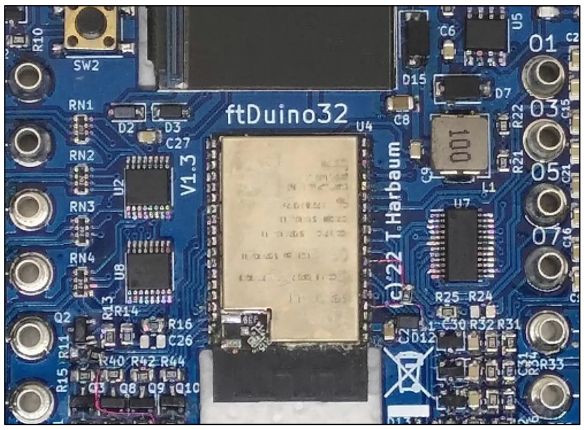
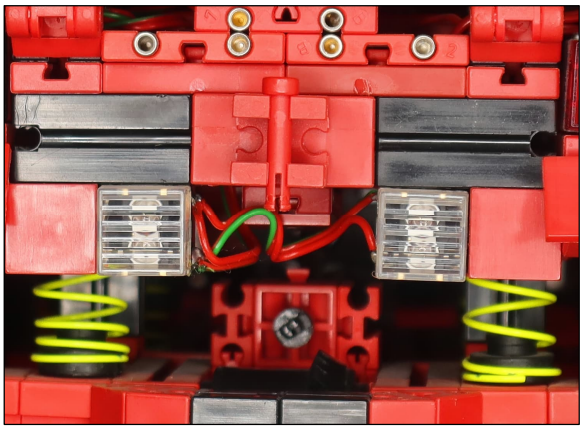
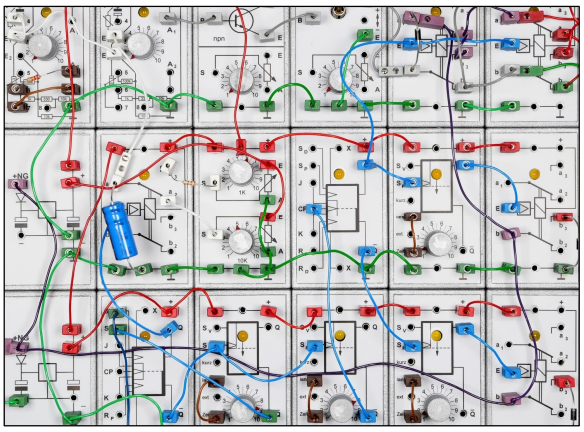
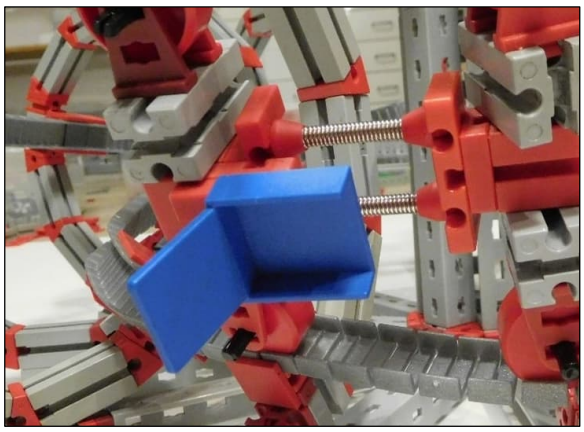
ft:pedia

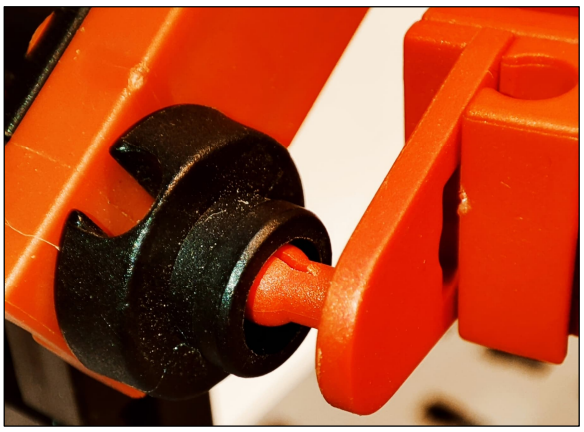
Heft 4/2023











Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

Die Zeit, die Zeit

Dirk Fox, Stefan Falk

Aus der an wunderbaren Modellideen reichen Bildersammlung der ft:community sticht ein Thema hervor, das fast jeden langjährigen fischertechniker schon einmal in Bann geschlagen hat: mechanische Uhren.

Die Gründe dafür sind sicher vielfältig, aber drei davon scheinen mir besonders gewichtig. So enthält eine Uhr komplexe Getriebe aus Zahnrädern, die eine Schlüsselrolle in der Geschichte der Technik spielen und bis heute aus mechanischen Antrieben nicht wegzudenken sind. Sie gehören zudem mit den Grundbausteinen zu den Kernelementen des fischertechnik-Systems.

Hinzu kommt, dass eine fischertechnik-Uhr einen unmittelbaren Nutzen hat – sie ist kein Modell, sondern ein funktionierendes Produkt, dessen Genauigkeit kommerziellen Lösungen sehr nahe kommt: Das hat Martin Romann mit seiner legendären Standuhr 2010 eindrucksvoll nachgewiesen.

Eine große Rolle scheint mir schließlich die besondere Faszination zu spielen, die von mechanischen Uhren als Präzisionsinstrument zur Zeitmessung ausgeht. Erst durch das Messen von Zeitabläufen erkannte die Menschheit die Gesetzmäßigkeiten hinter Tag und Nacht, Mondfinsternissen und sich wiederholenden Jahreszeiten. Die von *Galileo Galilei* (1564-1641) im Jahr 1640 erfundene und 1657 von *Christiaan Huygens* (1629-1695) entwickelte Pendeluhr revolutionierte die Zeitmessung: Diese mechanischen Uhren erreichten eine für die damalige Zeit beeindruckende Ganggenauigkeit von 10 Sekunden/Tag (0,01 %) und ermöglichten so auch an bedeckten Tagen eine präzise Zeitbestimmung.

Schon vor 300 Jahren hatte die genaue Zeitmessung eine immense Bedeutung, wie die Beschäftigung mit dem Längengradproblem (der Bestimmung der geografischen Länge auf See) zeigt: Im Jahr 1714 lobte das britische Parlament das gigantische Preisgeld von 20.000 £ für eine auf ein halbes Grad genaue Längengradbestimmung aus – das entspricht einer Abweichung von max. 55 km (am Äquator) und erfordert eine Uhr mit einer Genauigkeit von 1 s/Tag auf See.

Die Lösung gelang dem gelernten Tischler *John Harrison* (1693-1776) im Jahr 1759 nach jahrzehntelanger Entwicklungsarbeit mit seinem Schiffschronometer H4. Die Uhr wich nach 81 Tagen auf See lediglich 5 Sekunden ab – das entspricht einer Gangabweichung von 0,00007 %. Wer nun einwendet, dass heute (fast) jeder einen GPS-Empfänger in der Hosentasche trägt, sollte daran denken, dass die Positionsbestimmung durch eine Entfernungsmessung zu mehreren Satelliten erfolgt – mit Hilfe der Zeitangabe im Satellitensignal: Werte, die von Atomuhren gemessen werden, mit einer Gangabweichung von 10^{-18}, also maximal einer Sekunde in 300 Mio. Jahren.

Mit einer fischertechnik-Uhr könnt ihr zumindest die Ankunft des Weihnachtsmanns auf wenige Sekunden genau voraussagen. Und bei deren Konstruktion zuvor stundenlang einfach mal die Zeit vergessen.

Beste Grüße,
Euer ft:pedia-Team

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

Inhalt

Termine

Was?	Wann?	Wo?
Bescherung	24.12.2023	Unterm Weihnachtsbaum

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, Ralf Geerken, Till Harbaum, Peter Krijnen, Claus Ludwig, Jeroen Regtien.

Modell

Möbius-Bahn mit fischertechnik

Florian Bauer

Ein Möbiusband [1] ist eine Fläche, die nur eine Kante und eine Seite hat. Ein Vehikel, das sich entlang dieser Fläche bewegt, dreht sich bei zwei Umläufen einmal um seine eigene Achse. Dieser Beitrag zeigt, wie man aus fischertechnik eine Bahn und geeignete Vehikel bauen kann.

Einleitung

Aus einem geschlossenen Band kann man ein Möbiusband herstellen, indem man es an einer Stelle zerschneidet und die Endstücke um 180° verdreht wieder zusammenklebt. Würde nun ein Käfer an einer Stelle auf dem Band starten, an dem seine Beinchen nach unten schauen, würde er nach einem Umlauf auf dem Kopf stehen und nach einem weiteren Umlauf wieder aufrecht orientiert sein.

Unser Käfer hat es hier leicht, da er über die den Insekten innenwohnende Fähigkeit verfügt, über Kopf zu krabbeln. Wollten wir ihn in diesem Gedankenexperiment durch ein Fahrzeug ersetzen, würde das Fahrzeug, das nur durch Schwerkraft auf der Bahn gehalten wird, herunterfallen.

Man könnte das Fahrzeug durch Magnete auf der Bahn halten, wie es Forscher vom Low Temperature Physics Lab des Ithaca College demonstriert haben [5]. Sie lassen ein supraleitendes „Shuttle“ auf einer Bahn aus Magneten schweben.

Als eine weitere Möglichkeit könnte man sich ein aufwändiges Bahnprofil mit einer geeigneten Haltenut vorstellen, durch die das Vehikel über Halteräder geführt wird, die auf und unter der Bahn auf dieselbe geklemmt werden.

Eine ein- oder beidseitige Umklammerung der Bahn ist konstruktionsbedingt schwer

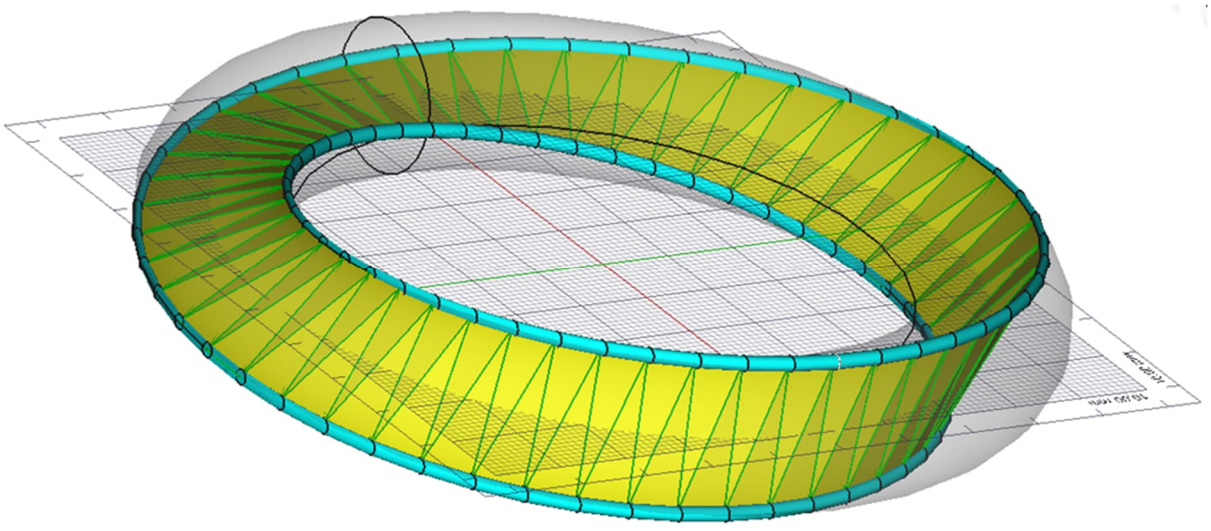


Abb. 1: Eine geschlossene Schiene (hellblau), zwischen der ein Möbiusband (gelb) aufgespannt ist

realisierbar, da man an bestimmten Stellen Halterungen platzieren müsste, die dann der Umklammerung nach einem Umlauf im Weg stünden.

Zu einer unkonventionellen Lösung kommt man, wenn man den Rand des Möbiusbandes betrachtet: Ersetzt man dessen Rand durch eine Schiene und lässt die Fläche dazwischen weg, kann man ein Fahrzeug dazwischen aufhängen, das sich dann entlang dieser Schiene auf einer Möbiusband-Fläche (Abb. 1) bewegt.

Diese Idee wurde bei der Imaginata-Single-Rail-Bahn in Jena realisiert [7]. Diese Bahn ist Grundlage für die hier realisierte Modellidee.

