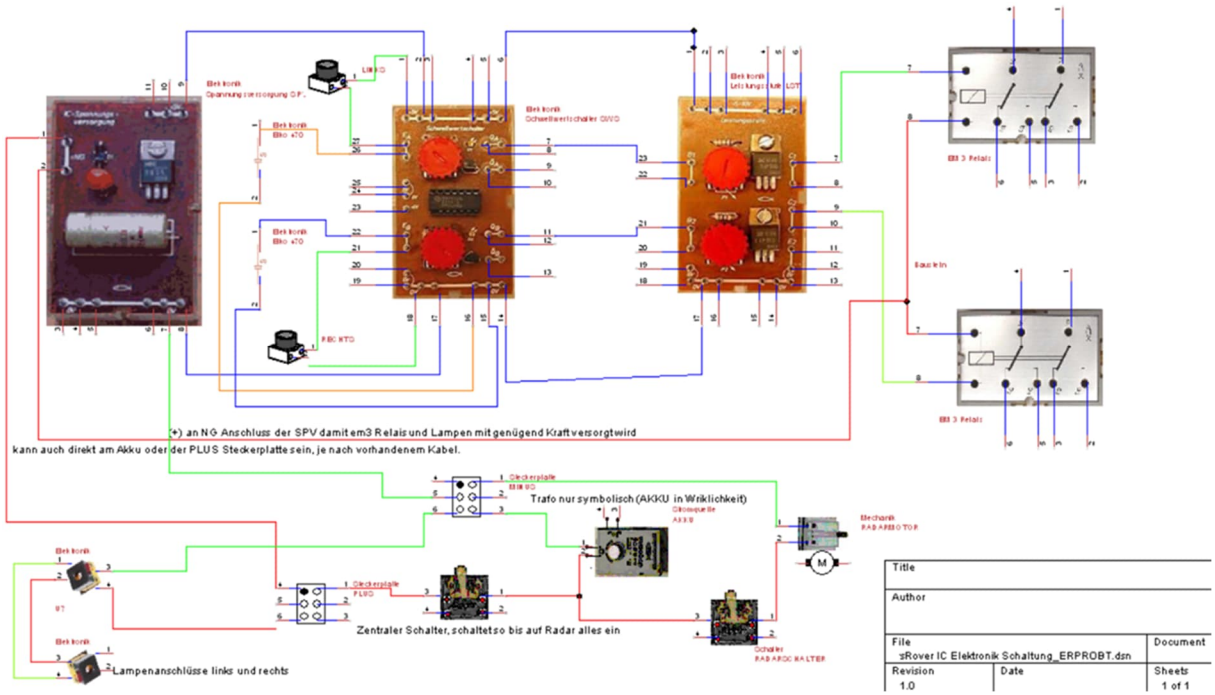


Abb. 10: Die IC-Schaltung

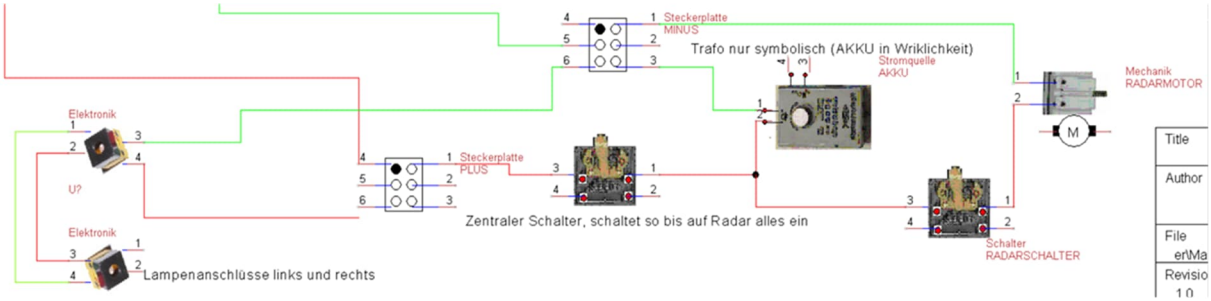


Abb. 11: Die Schalter, Lampen, Radaranschlüsse.

der kleinsten Lichtöffnung verwenden, die genügen völlig. Bekommt ein FW genug Licht, fließt (+) über *Ea* in den Elko 470 µF und lädt ihn auf. Je länger der FW über dem Licht bleibt, desto mehr Strom bekommt er und desto länger ist seine Entladezeit. Folglich kann das Gegenlenken unterschiedlich lang ausfallen. Außerdem benötigen die FW doch etwas unterschiedliche SWS-Poti-Einstellungen, damit jeder bei der gleichen Lichtstärke reagiert. Deshalb entladen sich die Elkos nicht exakt gleich schnell, und darum kann der Rover, auch wenn er frontal an einem Randstreifen stößt, etwas schräg zurücksetzen und seine Spur wiederfinden. Er passt sich also flexibel der Fahrbahn an und fährt nicht starr nach dem immer gleichen Schema. Es sieht sogar so aus, als würde er völlig selbstständig je nach Situation reagieren – und das, obwohl nur einfache Elektronik hinter dem Ganzen steckt.

Die Lampen für die FW (am besten Linsenlampen) sollten maximal hell sein und deshalb direkt am Akku angeschlossen werden. Dasselbe gilt für den Antrieb. Abb. 7 zeigt auch die Verkabelung und die zwei Schalter, mit denen das Radar und der Rover unabhängig voneinander ein- und ausgeschaltet werden können.

Am Schalter für den Rover sind die Motoren, die Lampen und die Elektronik über eine Steckplatte angeschlossen.

Die em3-Relais müssen ebenfalls über den Akku direkt versorgt werden, sonst reicht die Leistung nicht, damit sie anziehen können. Deren (+)-Anschlüsse dürfen also auch nicht an die (+)-Schienen der IC Bausteine angeschlossen werden. Ihre (-)-Verbindungen verlaufen dagegen natürlich über die LST Anschlüsse als gemeinsame Masse.

Mit dem zweiten Schalter wird das motorisierte Radar ein- und ausgeschaltet. Auch das hängt direkt am Akku, was aber nicht zwingend sein muss. Man kann es auch an die (+)-Schiene der IC-Elektronik-Bausteine anschließen (dann dreht es nur langsamer).

Da Elektronik, Lampen und Motoren gleichzeitig an- und ausgeschaltet werden, wird der erste Schalter zwischen (+) am Akku und einer Steckplatine geschaltet, an dem alles angeschlossen ist. Davon brauchen wir zwei Stück. Sie helfen, alle Steckplätze an einer Stelle zu bündeln und alles bequem zu verkabeln. Abb. 11 zeigt den unteren Teil mit den Schaltern etwas vergrößert.

Die Motorsteuerung

Die Relais sind so an die Motoren anzuschließen, dass sie für Vorwärtsfahrt sorgen, solange beide FW über der dunklen Fahrbahn liegen und die SWS damit ausgeschaltet sind. Kommt der rechte FW von der Fahrbahn auf den Randstreifen, schalten sich der entsprechende SWS und das über die LST angeschlossene Relais ein.

Das muss in dieser Situation das für den linken Motor sein, der damit umgepolt und auf Rückwärtsfahrt geschaltet wird. Der andere läuft normal weiter, und so beginnt

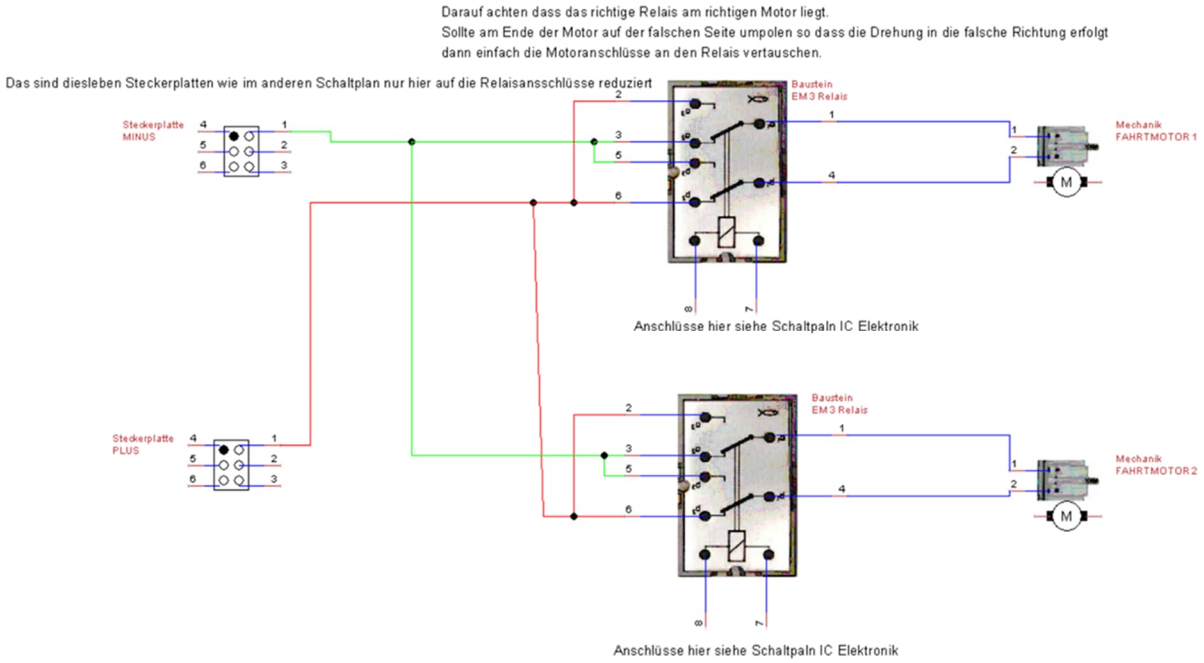


Abb. 12: Motorsteuerung

der Rover sich vom Randstreifen wegzudrehen. Kommt der linke FW auf den Randstreifen, schaltet sich das Relais für den rechten Motor ein und polt um auf Rückwärtsfahrt. Abb. 12 zeigt die Relaisanschlüsse.

Damit ist der Rover startklar – fehlt nur noch eine passende Fahrbahn. Abb. 11 zeigt, wie die aussehen könnte. Sie ist so gestaltet, dass der Rover am Ende eine 180°-Wende vollzieht und zum Anfang zurückfährt. Dort bleibt er stehen bzw. verbleibt in der Parkbucht. Er fährt nicht über den Randstreifen.

Viel Spaß beim eventuellen Nachbauen.

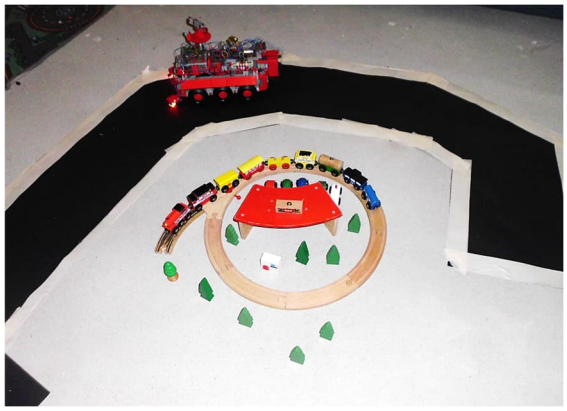


Abb. 13: Fahrbahn

Quellen

[1] fischertechnik: *Elektronik*. Baukasten und Anleitung auf ft-datenbank.de, 1981.

Modell

Seilroboter mit fischertechnik

Florian Bauer

In diesem Beitrag geht es um Seilroboter, ihre Eigenschaften, wie sie aufgebaut sind und wie sie funktionieren. Es werden ein fischertechnik-Modell mit drei translatorischen Freiheitsgraden und die zugehörige Kinematik vorgestellt.

Hintergrund

Seilbetriebene Mechanismen zur Bewegung und Positionierung von schweren Lasten wurden bereits in der Antike benutzt. Solche Mechanismen wurden bereits beim Bau der Pyramiden eingesetzt. Aber auch heutzutage sind sie in Form von Kränen allgegenwärtig.

Durch die heute vorhandenen Möglichkeiten, Ansteuerungen mittels Computer zu automatisieren, ist es naheliegend, Seilzugmechanismen zum Antrieb von Robotern zu verwenden. Dies hat zu einem neuen Typ von Robotern bzw. Manipulatoren geführt: den *Seilrobotern* oder im Englischen auch *„Cable bots“* genannt. Eine gute bebilderte Übersicht und die Geschichte ihrer Entwicklung findet sich zum Beispiel in [9].

Ein bekanntes Beispiel, das jeder schon mal im Einsatz erlebt hat, ist die sogenannte *Skycam*. Dies ist ein kabelbetriebenes Kamera-System, das bei Groß- und Sportveranstaltungen eingesetzt wird, um Aufnahmen aus der Vogelperspektive zu machen.

Seilroboter können mit relativ geringem konstruktiven Aufwand große Arbeitsbereiche abdecken. Ihre Seile werden von Winden aufgewickelt, die im festen Bezugssystem montiert sind. Dadurch erreicht man ein geringes bewegtes Gewicht, was schnelle Bewegungen ermöglicht. Einen Eindruck von den mit Seilrobotern erzielbaren Geschwindigkeiten vermittelt das Video in [2].

Natürlich haben Seilroboter auch einige Nachteile. Da Seile nur auf Zug beanspruchbar sind, muss der zu bewegende Endeffektor oder die Last mit Seilen, die in verschiedene Richtungen wirken, in Position gehalten werden. Um eine höhere Stabilität zu erreichen, wendet man gerne ein überbestimmtes Verspannungsschema an.

Da die Seile durch den Arbeitsbereich laufen, dürfen sich keine Hindernisse wie Personen darin befinden, die durch die Seile bei ihrer Bewegung getroffen werden könnten. Dies muss unbedingt bei der Arbeitssicherheit berücksichtigt werden, wie auch die Möglichkeit, dass ein unter Last stehendes Seil plötzlich reißen und geschoßartig unkontrolliert in der Gegend herumfliegen kann.

Komponenten

Für einen Seilroboter benötigt man:

- Starre Seile mit möglichst geringer Elastizität
- Stabile Aufnahme-Gestelle für die Seilrollen
- Beweglich gelagerte Seilrollen, die die Seile in den Arbeitsbereich leiten
- Seilwinden, die die Seile gleichmäßig auf- und abwickeln können

- Vorrichtungen zur Messung der aufgewickelten Seillängen
- Optionale Vorrichtungen zur Messung von Seilspannungen
- Motor-Treiber für die Windenmotoren
- Eine Microcontrollersteuerung, die den Endeffektor unter Verwendung der inversen Kinematik zur Zielposition steuert.

Die Inverse Kinematik ist die Anwendung des kinematischen Modells des Seilroboters, um aus einer gegebenen Zielposition die erforderlichen Seillängen bzw. Windenumdrehungen zu berechnen.

Arten von Seilrobotern

Ein sehr guter Überblick über die Arten und Funktionsweise von Seilrobotern findet sich in der Dissertation von Tobias Bruckmann [3].

Seilroboter werden nach ihren Bewegungsfreiheitsgraden bzgl. Verschiebung (*Translation* T) und Drehung (*Rotation* R) eingeteilt. Je nach Typ wird eine unterschiedliche Zahl von Seilen benötigt.

Wenn man die Schwerkraft nutzt, kann i. A. ein Seil eingespart werden: Ein einfaches Beispiel ist ein Kran, bei dem das Seil durch die Last gespannt wird. In der Horizontalen bräuchte man dagegen zwei Seile, um sie zu bewegen.

fischertechnik-Modell

Seilroboter-Geometrie

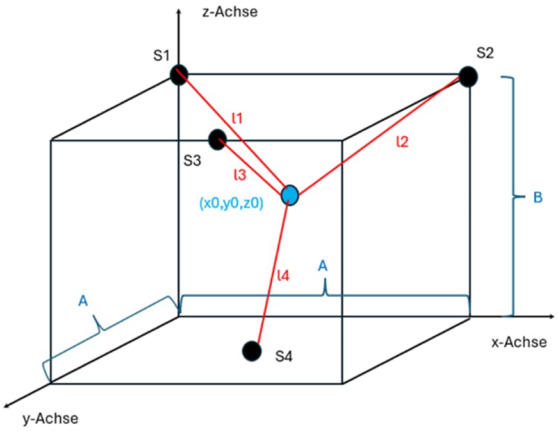


Abb. 1: Schemazeichnung eines 3T-Seilroboters mit punktförmigem Endeffektor. Die Seile werden an punktförmigen Seilrollen S1 ... S4 eingeleitet

Für mein fischertechnik-Modell habe ich mich für einen 3T-Mechanismus entschieden (Abb. 1). Das Gerüst ist ein Würfel von ca. 50 cm Kantenlänge. An der Oberkante werden drei Seile über Umlenkrollen in den Arbeitsbereich geführt, die an einem Endeffektor befestigt sind. Ein weiteres Seil kommt von der Mitte der Unterseite des Würfel-Gestells. Dieses Seil könnte man sich zwar wegen der Schwerkraft sparen. Jedoch verleiht das vierte Seil dem Modell mehr Stabilität.

Typ	Translations-Freiheitsgrade	Rotations-Freiheitsgrade	Arbeitsbereich	End-Effektor	Anzahl der Seile (typisch)
1T	1	-	linear	Punkt	2
2T	2	-	planar	Punkt	3
3T	3	-	räumlich	Punkt	4
1R2T	2	1	planar	Körper	4
2R3T	3	2	räumlich	Stab	6
3R3T	3	3	räumlich	Körper	8

Tab. 1: Klassen einiger Seilroboter

Der reale Endeffektor ist allerdings nicht punktförmig, sondern aus technischen Gründen ein kleines Prisma mit dreieckiger Grundfläche (Gleichseitiges Dreieck mit Kantenlänge C) und Dicke d.

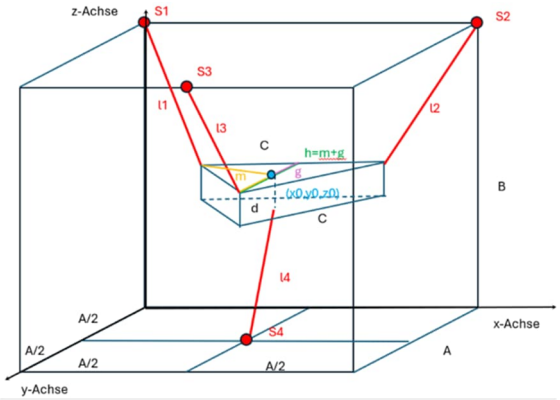


Abb. 2: Schemazeichnung eines „realen" 3T-Seilroboter mit prismenförmigem Endeffektor mit gleichseitig-dreieckiger Grundfläche mit Kantenlänge C und Dicke d

Arbeitsbereich

Der stabile Arbeitsbereich („Wrench Controllable Workspace", S. 60 in [3]) des Roboters ist bestimmt durch die Pyramide durch die Punkte S1, S2, S3 und S4. Innerhalb dieser Pyramide sind alle Seile immer gespannt. Lässt man auch die Schwerkraft als Akteur und ungespannte Seile zu, vergrößert sich der Arbeitsbereich (Arbeitsbereich des stabilen Gleichgewichts, S. 61 in [3]) auf ein Prisma mit dreieckiger Grundfläche S1, S2 und S3 bis zum Boden des Gestells.

Beispielsweise kann man Seile 1 und 3 so einstellen, dass der Endeffektor in der Mitte der auf der auf die Grundfläche senkrecht nach unten projizierten Verbindungsline von S1 und S3 am Boden ist. Die Seile 2 und 4 müssen dann allerdings vollkommen schlaff sein. In der Praxis erreicht man diesen Punkt nur, wenn das Eigengewicht des Endeffektors sehr viel größer als das Gewicht des Seils 2 ist. Ansonsten zieht das Eigengewicht von Seil 2 den Endeffektor in Richtung Mitte.

Kinematisches Modell

Das kinematische Modell kann aus Abb. 2 einfach durch Anwendung des Satzes von Pythagoras in drei Dimensionen abgeleitet werden und stellt bereits die inverse Kinematik dar. Zu einem gegebenen Zielpunkt (x_0, y_0, z_0) kann man damit die Seillängen $l_1 \ldots l_4$ berechnen:

$$l_1^2 = (B - z_0)^2 + (y_0 - g)^2 + (x_0 - \frac{C}{2})^2$$

$$l_2^2 = (B - z_0)^2 + (y_0 - g)^2 + (A - x_0 - \frac{C}{2})^2$$

$$l_3^2 = (B - z_0)^2 + (A - y_0 - m)^2 + (\frac{A}{2} - x_0)^2$$

$$l_4^2 = (z_0 - d)^2 + (y_0 - \frac{A}{2})^2 + (\frac{A}{2} - x_0)^2$$

Die Dreieckshöhe des Endeffektor-Dreiecks ist damit:

$$h = \sqrt{\frac{3}{4}} \cdot c$$

Der Abstand der Ecke zur Mitte des Dreiecks beträgt:

$$m^2 = \frac{c^2}{2 - 2 \cdot \cos(120°)}$$

Der Abstand der Seilmitte zur Mitte des Dreiecks beträgt:

$$g = h - m$$

Die Vorwärts-Kinematik erfordert die Auflösung der o. g. Ausdrücke nach den Zielpunkt-Koordinaten (x_0, y_0, z_0) oder die numerische Lösung von drei simultanen Gleichungen. Die Vorwärts-Kinematik wird für die Steuerung des Modells nicht benötigt, wohl aber, wenn man die Kinematik in einem Computermodell simulieren und visualisieren will. Für die numerische Lösung der Gleichungen für die Vorwärts-Kinematik eignet sich die *rootfinder*-Funktion der Python-Library *CasADi* [10], einer numerischen Bibliothek mit algorith-

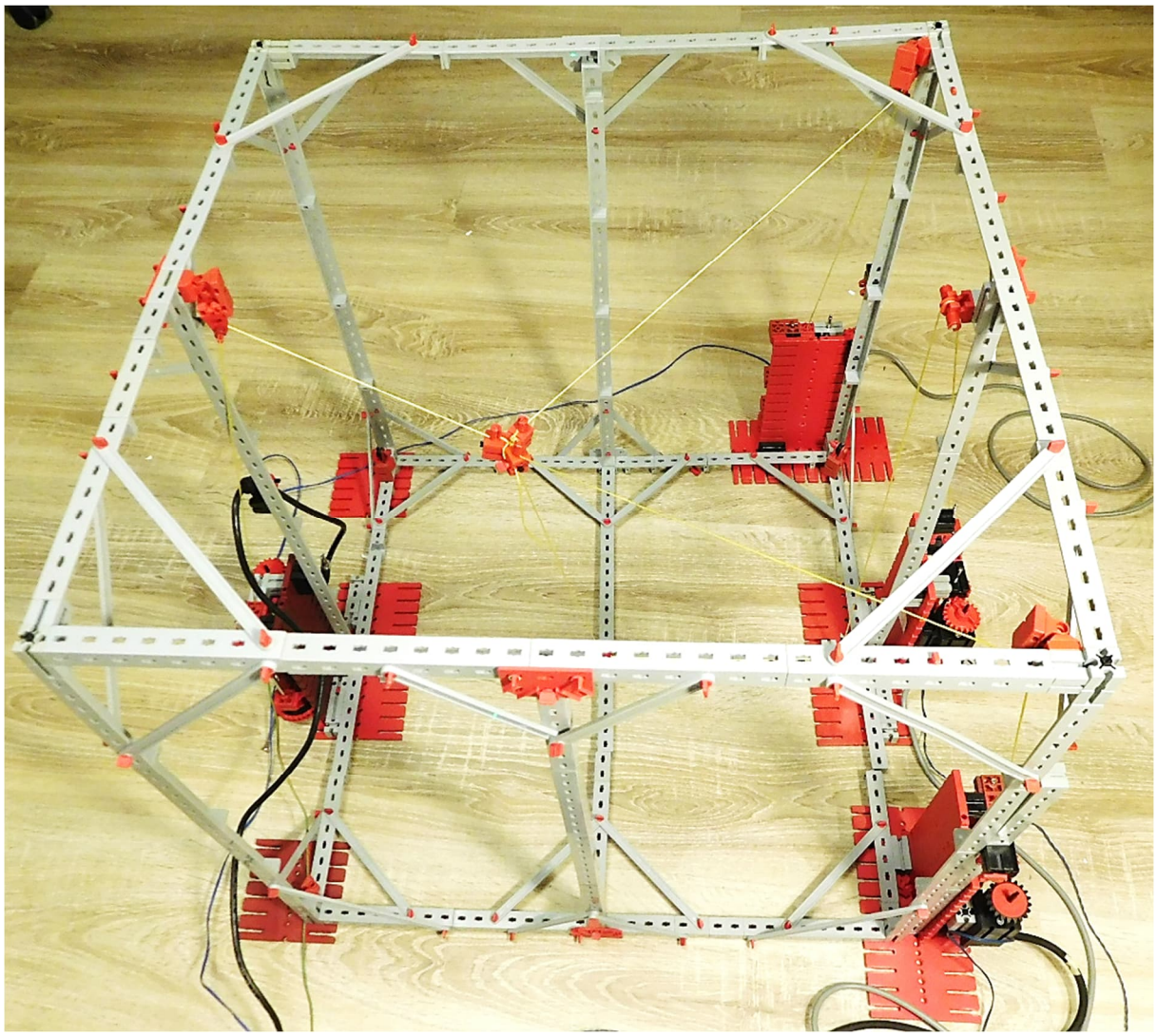


Abb. 3: fischertechnik-Modell des 3T-Seilroboters von oben

mischer Differentiation. Das bedeutet, dass Ableitungen von Funktionen, die für numerische Verfahren benötigt werden, nicht durch Differenzenquotienten angenähert werden, sondern durch symbolische Differentiation, die von der Library programmatisch vorgenommen wird. In der Code-Beilage zu diesem Beitrag ist ein Python-Skript *seilbot3t.py* enthalten, aus der die Anwendung ersichtlich ist.

fischertechnik-Aufbau

In Abb. 3 Ist das fertige fischertechnik-Modell zu sehen. Das Würfel-Gestell ist aus verstrebten Statik-Trägern aufgebaut. Die Verstrebungen sind sehr wichtig, da die Seilwinden starke Kräfte erzeugen, die das Gestell sonst stauchen. Der Rahmen ist so universell, dass sich verschiedene Seilroboter-Geometrien ausprobieren lassen.

Endeffektor

Abb. 4 zeigt den dreieckigen Endeffektor. Die oberen Seile werden mit Klemmringen gehalten; das untere Seil läuft von unten nahezu mittig in den Endeffektor und wird mit einem Statik-Riegel an der Seite fixiert.

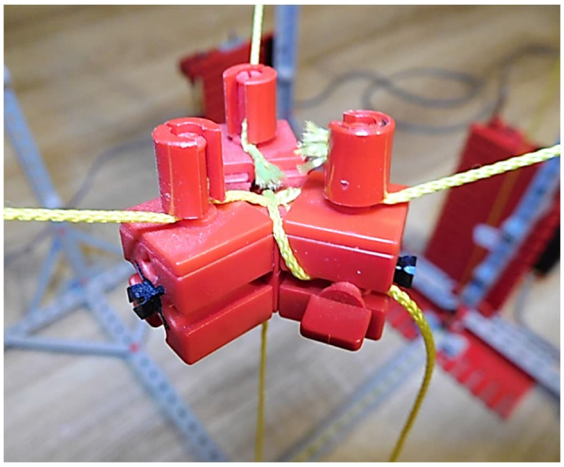


Abb. 4: Dreieckiger Endeffektor des 3T-Seilroboters

Seilrollen-Halterungen

In Abb. 5 ist eine der schwenkbaren Seilrollenhalterungen zu sehen. Durch das Gelenk wird die Seilrolle in Richtung des Seils ausgerichtet. Hier ließe sich auch eine Sensorhalterung für einen Abstandssensor montieren, die mitschwenkt und deren Kippwinkel zur Basisebene von der Seilrichtung bestimmt wird, so dass der Abstandssensor immer auf den Endeffektor ausgerichtet ist.

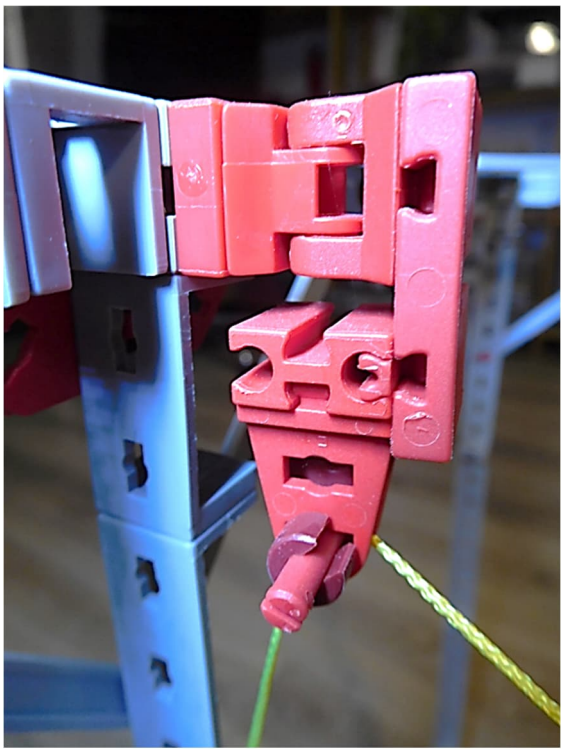


Abb. 5: Schwenkbare Seilrollenhalterung

Der endliche Durchmesser einer Führungsrolle führt dazu, dass die effektive Seillänge vom Winkel des Seils abhängt. Im Prinzip kann man diesen Effekt rechnerisch berücksichtigen (S. 20 in [3]). Allerdings ist die kinematische Berechnung für punktförmige Seilrollen einfacher. Deshalb habe ich mich dazu entschieden, diesen Effekt zunächst nur durch Verzicht auf eine Führungsrolle zu minimieren: Das Seil läuft direkt über eine Achse. In Abb. 6 ist die Seilrolle für das vierte Seil in der Bodenmitte des Gestells zu sehen.

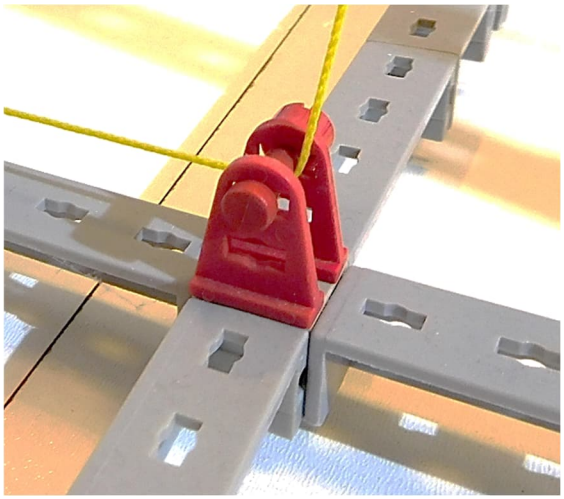


Abb. 6: Seilrollenhalterung an der Bodenmitte

Seilwinden

In Abb. 7 ist eine der vier Seilwinden gezeigt. Sie wird von einem XM-Encoder-Motor angetrieben. Für eine gleichmäßige Aufwicklung dient eine schnecken-gesteuerte Seilführung. Alle Encoder-Motoren wurden mit einem zusätzlichen Quadraturausgang ausgestattet, um Fehlzählungen zu eliminieren [11].

Mikrocontroller-Steuerung

Zur Ansteuerung wird ein Arduino-MEGA mit einem Adafruit Motor Shield [4] verwendet, das bis zu vier Motoren treiben kann. Für die Zählung der Impulse der Encoder-Motoren verwende ich zwei ic-MD Bausteine [5], die über SPI ausgelesen werden können. Die Bausteine sind sehr praktisch, da man sich nicht mit Interrupts

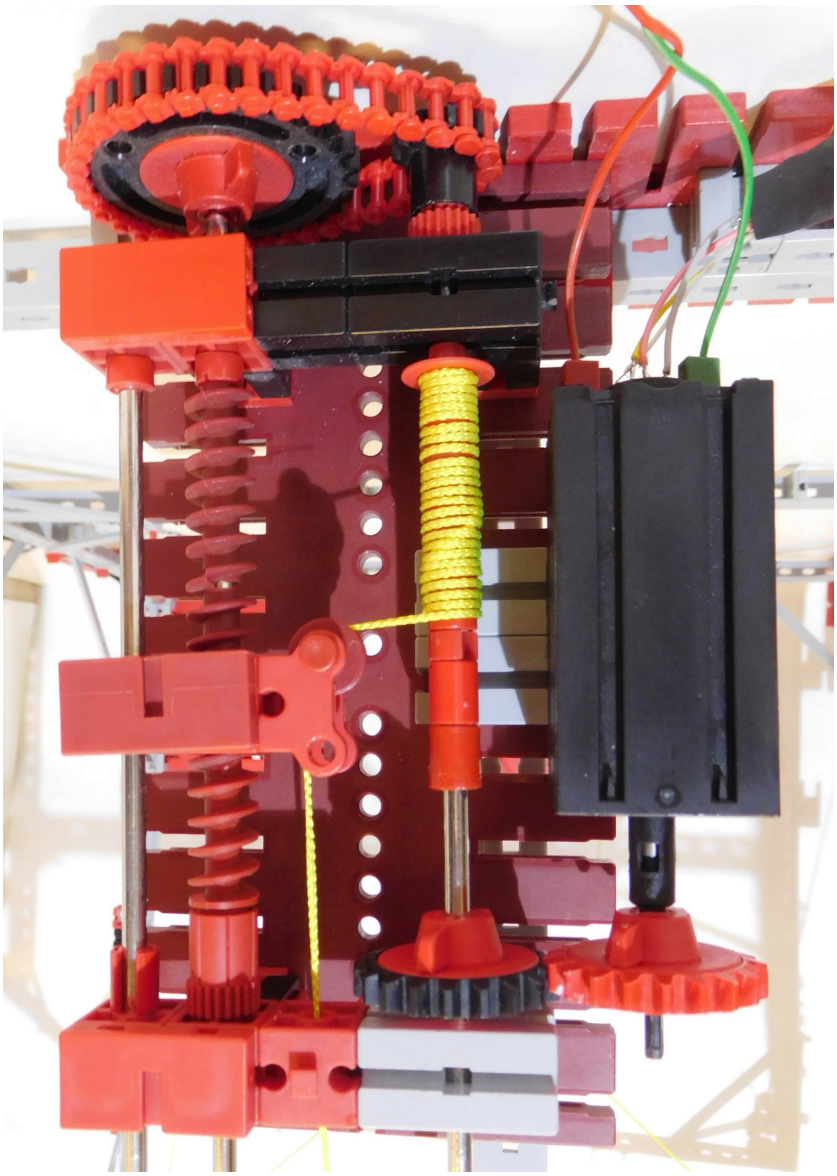


Abb. 7: Motorisierte Seilwinde mit Seilvorschub für gleichmäßige Wicklung, angetrieben von einem XM-Encoder-Motor

troller hinbekommen kann. Ich habe das allerdings nicht selbst ausprobiert und es würde mich interessieren, ob es jemand gelingt.

Wenn man die Messung der Seilwinden-Umdrehungen mit Magnet-Encodern realisieren würde, könnte man auch normale günstigere XM-Motoren verwenden. Diese Option hatte ich für den Fall vorgesehen, dass ich ein Modell mit noch mehr Motoren betreiben möchte.

Das Steuerprogramm

Das Auslesen der ic-MD-Zähler-ICs und die Kinematik des Modells sind in Arduino-Libraries ausgelagert. Für die Positionsregelung verwende ich eine PID-Regler-Library namens *PID_v2_bc*.

Die Kinematik-Library für den 3T-Seilroboter übernimmt auch die Umrechnung der Seillängen in die Motor-Counts für die Winden. Die Programme stelle ich mit diesem Beitrag zur Verfügung.

Das Steuerprogramm unterstützt folgende Kommandos:

- **- <motor number>**
  Seil nachgeben um 500 counts
- **+ <motor number>**
  Seil aufwickeln um 500 counts
- **Z <x0>,<y0>,<z0>**
  Ausgangsposition festlegen

oder Pin-Polling herumquälen muss – immerhin muss man acht Signale auswerten.

Die Schaltung ist auf einem Protoboard aufgesteckt. Die größte Herausforderung stellt hierbei die Verkabelung dar. Über fest montierte Lüsterklemmen und fischertechnik-Rollenhalter habe ich die Anschlussdrähte und Kabel zugentlastet.

Ich nehme an, dass man den Aufbau auch mit einfachen Encoder-Motoren ohne Quadraturausgang und Auslesen via Mikrocon-

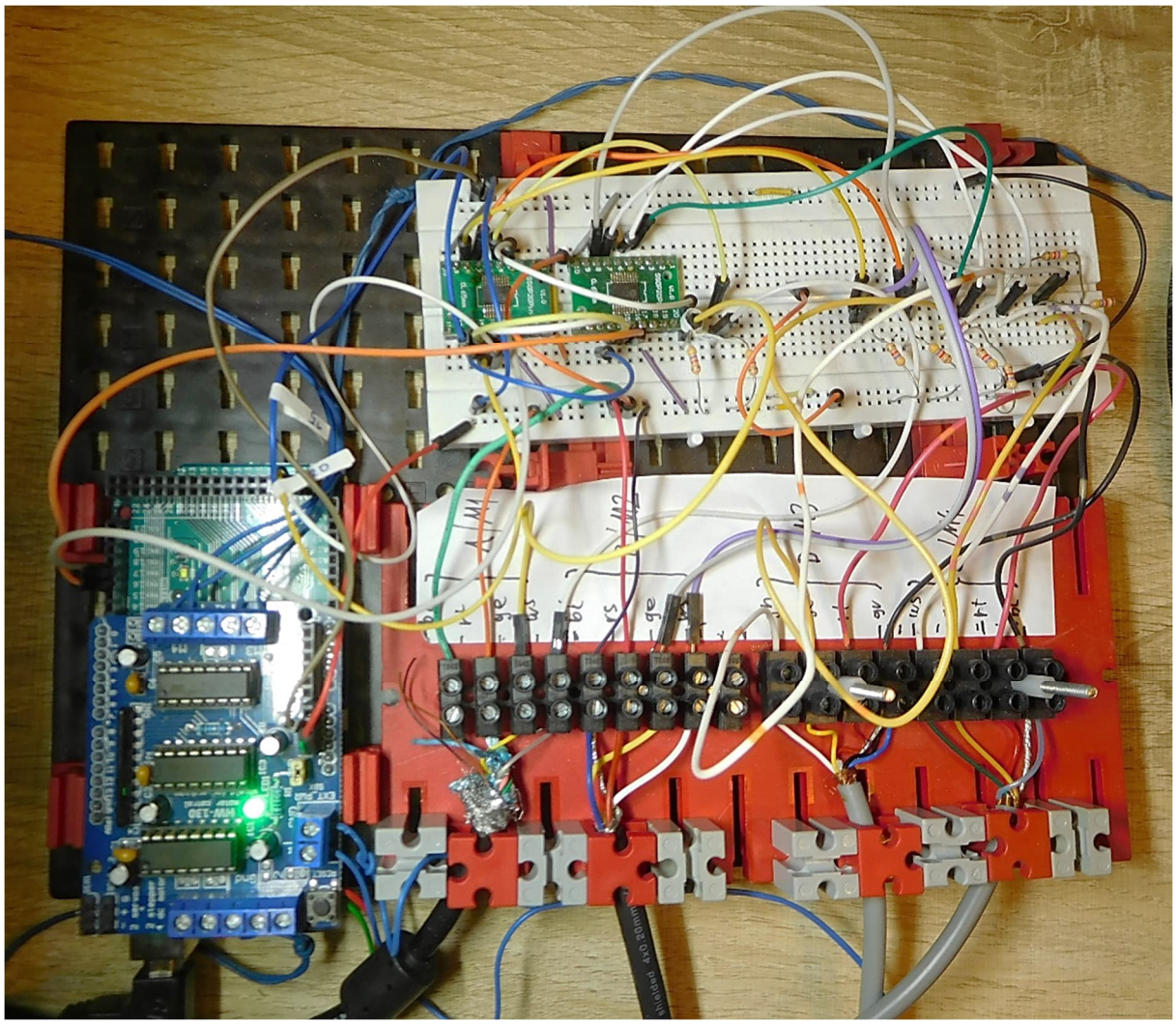


Abb. 8: Microcontrollerschaltung zur Ansteuerung des Seilroboters. Links: Arduino-Mega mit AFMotor-Shield für 4 Motoren, oben: Steckboard mit ic-MD-Zählerbausteinen; unten: Verteilerleiste mit Lüsterklemmen und Zugentlastung für die Encoder-Leitungen

- **z**
  Soll-Position als Ist-Position übernehmen
- **m <x>,<y>,<z>**
  Bewegung zu Position (x,y,z)

Für die Einrichtung des Seilroboters sind die +/- Befehle hilfreich. Mit ihnen kann man eine einzelne Seilwinde gezielt auf oder abwickeln lassen. Sind die Seile alle gespannt und die Bewegungsrichtungen korrekt (d. h. Aufwickeln mit „-“, Abwickeln mit „+“) muss man den Roboter kalibrieren. Dazu muss zunächst die Mittenposition (mit den +/- Kommandos) angefahren werden, die Seile gespannt und die Startposition mit dem Z-Befehl in cm eingegeben werden. Danach kann der Seilroboter mit dem m-Kommando neue Sollpositionen anfahren.

Verbesserungspotential

Im zentralen Bereich arbeitet der Roboter zufriedenstellend. Je weiter man nach außen kommt, desto ungenauer wird die Positionierung. Der Endeffektor kippt auch etwas. Durch unterschiedliche Seilspannungen wird das Seil manchmal nicht gleichmäßig aufgewickelt, was dann die Wiederholgenauigkeit der Position etwas mindert. Dies tritt vor allem in den Bereichen auf, bei

denen die Seile nur schlaff hängen. Eine Seilspannungsmessung und -regelung wäre sicher eine lohnende Erweiterung. Man könnte zum Beispiel die Seillängen von S1 bis S3 einstellen und die Regelung so gestalten, dass Seil S4 gespannt wird. Eine direkte Seillängenmessung über Ultraschall oder Laser wäre ebenfalls eine vielversprechende Variante.

Zusammenfassung und Ausblick

In diesem Beitrag habe ich nach einer kurzen Einführung zu Seilrobotern ein fischertechnik-Modell vorgestellt, das drei translatorische Freiheitsgrade besitzt. Es wurden die Kinematik und die notwendigen Komponenten beschrieben, die dazu benötigt werden. Die Modifikation der Encoder-Motoren und die Verwendung des speziellen Zählerbausteins ic-MD soll niemanden abschrecken, diesen Seilroboter zu bauen. Es gibt sicher alternative und einfachere Möglichkeiten – man muss sie nur finden. Es würde mich sehr interessieren, wenn jemand Alternativ-Vorschläge machen würde.

Auf einen sehr interessanten alternativen Seilroboter haben mich Mitglieder des Forums hingewiesen: Der Hexapod von Remadus ist ein eindrucksvolles Modell [6]. Aber auch der Winchbot von HomoFaciens ist eine Beachtung wert [7].

Für sehr ausführliche Informationen zu Seilrobotern, die insbesondere auch die dynamischen Eigenschaften und Kraftberechnungen betrachten, sei verwiesen auf [3, 8, 9].

Quellen

[1] Wikipedia: *Skycam*.

[2] Christopher Reichert: *Extreme Fast Cable-Driven Parallel Robot.* Auf YouTube, 2016.

[3] Tobias Bruckmann: *Auslegung und Betrieb redundanter paralleler Seilroboter*. Dissertation bei der Universität Duisburg-Essen, auf duepublico2.uni-due.de, 2010.

[4] Adafruit Industries: *Adafruit Motor Shield*. PDF auf cdn-learn.adafruit.com, 2025.

[5] iC-Haus GmbH: *iC-MD 48-Bit Quadrature Counter with RS422 Receiver and SPI and BiSS Interface*. Auf ichaus.de.

[6] Martin Romann (remadus): *Hexapod*. Auf ftcommunity.de, 2004.

[7] Norbert Heinz (HomoFaciens): *WinchBot 2 0 - Ein Roboter aus Seilwinden*. Auf YouTube, 2018.

[8] Philipp Tempel: *Dynamics of Cable-Driven Parallel Robots with Elastic and Flexible, Time-Varying Length Cables*. Dissertation bei der Universität Stuttgart, auf elib.uni-stuttgart.de, 2019.

[9] Sen Quian, Bin Zi, Wei-Wei Shang, Qing-Song Xu: *A Review on Cable-driven Parallel Robots*. Auf researchgate.net, 2018.

[10] CasADi (open-source tool for nonlinear optimization and algorithmic differentiation): https://web.casadi.org/

[11] Florian Bauer: *Quadratur-Encoder für fischertechnik-Encoder-Motoren*. ft:pedia 2/2023, S. 41–46.

Modell

Großprojekt Seilbahn (Teil 11): Die Zukunft

Tilo Rust

Diese Beitragsserie begleitet das Großprojekt „Kuppelbare Einseilumlaufbahn / Doppelmayr (10-MGD)“ von Anfang an bis…

Derzeit konzentrieren wir uns auf die Verschalung und Transportsicherung der Anlage. Der Sinn dahinter ist simpel: Eine Anlage, die nicht sicher transportabel ist (vgl. BUGA23) und erheblichen Arbeitsaufwand erfordert, um ausgestellt zu werden, ist nicht rentabel für Ausstellungen, die nur wenige Tage dauern. Und eine Anlage, die nicht vorgeführt werden kann und nicht ausgestellt wird, ist nicht rentabel für etwaige Sponsoren. Ohne Sponsoren gibt es kein Geld zum Weiterbauen, und ohne ein durchdachtes Transportsystem gibt es kein Interesse von Veranstaltern, die Anlage auszustellen.

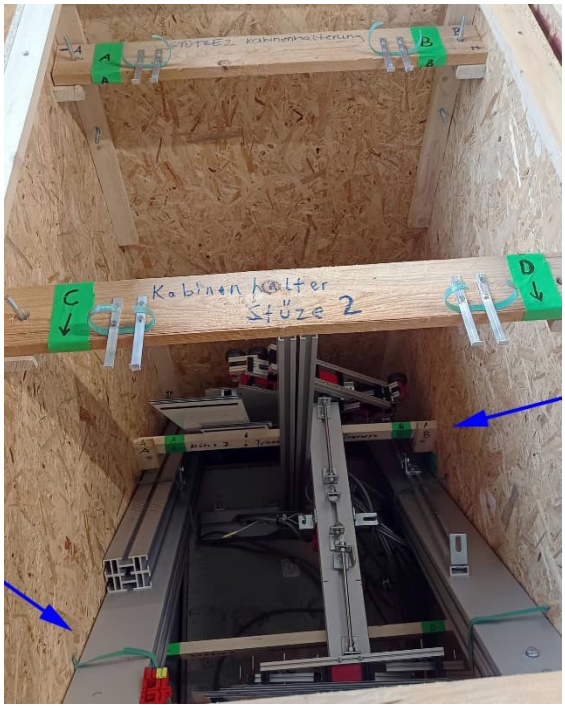


Abb. 1: Eine der Transportboxen

Verschalung und Transport

Mit Hilfe eines befreundeten Schreiners und eines Schweißers konnten wir die Transportboxen der Stützen erweitern und Sicherungen für bisher lose gelagerte Teile einbauen, dazu Platz für bis zu 20 Gondeln und Dekomaterial.

Abb. 1 zeigt die offene Transportbox einer Stütze von oben. Oben erkennt man zwei eingelegte Bretter, an denen vier Gondeln eingehängt werden können. Darunter sind rechts und links die Industrieprofile gesichert, mit denen die Stütze ihre eigentliche Höhe erreicht. Unten quer (blaue Pfeile) die Halterung, auf der die Traverse der Stütze befestigt ist.

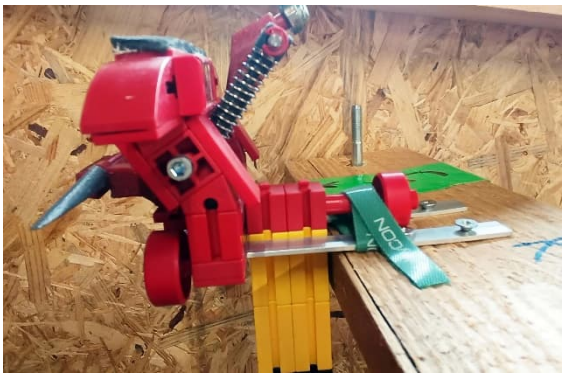


Abb. 2: Detail der Gondelhalterung

Auf den eingehängten Brettern sind Alu-Haken befestigt, die wie eine Gabel die Gondel am Klemmenwagen halten (Abb. 2). Ursprünglich waren Stahlseile vorgesehen, an denen die Klemmen eingehängt werden sollten. Das hätte aber den großen

Nachteil, dass alle Stöße direkt auf unser wertvollstes Bauteil gegangen wären: die Klemme selbst. Die neue Konstruktion ist aber nicht nur stabiler, sondern leitet die Kräfte an der stabilsten Stelle ein.