Realisierung mit fischertechnik

Für die Realisierung der Idee sind folgende miteinander zusammenhängenden Teilprobleme zu lösen:

1. Streckenführung und Halterungsstruktur
2. Auswahl der Schienen und deren Befestigung
3. Design eines Vehikels mit entsprechendem Fahrwerk
4. Antrieb bzw. Motorisierung

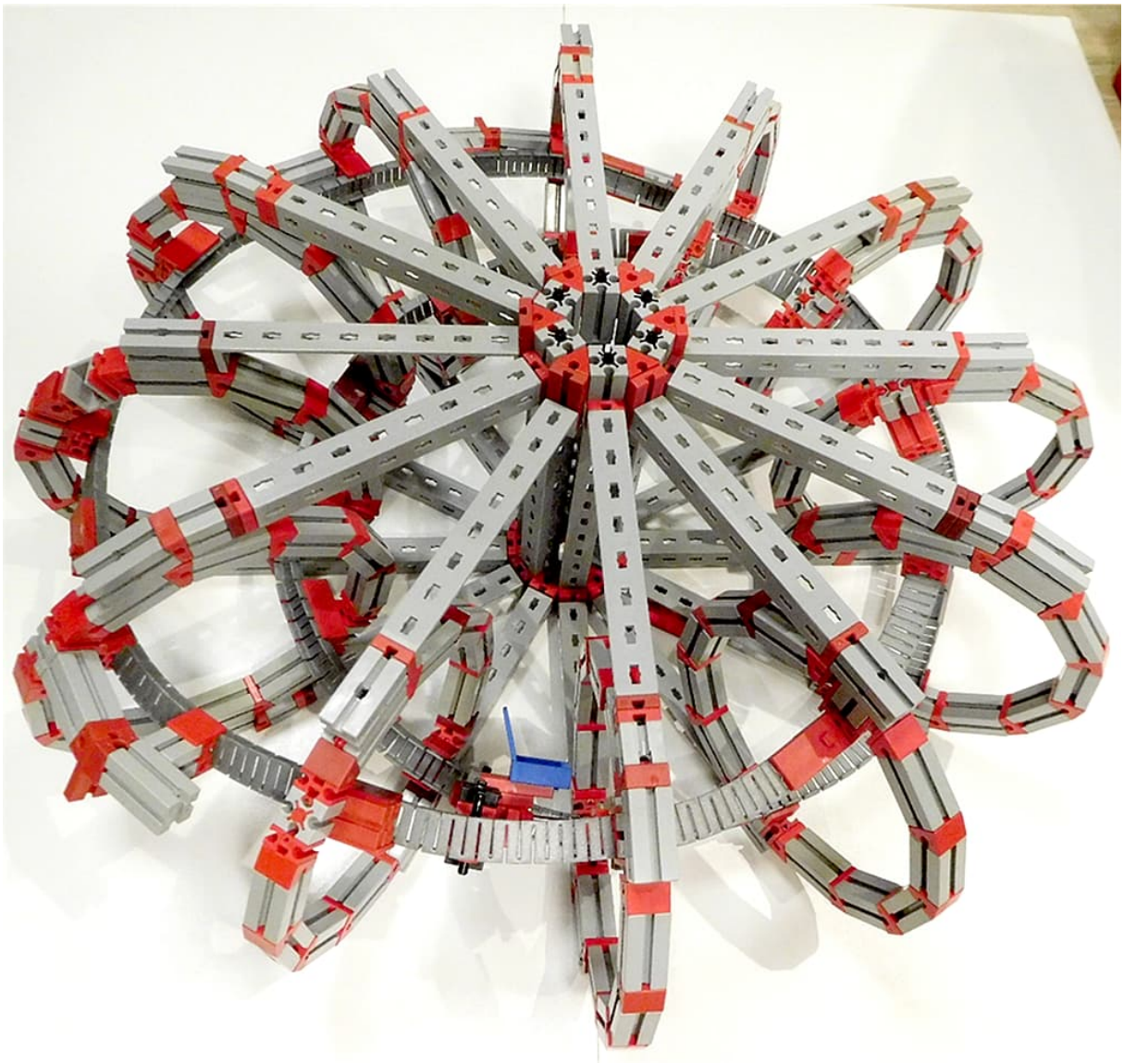


Abb. 2: Erste „kleine“ Möbius-Bahn mit Flex-Schienen

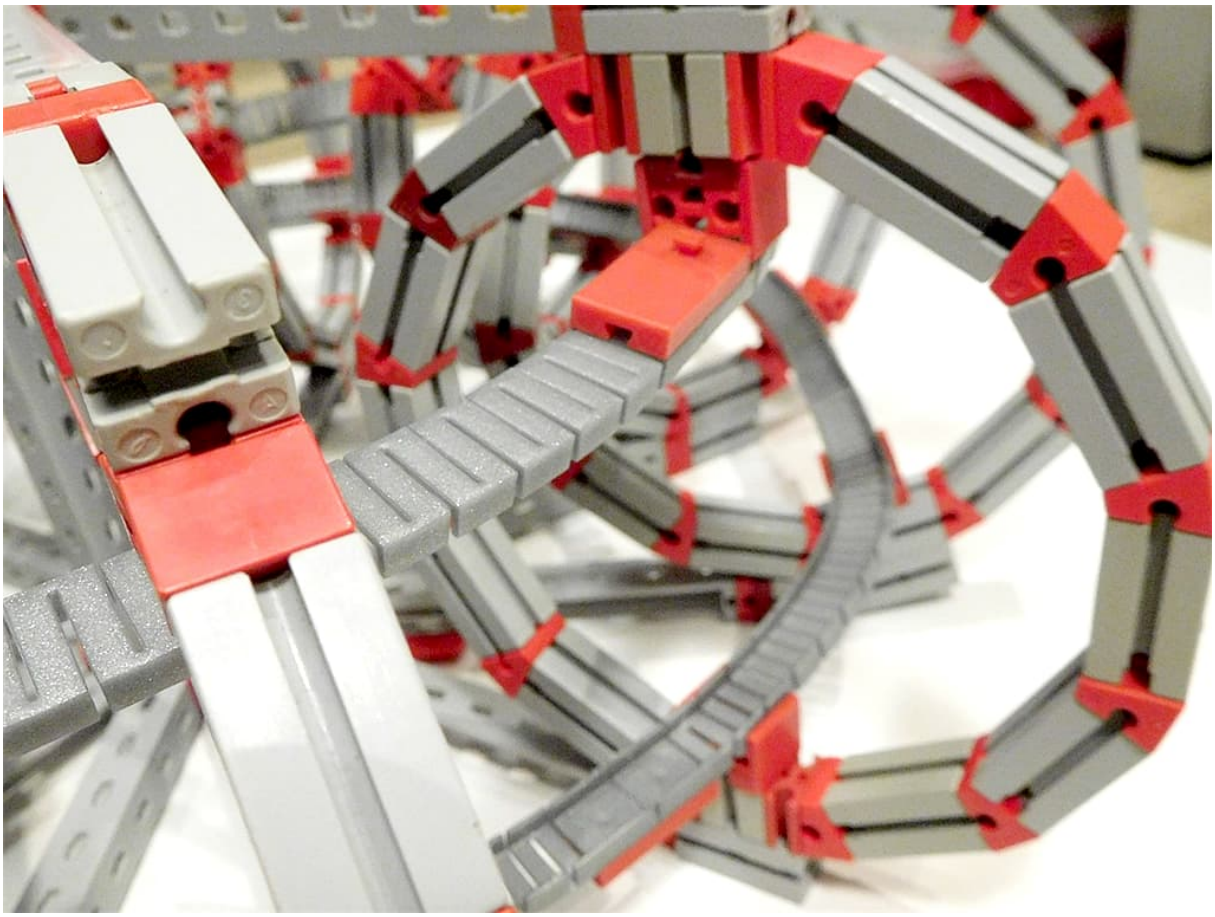


Abb. 3: Halteringe mit Halterungsblöcken für die Flex-Schienen

Die Halterungsstruktur ist bei allen vorgestellten Modellen ein Torus aus 12 kreisförmig angeordneten Ringen, an denen die Schienen befestigt sind (Abb. 2). Alle 30° befindet sich ein Halterungsring mit einem Innendurchmesser von 12,8 cm, der mit Statik-Streben an einer zentralen Nabe befestigt ist. Der Abstand von Nabe zur Mittellinie des Torus beträgt 24 cm.

Die Länge der Statik-Streben legt den Umfang des Torus und damit den Schienenbedarf fest. Das Verhältnis des kleinen zum großen Durchmesser (Haltering) bezeichnet man als *Aspektverhältnis*. Es beträgt für mein Modell 0,58.

Über der Konstruktion kann bei größeren Durchmessern zur besseren Einhaltung der Geometrie ein Ring aus Statik-Rundbögen befestigt werden. Die Halterungen für die Schiene sind an jedem Ring jeweils gegenüber positioniert und im Vergleich zu jedem vorigen Ring um 30 Grad verdreht angebracht.

Modell mit Flex-Schienen

Das erste Modell (Abb. 2) basiert auf fischertechnik-Flex-Schienen, die in dem Torus auf einer geschlossenen Strecke geführt werden. Zur Befestigung der Flex-Schienen an den Halteringen dienen Halterungsblöcke, in welche die Flex-Schienen eingedrückt werden und deren Orientierung an die aktuelle Krümmung dadurch angepasst werden kann, dass sie auf 15-er Steinen mit rundem Zapfen gesteckt sind (Abb. 3). Ggf. muss man die Flex-Schienen noch zusätzlich zur Klemmung durch doppelseitiges Klebeband an den Halterungen befestigen.

Da die Flex-Schienen sehr weich sind, wurde die Halterungs-Konstruktion durch

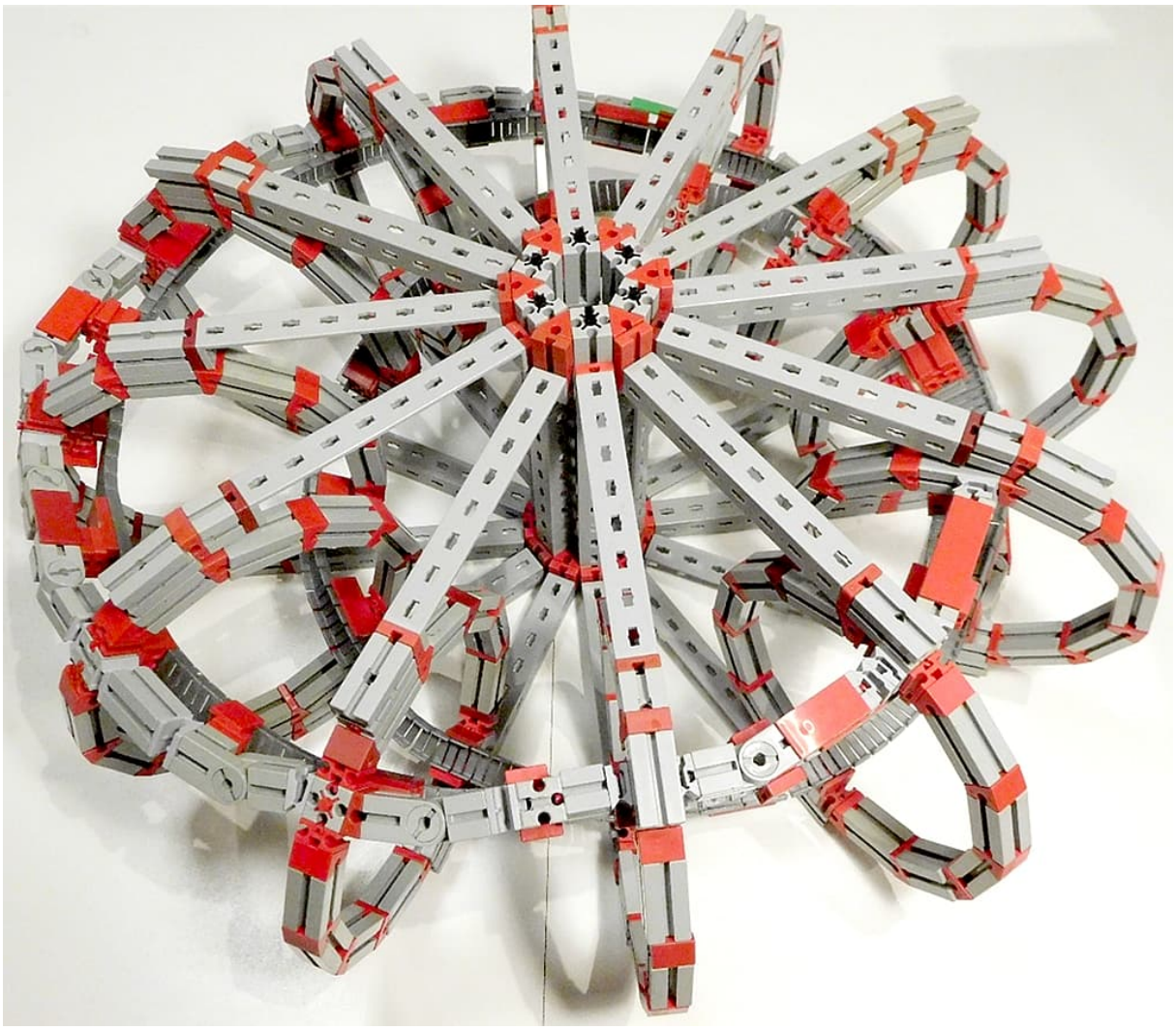


Abb. 4: Möbius-Bahn mit Verstärkungselementen für Flex-Schienen

entsprechende Zusatzhalterungen zwischen den Halteringen ergänzt (Abb. 4). Diese Halterungen sind nötig, da ein in diesen Schienen geführtes Vehikel die Schienen aufdrücken und es somit zur Entgleisung kommen würde.

Alternativ könnte man die Flex-Schienen, wie von Karl in [11] vorgeschlagen, auch mit einem Draht versteifen oder starrere kompatible Flex-Schienen aus dem 3D-Drucker verwenden (z. B. [11]).

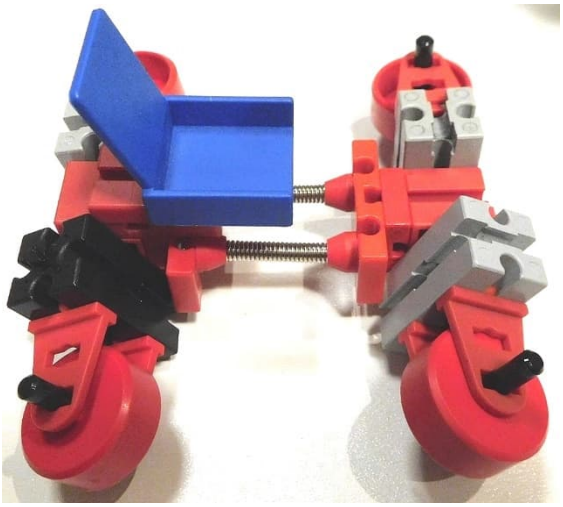


Abb. 5: Vehikel für Flex-Schienen-Möbius-Bahn

Das Vehikel (Abb. 5) besitzt Rollen, die in den Flex-Schienen laufen. Vorder- und Hinterachse sind durch eine Federkonstruktion miteinander verbunden, die eine Anpassung der Achsen an die jeweilig aktuelle Krümmung der Schiene erlaubt.