Dann nahmen wir uns die Antriebsstation (AST) vor. Hier wurden die Boxen komplett neu gebaut (mit vorhandenem Holz). Denn zur Cable Car World 2024 (CCW24) in Essen [3] mussten diese mit über 200 Spaxschrauben zusammengezimmert und vor Ort wieder auseinandergeschraubt werden – dasselbe für die Rückreise. Jetzt ist es möglich, die Station ganz alleine in weniger als einer Stunde komplett zu verpacken.

Weitere Transportsicherungen für Dekomaterial und bisher lose Teile, z. B. die Dibond-Abdeckplatten, die nun fest verschraubt sind, runden die Sache ab und beschleunigen den Verpackungsvorgang.

Der Steuerstand bekam nun auch eine eigene Box. Monitor und anderes Zubehör finden auch hier sicheren Platz. Das Verpacken kann von einer Person in weniger als 15 Minuten erfolgen.

Anhand der Beschriftungen lassen sich die Boxen nicht nur identifizieren, sondern mit Hilfe einer Symbolanleitung auch sicher zusammensetzen und öffnen [2].

Abb. 3: Transportboxen

Die neue Transportbox der Antriebsstation kann in 15 Minuten geschlossen und noch schneller geöffnet werden – von einer Person. Links in Abb. 3 sieht man noch unsere Werkstattpalette, die ebenfalls neu beschriftet wurde und einen geordneten Platz für schnell benötigte Arbeitsmittel bekam.

Die nächsten Schritte sind nun der Umbau der Transportboxen für die Bergstation (UST), die zwar fast fertig ist, aber mit Hilfe neu geschweißter Halterungen auch sehr viel einfacher und schneller zu verpacken sein wird. Dann kommt die Stütze 4 an die Reihe, für deren Transportbox ich noch einen Spender suche [5], um das Holz zu beschaffen.

Auf der CCW24 haben wir bewiesen, dass wir mit vier Leuten in der Lage sind, das Modell innerhalb von zehn Stunden auszupacken und aufzubauen sowie in etwa acht Stunden wieder abzubauen und zu verpacken (siehe Videobeitrag auf unserem YouTube-Kanal [4]). Allerdings noch ohne Steuerstand und Stütze 4, ohne Arbeitspause und mit Hilfe hunderter Spax-Schrauben…

In Zukunft soll die Anlage mit zwei Personen innerhalb von acht Stunden komplett ausgepackt und in Funktion sein und in sechs Stunden sicher wieder verpackt werden können. Und das ohne Akkuschrauber oder fischertechnik-Fachkenntnisse. Denn wenn das Modell leicht zu verpacken und transportieren ist, kann es von Speditionen ohne unser Zutun an jeden Ort der Welt gebracht werden. Dann kann ein Museum den Platz kurzfristig freigeben und nach einer Zwischennutzung wieder mit dem Modell belegen. Das schafft Möglichkeiten für Museen (oder andere Ausstellungsorte) und für uns, das Modell an fernen Orten zu präsentieren.

Die Zukunft

Dass ein Projekt wie diese Seilbahn riesig ist, das wussten wir von Anfang an. Dass es *so* riesig wird, das konnten wir höchstens erahnen. Dennoch war es machbar und ist in der Art, wie wir es realisierten, viel mehr als nur eine Seilbahn: Es ist eine komplette Seilbahn-Ausstellung.

Über 6.500 Arbeitsstunden hat das Team in das Großprojekt investiert. Es hat Unglaubliches geleistet und das komplexeste fischertechnik-Modell gebaut, das je entstanden ist. Es hat unglaubliches Improvisationstalent und Durchhaltevermögen bewiesen – auch unter widrigsten Bedingungen wie der Hitzewelle auf der BUGA23.

Und die Seilbahn läuft. Geschätzte weitere rund 100 Stunden Tüftelei wären erforderlich, um die letzten technischen Herausforderungen zu meistern – manchmal bleiben Gondeln in der Kurve hängen, ab und an setzt die Steuerung aus und die Türen sind noch nicht angebracht. Dennoch ist die Seilbahn inzwischen so attraktiv und sehenswert, dass auf der CCW24 die Gespräche mit den Fachbesuchern unbeeindruckt von einem Stillstand wegen eines technischen Defekts weitergeführt wurden.

Gegen Ende der Süd-Convention 2024 erfuhren wir, dass der Platz im Fördertechnikmuseum gebraucht wird und wir bis Ende 2024 einen anderen Ort für unser Seilbahn-Modell finden müssten. Da wir auf die Schnelle keinen Lagerort finden konnten, wurde uns die Möglichkeit eingeräumt, das zurückgebaute Modell bis Mitte 2025 im Museum zu lagern. Wie aber geht es danach weiter?

Verschrotten?

Verschrotten kostet Geld. Verschrotten ist Arbeit, denn die fast 100.000 fischertechnik-Teile müssten von Industrieprofil und anderen Komponenten getrennt werden. Und: Verschrotten bricht einem Modellbauer das Herz.

Verschrotten, obwohl wir vor einem Jahr noch so viel vorhatten? Mehrere Ausstellungstermine wie eine internationale Seilbahn-Fachmesse und die Nürnberger Spielwarenmesse waren bereits gebucht, teilweise Sponsoren gefunden und die Logistik geplant. Verschrotten müssten wir auch die Ideen und kreativen Lösungen der Teammitglieder, um die Seilbahn mit zusätzlichen Features auszustatten.

Verschrotten ist keine Option.

Museum gesucht

Daher suchen wir einen neuen Platz für unser Modell. Das ist natürlich aufgrund der Größe und der speziellen Thematik nicht unbedingt einfach.

Abb. 4: Wir suchen ein Museum!

Auf YouTube habe ich daher ein Video mit dem Titel „Suche Museum“ [6] veröffentlicht, das die Anlage und Möglichkeiten zeigt und gerne Museen oder Firmen mit Ausstellungsräumen gezeigt werden darf.

Ein paar Kontakte hatten wir bereits:

- Zwei Kuratoren des Technoseum Mannheim besuchten mich, um sich das Modell und die Optionen vor Ort anzuschauen. Prinzipiell besteht Interesse an einer Ausstellung, jedoch gibt es derzeit keinen ausreichenden Platz.
- Das Deutsche Museum in München hat leider abgelehnt.
- Die Technikmuseen in Sinsheim und Speyer haben wir bisher nicht erreicht: An der Kasse wurde ich abgewiesen und die mehrfachen E-Mail-Anfragen blieben unbeantwortet. Wer es also über die Vorzimmerschwelle schafft, dem wäre ich dankbar, wenn ein Kontakt zustande käme. Ein Verbleib der Seilbahn in Sinsheim wäre jedenfalls der optimale Fall für uns.

Folgende drei Optionen sehe ich zurzeit:

Idealfall

Du hast ein Museum oder Ausstellungsfläche mit Publikumsverkehr rund um den Rhein-Neckar-Raum, auf der wir unser Modell aufbauen können? Dann bekommst du nicht nur die Seilbahn und eine komplette Ausstellung, die wir auch mit deinen Inhalten branden können. Du bekommst eine Attraktion, an der ständig weitergearbeitet wird, die mit deinen Besuchern interagiert und Klein und Groß fasziniert. Wir kommen mit einer eigenen Werkstatt und benötigen nicht viel: Ca. 25 m × 3 m Platz, bei einer Raumhöhe von mindestens 2,20 m (idealerweise > 5 m), zwei Steckdosen und ein trockenes Dach, nicht kalt und nicht in der prallen Sonne.

Minimal-Lösung

Uns würde auch ausreichen, wenn wir auf ca. 8 m × 3 m (Raumhöhe unwichtig) nur die Talstation und die erste Stütze aufstellen könnten. Wir würden ein stehendes Seil einhängen und dieses an einer Wand / Holzwand enden lassen. Dann würde die Anlage zwar nicht fahren, aber wir könnten daran weiterarbeiten und sie stünde dem Publikumsverkehr offen.

Das könnte zum Beispiel der Empfangsbereich einer Firma sein, die einen Eyecatcher sucht, oder das Rathaus einer jener Städte in Deutschland, die planen, eine Seilbahn im urbanen Nahverkehr einzusetzen. Hier wären wir am genau richtigen Ort, als Botschafter den Bürgern die Sinnhaftigkeit dieses Verkehrsmittels zu zeigen. Die Ausstellung hierfür ist individuell gestaltbar.

Notlösung

Als Notlösung benötigen wir eine Lagerfläche, auf der wir 18 Euro-Paletten trocken und sicher lagern dürfen (möglicherweise auch auf mehrere Orte verteilt). Wir können dafür weder Miete zahlen, da wir kein Budget haben, noch können wir dort am Modell weiterbauen oder es zukünftigen Sponsoren oder Museen präsentieren. Als Übergangslösung wäre das aber immer noch besser als eine Verschrottung.

Herausforderung Logistik

Spätestens Mitte 2025 muss die Anlage auf einen LKW verfrachtet und an einen anderen Ort gebracht werden – wo auch immer dieser sein mag. Das ist nicht schwer, schließlich ist alles sicher in Kisten verpackt, ordentlich beschriftet (bestempelt) und kann mit Hubwagen bzw. Gabelstaplern verladen werden. Das Problem: Das Projekt hat kein Geld.

Wiederbelebung

In einem Punkt bin ich mir jedoch schon jetzt sicher: Sobald wir einen ordentlichen Ausstellungsplatz haben und die Anlage steht, dann wird es auch gelingen, das Projekt wiederzubeleben und weiter zu tüfteln, bis die Seilbahn sauber fährt und die Anlage glänzt. Daher bitten wir euch um Unterstützung: **Habt ihr Kontakte oder Ideen für eine Lösung oder einen Sponsor?**

Eines ist jedenfalls klar: Verschrotten ist keine Option.

Referenzen

[1] Tilo Rust: *Großprojekt Seilbahn (Teil 9): Energie!* ft:pedia 1/2024, S. 40–50.

[2] Tilo Rust: *Großprojekt Seilbahn (Teil 10): Messefertig*. ft:pedia 2/2024, S. 46–68.

[3] Cable Car World GmbH: *Cable Car World 2024*. Fachmesse und Kongress in Essen, 04./05.06.2024: https://www.cablecarworld.com/

[4] Technik Fischer – fischertechnik Seilbahn-Projekt: YouTube-Kanal.

[5] Kontakt zur Projektgruppe: Tilo Rust, ft.seilbahn@gmail.com

[6] Technik Fischer – fischertechnik Seilbahn-Projekt: *Hilfe: Wir suchen ein Museum!* Auf YouTube, 2025.

Getriebe

Getriebe für Mecanum-Räder

Dirk Fox

Mit den Baukästen „Robotics Smarttech“, „Robotics Hightech“ und dem MINT-Baukasten „Robotics Add-On: Omniwheels“ hat fischertechnik 2021 das Programm um Mecanum-Räder erweitert [1]. Im vergangenen Jahr ist der Baukasten „Maker Kit Omniwheels“ dazugekommen. Die Baukästen werden mit vier XS- bzw. Encoder-Motoren ausgeliefert, um die Mecanum-Räder unabhängig voneinander antreiben zu können. Doch halt: Benötigt man dafür wirklich vier Motoren?

Hintergrund

Mecanum-Räder ermöglichen einem Fahrzeug sehr flexible Bewegungen: Es kann auf der Stelle drehen, aus dem Stand in alle Richtungen fahren (vorwärts, rückwärts, seitlich, schräg) und Kurven mit beliebigen Radien „abfahren“. Daraus ergeben sich 18 verschiedene mögliche Bewegungen, bei denen die Räder jeweils unterschiedlich angetrieben werden müssen.

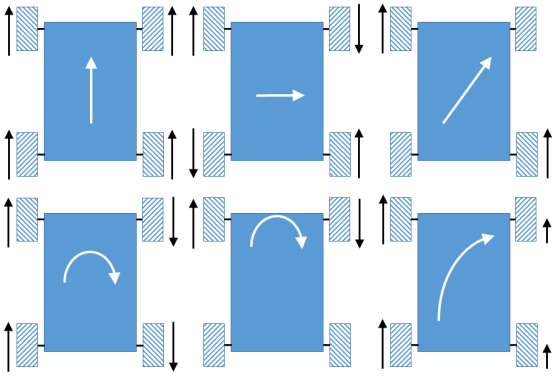


Abb. 1: Bewegungsarten eines Fahrzeugs mit Mecanum-Rädern [1]

Diese 18 Bewegungsmöglichkeiten lassen sich in sechs „Bewegungsarten“ zusammenfassen (Abb. 1) [1]. Durch Änderung der Drehrichtung der Räder oder Spiegelung des Antriebs zwischen den linken und rechten oder den vorderen und hinteren Rädern erhält man alle 18 möglichen Bewegungen. Üblicherweise werden alle vier Mecanum-Räder von einem eigenen Motor angetrieben. Durch Synchronisation der Motoren können die 18 Bewegungsarten recht präzise eingestellt werden.

Abb. 2: 4-Motoren-Chassis nach dem „Maker Kit Omniwheels“ (571901)

Ein Nachteil dieses Einzelradantriebs der Mecanum-Räder ist, dass alle vier Motorausgänge des RX-/TXT-Controllers dafür benötigt werden. Will man einen Gabelstapler mit Mecanum-Rädern bauen, muss man den Hub manuell vornehmen – oder einen zweiten RX/TXT oder eine Fernsteuerung hinzunehmen.[1] Außerdem wiegt jeder Motor etwas über 100g – viel Gewicht

[1] Eine trickreiche Beschaltung zur Steuerung eines Mecanum-Fahrzeugs mit der fischertechnik-Fernsteuerung hat Thomas Püttmann im Forum im Thread „Nutzung von Omniwheels ohne Controller“ vorgeschlagen.

für ein Modellfahrzeug. Und Platz benötigen sie schließlich auch (Abb. 2).

Eine interessante Frage ist also, ob sich ein Fahrzeug mit Mecanum-Rädern auch mit weniger als vier Motoren antreiben lässt. Ein Indiz dafür, dass das gelingen könnte, ist die Tatsache, dass bei jeder Bewegungsart mindestens zwei Motoren synchron derhen müssen – die Ansteuerung der Motoren ist also keineswegs voneinander unabhängig.

Ein-Motor-Antrieb

Mit einem Motor allein können wir nur eine einzige Bewegungsart ansteuern, z. B. die Vor- und Rückwärts- oder die Seitwärtsfahrt.

Die Vor- und Rückwärtsfahrt ist einfach: Wir müssen dafür lediglich zwei parallele Achsen gleichsinnig antreiben, damit sich alle vier Mecanum-Räder in dieselbe Richtung drehen. Auch das Drehen auf der Stelle lässt sich leicht verwirklichen – die beiden parallelen Achsen müssen nur gegensinnig angetrieben werden.

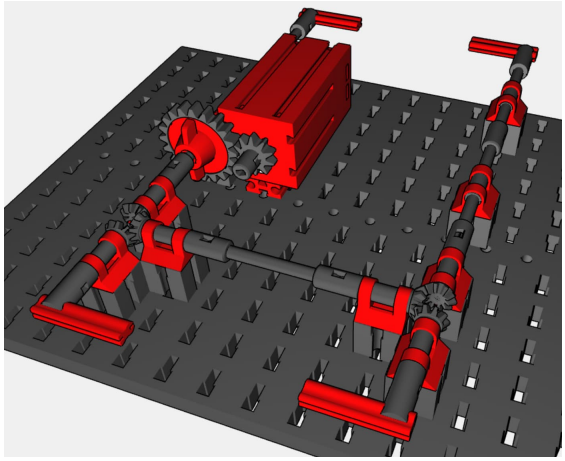


Abb. 3: Gegensinniger Antrieb der Räder mit einem Motor (fischertechnik-Designer)

Für die Seitwärtsfahrt genügt uns ein einfaches Getriebe, bei dem die einander gegenüberliegenden Räder jeweils in die entgegengesetzte Richtung drehen. Auf diese Weise können wir das Fahrzeug mit Mecanum-Rädern seitlich bewegen. Das gelingt leicht mit Kegelzahnrädern.

Abb. 3 zeigt eine solche Getriebekonstruktion. Statt der Mecanum-Räder sind im Modell an den vier Achsenden Rastadapter 20 (36227) mit Verbindungsstücken 30 (31061) montiert.

Ein einzelner Motor genügt also nicht, um mehr als eine der in Abb. 1 gezeigten Bewegungsarten der Mecanum-Räder umzusetzen. Möglich wäre es, ein Schaltgetriebe zu konstruieren, mit dem zwischen verschiedenen Antrieben umgeschaltet werden kann. Aber dafür benötigen wir einen zweiten Motor.

Zwei-Motoren-Antrieb

Mit zwei Motoren können wir immerhin die beiden wichtigsten Bewegungsarten steuern – vorwärts/rückwärts und seitwärts. Das gelingt, indem wir mit einem Motor alle vier Räder (für die Seitwärtsfahrt) gegenläufig antreiben. Mit Hilfe eines zweiten Motors können wir über ein Überlagerungsgetriebe mit zwei Differentialen die Räder alle in dieselbe Richtung drehen lassen (für die Geradeausfahrt).

Abb. 4 zeigt den Prototyp einer solchen Getriebekonstruktion. Dafür benötigen wir zwei Rast-Differentiale (75218) mit Differential-Zahnrad Z20 (35983) – glücklich, wer davon mehrere sein Eigen nennt, denn sie werden von fischertechnik unerklärlicher Weise nicht mehr produziert.

Schauen wir uns die Funktion des Getriebes einmal etwas genauer an:

- Arbeitet nur der linke Motor, sind die beiden Differentialkäfige durch das Z10 des rechten Motors blockiert. Daher drehen sich die Räder auf der linken Achse gegensinnig. Dasselbe gilt für die rechte Achse: Die mittlere Achse treibt den vorderen Teil der rechten Achse an; damit dreht sich das hintere Achsende in die entgegengesetzte Richtung. Die vorderen Räder drehen sich ebenfalls gegensinnig zu den hinteren.

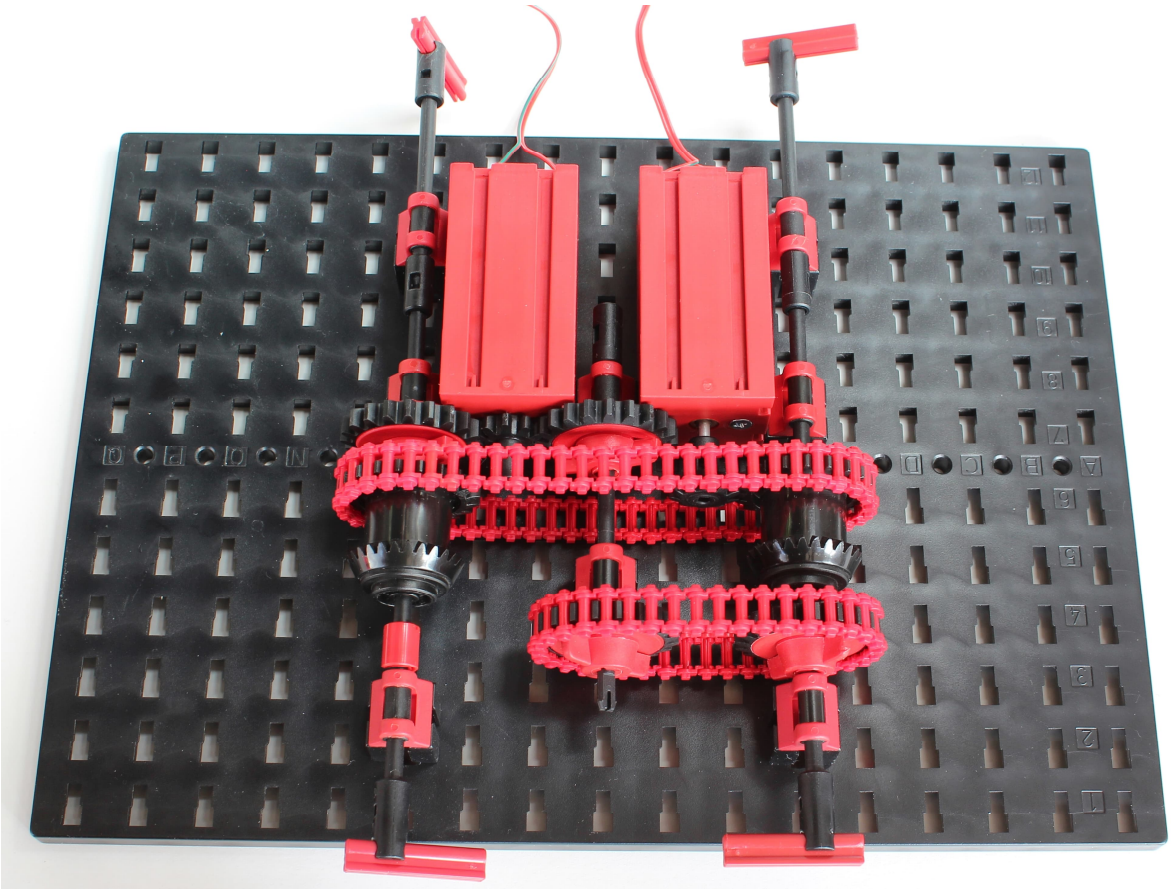


Abb. 4: Überlagerungsgetriebe zum Antrieb der Meccanum-Räder mit zwei Motoren

Damit bewegt sich das Fahrzeug nach rechts oder bei Umkehrung der Drehrichtung beider Motoren nach links (zweite Bewegungsart in Abb. 1).

- Treibt nur der rechte Motor das Getriebe (also die beiden Differentialkäfige) an, stehen das hintere linke und vordere rechte Rad still; das vordere linke und das hintere rechte Rad drehen sich gleichsinnig. Damit lässt sich das Fahrzeug entlang der Diagonalen von links vorne nach rechts hinten bewegen (dritte Bewegungsart in Abb. 1).
- Um das Fahrzeug entlang der Diagonalen von rechts vorne nach links hinten zu bewegen, treiben wir den zweiten Motor mit doppelter Geschwindigkeit und gleicher Drehrichtung) an: Dadurch überlagert der Antrieb der Achsen den der Differentialkäfige, und das vordere linke und das hintere rechte Rad stehen still.
- Arbeiten beide Motoren in derselben Geschwindigkeit und in entgegengesetzter Richtung, drehen sich alle Räder gleichsinnig. Das Fahrzeug fährt vorwärts und (bei Umkehrung der Drehrichtung der Motoren) rückwärts (erste Bewegungsart in Abb. 1).

Bei der Geradeausfahrt (vor-/rückwärts) müssen die beiden Motoren mit exakt derselben Umdrehungsgeschwindigkeit arbeiten. Das lässt sich mit einem Controller (TXT) und den Encodermotoren sicherstellen, indem die Motoren synchron gesteuert werden.

Abb. 5 zeigt das Getriebe ohne Ketten. Damit sollte der Nachbau keine Schwierigkeiten bereiten. Auf der Download-Seite zu dieser Ausgabe der ft:pedia findet sich die zugehörige fischertechnik-Designer-Datei.

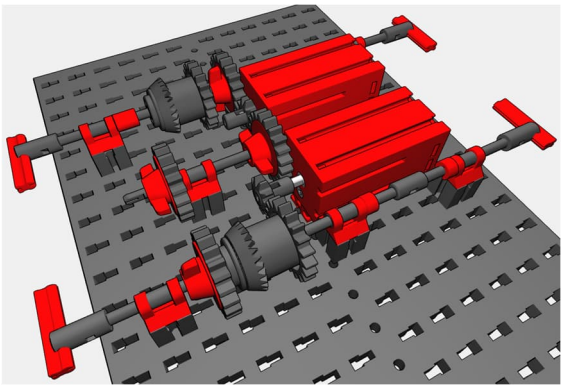


Abb. 5: Überlagerungsgetriebe ohne Ketten (fischertechnik-Designer)

Mit den „historischen“ Differentialen (31043) lässt sich dieses Getriebe recht kompakt in einem Chassis mit Mecanum-Rädern verbauen (Abb. 6). Auf der Stelle drehen kann das Getriebe ein Fahrzeug jedoch nicht. Dazu benötigen wir einen weiteren Motor.