In Abb. 6 und 7 ist das Vehikel in der Bahn vor und nach einem Umlauf gezeigt. Nach einem Umlauf hat es sich um seine Längsachse um 180° gedreht. Im YouTube-Video [2] kann man sehen, wie der Wagen durch den Torus geschoben wird.

Ein Problem dieses Fahrwerks ist, dass bei dem Vehikel in waagrechter Position die Seitenflächen der Rollen an den Seitenflanken der Flex-Schienen gleiten bzw. schleifen. Darüber hinaus ist es schwierig, genügend Reibung zwischen den Rädern und der Lauffläche der Flex-Schienen zu erzeugen, damit im Falle einer Motorisierung das Vehikel ausreichend Vorschub hat.

Das ist der Grund, warum sich diese Modellvariante für eine Elektrifizierung weniger eignet. Es ist gut möglich, dass man ein Fahrwerk finden kann, wie man sie aus dem Achterbahn-Bereich kennt [9], bei welchem also jede Radgruppe Räder in drei Richtungen für Side-Friction, Upper-Friction und Under-Friction besitzt. Es würde mich interessieren, ob das jemand hinbekommt.

Modelle mit Rundschienen

Prinzipiell lassen sich verschiedene Schienenprofile (z. B. auch T-Profile) einsetzen. Die Herausforderung besteht darin, die Schienenprofile in drei Richtungen zu biegen: Links/Rechts, Rauf/Runter, Torsion.

Im Allgemeinen müssen die Profile senkrecht zur Längsachse geschlitzt werden. Bei

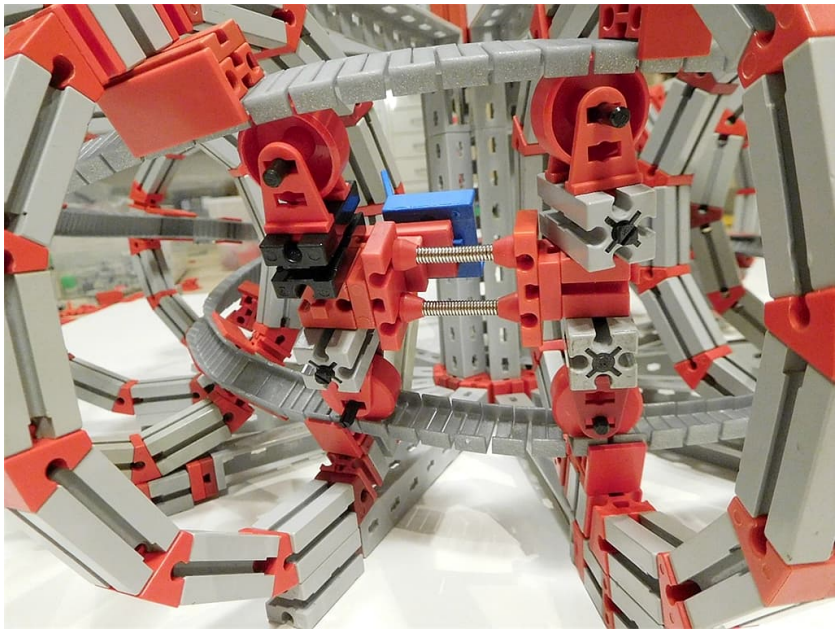


Abb. 6: Vehikel in der Flex-Schienen-Möbius-Bahn – Startposition

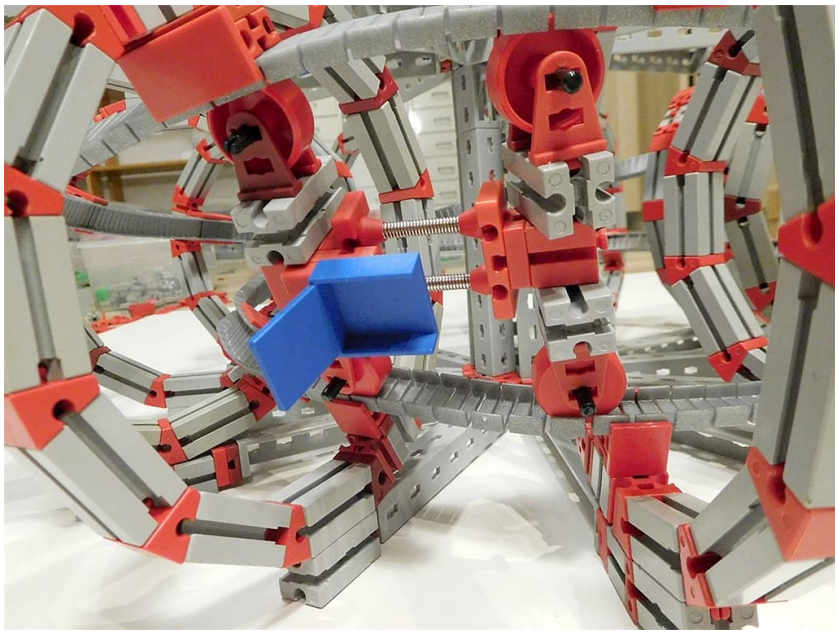


Abb. 7: Vehikel in der Flex-Schienen-Möbius-Bahn nach einem Umlauf

fischertechnik-Flex-Schienen sind Biegemöglichkeiten in allen Richtungen gegeben. Verwendet man ein Rundprofil, fällt die Notwendigkeit für die Torsion weg.

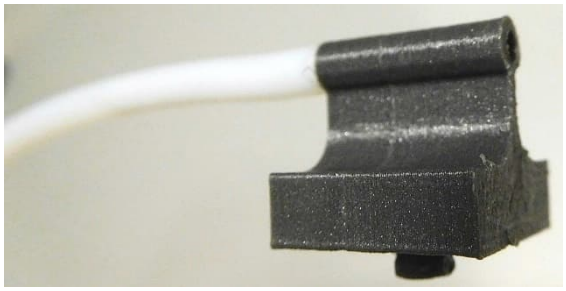


Abb. 8: 3D-gedruckte Halterung für Teflonschlauch

Als Rundprofil kann man Rundstäbe aus geeignetem Material (z. B. Messing, Aluminium oder Kunststoff) verwenden, die man dann entsprechend zurechtbiegen muss. Plastik-Rundmaterial kann unter Verwendung von Heißluft leicht verformt werden. Alternativ kann man einen Teflon-Schlauch verwenden. Dieser ist zum einen flexibel und hat zum anderen genügend Steifigkeit für einen Schienenstrang.

Ich habe mich für einen 4-mm-Teflon-Schlauch mit 2 mm Innendurchmesser entschieden, den ich als Ersatz-Filament-Führungsschlauch meines 3D-Druckers zur Hand hatte.

Für die Befestigung des Schlauches an den Halterungsringen der Torus-Konstruktion habe ich ein spezielles 3D-Teil entworfen [4], an das mit Hilfe eines Stückes Aluminium-Bindedrahts (2 mm Durchmesser) beidseitig Schlauchstücke angesteckt werden können (Abb. 8).

Da die Länge der Schienenstücke zwischen den Halterungsringen nicht konstant ist (Abb. 9), musste ich den Schlauch in verschieden lange Stücke zerschneiden.

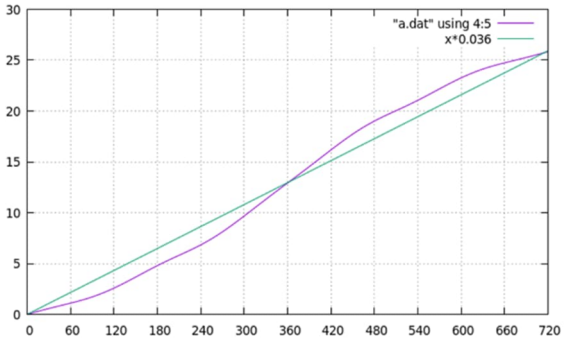


Abb. 9: Laufende Schienenlänge als Funktion des Umlaufwinkels

Die Anpassung der Längen habe ich manuell durch Ausprobieren vorgenommen, wodurch sich eine gewisse Ungenauigkeit in der Spurbreite ergab, die aber durch axiales Spiel der Laufrollen kompensiert werden kann. Versuche mit Halterungen für einen unzerteilten Schlauch habe ich nicht weiterverfolgt. Eine Bahn mit Schlauchschienen ist in Abb. 10 zu sehen.

Abb. 10: Möbius-Bahn mit Schlauchschiene – Der Schlauch ist mit gegenüberliegenden Schlauchhaltern an den Halteringen befestigt. Von Ring zu Ring sind die Halterungen um 15° verdreht angebracht, sodass der Schlauch sich nach zwei Umrundungen wieder schließt.

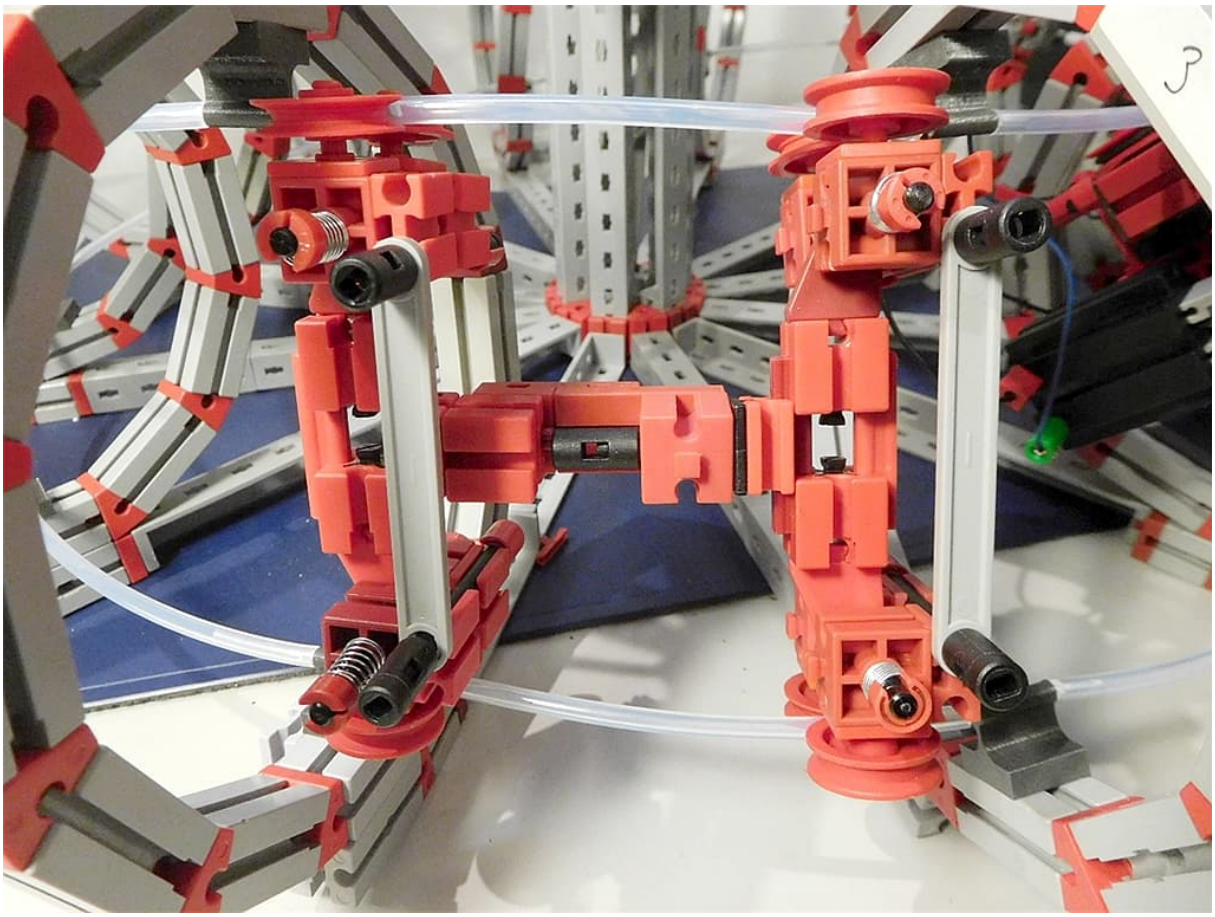


Abb. 11: Modell 0 - Basisvehikel für eine Möbius-Bahn mit Rundschienen mit Achsschenkellenkung und Axiallager zwischen Heck und Bug

Modell 0

Eine einfache unmotorisierte Basisversion eines Vehikels für die Teflon-Schiene ist in Abb. 11 gezeigt.

Dieses Vehikel besitzt vier Rad-Paare. Jedes Paar besteht aus zwei Nutrollen, zwischen denen der Schlauch läuft bzw. geklemmt wird. Vorder- und Hinterachse haben eine Achsschenkellenkung (Ackermann-Geometrie), welche die Krümmung der Bahn ausgleicht. Vorder- und Hinterachse sind mittels eines axialen Gelenkes miteinander verbunden. Mit einer Achsschenkellenkung sind auch Vehikel mit größerem Radstand möglich.

Motorisierte Vehikel

Für einen elektrischen Antrieb treibt man die Räder einer Achse über ein Differential mit einem Motor an. Das Differential ist nötig, da bei dem von mir gewählten kleinen Aufbau starke Krümmungen vorliegen, sodass Außen- und Innenradpaare unterschiedlich schnell laufen müssen.

Entscheidend wird das Differential bei starrer Antriebsachse ohne Achsschenkellenkung, da das Vehikel sonst unweigerlich entgleist.