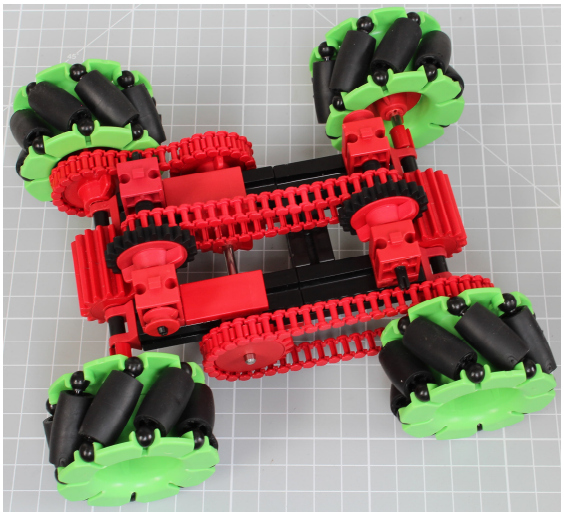


Abb. 6: Chassis mit Mecanum-Rädern und Überlagerungsgetriebe

Drei-Motoren-Antrieb

Mit drei Motoren und der Getriebekonstruktion in Abb. 7 können wir alle Bewegungsarten des Mecanum-Fahrzeugs ausführen. Schauen wir uns das Getriebe ein wenig genauer an:

- Wird nur der linke Motor angetrieben, sperrt der (linke) Differentialkäfig und das obere linke und das untere linke Rad drehen sich gegensinnig. Damit dreht das Fahrzeug um den Mittelpunkt der Vorderachse (fünfte Bewegungsart in Abb. 1).
- Wird nur der mittlere Motor angetrieben, sperrt der (rechte) Differentialkäfig und das obere rechte und untere rechte Rad drehen sich gegensinnig. Damit dreht das Fahrzeug um den Mittelpunkt der Hinterachse.

Werden beide Motoren mit gleicher Geschwindigkeit angetrieben, lassen sich zwei Bewegungsarten ausführen:

- Bei gleicher Polung der Motoren drehen sich alle einander gegenüberliegenden Mecanum-Räder gegensinnig und das Fahrzeug vollzieht eine Seitwärtsfahrt (zweite Bewegungsart in Abb. 1).
- Bei umgekehrter Polung der Motoren drehen sich die beiden unteren Räder in die eine, die beiden oberen in die entgegengesetzte Richtung und das Fahrzeug dreht auf der Stelle (vierte Bewegungsart in Abb. 1).

Eine weitere Bewegungsart gelingt mit Einsatz des rechten Motors:

- Wird nur der rechte Motor angetrieben, halten die beiden anderen Motoren das linke obere und das rechte untere Rad fest. Damit drehen sich das linke untere und das rechte obere Rad in dieselbe Richtung und das Fahrzeug unternimmt eine Diagonalfahrt (dritte Bewegungsart in Abb. 1).
- Überlagern wir den Antrieb des rechten Motors mit dem des mittleren (doppelte Geschwindigkeit, umgekehrte Drehrichtung) und dem des linken (doppelte Geschwindigkeit, gleiche Drehrichtung), so unternimmt das Fahrzeug eine Diagonalfahrt entlang der entgegengesetzten Diagonalen.

Fehlt noch die erste (und wichtigste) Bewegungsart aus Abb. 1 – die Geradeausfahrt. Auch dafür benötigen wir alle drei Motoren:

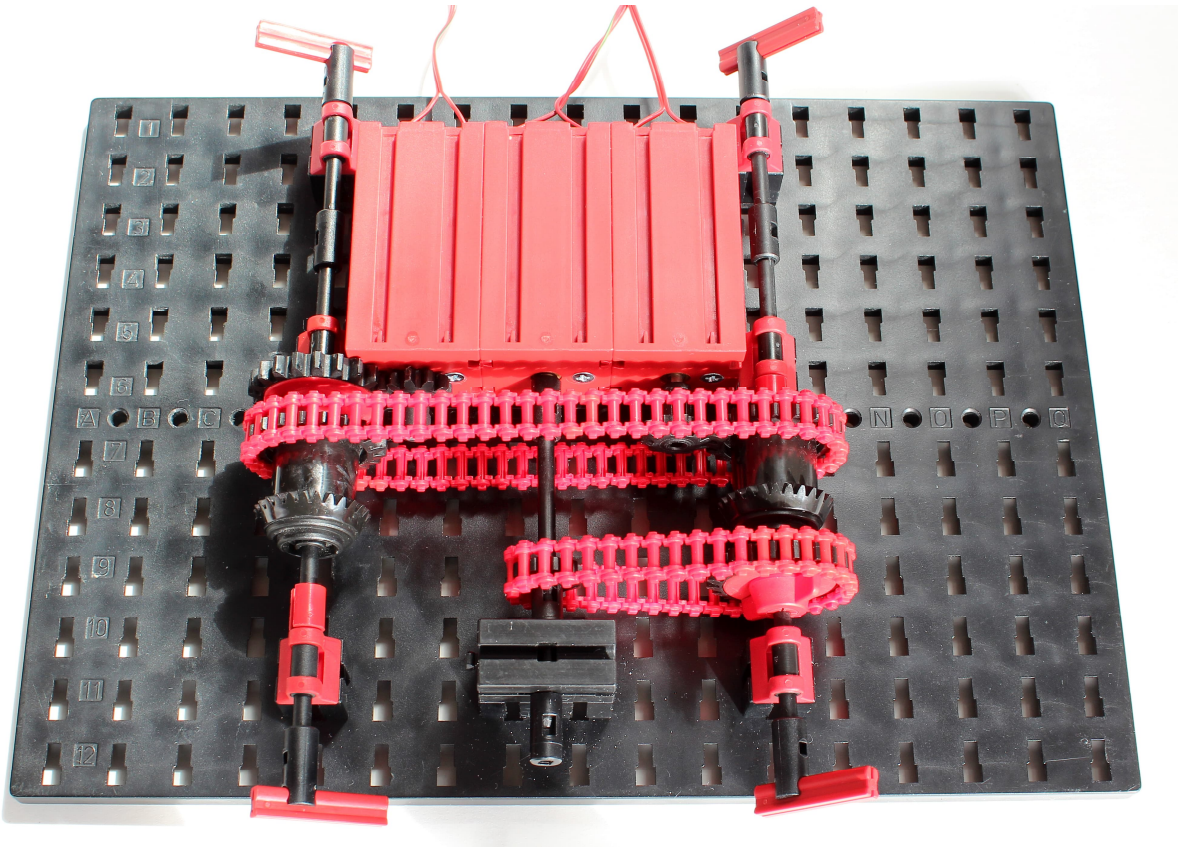


Abb. 7: Überlagerungsgetriebe mit drei Encoder-Motoren

- Der linke und der rechte Motor drehen bei gleicher Geschwindigkeit und Drehrichtung die Räder der linken Achse in dieselbe Richtung. Das hintere rechte Rad dreht dabei mit doppelter Geschwindigkeit in dieselbe Richtung; überlagern wir diese Bewegung mit dem mittleren Motor (gleiche Geschwindigkeit, entgegengesetzte Drehrichtung) drehen sich alle Räder in dieselbe Richtung.

In Abb. 8 ist das Überlagerungsgetriebe noch einmal ohne Ketten zu sehen; damit dürfte ein Nachbau leicht gelingen.

Fazit

Vier Motoren sind für den Antrieb der Mecanum-Räder bequem, aber übertrieben – die verschiedenen Bewegungsarten lassen sich auch mit drei Motoren und einem Differentialgetriebe erzeugen. Verzichtet man auf die Drehungen, genügen sogar zwei Motoren – und das Gewicht des Chassis sinkt schlagartig um 200g.

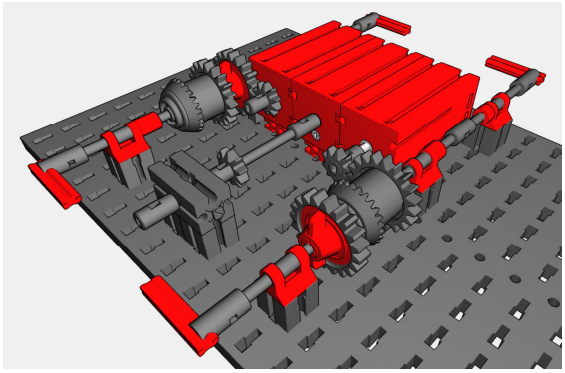


Abb. 8: Überlagerungsgetriebe mit drei Motoren (fischertechnik-Designer)

Als kleine Herausforderung bleibt die elegante Konstruktion des Fahrzeugs – das überlasse ich eurer Experimentierfreude.

Referenzen

[1] Dirk Fox: *Mecanum-Räder und Omniwheels*. ft:pedia 1/2021, S. 66–74.

[2] Dirk Fox, Thomas Püttmann: *Technikgeschichte mit fischertechnik*. dpunkt-Verlag, 2015 (3. Aufl. 2021).

Elektronik

Sensor-Adaptermodul

Arnoud van Delden

Heutzutage kann man kleine Leiterplatten mit allen möglichen Sensoren für wenig Geld online kaufen. Diese Sensormodule sind in erster Linie für den Einsatz mit 5-Volt-Mikrocontrollern konzipiert, lassen sich aber nicht so gut mit den fischertechnik-Elektronikmodulen und den klassischen „Silberlingen" aus dem 9-Volt-Bereich verwenden. Die hier besprochene Adapterplatine wurde entwickelt, um die Kompatibilität und die Einsatzmöglichkeiten dieser Sensoren zu verbessern. Sie erleichtert die recht unterschiedliche Pinbelegung der 3-poligen Stecker der verschiedenen Platinen. Außerdem stellt sie die (niedrigere) Versorgungsspannung für die Sensorplatine zur Verfügung und bietet Anschlussmöglichkeiten für die üblichen 2,5-mm-fischertechnik-Stecker.

Kleine Sensorplatinen

Die Sensorplatinen in Abb. 1 sind für den Einsatz in Projekten mit universellen Mikrocontroller-Boards wie Arduino, micro:bit, ESP-32 oder Raspberry Pi vorgesehen. Das Foto zeigt unter anderem einen PIR-Bewegungs-, einen Temperatur- und einen Feuchtigkeitssensor. Der passive Reed-Schalter und der Vibrationssensor sind auch als Platine mit 3-Pin-Stecker erhältlich.

Ich habe untersucht, welche Modifikationen sinnvoll wären, damit diese Sensorplatinen einfacher mit den fischertechnik-eigenen TX- und TXT-Controllern oder den Elektronikmodulen („Silberlingen") verwendet werden können. Dabei bin ich auf einige Inkompatibilitätshürden gestoßen.

Versorgungsspannung

Die Sensorplatinen benötigen ausnahmslos eine Versorgungsspannung von 5V. In der Vergangenheit habe ich untersucht, wie Hall-Effekt, LDR, Berührungsschalter und IR-Hindernissensoren nach diesem Prinzip nutzbar gemacht werden können [1]. Um die 5V-Versorgungsspannung bereitzustellen habe ich einige Stromversorgungsmodule entworfen [2] (Abb. 2).

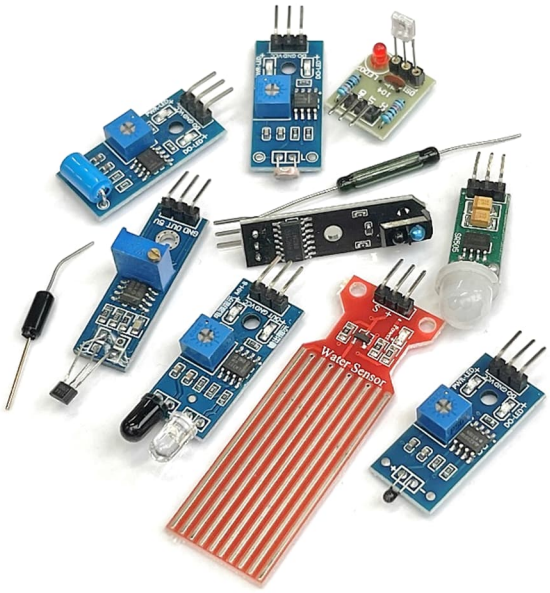


Abb. 1: Verschiedene preiswerte Sensorplatinen

Steht eine 5V-Versorgungsspannung zur Verfügung, lassen sich die Sensorplatinen gut an den Eingängen normaler Mikrocontroller und der TX- und TXT-Controller verwenden. Für die Verwendung unmodifizierter Platinen mit den Silberlingen muss allerdings ein Grundbaustein zwischenge-

schaltet werden, um den Ausgangssignalpegel detektierbar zu machen [3].

Abb. 2: Verschiedene Lösungen zur Bereitstellung einer 5V-Versorgungsspannung

Umstellung auf 9V

Eine andere Möglichkeit ist daher, die Sensorplatinen für eine Versorgungsspannung von 9V bis 11V geeignet zu machen, die direkt von der TXT-Steuerung oder von Silberlingen bezogen werden kann.

Einige Sensormodule, wie z. B. der Infrarot-Hindernissensor, sind mit einem LM393-Komparator ausgestattet, der eine schärfere (manchmal einstellbare) Erkennungsstufe und weniger Rauschen um die Erkennungsschwelle herum ermöglicht.

Der LM393 verträgt eine Versorgungsspannung von bis zu 36V, sodass er theoretisch ohne Probleme direkt an einer höheren Versorgungsspannung verwendet werden könnte. Das Problem ist jedoch, dass die Vorwiderstände der eingebauten Anzeige- (oder IR-) LEDs offensichtlich nicht darauf abgestimmt sind. Ohne Anpassungen ist die Wahrscheinlichkeit groß, dass sie bei dieser Betriebsspannung innerhalb kurzer Zeit durchbrennen.

Einige Möglichkeiten zur Anpassung der Hindernissensoren für den Betrieb mit 9V wurden von mir in der jüngeren Vergangenheit untersucht [4]. Abb. 3 zeigt einige dieser Versuche mit IR-Hindernissensoren. Allerdings sind die Ausgänge dieser modifizierten Sensoren immer noch nicht elektrisch kompatibel mit den Silberlingen, sodass es weiterhin notwendig ist, einen Grundbaustein pro Sensor dazwischen zu schalten.

Abb. 3: Verschiedene Arten von Anpassungen auf 9V-Betrieb

Inkompatibilität der Ausgänge

Wer die verschiedenen Sensorkarten ohne Mikrocontroller sinnvoll einsetzen will, muss auch die Inkompatibilität zwischen Ausgangs- und Eingangssignalen berücksichtigen. Mikrocontroller haben oft genügend Eingänge mit einer hohen Eingangsimpedanz. Diese Eingänge ziehen kaum Strom und tasten eigentlich nur den Spannungspegel ab, den der angeschlossene Sensorausgang bietet. Ob ein logisches „High“- oder aktives Signal durch +5V oder durch 0V repräsentiert wird, ist dabei sogar irrelevant. Denn die logische Interpretation („positive“ oder „negative“ Logik) der physikalischen Spannungspegel des Ausgangs kann in der Software leicht angepasst werden.

Die klassischen fischertechnik-Elektronikmodule, wie z. B. die Silberlinge, benötigen jedoch an ihren Eingängen ein stromschaltendes Eingangssignal. Diese Module sind elektronisch nach der sogenannten Dioden-Transistor-Logik aufgebaut [13]. Dies hat zur Folge, dass die Silberlinge die sogenannte „negative Logik“ verwenden. Hier muss für eine logische „1“ das Eingangssignal über den Eingang auf 0V gezogen werden. Die Ausgänge der Sensorplatinen können dies nicht tun. Ihr logisches Ausgangssignal wird durch einen Spannungspegel repräsentiert, der nicht in der Lage ist,

den Eingang eines Silberlings auf einen ausreichenden Nullpegel zu ziehen.

Es wäre also praktischer, wenn die Sensorplatinen „Stromausgänge“ hätten. Das würde die Verwendbarkeit für moderne Mikrocontroller nicht beeinträchtigen, sie aber auch für den Einsatz mit den Silberlingen brauchbar machen. Es wäre dann sogar möglich, sogenannte „verdrahtete ODERs“ zu verwenden [5]. Und wenn jedes Ausgangssignal auch invertiert zur Verfügung steht, ist es auch einfacher, „positive Logik“ und „negative Logik“ mit Modulen zu mischen.

Ein universelles Adaptermodul

Wenn wir ein universelles Adaptermodul für die verschiedenen Sensorplatinen entwickeln würden, sollte es idealerweise folgende Probleme lösen:

- **Versorgungsspannung:** Für eine optimale Kompatibilität sollte das Modul mit einer Versorgungsspannung von 5 bis 12 V betrieben werden können, so dass eine separate 5V-Versorgungsleitung nicht erforderlich ist. Dadurch wird der Sensor auch besser vor Störungen auf der allgemeinen Versorgungsspannung durch z. B. Motoren geschützt.
- **Elektrische Kompatibilität:** Die Ausgänge des Moduls sollten vorzugsweise stromgesteuert und nicht spannungsgesteuert sein. Dadurch können sie sowohl an modernen Mikrocontroller-(Spannungs-)Eingängen als auch an stromgesteuerten Eingängen der Silberlinge verwendet werden. Ein weiterer Vorteil wäre, dass logische Gatter eingespart werden könnten, da verdrahtete ODERs (für negative Logik) und verdrahtete UNDs (für positive Logik) verwendet werden könnten.
- **Auch invertiertes Ausgangssignal:** Bei Verwendung mit einem Mikrocontroller kann die interpretierte Logik des Ausgangssignals leicht per Software geändert werden. Für eine optimale Schaltflexibilität bei Schaltungen, die aus diskreten Bauteilen oder Elektronikmodulen bestehen, sollte das Sensormodul daher idealerweise sowohl den normalen Logikausgang als auch das logisch invertierte (komplementäre) Signal bereitstellen. Dies ist ohnehin bei Sensoren sinnvoll, die ein inverses Schaltverhalten aufweisen (wie die meisten Temperatursensoren). Es kann oft durch geschickte Anwendung der Morgan-Logikgatterregeln [14] oder durch Inverter im logischen Schaltschema eingespart werden.
- **Unterschiede in der Pin-Belegung:** Alle untersuchten Sensorplatinen haben eine dreipolige Dupont-Stiftleiste. Die Belegung dieses Steckers ist jedoch keineswegs genormt. Die Position von „+“, „-“ und Signalausgang scheint völlig willkürlich zu sein. Die zu entwerfende Platine sollte daher mehrere Anschlusspositionen mit unterschiedlichen Pin-Reihenfolgen bieten, so dass immer ein Platz für den Anschluss eines

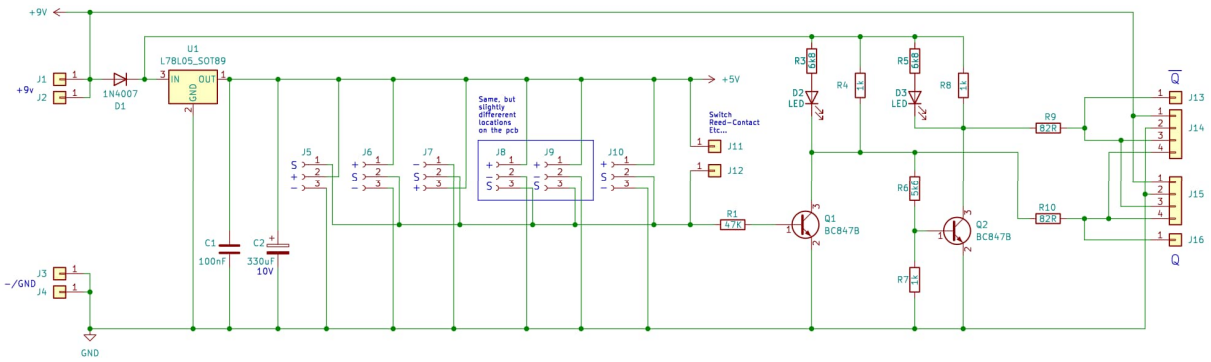


Abb. 4: Schaltplan der Universaladapterplatine

bestimmten dreipoligen Sensors vorhanden ist.

- **Anzeige-LEDs:** Nicht alle Sensorplatinen haben Anzeige-LEDs. Für diese Sensorplatinen wäre eine Option nützlich, diese separat auf der Platine anzubringen.
- **Auch für Tastschalter:** Es wäre schön, wenn die kürzlich diskutierten Berührungssensoren (sowohl das kleine rote HW-763-Board als auch die HTTM-Serie mit LED-Touch-Oberfläche) auch direkt auf dem Board verwendet werden könnten.
- **Sensoren ohne Platinen:** Passive Schaltkomponenten wie Reed-Kontakt und Drucktaster sollten auch direkt auf dem Modul verwendbar sein.
- **fischertechnik-Stecker:** Idealerweise sollte das Modul mit fischertechnik-Steckern verwendbar und somit mit 2,5-mm-Buchsen oder Löchern ausgestattet sein.
- **Durchschleifen und anschließen:** Es wäre schön, wenn es neben den Standard-fischertechnik-Buchsen auch eine kompaktere Möglichkeit gäbe, die Stromversorgung durchzuschleifen und die Ausgänge anzuschließen, z. B. mit JST-Steckern.

Abb. 4 zeigt das Schema der Universaladapterplatine, mit der alle oben genannten Anforderungen erfüllt werden können. Um eine optimale Kompatibilität mit den Silberlingen zu gewährleisten, wurde das ursprüngliche Design der Endstufe mit invertierender Folgestufe dieser Module als Ausgangspunkt genommen.

Der kleine Widerstand von 82 Ω an den Ausgängen scheint ein einfacher Strombegrenzer zu sein, der verhindern soll, dass ein Ausgang direkt an die positive Versorgungsspannung angeschlossen wird. Die Silberlinge verwendeten jedoch ursprünglich BC238-Transistoren mit einem maximalen Kollektorstrom von 100 mA. Dies würde bedeuten, dass der Ausgang (kurzzeitig) bis zu einer Versorgungsspannung von 8,2 V „geschützt“ wäre. Die Wahl des heute gebräuchlicheren (und billigeren) universellen NPN-Transistors BC547 in selbstgebauten Silberlingen ist daher eine gute Wahl. Das Schaltverhalten der Module wird dadurch nicht verändert, wobei dieser Transistor einen noch höheren maximalen Kollektorstrom hat.

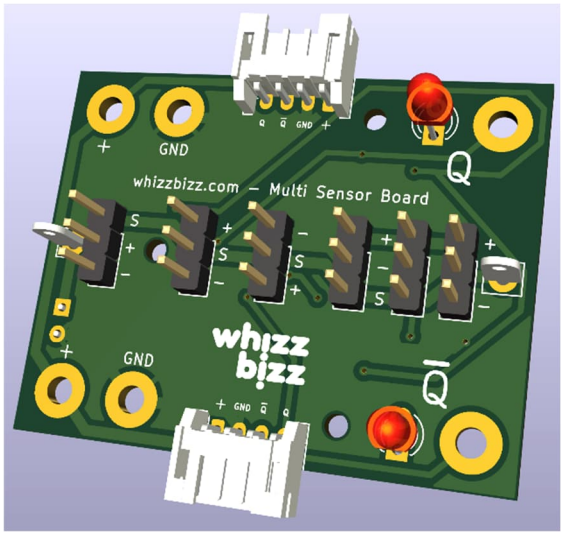


Abb. 5: PCB-Design mit den SMD-Komponenten (verdeckt) auf der Unterseite

Die Spannung des Ausgangssignals entspricht in etwa der Versorgungsspannung und das Ganze kann trotz des Spannungsreglers in der Praxis meist noch mit 5V betrieben werden. Der Schaltplan führte zu dem Platinenentwurf in Abb. 5, der die möglichen Positionen (mit Permutation der Anschlüsse) für einen Stecker zeigt.

Falls die Sensorplatine dies nicht bereits von sich aus leistet, können optional Kontroll-LEDs angebracht werden, an denen die Ausgänge abgelesen werden können. Die Stromversorgung und der Anschluss der Ausgangssignale sind über fischertechnik-Stecker oder über den optional angebrachten weißen JST-Stecker möglich. Da bei der Verwendung von waagerechten (abgewinkelten) JST-Steckern Ausschnitte in den Seitenwänden vorhanden sein müssen, habe ich mehrere Varianten des Gehäuses entworfen. Nach der Montage der

Platine befinden sich die SMD-Teile im unteren Teil des Halters. Das Gehäuse enthält eine halbkreisförmige Aussparung für den kleinen radialen Elektrolytkondensator C2 auf der Unterseite der Platine. Dieser Elko ist in der Praxis nicht notwendig und wurde nur zum Schutz vor einer z. B. durch Motoren stark „verrauschten“ Eingangsspannung konzipiert.

Zusammenbau

Die Sensorplatine der Wahl wird einfach an die entsprechende Anschlussstelle gelötet. Manchmal ist eine kleine Vorbereitung der Sensorplatine erforderlich. Wie in Abb. 1 gezeigt, werden viele dieser Sensoren mit oft abgewinkelten, aufgelöteten 3-Pin-Steckern geliefert. Wenn keine Einstellmöglichkeiten auf der Oberseite der Sensorplatine zugänglich bleiben sollen, kann die Sensorplatine kopfüber in den 3-poligen Steckverbinder mit der richtigen Pin-Reihenfolge eingesetzt werden (Siehe Abb. 6).

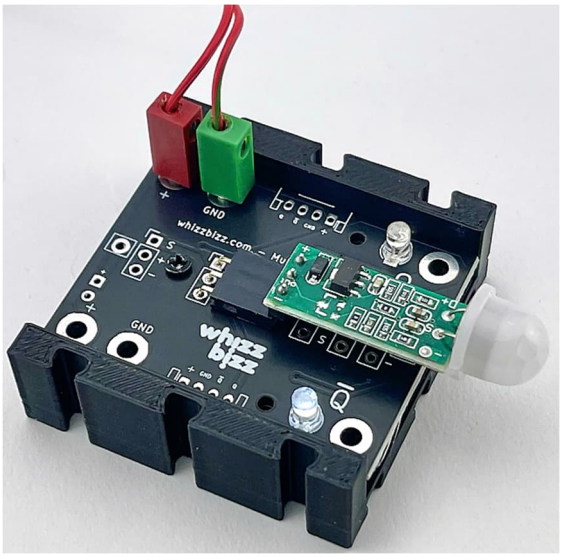


Abb. 6: Sensorplatine in Dupont-Stecker

Platzsparender, aber etwas aufwendiger ist die Lösung, bei der der montierte Stecker entfernt und die Verbindung zur Platine mit drei kurzen Drähten hergestellt wird. Um die Sensorplatine auf der Modulplatine zu befestigen habe ich mit einer Klebepistole für beide Montagevarianten gute Ergebnisse erzielt. Auf diese Weise konnte sogar der in Abb. 7 gezeigte Wassersensor sicher befestigt werden.

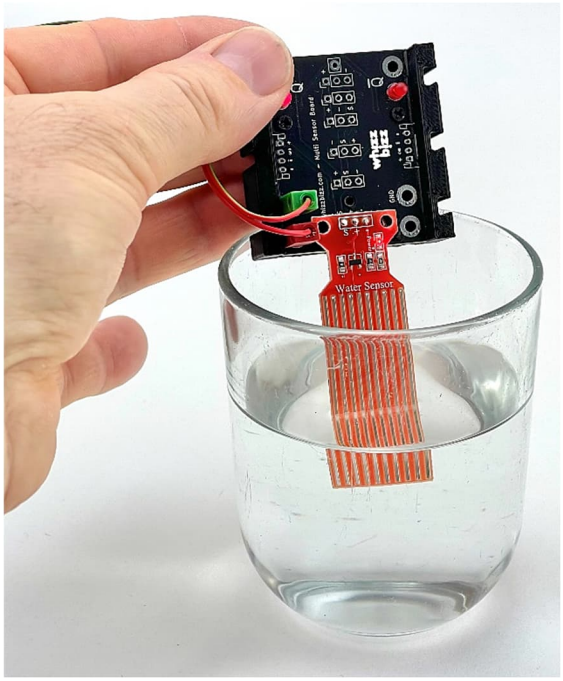


Abb. 7: Wassersensor auf der Adapterplatine

Anschlussmöglichkeiten

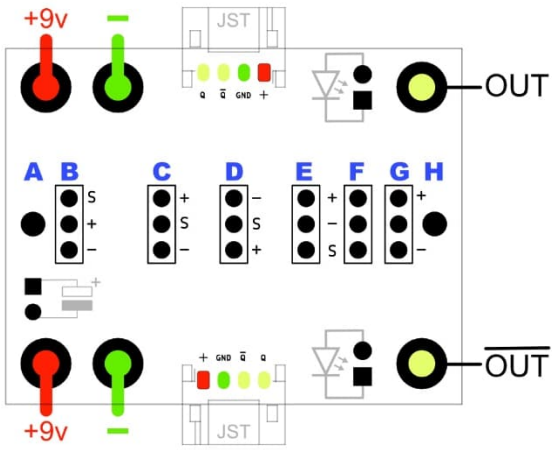


Abb. 8: Die verschiedenen Anschlussmöglichkeiten und Steckerpositionen des Sensormoduls

Die verschiedenen Anschlussmöglichkeiten sind in Abb. 8 dargestellt. Die blauen Positionen A bis H zeigen die möglichen Anschlusspunkte für die möglichen Sensorplatinen an. Der optionale Elektrolytkondensator, der auf der Unterseite der Platine angelötet werden kann, ist grau dargestellt.

Die LEDs an den Ausgängen sind ebenfalls optional und daher grau dargestellt. Sie werden nicht benötigt, wenn die verwendete Sensorplatine bereits über eigene Anzeige-LEDs verfügt.

Die Stromversorgung und die Ausgänge können bei Bedarf mit fischertechnik-Steckern in den entsprechenden Anschlusslöchern auf der Platine angeschlossen werden. Diese Anschlussstellen sind doppelt vorhanden, so dass mehrere Sensormodule einfach durchgeschleift werden können. Lose Steckverbindungen können korrigiert werden, indem man die Lamellen des Steckers mit einem Messer vorsichtig weiter auseinanderbiegt und den Stecker nicht ganz bis zur Platine durchdrückt.

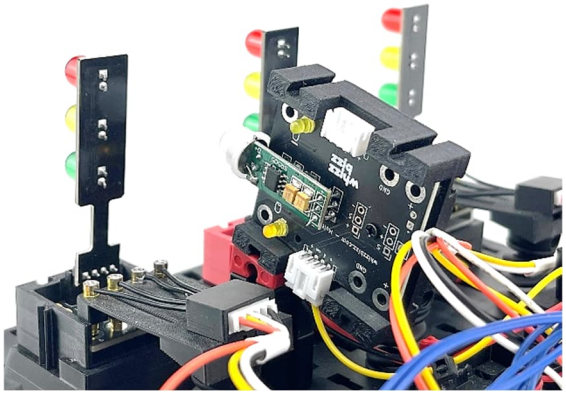


Abb. 13: PIR-Bewegungssensor, angeschlossen mit JST-Stecker

Wer Anschlusshöhe oder fischertechnik-Stecker sparen möchte und die Möglichkeit hat, eigene Kabel mit JST-PH-Steckern herzustellen, kann statt der fischertechnik-Stecker wahlweise (einen oder) zwei stehende oder liegende vierpolige JST-PH-Stecker verwenden. Im Gehäuse gibt es drei Stellen, an denen die so bestückte Adapterplatine nach Belieben mit einer kleinen 6-mm-M2-Schraube befestigt werden kann.

Verschiedene Positionen der Sensorplatine

Passive Sensoren wie der Vibrationssensor, ein Reedkontakt oder ein (verdrahteter) Taster können zwischen den Positionen A und H in Abb. 8 angeschlossen werden. Die verschiedenen dreipoligen Anschlusspunkte bieten eine Reihe von Anschlussmöglichkeiten für die aktiven Sensorplatinen, die eine (5V) Versorgungsspannung benötigen.

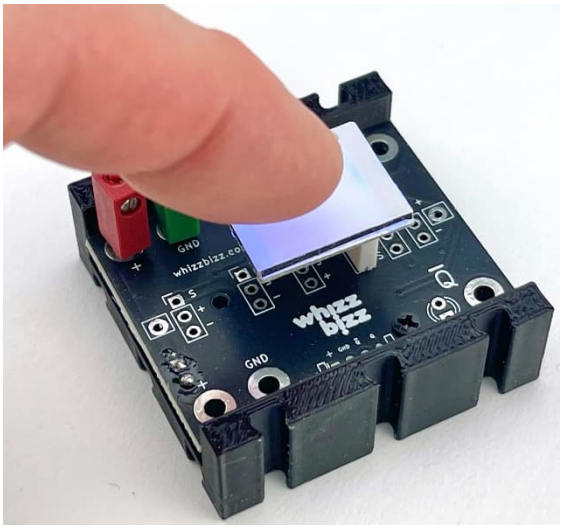


Abb. 14: Die Touch-Schalter verdienen ein weniger zerbrechliches Gehäuse

Die Anschlussmöglichkeit E kann für LDR-, IR-Hindernis- und Temperatur-

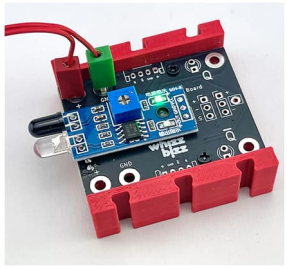


Abb. 9: IR-Hindernis-Sensor

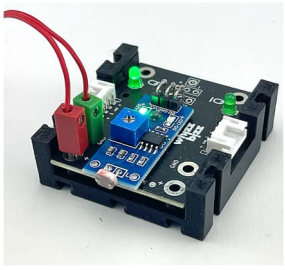


Abb. 10: LDR-Sensor

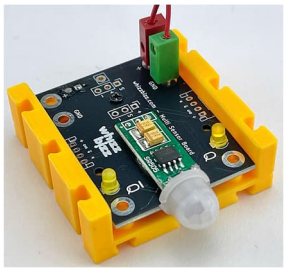


Abb. 11: Infrarot-Bewegungsmelder

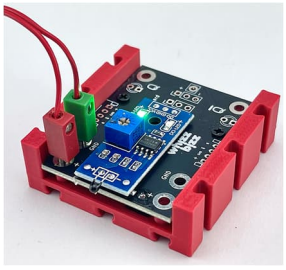


Abb. 12: Adapterplatine mit Temperatursensor

sensorplatinen sowie für die LED-Touch-Schalter verwendet werden. Die Position F ist im Grunde die gleiche, aber der Unterschied in der Position ermöglicht es, längere Varianten dieser Sensorplatinen zu montieren oder das Ganze in einem geschlossenen Gehäuse unterzubringen. Dies war auch die richtige Anschlussstelle für die Vibrations-/Bewegungssensorplatine.

Meine Hall-Effekt-Sensorplatinen habe ich an Position G platziert, während der Feuchtigkeitssensor an Position B am Rand der Platine Platz finden konnte (siehe Abb. 7).

Für die kleinen PIR-Bewegungssensoren ist die Position D die beste Wahl; die Sensorplatine reicht dann genau bis zur rechten Seite. Dieser Sensor kann aber auch kopfüber mit einem Steckersockel z. B. an Position C angeschlossen werden, wie in Abb. 6 gezeigt. Kurzum, es gibt immer einen Platz für eine bestimmte Sensorplatine auf der Adapterplatine.

Anwendungen

Ein Modul mit einem PIR-Bewegungssensor erwies sich beim letzten Treffen des niederländischen fischertechnik-Clubs als nützlich, um ein Demonstrationsmodell mit vier LED-Ampeln [6] auf Kommando zu starten. Dieses Modell, das in Abb. 15 zu sehen ist, startet eine kleine Lichtshow, sobald ein Interessent an den Tisch herantritt. Dieses Modell wird mit einem Arduino Mega Pro Mini [7] gesteuert, wobei die verschiedenen LED-Ampeln über einige Module mit kurzschlussfesten Stromausgängen [8] versorgt werden. Aber mit der hier besprochenen Adapterplatine ist es genauso einfach, eine solche Anordnung mit dem TXT-Controller oder sogar mit den traditionellen Silberlingen zu bauen.

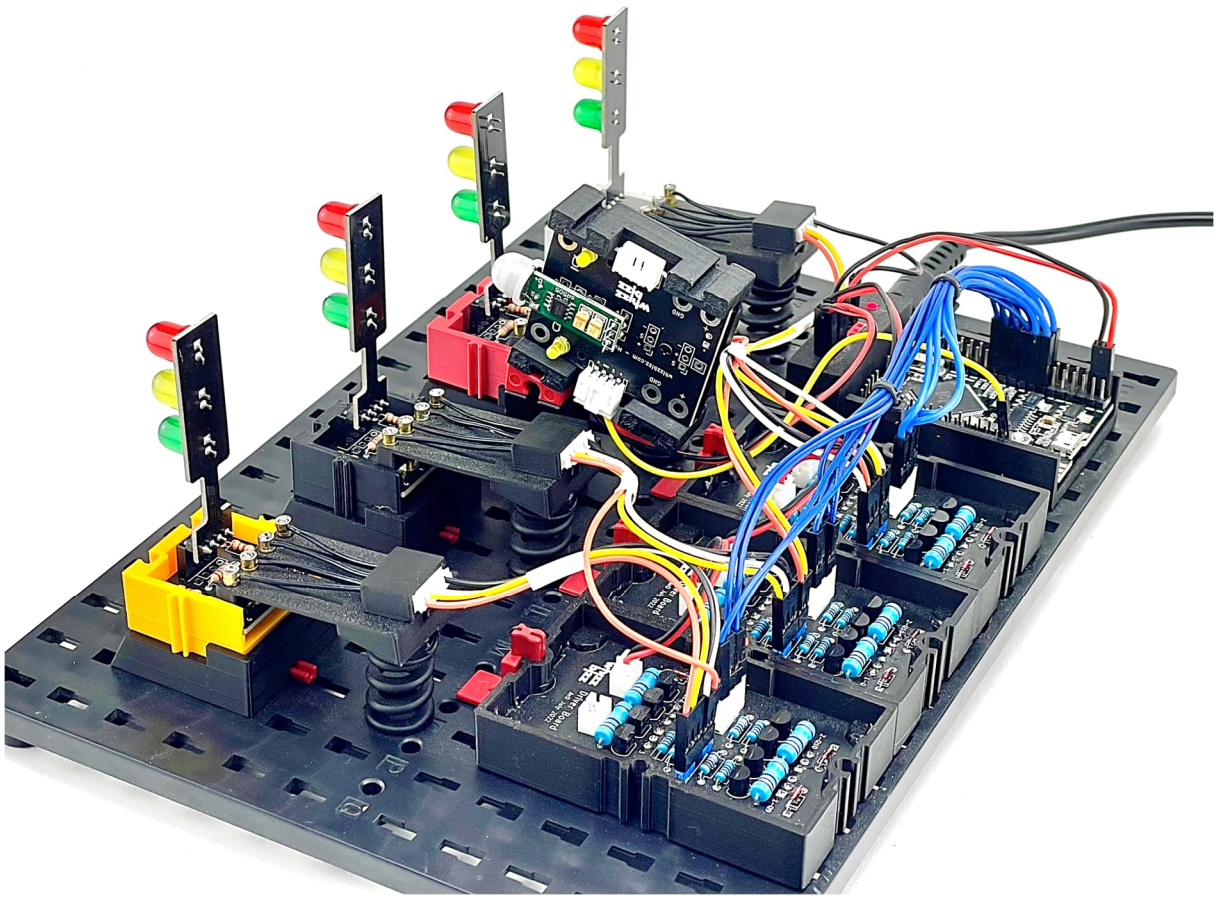


Abb. 14: LED-Ampel-Demonstration, selbststartend mit Bewegungssensor

In der Vergangenheit habe ich IR-Hindernissensoren verwendet, um die Drehzahl mit einem Doppeltachometer zu messen [9]. Abb. 16 zeigt ein mögliches Tachometer mit dem Hall-Effekt-Sensor im Adaptermodul. Eine Bewegung des Magneten am Sensor vorbei löst einen Hardware-Interrupt aus, mit dem die Drehzahl ermittelt werden kann. Das gleiche Prinzip wird verwendet, wenn das Impulssignal des fischertechnik-Encoder-Motors (153422) mit einem Zählereingang eines TXT-Controllers verbunden wird. Eine solche externe Montage kostet zwar mehr Platz, dafür ist die Messung (zumindest theoretisch) genauer. Denn dadurch, dass man näher am tatsächlichen Endpunkt der Bewegung misst statt am Ursprung im Motor, wird ein möglicher Schlupf oder ein Spiel des (Zahnrad-) Getriebes automatisch einkalkuliert.

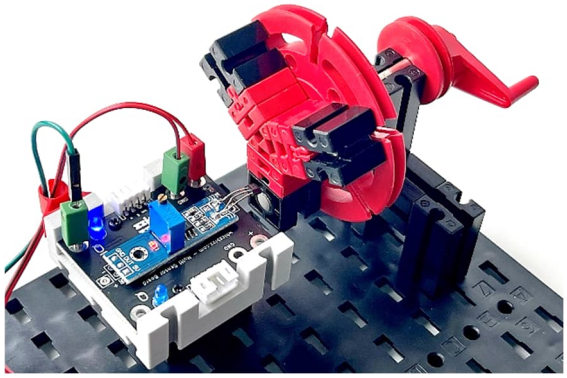


Abb. 15: Drehzahlmessung mit Hall-Effekt-Sensor

Abschließende Worte

Das Adaptermodul hat sich bereits in verschiedenen Modellen bewährt. Mehr Informationen findet ihr auf der Projektseite auf meiner Website [10]. Die 3D-Druckdateien einiger Varianten des Gehäuses, das ich für die Adapterplatine entworfen habe, stehen für alle, die sie ausdrucken (oder ausdrucken lassen) möchten, zum Download bereit [11]. Im Moment sind noch viele Einzelplatinen mit den fertig bestückten SMD-Bauteilen bei mir auf Lager [12].

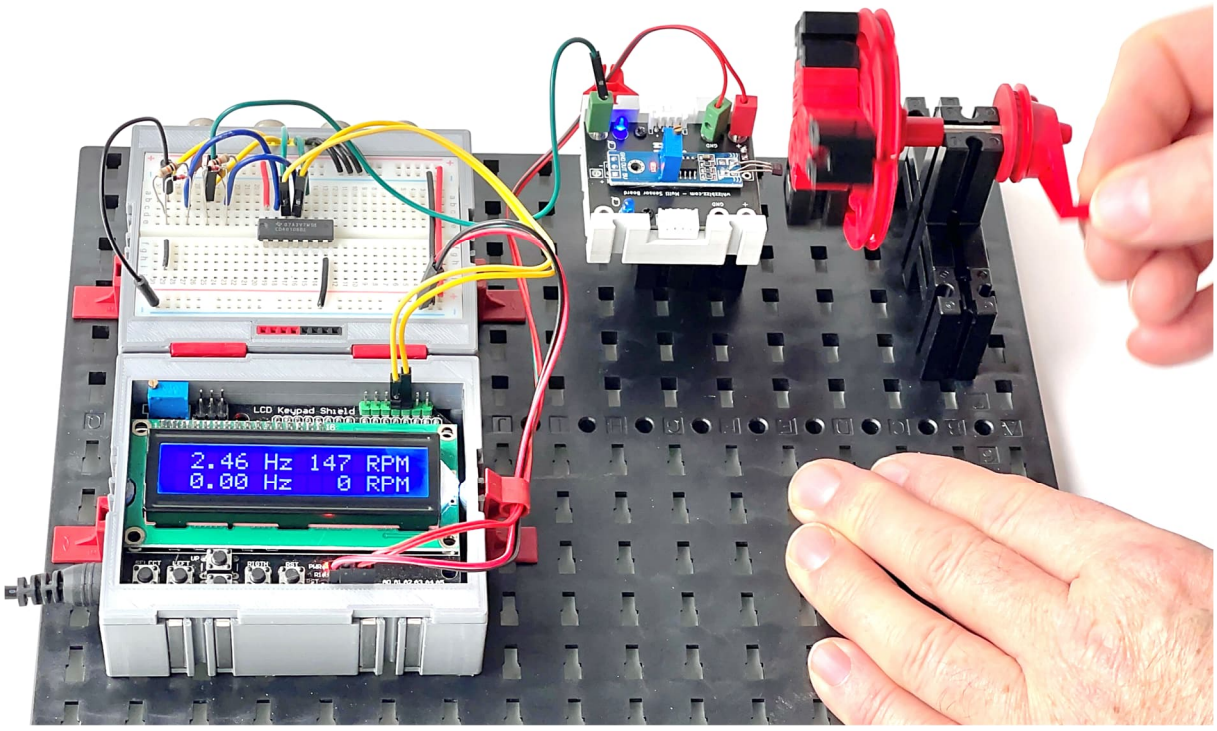


Abb. 16: Tachometer mit Hall-Effekt-Sensor im Sensor-Adaptermodul

Für diejenigen, die das (PCB-)Design noch verbessern wollen, sind der Schaltplan, das PCB-Design und die 3D-Druckdateien auf GitHub verfügbar [11], aber wer gleich loslegen will, dem helfe ich natürlich gerne mit gebauten und geprüften Adaptermodulen. Alle Sensoren mit Gehäuse sind auf Anfrage in (fast) jeder Farbe erhältlich.

Quellen

[1] Arnoud van Delden: *Kontaktlose Schalter (Teil 3)*. ft:pedia 4/2023, S. 46–52.

[2] Arnoud van Delden: *Eine zukunftssichere Stromversorgung (Teil 2)*. ft:pedia 3/2023, S. 63–69.

[3] Arnoud van Delden: *Kontaktlose Schalter (Teil 2)*. ft:pedia 4/2023, S. 37–45.

[4] Arnoud van Delden: *Der Zauberling (Teil 3): Ein erster Trick*. ft:pedia 4/2021, S. 52–57.

[5] Fischer-Werke: *2 Lichtschranken steuern den Relaisbaustein*. In *hobby 4 Band 3*, 1973, S. 25.

[6] Arnoud van Delden: *Ampelschaltungen*. ft:pedia 3/2024, S. 49–62.

[7] Arnoud van Delden: *Casing for Arduino Mega Pro Mini for fischertechnik.* Auf printables.com, 2024.

[8] Arnoud van Delden: *Kurzschlussfeste Leistungsausgänge für DIY-Elektronikmodule*. ft:pedia 2/2024, S. 84–91.

[9] Arnoud van Delden: *Measure rotational speed with IR sensors.* Auf whizzbizz.com.

[10] Arnoud van Delden: *Sensor Adaptor Module*. Auf whizzbizz.com.

[11] Arnoud van Delden: *Zwei Varianten des Sensor-Adapter-Board-Gehäuses.* Auf github.com, 2025.

[12] Arnoud van Delden: *Elektronik, Module und Sensoren.* Online-Katalog auf whizzbizz.com.

[13] Wikipedia: *Diode-Transistor-Logik*.

[14] Wikipedia: *De-morgansche Gesetze*.

Computing

BT Smart Controller – der Universelle

Axel Chobe

Der BT Smart Controller ist der günstigste (Einsteiger-) Mikrocontroller von fischertechnik. Er kann inzwischen aus unterschiedlichen Programmierumgebungen angesteuert werden. Die Programmierbarkeit über Bluetooth (BT) ermöglicht die Steuerung von mobilen Modellen, obwohl der Controller über keinen eigenen Programmspeicher verfügt.

Einleitung

Seit 2017 gibt es den BT Smart Controller (161944). Er besitzt vier Eingänge für Sensoren und zwei Ausgänge für Motoren oder für vier Lampen. Die Verbindung zur Programmieroberfläche erfolgt über USB oder Bluetooth. Einen eigenen Programmspeicher besitzt der BT Smart Controller nicht. Das Beginner Set (540586) ist preisgünstig (um 160 €) zu erwerben und damit für Einsteiger besonders empfehlenswert.

Abb. 1: BT Smart Controller (161944)

Die Besonderheit dieses Controllers ist, dass er außer mit ROBO Pro Light auch mit Scratch und seit neuestem auch mit ROBO Pro Coding programmiert werden kann.

Experimentell gibt es auch ein Testprogramm unter dem Raspberry Pi in Verbindung mit der Community-Firmware. Eine weitere Demo beschäftigt sich mit der direkten Steuerung in einem Web-Browser.

Im Folgenden werden alle diese Möglichkeiten hier kurz beschrieben.

BT Smart Beginner Set

Abb. 2: fischertechnik-Baukasten BT Smart Beginner Set (540586)

- Entwicklungsumgebung: ROBO Pro Light
- Programmierung auf einem PC oder mit Android-App über Bluetooth bzw. USB
- Begleitheft, Bauanleitung

ROBO Pro Light ist die abgespeckte Version von ROBO Pro [1]. Das Programm kann kostenlos von der Webseite von fischertechnik heruntergeladen werden.