Um die für den Antrieb notwendige Klemmwirkung und Friktion zu erzeugen, besitzt jedes Rad-Paar aufgezogene Gummiringe (O-Ringe) und Gegenzahnräder (Z15) für eine gegensinnige Drehrichtung der Nutrollen. Nutrollen und Zahnräder können mit Papierstreifen oder Schrumpf-

schlauchstücken auf die Achsen geklemmt werden.

Basierend auf diesem Prinzip ist eine Reihe von Vehikeln für Rundschienen aus 4-mm-Teflon-Schlauch entstanden:

Modell 1

Dies war das erste motorisierte Vehikel-Modell. Es hat Rad-Dubletten mit Gegenzahnrädern in der Antriebsachse und wird durch einen XS-Motor über ein Differential angetrieben. Die Vorderachse ist mit einer Achsschenkellenkung versehen. Vorder- und Hinterachse sind über eine Feder verbunden, welche die notwendige Torsion zwischen Vorder- und Hinterachse ermöglicht. Das Modell ist in Abb. 12, 13 und 14 gezeigt. In Abb. 15 sieht man das Vehikel in der Bahn.

Abb. 12: Modell 1 – Vehikel mit XS-Motor; Ansicht von unten

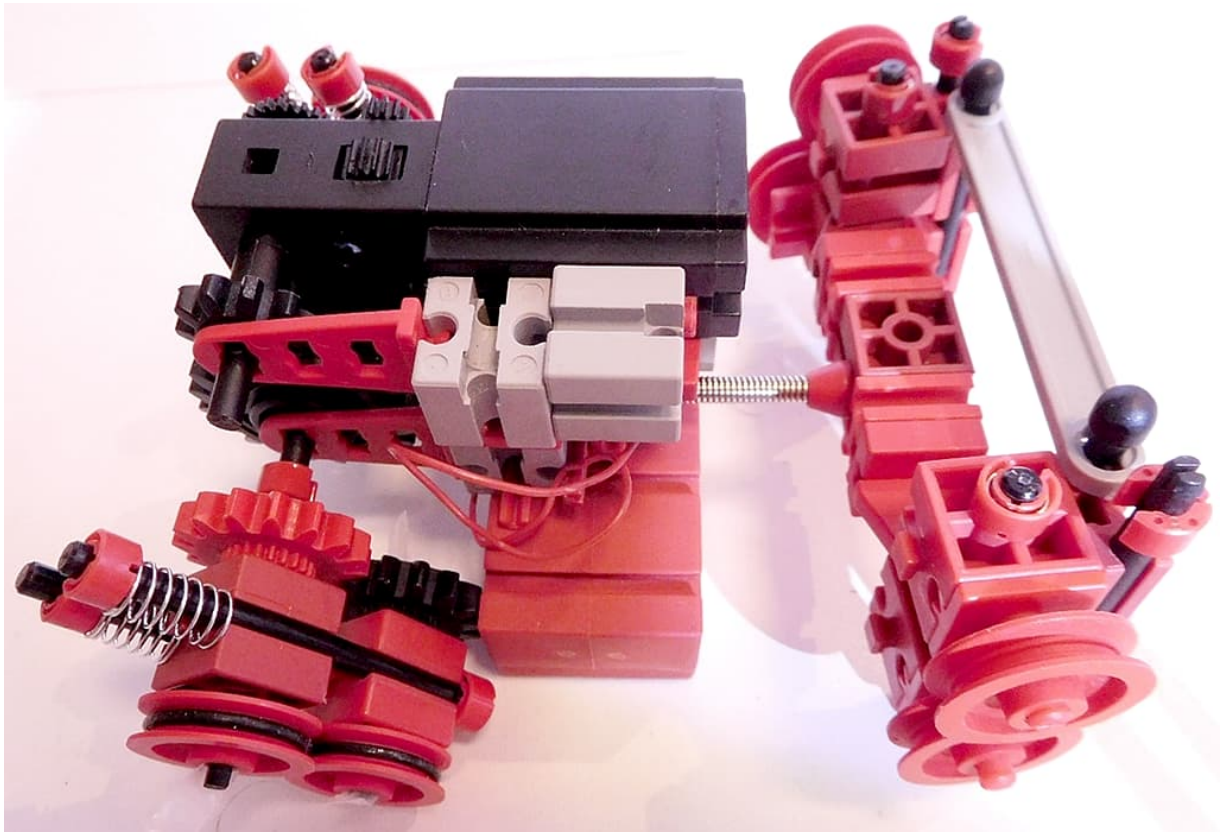


Abb. 13: Modell 1 – Vehikel mit XS-Motor; Aufsicht

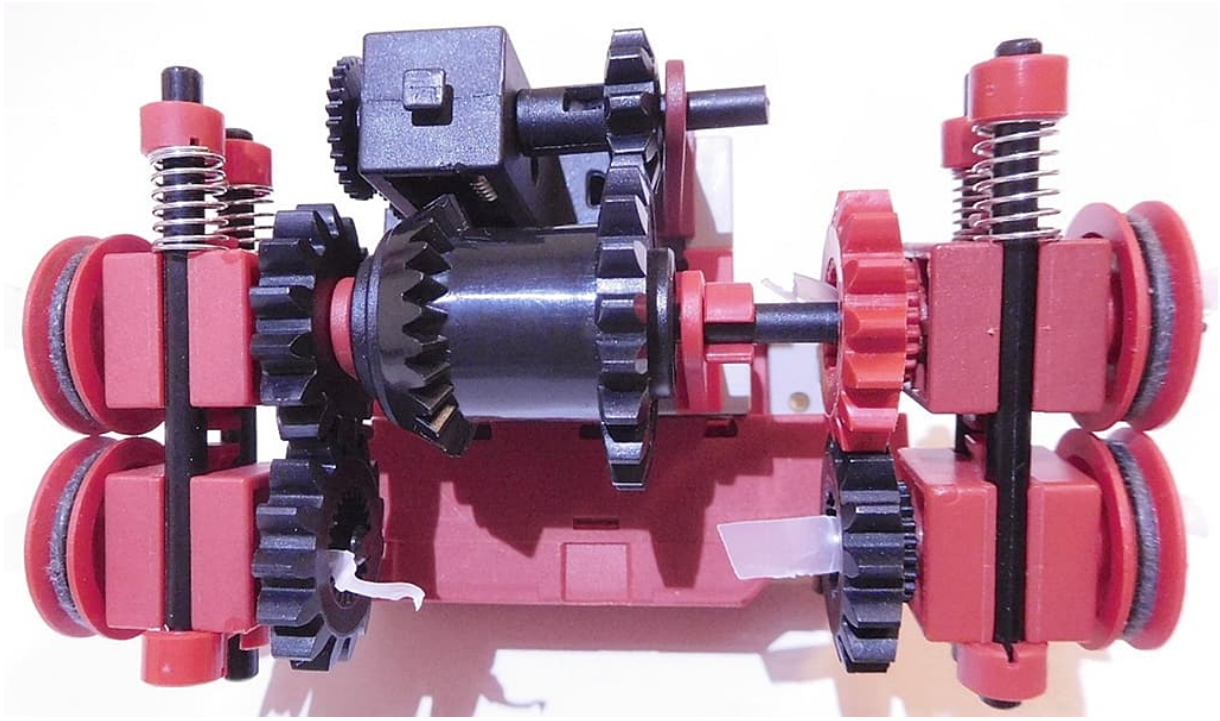


Abb. 14: Modell 1 – Vehikel mit XS-Motor, Getriebeblock, Differential und Klemmantrieb; Heckansicht

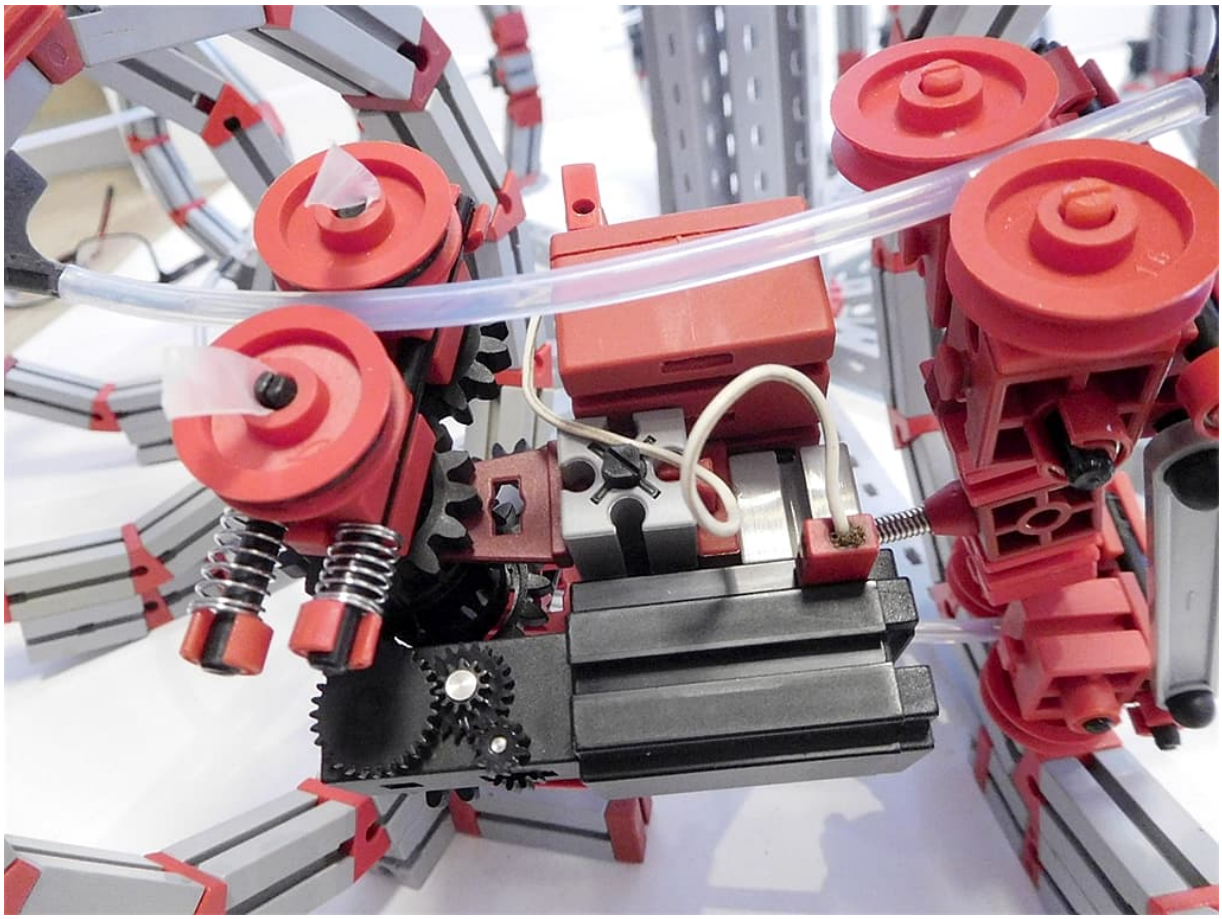


Abb. 15: Modell 1 in der Möbius-Bahn mit Rundschiene. Die Klemmung der Nutrollen durch die Schraubenfedern hält das Vehikel auf der Bahn und erzeugt die notwendige Reibung für den Vorschub

Modell 2

Das zweite Modell unterscheidet sich in zwei Punkten von Modell 1: Für seinen Antrieb wird ein XM-Motor verwendet und Vorder- und Hinterachse sind über ein Gelenk mit drei Freiheitsgraden miteinander verbunden. Das kann entweder über eine fischertechnik-Feder, eine Kombination aus einem Kardangelenk und einem axialen Lager oder über ein 3D gedrucktes Kugelgelenk realisiert werden. Abb. 16 zeigt das XM-Vehikel in der Bahn von oben und Abb. 17 nach einer Umrundung von unten.

Dieses Modell kann mit unterschiedlichen Übersetzungen ausgestattet werden und schafft mit einem Z30 am Motorausgang, das das Z10 für das Differential (ohne Kette) treibt, eine Runde in ca. 6 Sekunden. Das Modell ist zudem auch sehr zugkräftig, sodass es als Lok für einen weiteren „Möbius-Waggon“ dienen kann. Ein Zug ist in Abb. 19 zu sehen, die Modell 3 zeigt.