Abb. 3 zeigt ein Beispielprogramm, das eine Lampe während der Betätigung eines Tasters blinken lässt.

Die Programmierung kann auch am Tablet oder auf einem Smartphone mit der App ROBO Pro Smart erfolgen (Abb. 4).

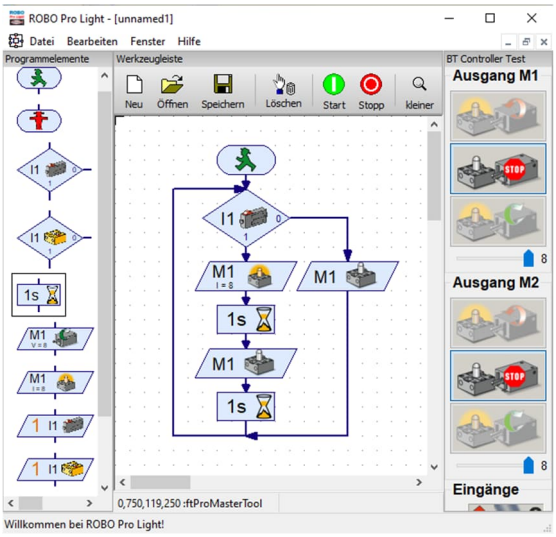


Abb. 3: ROBO Pro Light

Darüber kann außerdem z. B. ein Fahrzeug direkt mit dem Cursor (unten rechts) ferngesteuert werden.

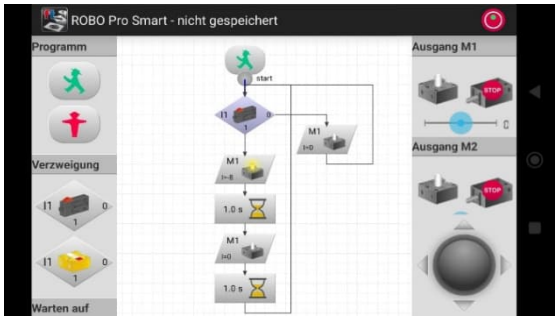


Abb. 4: App ROBO Pro Smart

STEM Coding Pro

Abb. 5: Baukasten STEM Coding Pro

- Programmierumgebung: Scratch [2]
- Programmierung mit der fischertechnik-Coding-Pro-App über Bluetooth bzw. USB
- Bauanleitung und didaktisches Begleitmaterial

Die Bauanleitungen, Hinweise zur Programmierung und Scratch sind in der Coding Pro App vereint. Diese App gibt es für Android, Windows sowie Mac und iOS. Hinter den ersten Schritten und Tutorials sind Videos zum Lernen abrufbar.

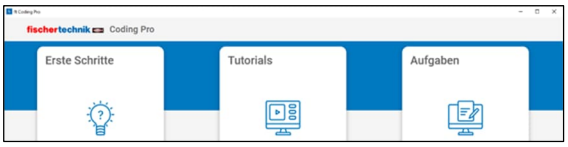


Abb. 6: Coding Pro App

Nach Auswahl eine Aufgabe kann die Bauanleitung Step-by-Step ausgeführt werden. Im Folgenden gibt es dann Hinweise zur Programmierung.

Abb. 7: Aufgabe im Lernmaterial

Von einigen der Seiten kann man zur Programmierumgebung Scratch wechseln. Dazu muss die Art der Verbindung ausgewählt werden (Abb. 8).

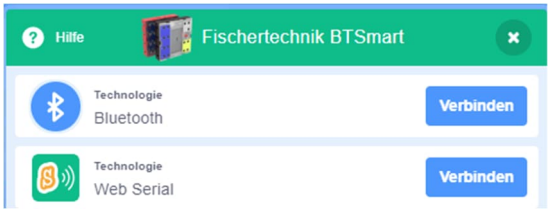


Abb. 8: Auswahl der Verbindung

Es erscheint die Scratch Oberfläche in der App. Links gibt es jetzt eine neue Gruppe von Programmelementen, um Ein- und Ausgänge des Controllers in die Programmierung einzubeziehen (Abb. 9).

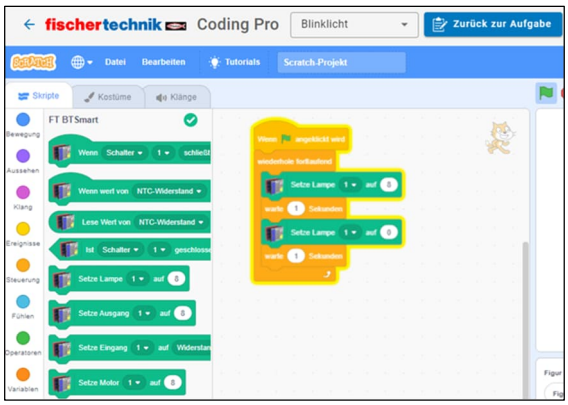


Abb. 9: Programmierumgebung Scratch

Unten das Beispielskript, das die Lampe blinken lässt, solange der Taster gedrückt wird.

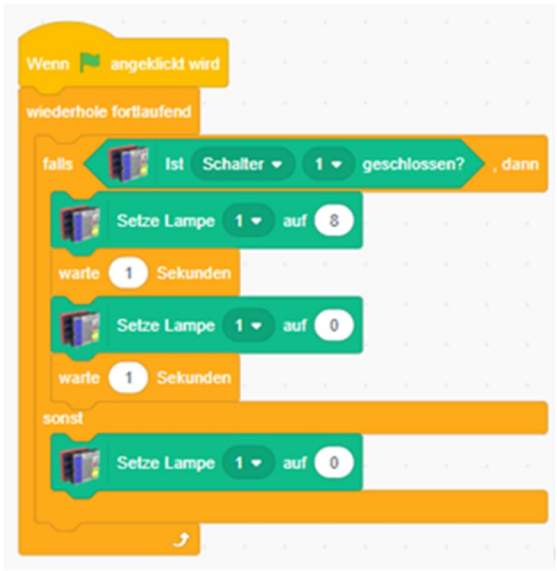


Abb. 10: Beispielskript

Scratch-Erweiterung technika

Von der Karlsruher Technik-Initiative (technika) wurde 2022 eine Scratch-Erweiterung für den BT Smart Controller entwickelt und als Open Source in Github veröffentlicht.

- Programmierumgebung “Scratch”
- Programmierung im Browser Chrome unter allen Betriebssystemen

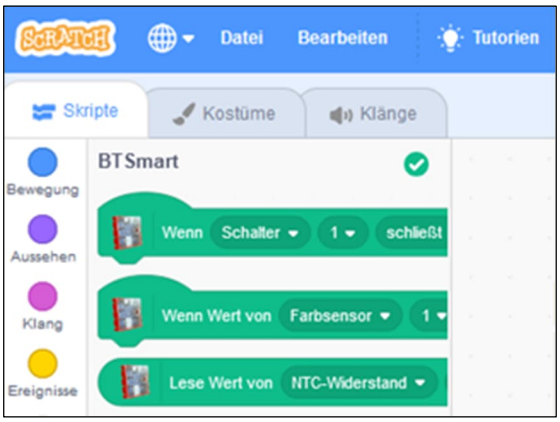


Abb. 11: technika-Integration von Scratch

Nach dem Aufruf der Internetseite muss man unten links den Button *Erweiterungen* betätigen, dann „BTSmart“ auswählen.

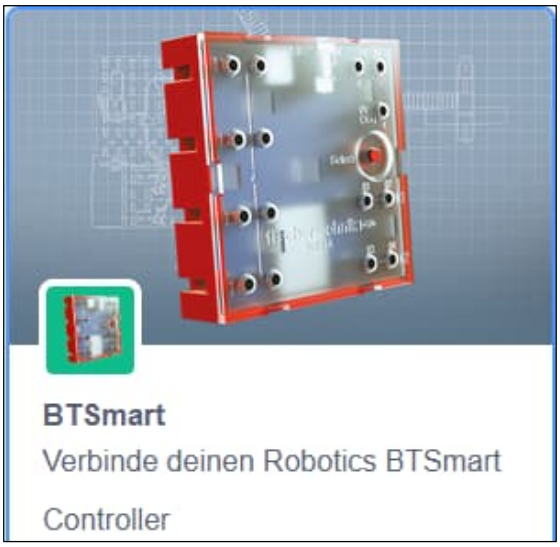


Abb. 12: BTSmart-Erweiterung auswählen

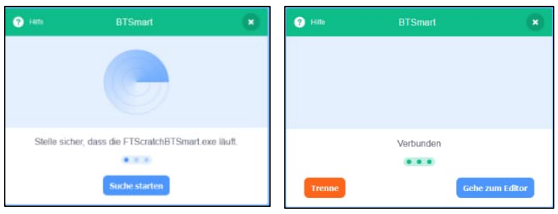


Abb. 13: Mit dem Controller verbinden

Links gibt es jetzt eine neue Gruppe von Programmelementen, um Ein- und Ausgänge des Controllers in die Programmierung einzubeziehen.

Abb. 14: Befehle für den BT Smart Controller

Das Beispielprogramm ist genauso aufgebaut wie bei STEM Coding Pro, da beide mit Scratch programmiert werden.

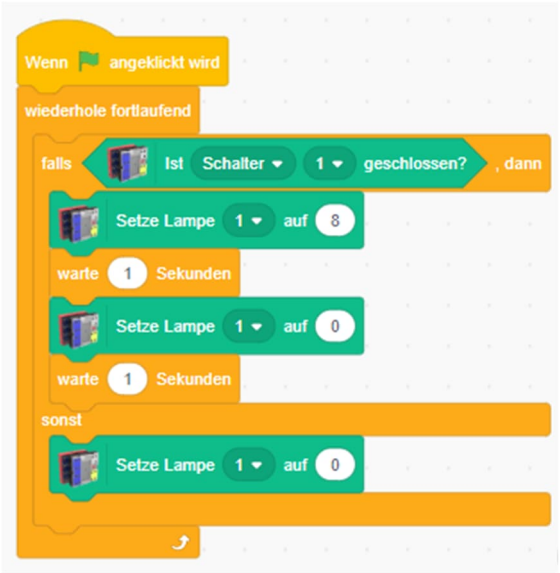


Abb. 15: Beispielprogramm

Smart Robots Pro

Abb. 16: fischertechnik-Baukasten Smart Robots Pro (569021)

- Programmierumgebung „ROBO Pro Coding“ [3] über BT bzw. USB
- Programmierung auf PC Windows, Apple, Android oder Linux
- Bauanleitung, Aufgabenbeschreibung

Einrichten

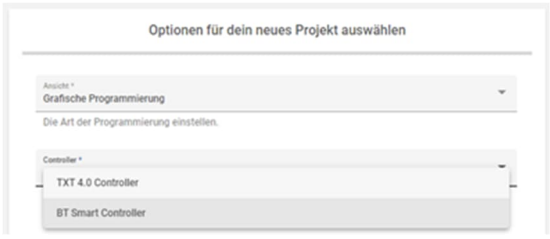


Abb. 17: Auswahl des Controllers in ROBO Pro Coding

Nach dem Öffnen von ROBO Pro Coding – Datei/Neu unter ‚Controller‘ den BT Smart Controller auswählen (Abb. 17).

Die nächste Abfrage wahlweise ‚Leer‘ oder ‚Beispiel‘. Es erscheint die Oberfläche von ROBO Pro Coding.

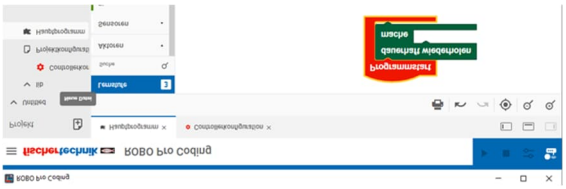


Abb. 18: Programmierumgebung von ROBO Pro Coding

Verbinden

Nach dem Drücken des Buttons im neuen Fenster USB oder Bluetooth wählen und Verbinden. Im folgenden Fenster (Abb. 19, rechts) muss der blaue Text (USB Serial Port) ebenfalls angeklickt werden.

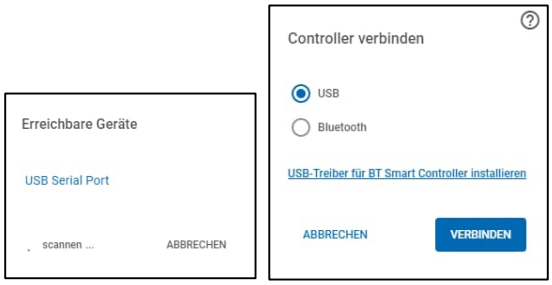


Abb. 19: Verbindung zum Controller aktivieren

Abschließend kann nun in der Controllerkonfiguration der BT Smart ausgewählt werden. Hier müssen noch die entsprechenden Ein- und Ausgänge konfiguriert werden.

Beispielprogramm

Solange der Taster gedrückt wird, blinkt die LED im Sekundentakt.

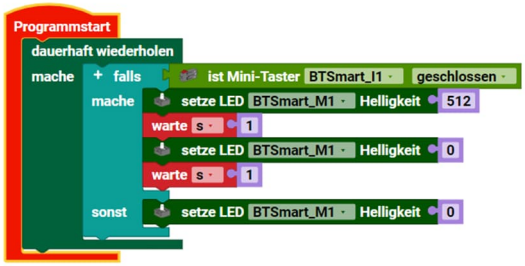


Abb. 20: Beispielprogramm

Community-Firmware mit Raspberry Pi

Grundlage ist ein Raspberry Pi mit einem TouchUI und der fischertechnik Community Firmware. Er ist ein Pi 3 Model B+ mit einem aufgesteckten Waveshare-3,5-Zoll-(A)-Bildschirm (Abb. 21).

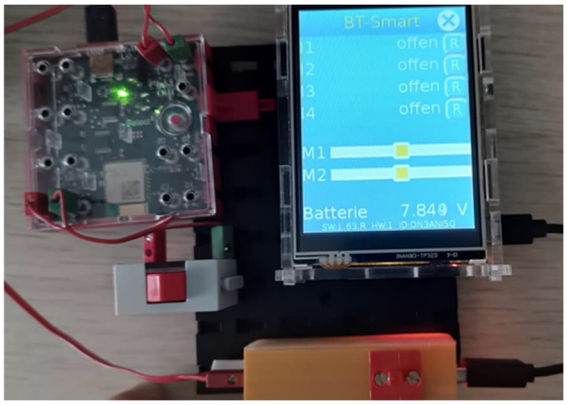


Abb. 21: Raspberry Pi mit Touch-Display

Die Software für die SD-Karte, das TX-Pi, ist ein komplettes Software-Setup für den Raspberry Pi. Es umfasst eine einfach zu benutzende Touchscreen-Oberfläche und kommt mit vorinstallierten Anwendungen sowie der Möglichkeit, weitere Anwendungen aus den App-Stores zu laden. Weitere einzelnen Apps können als Zip-Dateien heruntergeladen werden.

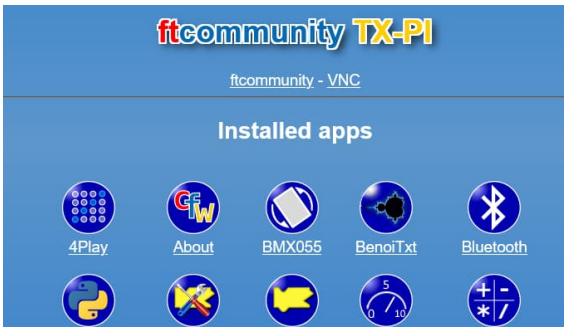


Abb. 22: Community Firmware

Die Verbindung zum PC wird durch den Aufruf der zugewiesenen IP-Adresse hergestellt. Über die Oberfläche des TX-PI auf dem PC können die Zip-Dateien auf den Raspberry geladen werden. Dann kann das Testprogramm BT-Smart aufgerufen werden.

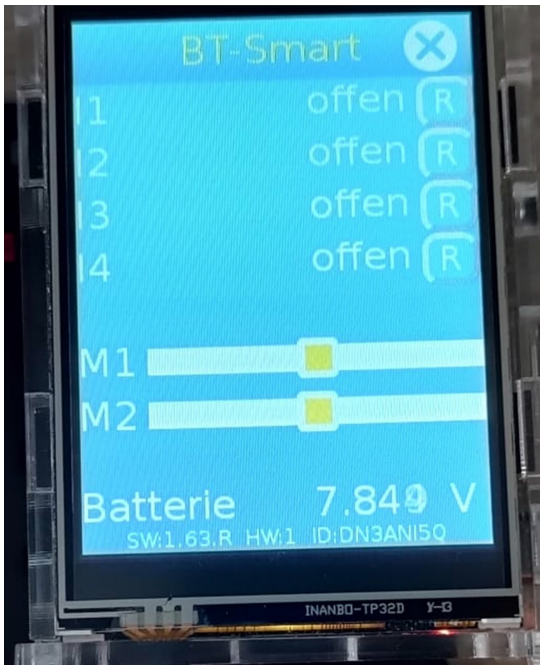


Abb. 22: Testprogramm für den BT Smart

Mit dem Testprogramm werden die Eingänge des BT Smart Controllers überwacht. Für die Motorausgänge kann mit Reglern der Vor- und Rücklauf getestet werden.

Für die Stromversorgung wurde ein 5V-USB-Pi-Versorgungsmodul gebaut (Abb. 23). Weitere Informationen finden sich auf der Webseite zur Community Firmware und dem TX-Pi.

Abb. 23: Stromversorgung

BT Smart Web

Abb. 24: BT Smart Web Demo

Demo zur Steuerung des fischertechnik BT Smart Controllers über einen Browser. Die Verbindung kann dabei über Bluetooth oder USB-Kabel erfolgen. Nähere Informationen zu dieser Demo finden sich auf Github.

Damit die USB-Version funktioniert, wird ein USB-auf-Mini-USB-Host-Modus-Kabel oder ein USB-on-the-go (OTG) auf Mini-USB-Adapter benötigt. Dieses Kabel sollte auf der einen Seite einen Anschluss haben, der zum Smartphone oder Tablet passt (z. B. USB-C bei neueren Handys), und auf der anderen Seite einen Mini-USB-Anschluss, der zum BT Smart Controller passt.

Die Demo läuft unter Chrome auf jedem aktuellen Android-Gerät. Sie wurde auch unter Windows, Linux, MacOS und Android getestet. Bisher lässt sich nach erfolgreicher Verbindung eine Lampe oder ein Motor auf Ausgang 1 ein- und ausschalten.

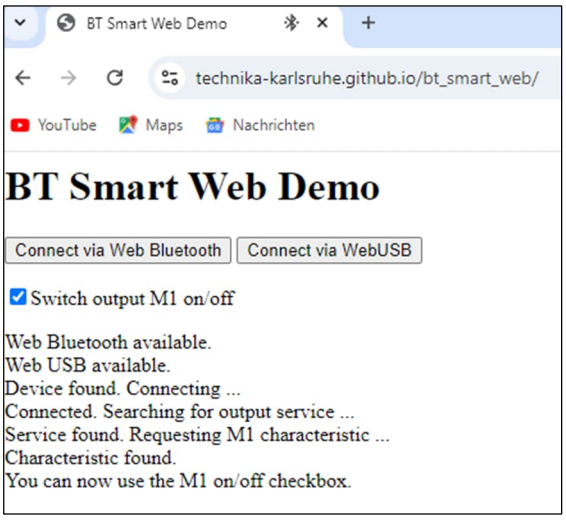


Abb. 25: BT Smart Web Demo (Screen)

Verweise

[1] Axel Chobe: *Übersicht der ROBO Pro Befehle*. 2013-2018

[2] Axel Chobe: *Einführung in die Programmiersprache Scratch*. 2019

[3] Axel Chobe: *Übersicht der Programmierung des TXT 4.0 mit ROBO Pro Coding (Kurzreferenz)*. 2024

Computing

Modelle steuern mit Python – auf dem Raspberry Pi

Peter Habermehl

Das Community-Projekt „TX-Pi" und der ftHAT/TxPiHAT wurden bisher mit keinem Beitrag in der ft:pedia gewürdigt – obwohl sie nicht nur die fischertechnik-Welt mit dem Raspberry Pi um einen verbreiteten Mikrocontroller bereichern, sondern auch einen einfachen Einstieg in die Programmierung mit Python ermöglichen.

Python

Python ist eine vielseitige, hochgradig lesbare und einfach zu erlernende Programmiersprache, die 1991 von *Guido van Rossum* (*1956) entwickelt wurde. Sie zeichnet sich durch eine klare und prägnante Syntax aus, die es Entwicklern ermöglicht, komplexe Programme mit wenigen Codezeilen zu schreiben. Python unterstützt mehrere Programmierparadigmen, darunter objektorientierte, imperative und funktionale Programmierung. Aufgrund ihrer umfangreichen Standardbibliothek und der großen Entwicklercommunity wird Python in vielen Bereichen wie Webentwicklung, Datenanalyse, Künstliche Intelligenz, Automatisierung und wissenschaftliches Rechnen eingesetzt. Python ist Open Source und plattformübergreifend, was es zu einer der beliebtesten Programmiersprachen weltweit macht.

Raspberry Pi

Der Raspberry Pi ist ein kostengünstiger, kreditkartengroßer Einplatinencomputer, der ursprünglich 2012 von der Raspberry Pi Foundation entwickelt wurde, um das Verständnis für Computertechnik und Programmierung zu fördern. Er bietet eine leistungsstarke Plattform für Bildung, Bastelprojekte und Prototypenentwicklung und wird weltweit von Hobbyisten, Studenten und Fachleuten genutzt. Der „Pi" unterstützt verschiedene Betriebssysteme, darunter eine spezielle Linux-Variante namens Raspberry Pi OS, und verfügt über zahlreiche Anschlüsse für Peripheriegeräte wie Tastaturen, Mäuse, Monitore und Kameras. Aufgrund seiner Flexibilität und Erschwinglichkeit hat sich der Pi auch in Bereichen wie dem Internet der Dinge (IoT), der Robotik und der Heimautomation etabliert. Zurzeit ist der Raspberry Pi 5 die neueste und leistungsstärkste Variante.

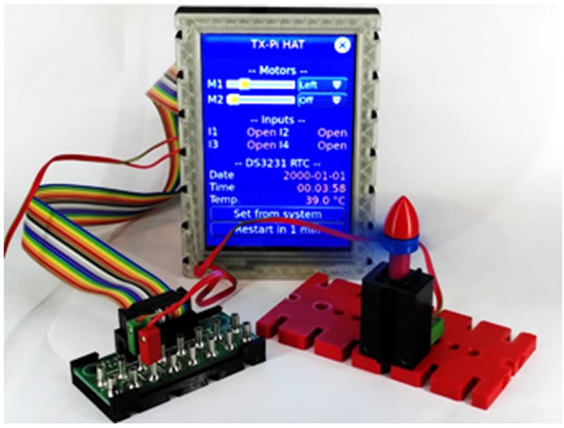


Abb. 1: Raspberry Pi mit ftHAT und 3,5-Zoll Touchscreen im 3D-Druck-Gehäuse

Till Harbaum hat vor einiger Zeit Gehäuse zum 3D-Druck für den damals aktuellen Raspberry Pi 3 konstruiert und drumherum ein Setup entwickelt, um die Community Firmware des fischertechnik TXT Controllers auch auf dem Pi laufen zu lassen.

Komplettiert mit einem kleinen Touchscreen wurde daraus das TX-Pi-Projekt, das im Jahr 2020 noch eine eigene Hardware in Form des ftHAT/TxPiHAT bekam.

Der ftHAT hat folgende Eigenschaften:

- Raspberry-Pi-Versorgung aus jeder fischertechnik-9-Volt-Spannungsquelle
- Versorgung durch Pi selbst kontrolliert (Selbstabschaltung)
- Versorgung durch Echtzeituhr (RTC) kontrolliert (zeitgesteuertes Aufwachen)
- Vier digitale, fischertechnik-kompatible Eingänge (I1 bis I4)
- Zwei analoge fischertechnik-kompatible Motorausgänge (M1 und M2)
- Zwei ftDuino-, fischertechik-TX- und TXT-kompatible I²C-Anschlüsse (3,3V und 5V Busspannung)

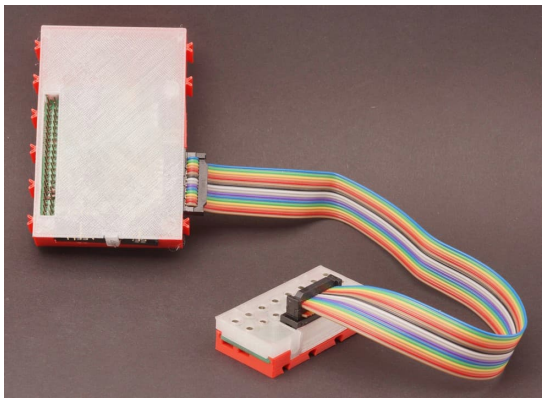


Abb. 2: ftHAT mit Breakout-Board für die I/O-Anschlüsse, im 3D-Druck-Gehäuse

Der Pi ist also geradezu prädestiniert dafür, im fischertechnik-Umfeld als Controller und auch als kleine Workstation zur Softwareentwicklung eingesetzt zu werden. In diesem Beitrag geht es um diesen Einsatz und um die Programmierung mit Python.