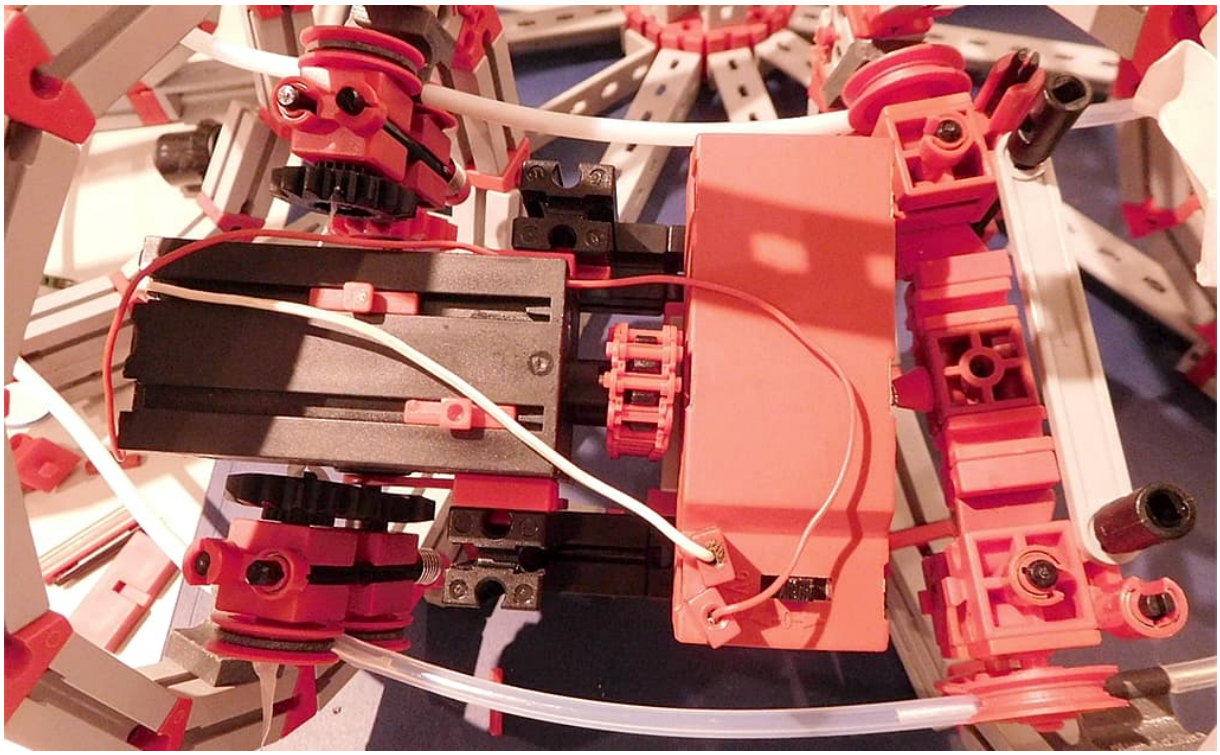


Abb. 16: Modell 2 – Vehikel mit XM-Motor in der Möbius-Bahn

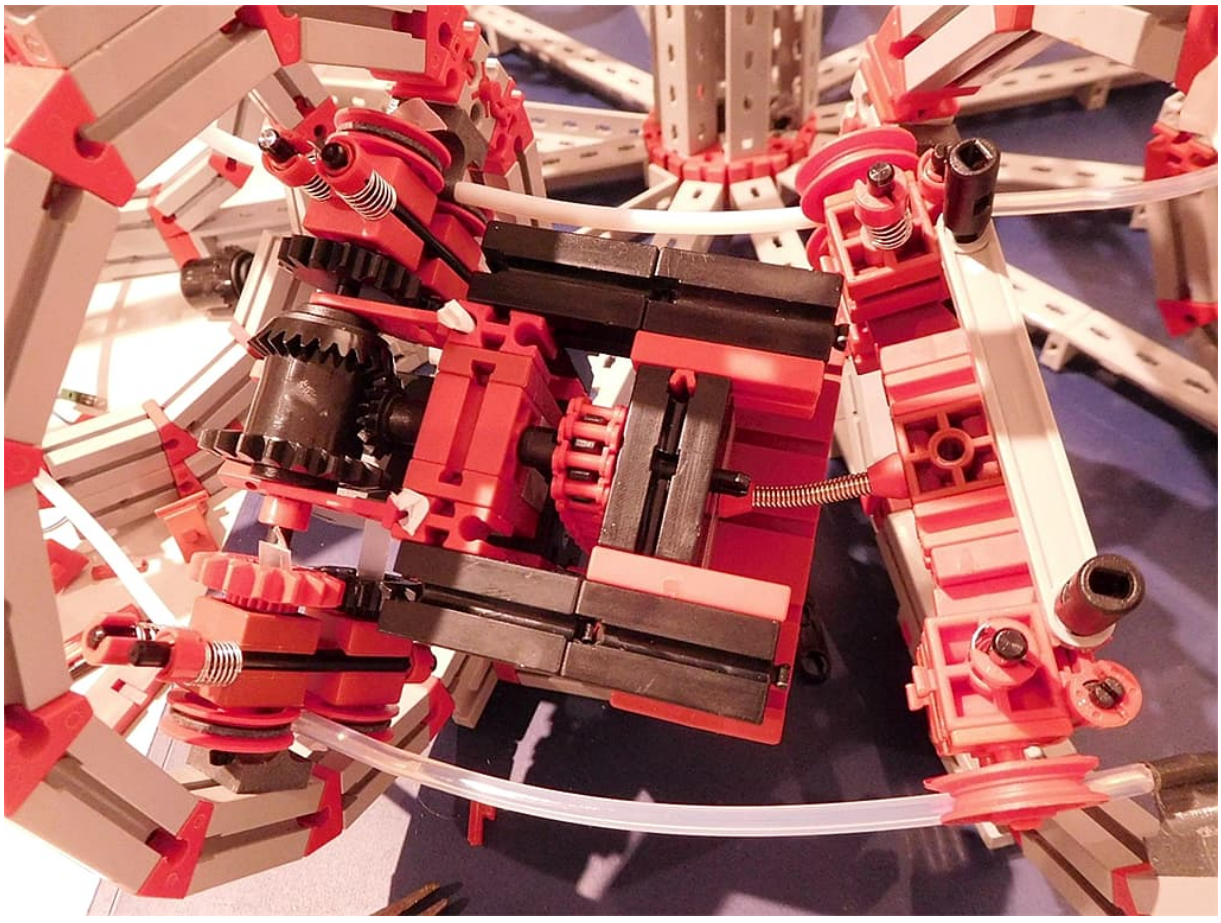


Abb. 17: Modell 2 – Vehikel mit XM-Motor; Unteransicht

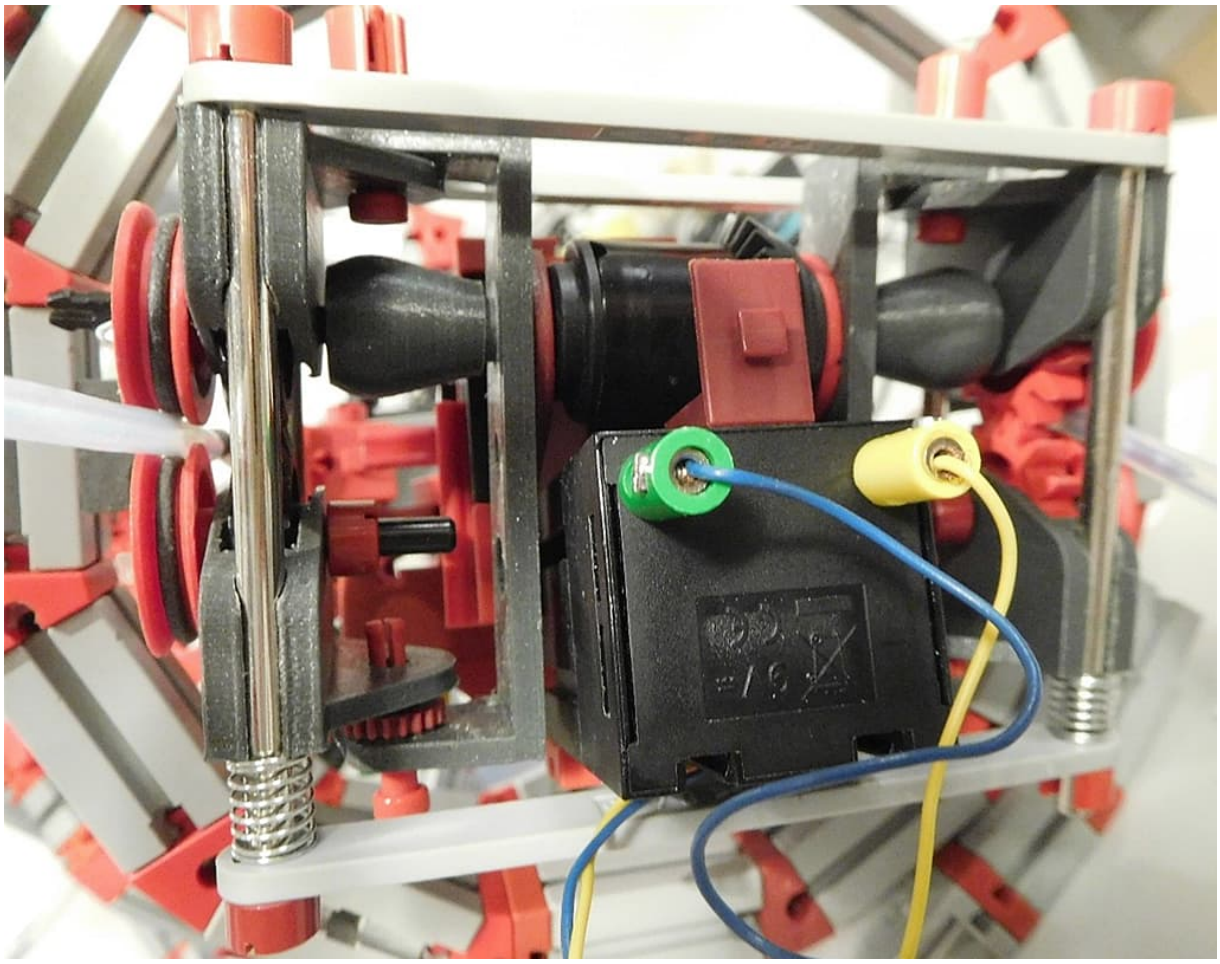


Abb. 18: Modell 3 – Vehikel mit XM-Motor und Achsschenkellenkung an der Antriebsachse

Modell 3

In Abb. 20 sieht man eine spezielle Weiterentwicklung von Modell 2 mit einer Achsschenkellenkung in der Antriebsachse. Wegen der limitierten Spurbreite meiner Bahn von ca. 100 mm habe ich auf den beschränkten Platzbedarf optimierte Spezialteile konstruiert (FreeCAD) und mit dem 3D-Drucker gedruckt. Die Antriebs-Sektion am Heck ist in Abb. 18 dargestellt.

Das entscheidende Bauteil ist ein abgewandeltes Um-die-Ecke-Kugelgelenk von Jan (juh) [9], welches durch einen Halterungsbügel für die „Radlager“ geführt ist. Jedes Radlager besitzt Führungsbohrungen für Metall-Klemmachsen und eine Tasche für ein Z15, dessen Achse beidseitig durch dünne Wände geführt wird, um einerseits den Platzbedarf für die Halterung und andererseits die Neigung zur Verkippung der Nutrollenachse zu verringern. Für den Anschluss von Differential und Kugelgelenken habe ich aus 4-mm-Rundmaterial kürzere Rastachsen erst aus Holz und später aus Messing gefeilt.

Modell 4

Ein Mini-Motor reicht wegen der hohen Klemmkraft und der damit verbundenen Reibung nicht aus, um das Modell anzutreiben - zwei Motoren schaffen es. In dem in Abb. 21 gezeigten Vehikel kommen zwei Mini-Motoren in diagonaler Anordnung am Heck und am Bug zum Einsatz. Sowohl Vorder- als auch Hinterachse haben eine Achsschenkellenkung und Bug und Heck sind mit einem Axiallager miteinander verbunden. Wegen der Achsschenkellenkung an beiden Achsen kann auf ein Differential verzichtet werden.

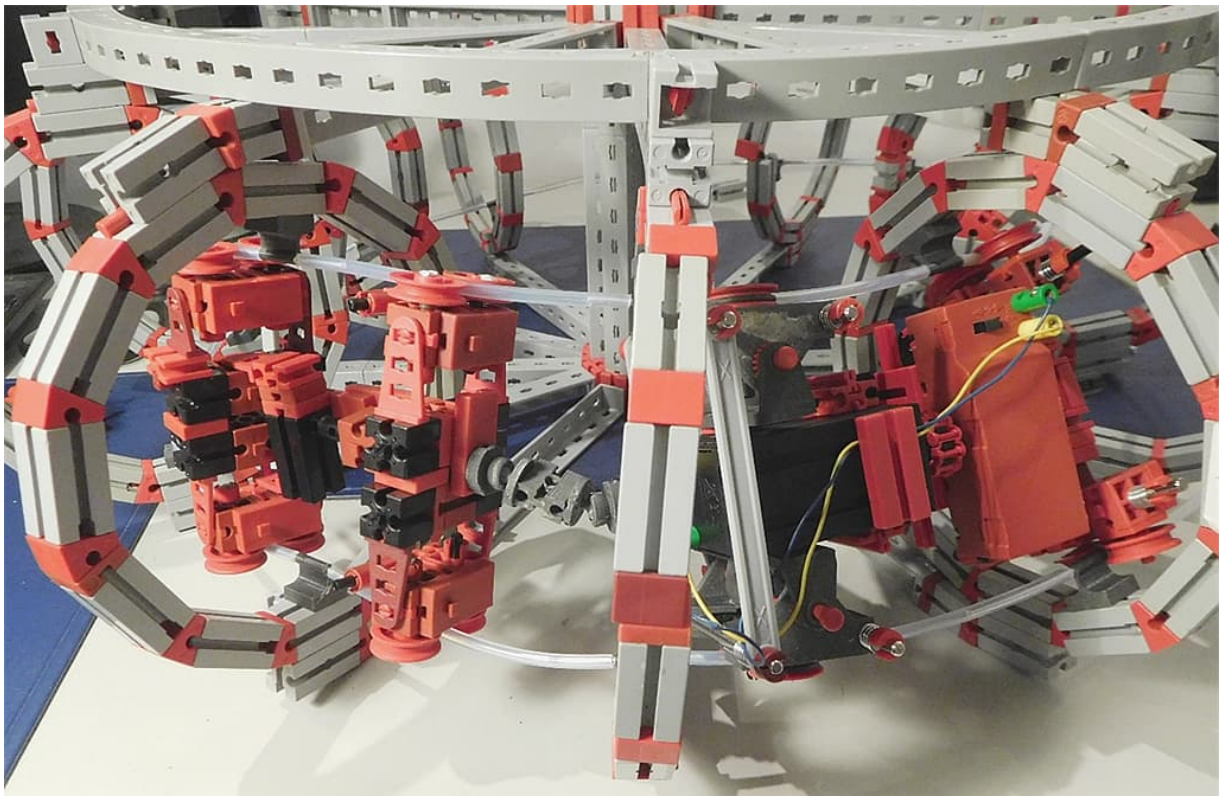


Abb. 19: Modell 3 als Lok für einen Möbiuswagen

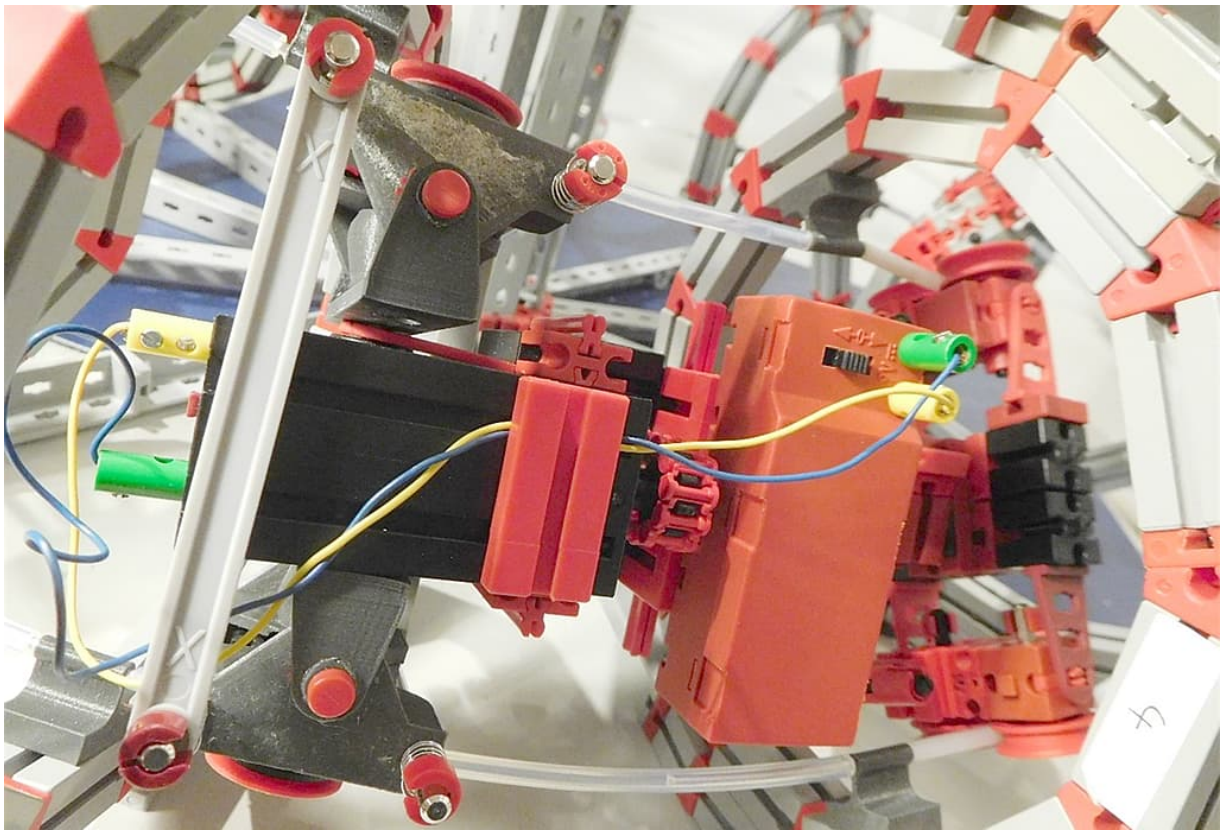


Abb. 20: Modell 3; Ansicht von oben

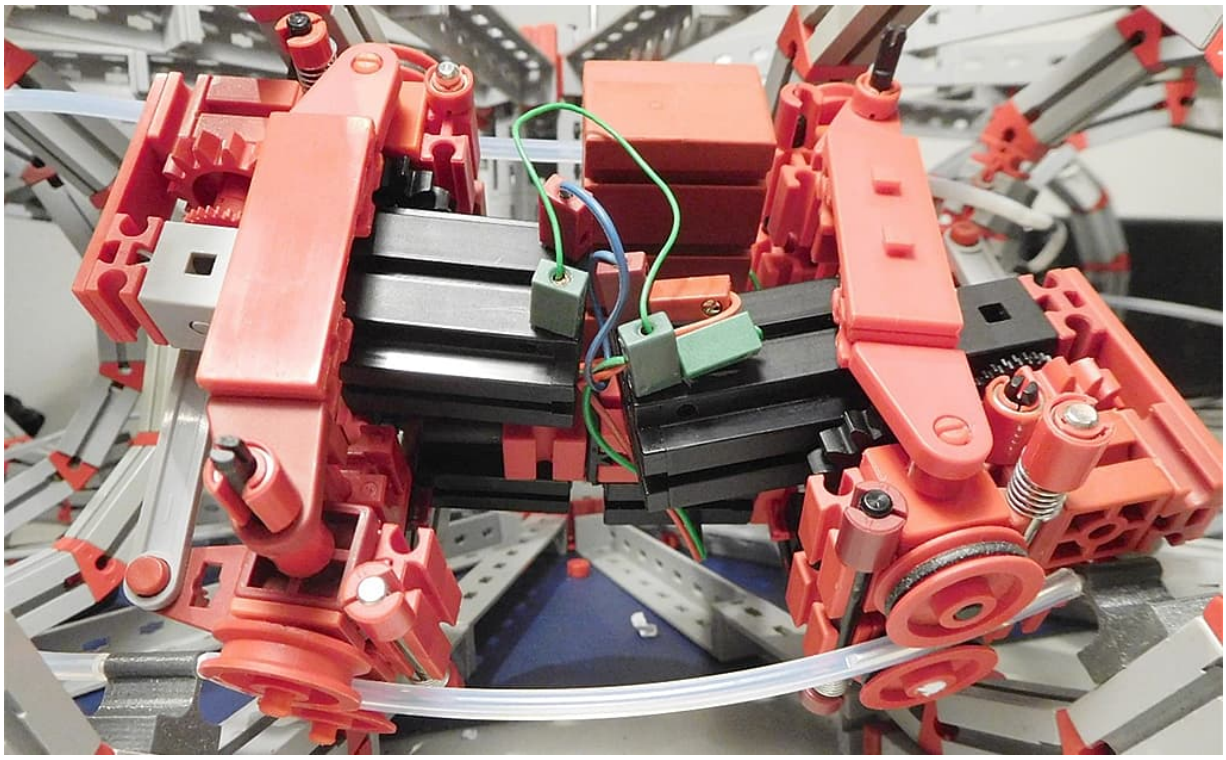


Abb. 21: Modell 4 mit voller Achsschenkellenkung und diagonaler Motoranordnung

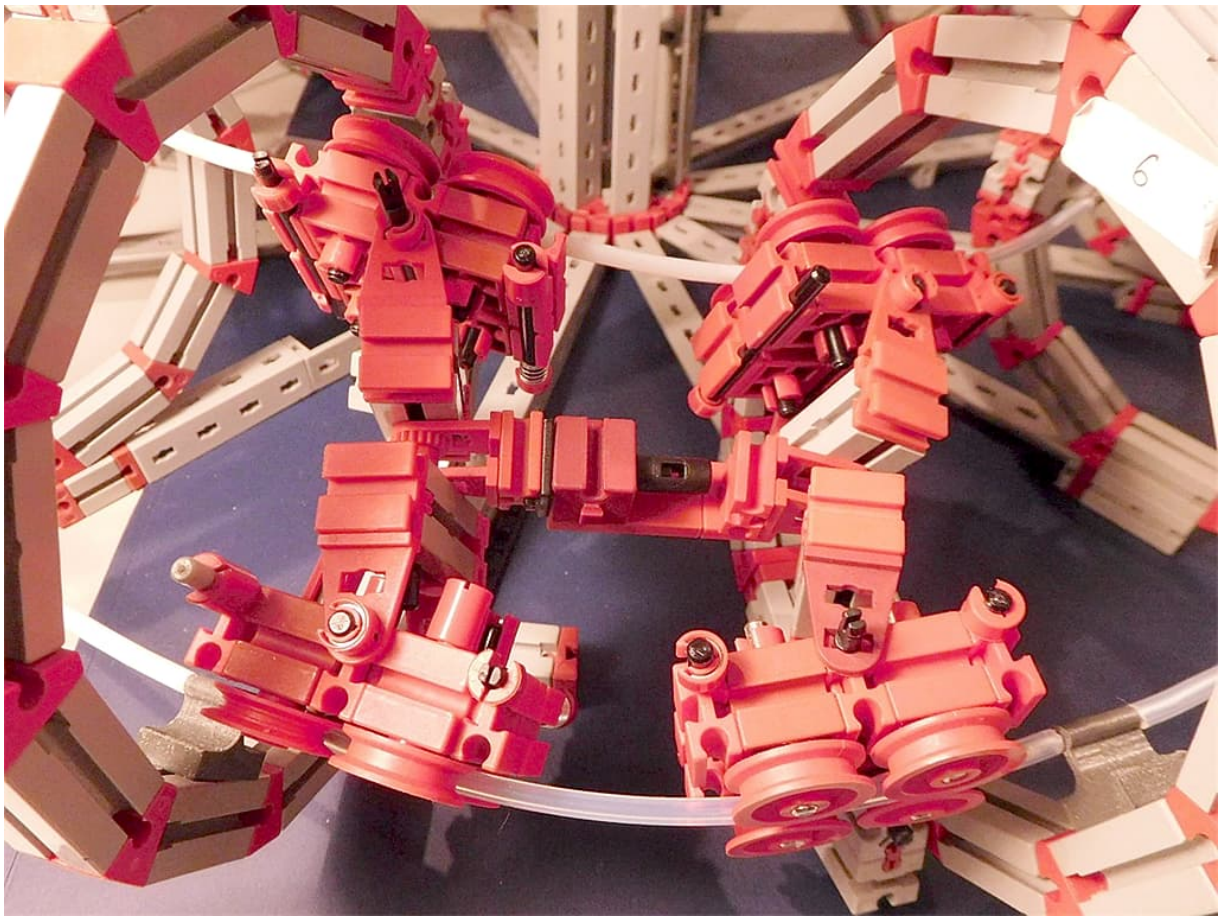


Abb. 22: Modell 5 mit Rad-Quadrupeln und Achsschenkellenkung

Modell 5

Die Idee hinter dem in Abb. 22 gezeigten Modell ist ein kinematisch „sauberes“ Fahrgestell in Anlehnung an die Radgruppen beim Achterbahn-Chassis. Es ist ein Vehikel mit in zwei Richtungen unabhängig drehbaren Radlager-Blöcken vorne und hinten. Die Rad-Paare sind jetzt Quadrupel, die an jeder Position eine „Tangentiale“ (eher Sekante) entlang eines Schienenstückes definieren. Man sieht schön, welche Freiheitsgrade auftreten können. Das Vehikel hat noch eine Längsachse zwischen Vorder- und Hinterteil. Mit den unabhängig voneinander beweglichen Radlager-Blöcken können Spurbreite-Schwankungen etwas ausgeglichen werden.

Modell 6

Die in Abb. 23 gezeigte motorisierte Variante greift die diagonale Motor-Anordnung von Modell 4 auf. Besonders knifflig waren die Radgruppen-Aufhängungen, die man dicht an das Mittelteil klemmen muss. Man kann keinen 15er-Lochstein dafür nehmen, da sonst die Achsklemmung den Rad-Paketen in die Quere kommt. Dann waren noch Lochplatten (grau) nötig, die das Verschieben der 15er-Lochsteine und der 7,5er-Steine verhindern, da die Klemmkraft der Federn, welche die Räder auf die Schienen klemmen, recht hoch ist. Die Lochplatten sind 3D-gedruckt. Man könnte sie aber auch leicht aus einem dünnen Kunststoff-Plättchen oder gar Pappkarton herstellen.

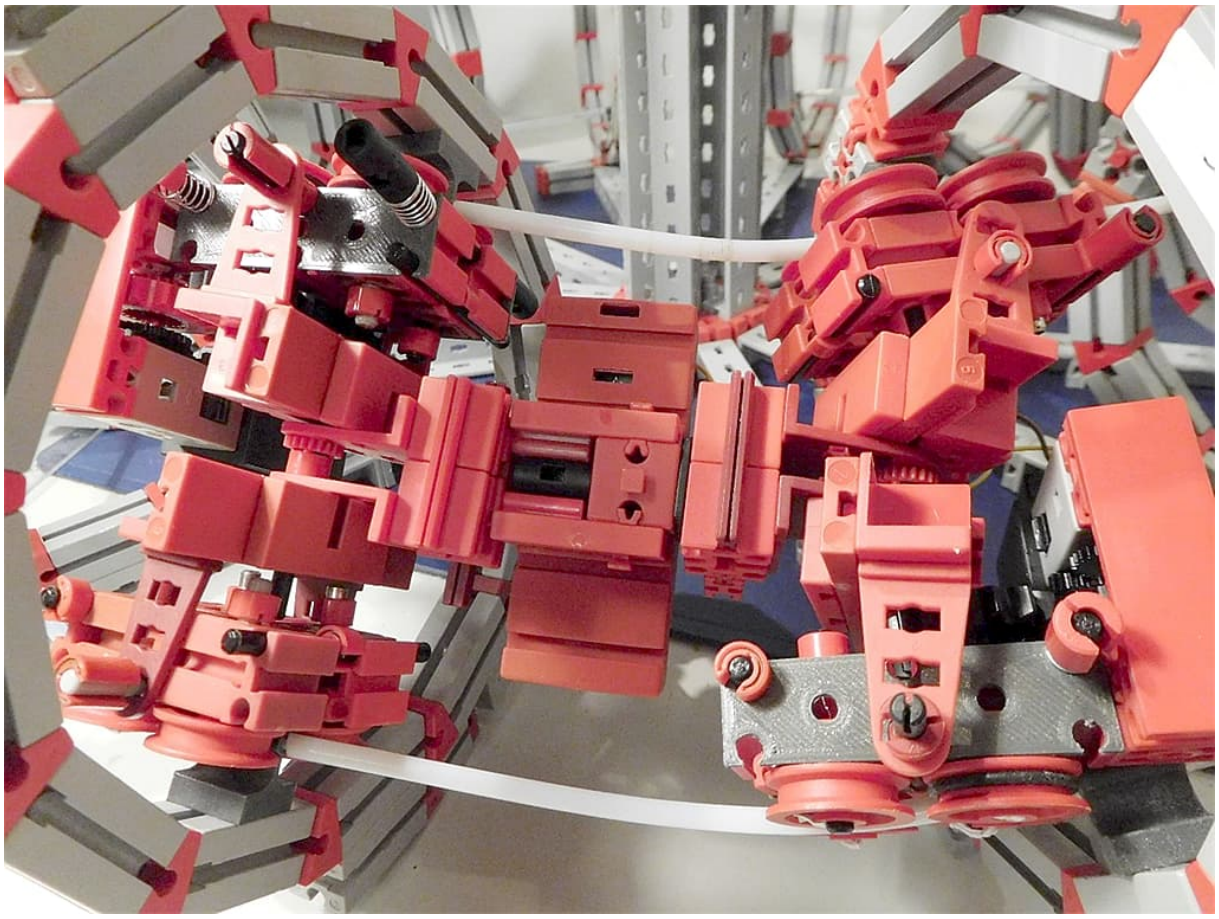


Abb. 23: Modell 6 – Motorisiertes Modell mit Rad-Quadrupeln und diagonaler Motoranordnung