Da die General-Purpose-Input-and-Output-Stiftleiste (GPIO) des Raspberry zwar eine Vielzahl von Möglichkeiten bietet, aber die Betriebsspannung von 5V nicht fischertechnik-kompatibel ist, wird eine sinnvolle Adaption auf die fischertechnik-Standards benötigt, um Modelle ansteuern zu können.

Dazu wird der vorgenannte ftHAT verwendet, der – wie bereits erwähnt – vier digitale Eingänge, zwei Motorausgänge sowie 3,3V- und 5V-I²C-Anschlüsse bietet.

Einrichten des Raspberry Pi als Entwicklungssystem

Prinzipiell sind alle Raspberry-Pi-Versionen geeignet. Für flüssiges Arbeiten sind dennoch die aktuellen Versionen 4 und 5 besonders zu empfehlen. Auch sollten mindestens 4 GB Arbeitsspeicher an Bord sein, da das RAM nicht aufrüstbar ist.

Beim Einsatz des ftHAT wird der Pi vom HAT mit Spannung versorgt, da das Gerät dann auch über den Taster am HAT gestartet werden kann. Ansonsten ist auf eine stabile Spannungsversorgung des Pi zu achten.

Als Betriebssystem empfiehlt sich die jeweils aktuelle Version des Raspberry Pi OS. Die Installation ist sehr gut dokumentiert. Man kann den Pi nun direkt mit Tastatur, Maus und Monitor betreiben und hat dann einen vollwertigen Linux-MiniPC.

Sehr praktisch ist aber auch die Variante, den Pi mit ftHAT und 3D-Druck-Gehäusen in das fischertechnik-Modell zu integrieren und den Rechner dann headless zu betreiben, also ohne Monitor und Eingabegeräte, und ihn von einem anderen Rechner aus fernzubedienen. Dazu muss auf dem Pi selbst der VNC-Server aktiviert und auf dem Remote-Rechner ein VNC-Client installiert werden. Raspberry-seitig ist die notwendige Software bereits mit dem Betriebssystem installiert.

Dann wird ein virtueller Desktop auf dem Remote-PC dargestellt, über den man auf dem Pi arbeiten kann. Voraussetzung dafür ist, dass Pi und PC im selben Netzwerk angemeldet sind.

Python auf dem Pi

Für die Programmierung in Python ist standardmäßig mit der Raspberry-Pi-OS-Installation alles Wichtige bereits enthalten. Insbesondere die IDEs Geanie und Thonny

sind vorinstalliert. Thonny ist direkt auf Python ausgerichtet und wendet sich eher an Einsteiger, Geanie beherrscht eine Vielzahl von Programmiersprachen und spricht eher Fortgeschrittene an. Abb. 3 zeigt die Thonny IDE mit einem „Hallo Welt“-Python-Script.

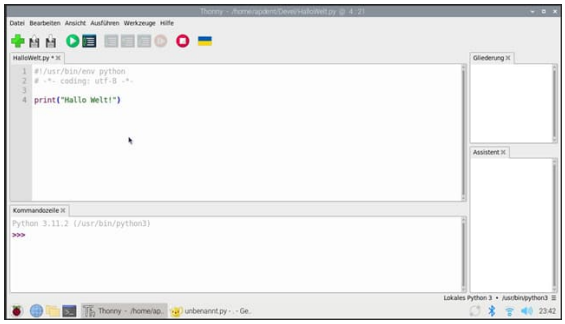


Abb. 3: Thonny auf dem Raspberry Pi Desktop

Als allgemeiner Einstieg in die Programmierung mit Python seien exemplarisch folgende Quellen empfohlen:

- https://www.python-lernen.de
- https://www.python.org/

Ansteuerung des ftHAT in Python

Um nun die Ein- und Ausgänge des ftHAT anzusprechen, kann man ein Raspberry-spezifisches Python-GPIO-Modul importieren und gemäß der HAT-Dokumentation direkt auf die entsprechenden Hardware-Pins zugreifen.

Wesentlich eleganter geht es mit dem txpihat-Modul, welches an den ftHAT angepasst ist und in der jeweils aktuellen Version auf der HAT-Produktseite bei MINTronics erhältlich ist.

Die Datei txpihat.py muss in den Suchpfad des Rechners kopiert werden. Die einfachste Variante ist, sie in dem Verzeichnis abzulegen, in dem man auch den Python-Programmcode speichert.

Das Hallo-Welt-Programm für den HAT, das das Setzen der analogen Ausgänge und das Lesen der digitalen Eingänge demonstriert, ist in Listing 1 wiedergegeben.

Mit dem ftHAT für den Raspberry Pi und dem txpihat.py-Modul für Python ist also der Einstieg in die Python-Programmierung für fischertechnik-Modelle schnell geschafft.

```
#! /usr/bin/env python3
# -*- coding: utf-8 -*-

from txpihat import * # Modul
laden
from time import sleep # Delay-
Funktion aus dem time-Modul laden

# create HAT object
try:
    hat = TxPiHAT()
except Exception as e: # falls das
nicht geklappt hat....
    print(e)
    exit()

# Die Ausgänge heißen "M1" und
"M2"
# Mögliche Status sind "Off",
"Right", "Left", "Brake"
# Sie werden mit m_set_mode()
gesetzt:

hat.m_set_mode("M1", "Brake") #
Motor 1 gebremst aus
hat.m_set_mode("M2", "Right") #
Motor 2 rechts

# Der PWM-Wert für einen Ausgang
kann 0 - 100 betragen
# Er wird mit m_set_pwm() gesetzt:

hat.m_set_pwm("M1", 0) # Motor 1
auf 0%
hat.m_set_pwm("M2", 75) # Motor 2
auf 75%

sleep(5) # 5 Sekunden warten

# Der gelesene Status eines
Einganges kann
# True oder False sein. Die
Eingänge heißen "I1" - "I4"
# Gelesen wird der Eingang mit
get_input():

for i in ["I1", "I2", "I3", "I4"]:
    print("Eingang " + i + " hat
den Status: ", hat.get_input(i))

print(„---Ende---“)
```

Listing 1: „Hallo Welt“ für den HAT

Beliebte Modelle wie z. B. eine Fußgängerampel mit rotem und grünem Signal sowie einem Bedarfs-Taster, ein Händetrockner mit Lichtschranke oder auch ein kleiner Buggy mit Hinderniserkennung oder Spurverfolgung sind mit den zur Verfügung stehenden I/O-Anschlüssen gut zu bedienen. Dies wird Inhalt einer folgenden Veröffentlichung in der ft:pedia sein.

Für komplexere Projekte können über die I²C-Bus-Anschlüsse des HAT weitere Sensoren oder Aktoren genutzt werden. Auch die Anbindung diverser Interfaces wie TXT, TXT4 (ftrobopy), Robo- und Intelligent Interface (libroboint, auch für Python) ist am Pi über USB möglich, allerdings mit unterschiedlichen und insgesamt eher höheren Schwierigkeitsgraden.

Nicht außer Acht zu lassen ist aber auch die Möglichkeit, eine oder gar mehrere Kameras an den Pi anzuschließen. Bilderkennung und Objektverfolgung lassen sich durch die gute Rechenleistung von Pi4 und Pi5 problemlos verwirklichen, entsprechende Bibliotheken wie Tensorflow oder OpenCV vorausgesetzt (siehe z. B. [1, 2]).

Und schlussendlich ist auch das Erstellen von grafischen Benutzerschnittstellen unter Python auf dem Pi möglich, hier sei exemplarisch PyQt erwähnt.

Wer Hilfe, Tipps und Anregungen sucht, kann sich neben der Vielzahl der Python-spezifischen Webseiten und Foren auch im Unterforum für community-Projekte auf der Webseite der fischertechnik community umsehen.

Ausblick

Zusammengefasst ist der Raspberry Pi, gerade in der aktuellen Version 5, ein sehr leistungsstarker, weit verbreiteter Mini-Computer, der als Desktop-Entwicklungssystem gut geeignet ist und insbesondere mit dem ftHAT ganz hervorragend in fischertechnik-Projekte integriert werden kann.

Eine Fortsetzung dieses Artikels mit einer Einführung in die Ansteuerung von I²C-Komponenten am Beispiel des I²C Mini Servo Adapters ist, ebenso wie der bereits erwähnte Artikel über die Steuerung einfacher Modelle, für die nächsten Ausgaben der ft:pedia geplant.

Referenzen

[1] Erik Andresen: *Von Kameras, Himbeeren und schwarzen Hundeknochen*. ft:pedia 2/2014, S. 40–47.

[2] Marco Ahlers: *Schau‘ mir in die Augen, Kleiner! Kamera am TX-Controller*. ft:pedia 2/2014, S. 48–56.

Computing

Der ft-RPI-sa – ein moderner BASIC-Controller für fischertechnik (2)

Robert Lippmann

Nach dem Ein- und Überblick im ersten Teil [1] möchte ich nun etwas detaillierter in die Anwendung und Programmierung sowie auf die Hardware-nahen Befehle und Funktionen des ft-RPI-sa Controllers unter MMBasic eingehen.

Erste Schritte

Bevor wir über die Programmierung sprechen, muss das Programm erst einmal in den Mikrocontroller kommen und wir müssen eine Verbindung zum Controller herstellen. Dafür brauchen wir einen PC oder einen Raspberry Pi, auf den der ft-RPI-sa aufgesteckt wird (HAT). Um die entsprechende Verbindung herzustellen befindet sich auf dem Board eine Reihe mit sieben Kontakten, die es zu konfigurieren gilt (Abb. 1).

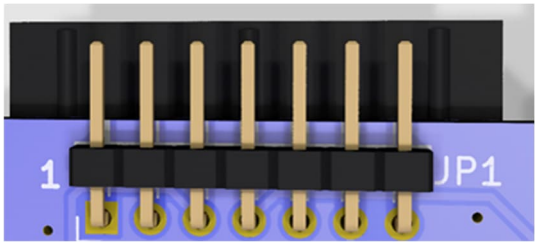


Abb. 1: UART-Konfiguration

Mittels Kurzschlussbrücken über den Kontakten 3-4 sowie 6-7 von JP1 können wir einstellen, dass der ft-RPI-sa die Verbindung zum Raspberry Pi verwendet. Dagegen verbinden die Brücken zwischen 2-3 und 5-6 den Controller mit der UART Verbindung am Peripherieanschluss J5 (Abb. 2).

Falls mit einem PC über die letztgenannte Verbindung zugegriffen wird, wird ein zusätzlicher USB2TTL-Adapter benötigt. Sobald die Verbindung konfiguriert wurde, kann ein Terminalprogramm der Wahl gestartet werden. Die Übertragungsparameter sind vorerst auf 115200 Baud und acht Datenbits, keine Parität und ein Stopbit (8N1) eingestellt. Die Baudrate kann später angepasst werden.

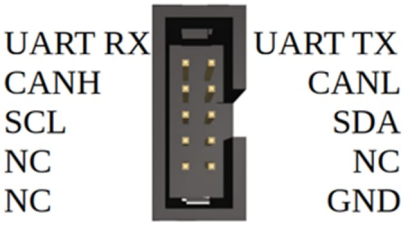


Abb. 2: Peripherieanschluss J5

Der Controller sollte nun über COMx (Windows) oder /dev/ttyUSBx (Linux USB) bzw. /dev/ttySx (Raspberry Pi) erreichbar sein und sich nach dem Einschalten bzw. Drücken des RESET-Knopfes entsprechend melden (Abb. 3).

```
ft-RPI-sa Controller MMBasic Ver 5.05.05
Copyright 2011-2021 Geoff Graham
S32K144 Adaption (C) 2022-2024 Robert Lippmann
Firmware version: Dec  8 2024/12:16:05

>
```

Abb. 3: ft-RPI-sa-„Welcome"-Meldung

Sobald dies erfolgreich war, kann durch Eingabe von EDIT oder Drücken der F4 Taste der eingebaute Editor (Abb. 4) gestartet und sofort mit dem Erstellen eines Programms begonnen werden.

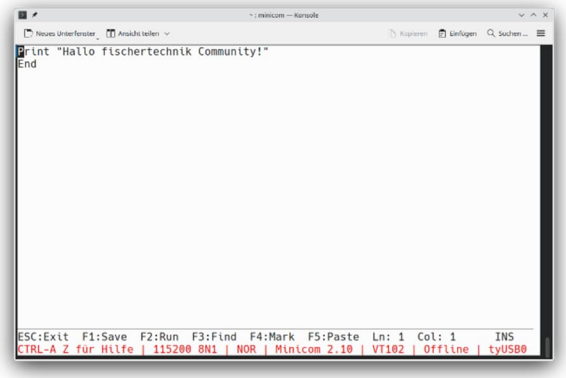


Abb. 4: Editor

Durch drücken von F1 wird das erstellte Programm in dem Speicher des Mikrocontrollers abgelegt und der Editor verlassen. Ein RUN an der Eingabeaufforderung startet dann das Programm.

Eine hübsche bzw. nützliche Erweiterung des Editors besteht darin, die Syntaxeinfärbung mittels

```
OPTION COLOURCODE ON
```

einzuschalten. Damit wird das eingegebene Programm lesbarer und übersichtlicher.

Konservieren

Sobald ein Programm abgeschlossen ist und man sich dem nächsten Projekt widmen will, muss das noch vorhandene Programm im Mikrocontroller irgendwo gespeichert werden, damit es auf Abruf wieder zur Verfügung steht. Dafür ist in MMBASIC ein Kommando implementiert, das das Übertragungsprotokoll XMODEM [2] unterstützt. Das zugegeben schon etwas ältere Protokoll (Geburtsjahr 1977) überträgt Daten zuverlässig mittels Prüfsumme von einem Punkt zum anderen, in unserem Fall zwischen Host und ft-RPI-sa. Dazu ist an der Eingabeaufforderung nur das Kommando

```
XMODEM SEND
```

Einzugeben. Nach Starten des Protokolls am Host wird das Programm aus dem Speicher des ft-RPI-sa an den Host übertragen, wo es gespeichert werden kann. Umgekehrt funktioniert das durch Eingabe von

```
XMODEM RECEIVE
```

am ft-RPI-sa und Starten des Sendevorgangs am Host.

Da die übertragene Datei im Klartextformat vorliegt, kann diese auch auf dem Host mit einem beliebigen Editor bearbeitet oder erstellt werden.

MMBASIC

Da das komplette Manual von MMBASIC einen Umfang von knapp 100 Seiten hat, gehe ich hier nur auf die generelle Art und Weise der Programmierung unter MMBASIC ein und erkläre etwas detaillierter die Befehle, die die implementierte Hardware unterstützen. Ich arbeite gerade daran, das originale MMBASIC-Manual von Geoff Graham auf das ft-RPI-sa relevante Material zu kürzen bzw. um die Befehle und Funktionen zu erweitern, die spezifisch für den Controller sind. Sobald ein brauchbares Manual für den ft-RPI-sa daraus geworden ist, wird es dies als Download geben.

Zuallererst: Have a BREAK

Vorab eine der wichtigsten Informationen überhaupt: Ein Programm und auch manche Befehle lassen sich mit CTRL-C respektive STRG-C abbrechen. Falls also mal ein Motor über das Ziel hinausschießen sollte oder das Programm nicht das macht, was man erwartet, kann dies damit verhindert werden. Nach Drücken der Kombination kommt man wieder an die Eingabeaufforderung.

Datentypen

MMBASIC unterstützt drei verschieden Typen von Variablen:

- Fließkommazahlen
- 64-bit Integer- oder auch Ganzzahlen
- Strings

Der Standardtyp ist die Fließkommazahl, der jedoch durch ein sogenanntes Suffix pro Variable geändert werden kann. So wird aus der Fließkommavariable A durch Anhängen

eines „%“ eine Integerzahl. Zur besseren Identifikation kann die Fließkommavariable auch A! benannt werden. Strings werden durch das Suffix „$“ deklariert. Variablen mit gleichem Namen aber unterschiedlichem Suffix sind verboten und erzeugen eine Fehlermeldung.

Der Standardtyp für Variablen ohne Suffix kann per Befehl

```
Option Default
[Integer|Float|String| None]
```

festgelegt werden.

Zahlen lassen sich mit bestimmten Präfixen aus anderen Zahlensystemen angeben: &H für Hexadezimal, &B für Binär und &O für Oktal. Somit ist &HF = &B1111 = &O17 = 15 (dez.).

Struktur macht das Leben leichter

Wie schon erwähnt lehnt sich das implementierte MMBASIC an das GW-Basic von Microsoft an. Die gefürchteten Zeilennummern – aus der Zeit der Homecomputer aus den 80er Jahren – sind hier optional, d. h. wir können das gefürchtete GOTO sehr wohl verwenden. Jedoch muss jede Sprungmarke mit einem Label versehen werden, wie es z. B. auch in Assembler oder in C gemacht wird:

```
L1:   var=var+1
      If var<100 Then
            Goto L1
      Endif
```

An diesem kleinen Codestück wird auch schon ersichtlich, dass MMBASIC Wert auf Struktur legt. `If-Then-Else`-Kontrukte werden durch ein ENDIF abgeschlossen, was die Verwendung über mehrere Zeilen erlaubt. Die Formatierung, sprich das Einrücken des Codes, wird von MMBASIC leider nicht automatisch vorgenommen. Dies kann jedoch alternativ zu Leerzeichen auch durch Einfügen eines Tabulators erfolgen.

Durch die Verwendung von Sub-Routinen wird ebenfalls eine saubere und übersichtliche Programmierstruktur unterstützt:

```
Sub Fehler
      Print "Fehler aufgetreten!"
End Sub
```

Diese Sub-Routine kann im Hauptprogramm durch Angabe des Namens aufgerufen werden:

```
If Var<0 Or Var>100 Then
      Fehler
Endif
```

Die Flexibilität im Zusammenhang mit Sub-Routinen kann durch Übergabe von Parametern erweitert werden:

```
Sub Fehler Text$
      Print "Fehler: " + Text$
End Sub
```

Der Aufruf erfolgt dann mittels:

```
Fehler "Zahl unzulaessig!"
```

Eine Kleinigkeit sollte ich an dieser Stelle erwähnen: Umlaute aus dem Deutschen wie ä, ü, ö oder das ß werden in MMBASIC nicht unterstützt, auch nicht als Text zur Ausgabe.

Neben Sub-Routinen gibt es auch die Möglichkeit sogenannte Funktionen zu definieren. Diese erlauben es, sowohl Werte aus dem Hauptprogramm zu übernehmen als auch Werte an dieses zurückzugeben:

```
Function quadrat(a)
      quadrat=a*a
End Function
```

Im Hauptprogramm wird dabei lediglich der Aufruf angegeben:

```
xyz=quadrat(12)
```

In MMBASIC gibt es mehrere Methoden um Schleifen zu programmieren. Zuerst die klassische Methode mittels `For-Next`:

```
For A=1 To 50 → (A=1,2,3,4,...)
      ...
Next A
```

Dabei ändert man die Schrittweite durch Hinzufügen eines `Step`:

```
For A=0 to 100 Step 10 →
(A=0,10,20,...)
```

Oder mittels `Do-Loop`:

```
Do While A<=100
      ...
      A=A+1
Loop
```

Diese Variante der Schleife würde beim Erreichen des `Do-While`-Konstruktes überhaupt nicht betreten werden, wenn A einen Wert >100 hat.

Die dritte Variante erweitert die zweite durch eine Abbruchbedingung am Ende der Schleife. Diese Schleife wird auf jeden Fall einmal durchlaufen, da die Abbruchbedingung erst an deren Ende geprüft wird:

```
Do
      ...
      A=A+1
Loop Until A>100
```

Bei der vierten Variante wird bei der `Do-Loop`-Anweisung einfach keine Abbruchbedingung angegeben und die Schleife läuft erst einmal ewig, kann aber mittels des `Exit Do`-Kommandos verlassen werden.

Ein- und Ausgaben

Um mit einem Programm interagieren zu können, bedarf es der Möglichkeit der Ein- und Ausgabe von Informationen bzw. Daten. Den Fall der Ausgabe von Daten haben wir oben schon kennengelernt: `Print`, das aber bei der Eingabe auch durch ein „`?`“ abgekürzt werden kann.

Damit können wir beliebige Informationen über die serielle Schnittstelle an das auf dem Host laufende Terminalprogramm schicken bzw. dort auf dem Bildschirm ausgeben; im einfachsten Fall durch ein

```
Print "Hallo ft:pedia Leser"
```

Um Daten auszugeben, werden Variablen in die Ausgabe eingebunden:

```
Print "Heute ist der ";Tag;". des
Monats"
```

Das Semikolon bindet die Variable `Tag` nahtlos in den Ausgabetext ein:

„Heute ist der 10. des Monats“

… und verhindert am Ende der Zeile einen Zeilenvorschub:

```
Print "Heute ist der ";
Print Tag;
Print " des Monats"
```

wird zu:

„Heute ist der 10. des Monats“

Bei einem Komma statt eines Semikolons wird ein Tabulator ausgegeben:

```
Print "Heute ist der ";Tag,"des
Monats"
```

„Heute ist der 10. des Monats“

Die Ausgabe hängt allerdings davon ab, welche Schrittweite für den Tabulator im Terminalprogramm eingestellt ist.

Um einem Programm Daten zu übergeben verwendet man den `Input` Befehl:

```
Input "Tag heute: "; Tag
```

Damit bekommen wir auf dem Bildschirm Folgendes zu sehen:

„Tag heute: ?“

Die einzugebende Zahl erscheint nach dem Tippen hinter dem Fragezeichen. Dieses Fragezeichen kann vermieden werden, wenn statt des Semikolons ein Komma verwendet wird.

Soweit der schnelle Einblick in die Art und Weise, wie unter MMBASIC programmiert wird.

Hardwarespezifisches

Um nun die Hardware unseres Controllers nutzen zu können und damit unsere fischertechnik-Modelle zu steuern, bedarf es spezieller Befehle und Funktionen für die implementierten Komponenten.

Den Anfang machen die Eingänge

Zum Anschließen von Sensoren bietet der ft-RPI-sa, wie bekannte Controller, acht separate Eingänge. Jeder dieser acht Eingänge verfügt über programmierbare Pull-Widerstände: Pull-Up (+5V), Pull-Down (0V bzw. GND) oder keinen Widerstand (offen bzw. Hi-Z). Der Wert eines Pull-Widerstands beträgt 2,7kΩ.

Um nun einen entsprechenden Zustand zu programmieren gibt es den EPull-Befehl:

```
EPULL <input>, <pull config>
```

Mit dem 8-bit Wert <input> gibt man die zu programmierenden Eingänge an, wobei jedes Bit den entsprechenden Eingang repräsentiert. Bit 0 für Eingang 1, Bit 1 für Eingang 2 usw.

Der 8-Bit-Wert <pull config> wählt den gewünschten Pull-Widerstand aus. „1“ steht für Pull-Up und „0“ für einen Pull-Down.

Wie bekommt man nun einen Eingang ohne Pull-Widerstand? Jeder Eingang der bei <input> auf 0 steht, bekommt keinen aktiven Pull-Widerstand und liegt somit auf Hi-Z, also nicht angeschlossen. Ein Beispiel:

```
EPull &B00001111,&B00001010
```

&B00001111 selektiert mit dem übergebenen Wert die Eingänge I1 bis I4, da Bit 0 bis Bit 3 auf 1 gesetzt sind. Die Bits 4 bis Bit 7 sind 0 und somit werden die Eingänge I5 bis I8 ohne Pull-Widerstand konfiguriert. Der Wert &B00001010 bestimmt für die Eingänge I4 & I2 einen Pull-Up, da die entsprechenden Bits auf 1 stehen, und für I3 & I1 einen Pull-Down-Widerstand, da deren Übergabewert 0 ist.

Um nun einen Eingang bzw. dessen Zustand zu erfahren, gibt es mehrere Möglichkeiten:

1. Der Befehl

```
Print E_Dig(<Eingang>)
```

gibt als Wert des <Eingang>s 1 aus, falls der Eingangsspannungswert > 1,88V liegt, ansonsten eine 0.