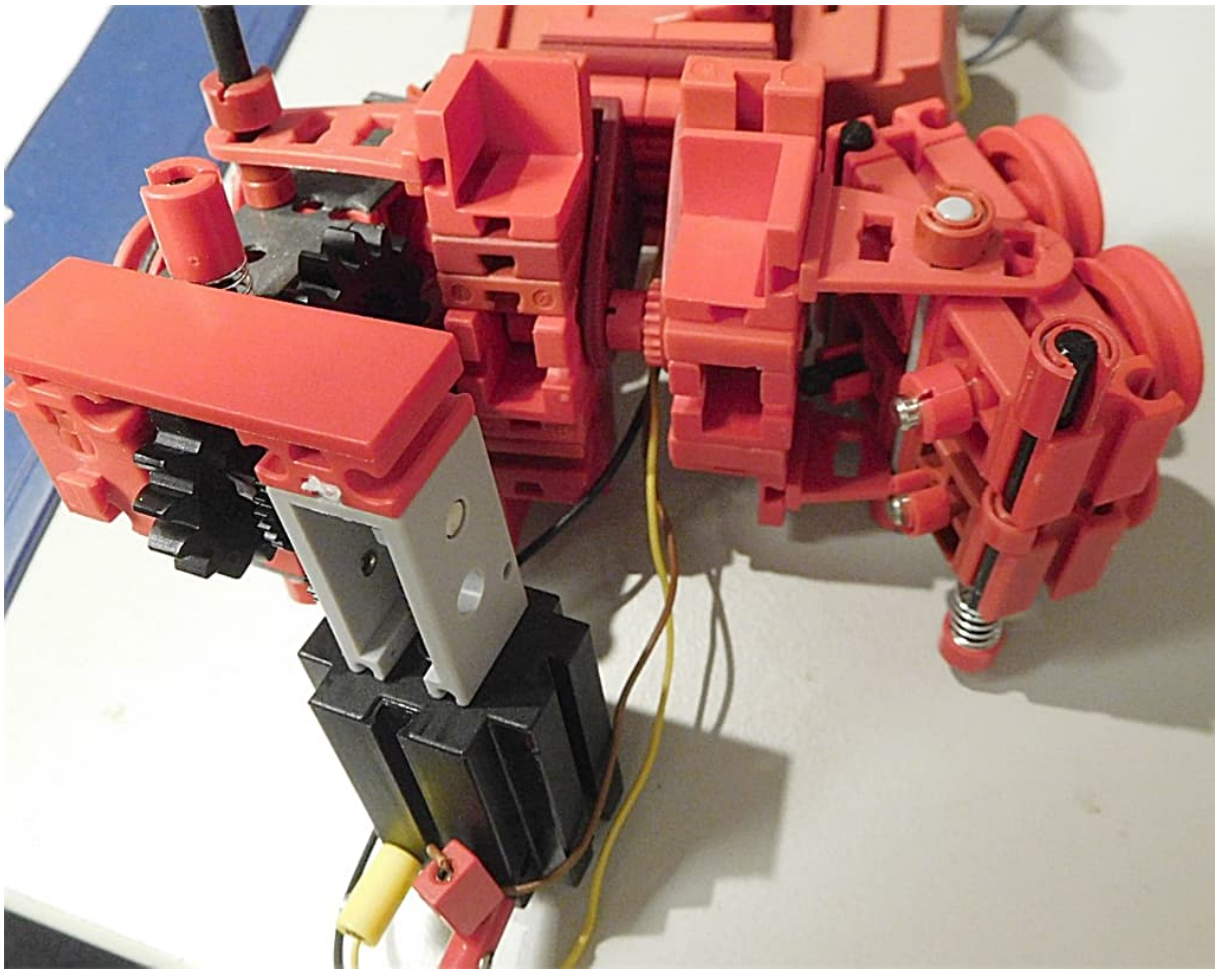


Abb. 24: Heckansicht von Modell 6 mit „Außenbordmotor“

Zusammenfassung und Fazit

Der Beitrag fasst die Entwicklungsschritte eines längeren Projektes zusammen, das zum Ziel hatte, eine Möbius-Bahn mit fischertechnik nach dem „Imaginata-Stil“ zu bauen und zu motorisieren. Die Entwicklung dieses Projektes kann im fischertechnik-Community-Forum nachvollzogen und diskutiert werden [10].

Begonnen habe ich mit einem unmotorisierten Modell mit fischertechnik-Flex-Schienen. Darauf folgte die Umstellung auf eine Bahn mit 4-mm-Teflon-Schlauch, der in Teilstücken mittels 3D-gedruckter Halterungen am fischertechnik-Torus-Gerüst fixiert ist. Für diese Bahn wurde eine Reihe von motorisierten Vehikeln vorgestellt, die zum Teil mit Sonderteilen aus dem 3D-Drucker ausgestattet waren.

Ich möchte mich hier bei der ft-Community bedanken, die dieses Projekt im Forum begleitet und wesentliche Ratschläge und Anregungen gegeben hat.

Ausblick

Auch wenn ich es nicht hinbekommen habe, sollte die Motorisierung eines Vehikels für eine Möbius-Bahn aus Flex-Schienen möglich sein. Andere Schienenprofile und insbesondere Bahnen mit größerer Spurbreite wären reizvoll, da man dann großzügiger bauen und vermutlich auf 3D-Teile verzichten kann. Man könnte eine Bahn mit metallischen Schienen bauen, die das Vehikel wie bei einer Modelleisenbahn mit Strom versorgen. Wenn man die Schienenschleife zweimal faltet, könnte man eine Bahn bauen, bei der das Vehikel zwischen drei Schienenabschnitten läuft und erst nach drei

Umläufen wieder in die ursprüngliche Position kommt.

Darüber hinaus denke ich, dass man sich noch viele Anregungen von den Chassis der Achterbahn-Technik holen kann. Als „ganz großes Ding“ schwebt mir eine Vertikalbahn mit Weichen vor, bei der das Vehikel in allen drei Dimensionen fahren kann.

Quellen

[1] Wikipedia: *Möbiusband*.

[2] Florian Bauer: *Moebius-Bahn mit fischertechnik (WB)*. Auf YouTube, 2023.

[3] Florian Bauer: *fischertechnik Moebius Track – Fast Shuttle*. Auf YouTube, 2023.

[4] Florian Bauer: *fischertechnik Schlauchhalter 4mm / 2mm*. Auf Thingiverse, 2023.

[5] Ithaca College Physics: *Superconducting Quantum Levitation on a 3π Möbius Strip*. Auf YouTube, 2016.

[6] Imaginata e. V.: *Möbiusbahn*. Auf imaginata.de.

[7] Padefeo: *Möbius-Schleife – Imaginata Jena*. Auf YouTube, 2019.

[8] Wikipedia: *Achterbahnfahrwerk*.

[9] Jan Hanson: *Print in place angular joint for fischertechnik.* Auf Thingiverse, 2023.

[10] Diskussion zur Möbiusbahn im fischertechnik-Community-Forum, 2023.

[11] Peter Habermehl: *fischertechnik compatible flex rail (Flexschiene) 155mm*. Auf Thingiverse, 2019.

Getriebe

Almond-Kupplung

Ralf Geerken

Um die Ecke (an)getrieben mit einer zahnradlosen bzw. getriebefreien Kupplung: Vor ein paar Wochen war ich mal wieder etwas antriebslos bei irgendwelchen „Shorts“ bei YouTube unterwegs und bin dabei zufällig über ein Eckgetriebe gestolpert. Es war gar nicht so einfach herauszufinden, um was für eine Sache es sich da handelte.

Der englische Ausdruck *Almond Coupling* bezeichnet eine Mechanik, die im Jahr 1884 von einem *Thomas R. Almond* (1855-1906) erfunden wurde. Das US-Patent hat die Nummer 304,156 und ist im Internet zu finden ([1], Abb. 1).

Auf allen Internet-Seiten, die ich fand, wurde diese Kupplung als rechtwinklig bezeichnet. Meistens wurde sie wohl in sogenannten *Transmissionen* verwendet. Das waren Bandantriebe in Werkstätten bzw. Maschinenhallen aus der Zeit der Industrialisierung. Dass es auch in allen – und damit meine ich wirklich in allen – Winkeln funktioniert, werde ich in meinem Nachbau zeigen. Auch die fischertechnik-Variante ist eine leise, homokinetische Verbindung und ist wahrscheinlich, wie das Original, nur für relativ niedrige Drehzahlen geeignet (wobei ich denke, dass ein normaler Auto-Antrieb trotzdem damit realisiert werden kann).

Mit meiner Variante können Wellen bzw. Achsen miteinander verbunden werden, wobei es eine Antriebswelle und sogar gleich mehrere Abtriebswellen geben kann. Die Abtriebswelle muss noch nicht einmal in einer Ebene mit der Antriebswelle liegen und kann obendrein noch einen geringfügig anderen Winkel in der Z-Ebene haben. In einem Fall habe einen Winkelstein 7,5° eingesetzt und es funktionierte alles immer noch gut.

Dass es mit fischertechnik unbegrenzte Möglichkeiten gibt, ist ja allgemein bekannt. Aber für einen einigermaßen originalgetreuen Nachbau der Almond-Kupplung musste ich doch ein wenig tüfteln.

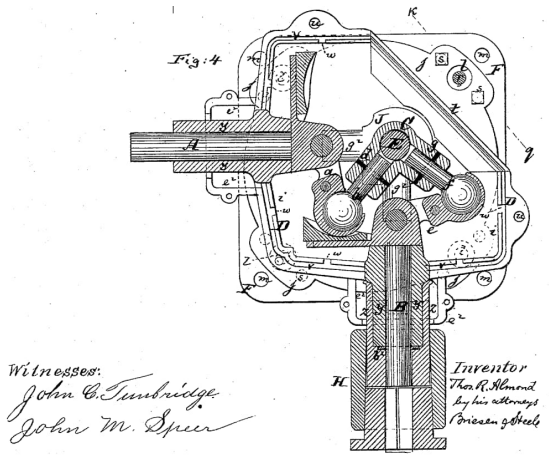


Abb. 1: US-Patent 304,156 von 1884 [1]

Für einen gleichmäßigen Lauf ist es sehr wichtig, dass die Teile nicht verkanten und alle Komponenten sehr leichtgängig sind. Des Weiteren sollten alle Teile recht fest miteinander verbunden sein. Lockere Nut- und Feder-Verbindungen verrutschen gerne mal und dann verändern sich Abstände. Das wiederum beeinträchtigt den Gleichlauf.

Beide Arme, zuführender sowie abgreifender, müssen gleich weit von der zentralen auf und ab pendelnden Führungsachse entfernt sein. Ist das nicht der Fall, dann hakt die Abtriebseite gerne und läuft nicht mehr gleichmäßig. Variante a) mit dem Baustein

Schneckenmutter (35973) war meine erste funktionsfähige Idee. Diese lief bei mir am gleichmäßigsten, bis es Variante g) gab, braucht aber einiges an Platz (Abb. 2 bis 6).