2. Der Befehl

```
Print E_Inp
```

gibt die digitalen Zustände aller acht Eingänge auf einmal aus, wobei wieder Bit 0 für Eingang 1, Bit 1 für Eingang 2 usw. stehen. Beispiel:

Rückgabewert 211 = &B11010011

d. h. die Eingänge 1, 2, 5, 7 und 8 liegen auf > 1,88V und die Eingänge 3, 4 und 6 liegen auf 0V bzw. GND.

3. Der Befehl

```
Print E_Ana(<Eingang>)
```

gibt einen 12-Bit digital Wert für den angegebenen <Eingang> aus. Die Spannungsreferenz für den Analog-Digital-Konverter (ADC) beträgt 3,3V. Das bedeutet, der Minimalwert von 0 repräsentiert eine Eingangsspannung von 0V und der Maximalwert von 4095 repräsentiert eine Eingangsspannung von 3,3V.

Um nun eine fischertechnik-übliche 9V Spannung zu verarbeiten, kommen die Pull-Widerstände (R_P) ins Spiel. Angenommen, wir wollen die Ausgangsspannung unseres ft-Netzgerätes messen, die mit 9V deutlich über den 3,3V liegt. Um diese in den Bereich des ADC zu konvertieren, schalten wir einen Pull-Down auf den entsprechenden Eingang, verbinden den Plus-Ausgang unseres Netzgerätes über einen Widerstand (R_V) mit dem Eingang und bauen uns einen Spannungsteiler [3]. Damit sieht das Ganze dann folgendermaßen aus:

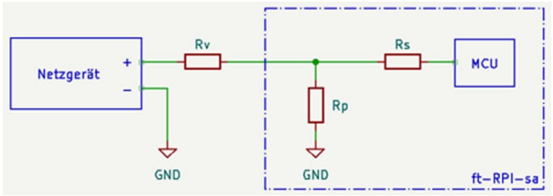


Abb. 5: Messen einer externen Spannung

Der Widerstand R_S ist ein Schutzwiderstand, der dafür sorgt, daß die MCU keinen Schaden nimmt, falls doch einmal eine höhere Spannung als 3,3V anliegen sollte.

Und jetzt wird es etwas akademisch. Wie vorhin erwähnt, haben die Pull-Widerstände, wie der hier gezeigte Pull-Down-Widerstand R_P, einen Wert von 2,7kΩ. An diesem R_P wollen wir eine maximale Spannung von 3V anliegen haben, sobald unser Netzgerät 9V an R_V abgibt.

Die Spannung von 9V, die am Spannungsteiler aus R_V und R_P nun anliegt, verteilt sich im Verhältnis der Widerstände:

$$\frac{U_V}{U_P} = \frac{R_V}{R_P}$$

Wenn wir jetzt die Formel nach R_V auflösen bzw. umstellen, können wir R_V berechnen:

$$\frac{U_V \cdot R_P}{U_P} = \frac{(9\text{V} - 3\text{V}) \cdot 2700\Omega}{3\text{V}} = 5400\Omega$$

Um uns das Leben leichter zu machen nehmen wir einen käuflich leicht erhältlichen 5,6kΩ Widerstand. Wenn an $R_P + R_V$ nun 9V anliegen, fließt durch beide gemäß Ohmschem Gesetz ein Strom von:

$$I = \frac{U}{R} = \frac{9V}{5600\Omega + 2700\Omega} \approx 1{,}08\text{mA}$$

Diese 1,08mA erzeugen an R_P einen Spannungsabfall von 2,928V. Unser ADC arbeitet mit 12 Bit, was insgesamt 4096 (2^{12}) Schritten entspricht. Teilen wir nun die Referenzspannung von 3,3V durch diese 4096 Schritte, erhalten wir pro Schritt 0,000805664V, also rund 806µV. Teilen wir die 2,928V durch diese 806µV bekommen wir einen Wert von 3633.[2]

Wann immer wir also als Rückgabewert vom ADC eine 3633 erhalten, wissen wir, dass unser Netzteil an R_V exakt 9V ausgibt.

Mancher wird sich nun fragen, ob der Anschluss von R_S das Ergebnis nicht beeinflusst. Zugegeben, da fließt ein Strom in den Analog-Digital-Wandler der MCU, um den internen (Sampling-)Kondensator aufzuladen. Das passiert aber nur sehr kurz, und im Vergleich zum Strom durch den Spannungsteiler ist der Strom durch R_S so gering, dass er vernachlässigbar ist.

Wir können auch einen Sensor direkt an einen Eingang anschließen. Nehmen wir z. B. einen Widerstand wie den rechts gezeigten fischertechnik-NTC-Widerstand 1,5kΩ des Typs 36437 und verbinden

ihn mit einem Eingang und GND (Abb. 6):

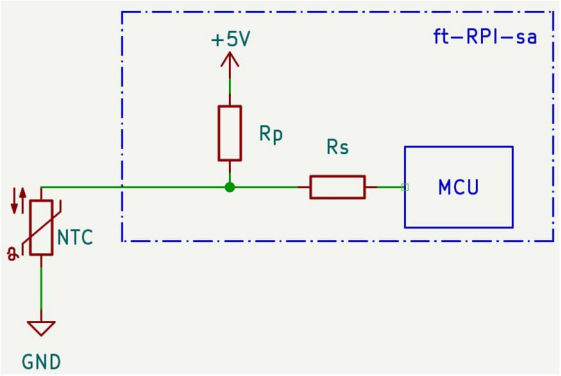


Abb. 6: Anschluss eines NTCs

Jetzt konfigurieren wir den Pull-Widerstand als Pull-Up und schon können wir die Temperatur über die Spannung am NTC messen.

Bei einer Raumtemperatur von 25°C beträgt der Widerstandswert des NTC 1,5kΩ. Die Pull-Spannung von +5V verteilt sich, wie zuvor, im Verhältnis der Widerstände:

$$\frac{U_{NTC}}{U_{PULL}} = \frac{R_{NTC}}{R_P + R_{NTC}}$$

Nach U_{NTC} umgestellt ergibt sich somit eine Spannung am ADC von 1,79V:

$$U_{NTC} = \frac{R_{NTC} \cdot U_{PULL}}{R_P + R_{NTC}} = \frac{1500\Omega \cdot 5\text{V}}{2700\Omega + 1500\Omega} = 1{,}79\text{V}$$

Um weitere Spannungen am ADC zu berechnen empfehle ich, das Datenblatt bzw. die Widerstandskurve des verwendeten NTC griffbereit zu haben. Ein Beispiel dazu findet ihr am Ende des Beitrags (Abb. 8).

Zählen will gelernt sein

Wie schon in Teil 1 des Beitrags erwähnt sind die Zählereingänge C1 bis C4 auf zwei unterschiedliche Arten implementiert. Die erste Art kompensiert das Prellen und ist somit auch langsamer, die zweite Art ver-

[2] *Randnotiz*: Bei einer exakten Berechnung und nur einer finalen Rundung des Ergebnisses landen wir zwar bei 3634, aber wir haben auch alle Bauteiltoleranzen vernachlässigt. Für unseren Zweck ist das Ergebnis völlig ausreichend, ich wollte es jedoch der Vollständigkeit halber erwähnen.

zichtet auf ein Entprellen und ist auf Geschwindigkeit bzw. hohe Frequenzen ausgelegt. Beiden gemeinsam ist, dass gezählt wird, sobald eine Zustandsänderung von 9V auf 0V stattfindet.

Das Auslesen der Anzahl der Impulse erfolgt mit der Counter-Funktion

```
Print Counter(<Input>)
```

wobei <Input> eine Zahl von 1 bis 8 sein kann. 1 bis 4 repräsentieren dabei die Zählereingänge mit automatischer Entprellung, 5 bis 8 sind die Eingänge ohne Entprellung, die die Impulse direkt durchreichen. Jede Zählerfunktion kann durch Zuweisen einer 0 zurückgesetzt werden; andere Werte können nicht geschrieben werden.

Interessant wird es, wenn man durch Drücken eines angeschlossenen Tasters die Werte von Counter 1 mit denen von Counter 5 vergleicht. Beide repräsentieren den Zählereingang C1, einmal mit und einmal ohne Entprellen.

Extras für die Eingabe

Neben den Eingängen für die Sensoren und den Zählereingängen befinden sich zwei weitere Extra-Komponenten auf dem Board, die eine Eingabe ermöglichen.

Die eine ist ein 5-facher DIP-Schalter, die andere ein Potentiometer. Erstere können durch

```
Print Dipswitch
```

ausgelesen werden. Auch hier steht wieder jedes Bit für einen Schalter – Bit 0 für Schalter 1, Bit 1 für Schalter 2 usw.

Um z.B. einen Schwellenwert oder die Geschwindigkeit eines Motors vorzugeben, kann das implementierte Potentiometer verwendet werden. Die Abfrage des Wertes geschieht mit der weiter oben schon erwähnten Funktion E_ANA(<Eingang>), nur dass <Eingang> hier den Wert 9 haben muss. Der Rückgabewert ist auch diesmal ein 12-Bit-Wert im Bereich von 0 bis 4095.

Treibendes

Nach dem Auslesen der Sensoren und dem Zählen der Impulse wollen wir unsere Aktoren nun entsprechend steuern.

Wie im ersten Teil erwähnt besitzt der ft-RPI-sa insgesamt vier vollwertige H-Brücken, die sich aufteilen lassen, sodass jeder Ausgang individuell steuerbar ist.

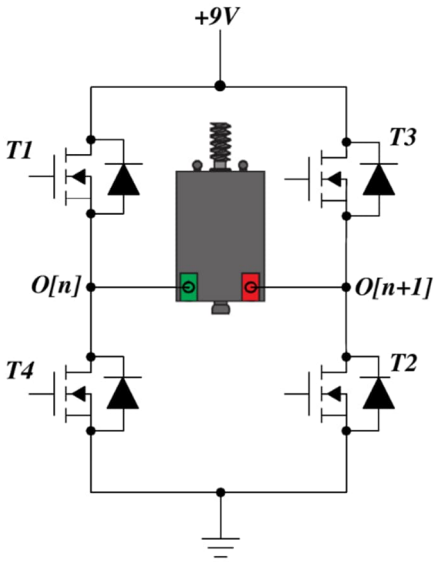


Abb. 7: H-Brücke mit Motor

In MMBasic wird ein Ausgang mit dem OUTPUT-Befehl konfiguriert:

```
Output Conf <Device>, <Mode>
```

<Device> ist entweder eine 1 oder 2, wobei eine 1 die Ausgänge 1-4 und eine 2 die Ausgänge 5-8 kontrolliert.

<Mode> ist entweder 1 oder 3, wobei Mode 1 für die volle Kontrolle eines Bürsten- oder Schrittmotors verwendet wird. Dieser Modus erlaubt es, einen Bürstenmotor im Rechts- bzw. Linkslauf zu betreiben, sowie ausgleiten oder abrupt anhalten zu lassen.

Es werden nur die Transistoren T[n] und T[n+1] (Abb. 7, mit n=1 oder 3) durchgeschaltet und somit beide Drehrichtungen ermöglicht. Zum Anhalten werden T2 und T4 geschaltet, wodurch die generierte Spannung des Motors, unabhängig der Drehrichtung, über einen der beiden Transistoren und einer der Dioden kurzgeschlossen wird und er dadurch extrem schnell abbremst.

Als letzte Möglichkeit wird keiner der Transistoren durchgeschaltet, was einem Anhalten des Motors durch Ausgleiten entspricht. Im Übrigen kümmert sich der Treiberbaustein automatisch darum, beim Umschalten der Polarität zwischen O[n] und O[n+1] eine kleine Totzeit einzulegen, um einem Kurzschluss zwischen +9V und GND vorzubeugen.

Der Mode 3 entkoppelt die beiden H-Brücken-Hälften, sodass jeder Ausgang unabhängig für Leuchten, Elektromagnete oder Bürstenmotoren in einer Drehrichtung verwendet werden kann.

Es gibt mehrere Befehle, mit denen ein Ausgang aktiviert werden kann.

```
Output Write <Wert>
```

<Wert> ist eine 8-Bit Zahl, bei der wieder jedes Bit einem Ausgang zugeordnet ist. Bit 0 für Ausgang 1, Bit 1 für Ausgang 2 usw.

Eine 1 schaltet den entsprechenden Ausgang ein und eine 0 schaltet ihn aus. Dieser Befehl ist nur wirksam, wenn die entsprechende H-Brücke im Modus 3 konfiguriert ist; andernfalls wird der Wert ignoriert. Modus 3 ist auch Voraussetzung für den folgenden Befehl:

```
Output Set <Output>,
[ON|OFF|<Level>]
```

Hier ist <Output> wieder eine 8-Bit Zahl mit entsprechender Ausgangszuordnung wie zuvor. Es besteht jedoch die Möglichkeit, nur ausgewählte, auf 1 gesetzte Ausgänge zu ändern. ON bzw. OFF bedarf keiner weiteren Erklärung, aber mit Angabe eines Prozentwertes von 0 bis 100 lässt sich mittels PWM die Spannung am Ausgang in Relation zur Versorgungsspannung einstellen.

Folgende Befehle benötigen eine H-Brücke in der Konfiguration Mode 1:

```
Output Motor <Bridge>,
<Direction>, [<Level>]
```

<Bridge> ist die Nummer der entsprechenden H-Brücke und liegt im Bereich 1 bis 4. H-Brücke 1 und 2 steuern die Ausgänge 1 und 2 sowie 3 und 4. H-Brücke 3 und 4 steuern die Ausgänge 5 und 6 sowie 7 und 8.

<Direction> variiert zwischen 1 und 4 und ist definiert als:

- 1: Motor dreht sich links oder rechts (abhängig von der Polung)
- 2: Motor dreht sich gegenläufig zu 1
- 3: Abrupter Halt
- 4: Motor gleitet aus

<Level> ist wieder der Ausgangspegel, angegeben in Prozent, und ist nur erlaubt bzw. Voraussetzung, wenn <Direction> 1 oder 2 ist.

Ein spezieller Befehl dient der Nutzung der Encodermotoren:

```
Output EncMotor <Bridge>, <Counter
Input>, <#Steps>, <MinSpeed>,
<MaxSpeed>
```

<Bridge> ist identisch mit dem gleichlautenden Parameter, der für den Output-Motor-Befehl verwendet wird.

<Counter Input> ist die Nummer des verwendeten Zähleingang C[1..4], an den der Encoder angeschlossen ist.

<#Steps> ist die Anzahl der durchzuführenden Schritte, die im Bereich ±10000 liegen darf.

Die Drehrichtung des Motors hängt von der Polung ab. Eine negative Anzahl <Steps> dreht den Motor in die entgegengesetzte Richtung wie bei einer positiven Anzahl.

<MinSpeed>, <MaxSpeed> sind jeweils eine Zahl im Bereich 1-100 und bestimmen den prozentualen Anteil der Maximalspannung:

- Sobald der Motor startet, beginnt dieser sich mit dem Pegel <MinSpeed> zu drehen; dieser Pegel erhöht sich bis zum angegebenen Pegel <MaxSpeed>.
- Sobald der Motor sich dem Ende der durchzuführenden Schritte nähert, verringert sich die Geschwindigkeit

wieder, bis <MinSpeed> erreicht ist, um den Motor exakt zu positionieren.
- Falls <MinSpeed> zu hoch gewählt wurde und der Motor nicht auf die exakte Position „geparkt“ werden konnte, wird eine Warnmeldung ausgegeben.
- Falls <MinSpeed> zu niedrig gewählt wurde, kann, je nach Last, die exakte Position nicht erreicht werden, da der Motor sich evtl. überhaupt nicht mehr dreht.

Die optimalen Parameter hierfür lassen sich nur durch Ausprobieren am zu steuernden Modell herausfinden.

Der letzte Befehl der Reihe ist für das Ansteuern eines Schrittmotors vorgesehen:

```
Output Stepper <Motor>, <#Steps>,
<Speed>
```

<Motor> ist die Nummer des anzusteuernden Motors und liegt im Bereich 1 bis 4. Um einen Schrittmotor anzusteuern, sind zwei vollwertige H-Brücken notwendig. Somit kann der ft-RPI-sa maximal zwei Schrittmotoren ansteuern.

Die angegebene Motornummer ist kodiert als:

- 1: Motor an Ausgang 1-4 (Vollschritt)
- 2: Motor an Ausgang 5-8 (Vollschritt)
- 3: Motor an Ausgang 1-4 (Halbschritt)
- 4: Motor an Ausgang 5-8 (Halbschritt)

<#Steps> ist die Anzahl der durchzuführenden Schritte und liegt im Bereich ±10000.

<Speed> ist die Frequenz, mit der die Schritte durchgeführt werden und ist limitiert durch den verwendeten Motor. Der erlaubte Bereich liegt zwischen 1-350Hz.

Alle Ausgänge sind selbstverständlich gegen Kurzschluss geschützt. Falls es doch einmal zu einem Kurzschluss kommen sollte, werden die entsprechenden Ausgänge abgeschaltet, die Status LED blinkt dreimal mit ungefähr 3Hz und nach einer anschließenden Pause von 1,5s wiederholt sich der Vorgang, bis der Kurzschluss für mehr als 100ms entfernt wurde.

Während der Kurzschluss signalisiert wird läuft das MMBASIC-Programm im ft-RPI-sa Controller weiter.

Beispielprogramm

Nachdem jetzt die meisten Steuerungsbefehle bzw. Funktionen zur Steuerung erläutert wurden, erklärt sich das im ersten Teil gezeigte Programm mehr oder weniger von selbst:

```
Option default integer
EPull 2,2
Output conf 1,3
Const Udiv!=3.3/4096
Const Slope!=(18-40)/(2051-799)
Do
  MyInput=E_Ana(2)
  Intc!=(5-MyInput*Udiv!)/2700

MyTemp!=40+Slope!*(MyInput*Udiv!/I
ntc!-799)
  If MyTemp!>25 Then
    Output set 1,ON
  Else
    Output set 1,OFF
  EndIf
Loop
```

An Eingang 2 befindet sich ein NTC, angeschlossen gemäß Abb. 6, und an Ausgang 1 und GND ein Motor, der einen Lüfter antreibt.

Mit `Option default integer` wird der Standardtyp für Variablen auf Integer gesetzt. Es wird von nun an kein Präfix „%“ für diesen Variablentyp mehr benötigt.

Die Befehle `Epull` und `Output Conf` wurden vorhin hinreichend erklärt.

Nun werden vorab die Konstanten für die Schrittweite des ADC in mV für 12-Bit berechnet und die Steigung der Widerstandskurve für den NTC im Bereich 18-40 Grad Celsius berechnet – Stichwort: *Lineare Interpolation* [5], da der Widerstandswert des NTC über die Temperatur nicht linear, also als Gerade im Diagramm, verläuft.

Falls eine als Konstante definierte Variable im weiteren Programmverlauf geändert werden sollte, erscheint eine Fehlermeldung.

In der DO-Schleife, die keine Abbruchbedingung besitzt, wird zuerst der Analogwert vom Eingang 2 eingelesen. Danach wird der Strom durch den NTC berechnet, der für die finale Berechnung der Temperatur benötigt wird. Abhängig von der berechneten Temperatur wird der Motor bzw. der Lüfter entweder ein- oder ausgeschaltet.

Soweit die Erklärung des gezeigten Programms. Es ist sehr einfach und die Regelung sehr minimalistisch.

Exkurs: Wie funktioniert die lineare Interpolation? [5, 6]

$$f(x) = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0)$$

mit

$$\frac{f_1 - f_0}{x_1 - x_0}$$

als schon berechnete Steigung in der Konstante `Slope!` und $f_1 = 18°C$, $f_0 = 40°C$, $x_1 = 2051\Omega$, $x_0 = 799\Omega$ (Abb. 8).

Das Ergebnis ist negativ, da die Kurve bzw. der Widerstandswert mit steigender Temperatur abfällt.

Es ergibt sich somit

$$f(x) = f_0 + Slope! \cdot (x - x_0)$$

mit x als der aktuelle NTC- Widerstandswert, berechnet mit:

$$R_{NTC} = \frac{E_Ana(2) \cdot U_{DIV}!}{I_{NTC}!}$$

R25=1.5KΩ						B25/50=3950K					
T	R	T	R	T	R	T	R	T	R	T	R
-30	27.36	-4	6.049	22	1.712	48	0.584	74	0.232	100	0.105
-29	25.7	-3	5.738	23	1.638	49	0.562	75	0.225	101	0.105
-28	24.149	-2	5.445	24	1.567	50	0.541	76	0.218	102	0.102
-27	22.697	-1	5.169	25	1.5	51	0.521	77	0.211	103	0.099
-26	21.339	0	4.935	26	1.435	52	0.502	78	0.204	104	0.097
-25	20.07	1	4.663	27	1.374	53	0.483	79	0.198	105	0.094
-24	18.882	2	4.431	28	1.316	54	0.466	80	0.192	106	0.092
-23	17.771	3	4.213	29	1.261	55	0.449	81	0.186	107	0.09
-22	16.731	4	4.006	30	1.208	56	0.433	82	0.18	108	0.088
-21	15.758	5	3.811	31	1.158	57	0.417	83	0.175	109	0.085
-20	14.847	6	3.627	32	1.11	58	0.402	84	0.17	110	0.083
-19	13.995	7	3.453	33	1.065	59	0.388	85	0.165	111	0.081
-18	13.197	8	3.288	34	1.021	60	0.375	86	0.16	112	0.079
-17	12.449	9	3.132	35	0.98	61	0.361	87	0.155	113	0.078
-16	11.748	10	2.985	36	0.94	62	0.349	88	0.15	114	0.076
-15	11.091	11	2.845	37	0.903	63	0.337	89	0.146	115	0.074
-14	10.475	12	2.712	38	0.866	64	0.325	90	0.142	116	0.072
-13	9.898	13	2.587	39	0.832	65	0.314	91	0.138	117	0.071
-12	9.356	14	2.468	40	0.799	66	0.304	92	0.134	118	0.069
-11	8.847	15	2.356	41	0.768	67	0.293	93	0.13	119	0.068
-10	8.369	16	2.249	42	0.738	68	0.283	94	0.127	120	0.066
-9	7.921	17	2.147	43	0.709	69	0.274	95	0.123	121	0.065
-8	7.499	18	2.051	44	0.682	70	0.265	96	0.12	122	0.063
-7	7.102	19	1.96	45	0.656	71	0.256	97	0.117	123	0.062
-6	6.73	20	1.873	46	0.631	72	0.248	98	0.113	124	0.061
-5	6.379	21	1.791	47	0.607	73	0.24	99	0.11	125	0.06

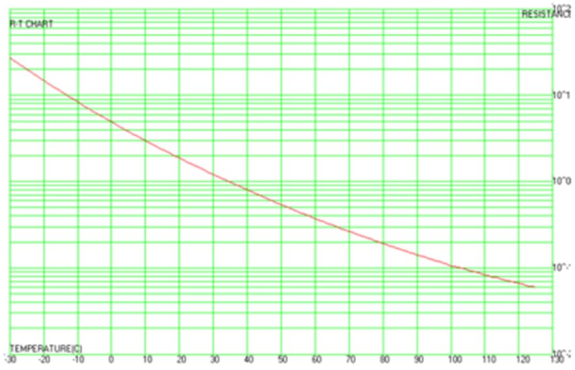


Abb. 8: NTC-Widerstand (Datenblatt [7])

Als Herausforderung könnten folgende Punkte verbessert bzw. eingebaut werden:

- Hysterese für das Ein- und Ausschalten des Motors
- PWM-Ansteuerung um den Regelungspunkt
- Verwendung einer Interpolationstabelle, um der Widerstandskurve besser zu folgen.

Weitere Verbesserungen überlasse ich eurer Phantasie.

Referenzen

[1] Robert Lippmann: *Der ft-RPI-sa – ein moderner BASIC-Controller für fischertechnik (1)*. ft:pedia 4/2024, S. 82–87.

[2] Wikipedia: *XMODEM*.

[3] Wikipedia: *Spannungsteiler*.

[4] ft-Datenbank: *NTC-Widerstand 1,5kOhm (36437)*.

[5] Wikipedia: *Lineare Interpolation*.

[6] Dirk Fox: *„Einmessen“ eines digitalen Messgeräts*. ft:pedia 1/2013, S. 39–48.

[7] fischertechnik: *36437-NTC-resistor*. Datenblatt, FT-T-KN, 07.08.2017.