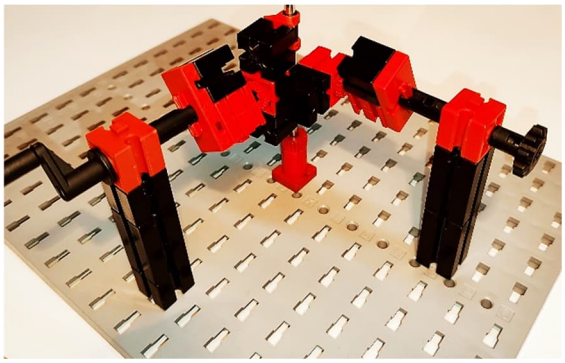


Abb. 2: Variante a)

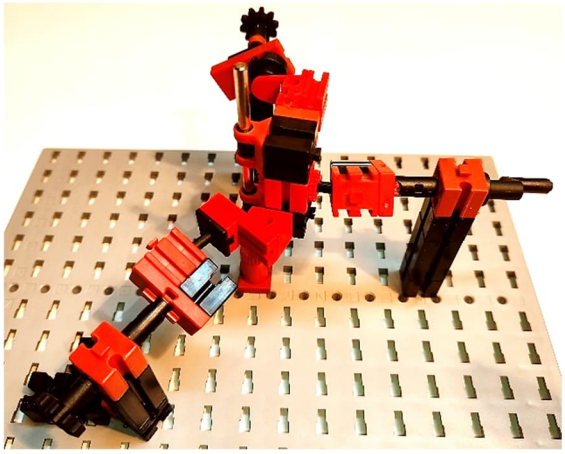


Abb. 3: Variante a) aus anderem Blickwinkel

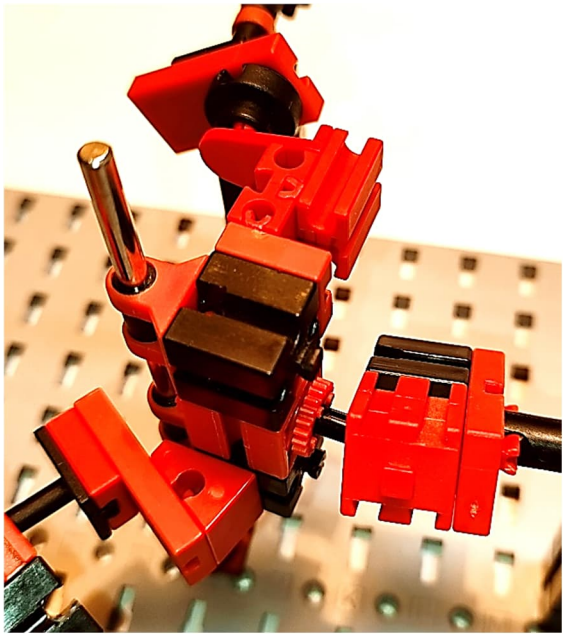


Abb. 4: Variante a) – Detail

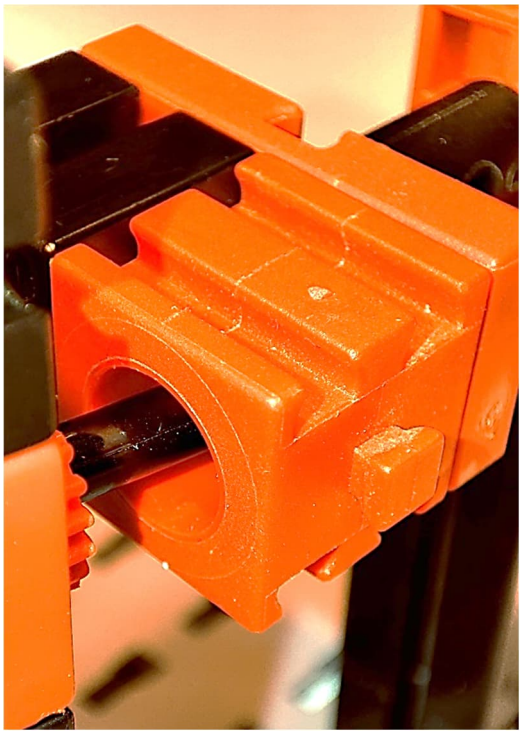


Abb. 5: Variante a) – Detail

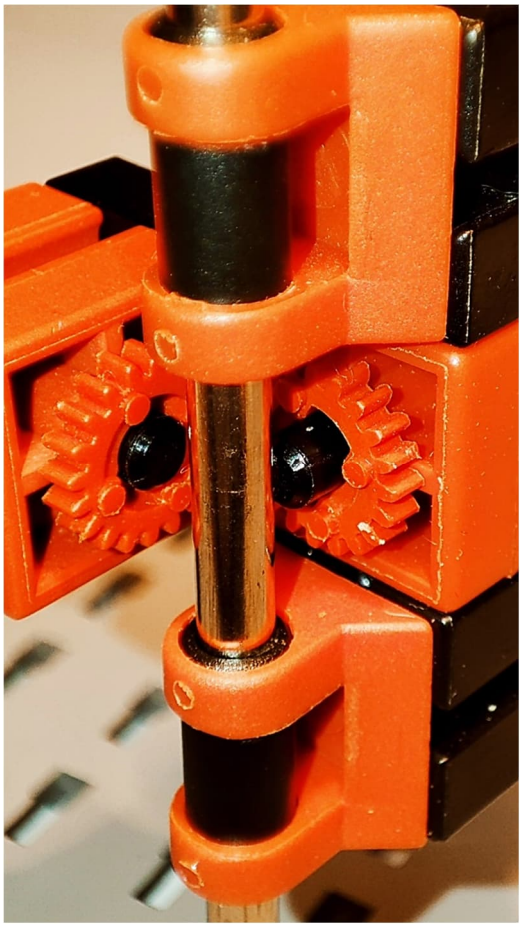


Abb. 6: Variante a) – Detail

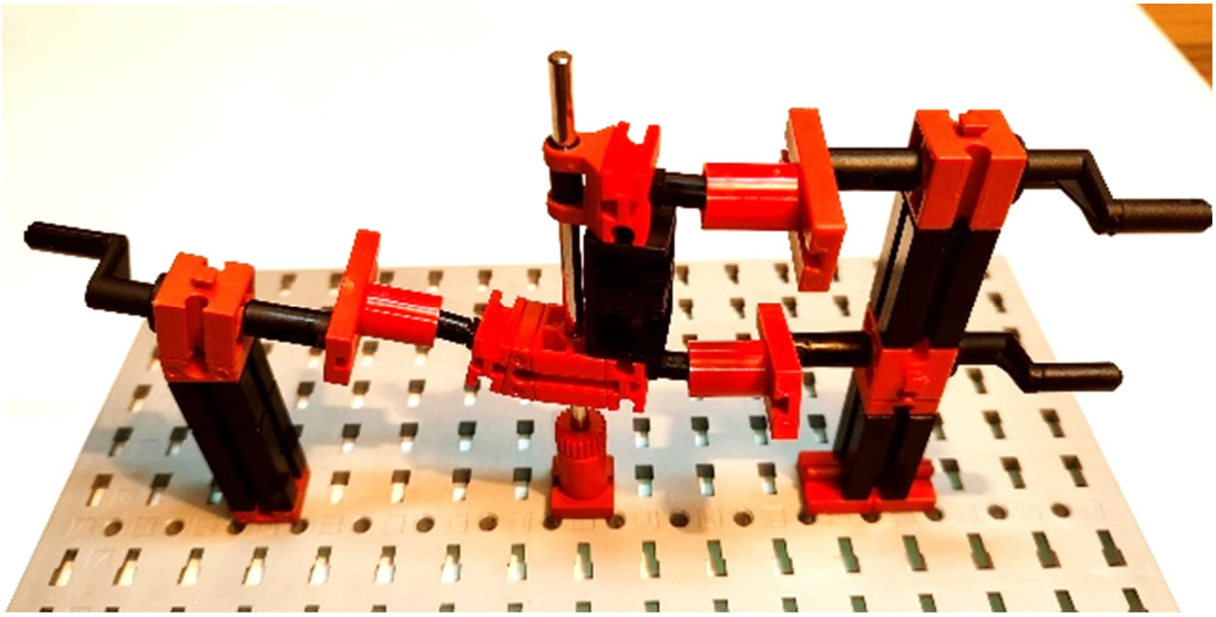


Abb. 7: Variante b)

Variante b) mit der Achsenverschraubung (38843) und der Lagerhülse schwarz (36819) auf dem geraden Schlauchanschluss (163204) läuft ebenfalls sehr leichtgängig (Abb. 7 bis 10).

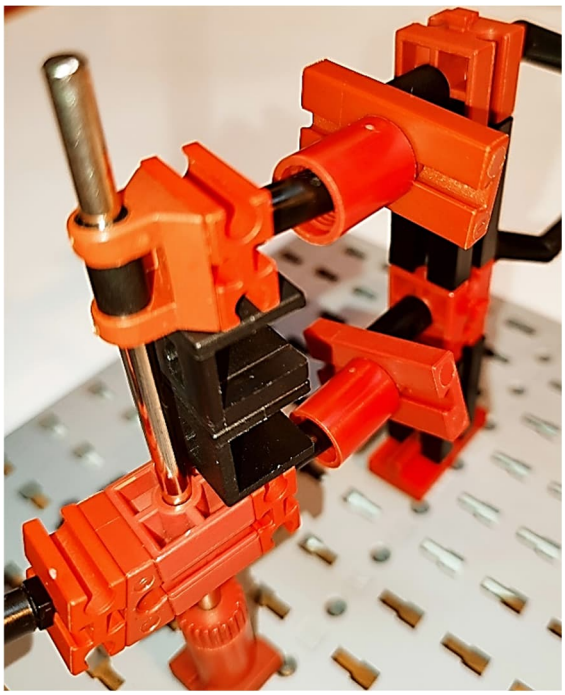


Abb. 8: Variante b) – Detail

Hierbei wird die Drehbewegung an zwei Abtriebswellen übertragen, die im Winkel 180° und 360° angebracht sind.

Noch dazu liegt der 360°-Teil in einer anderen Z-Ebene.

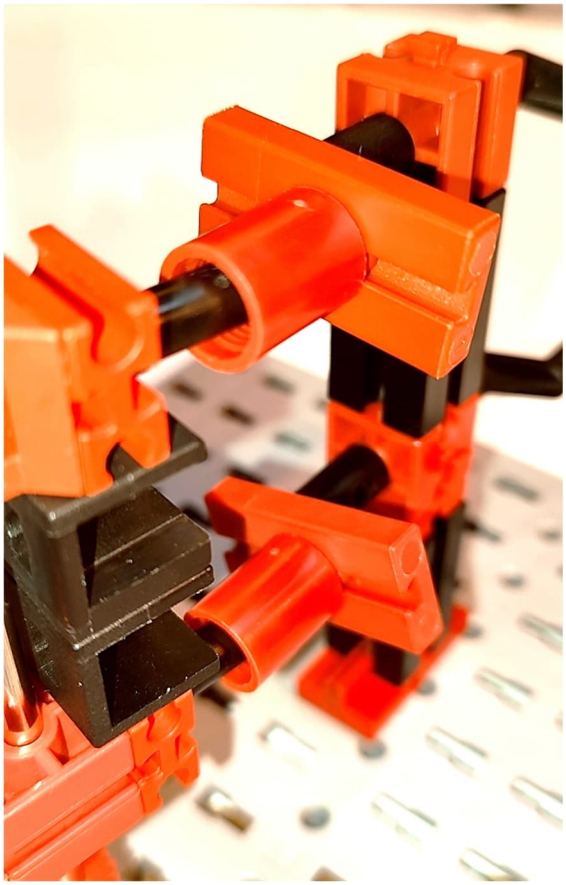


Abb. 9: Variante b) – Detail

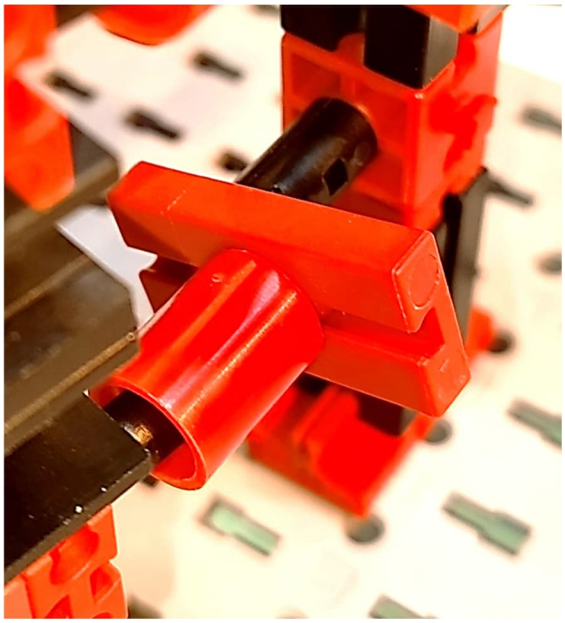


Abb. 10: Variante b) – Detail

Variante c) mit dem Winkelträger 15 gehört zu den einfachsten Varianten, enthält nur Standardteile und läuft erstaunlicherweise auch sehr ruhig bzw. ohne zu haken. Hierbei ist zu beachten, dass die Winkelträger 15 quer eingebaut sind. Bei senkrechter Lage beginnen die Rastkurbeln zu haken, kurz bevor sie die 90° erreichen (Abb. 11 bis 13).

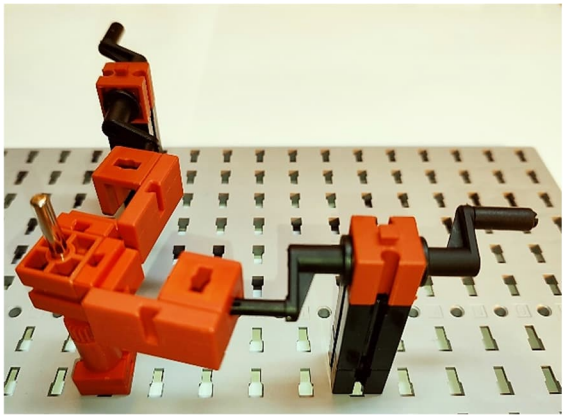


Abb. 11: Variante c)

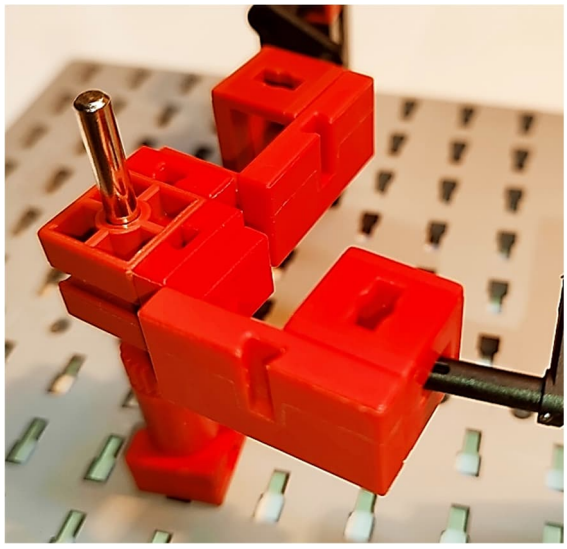


Abb. 12: Variante c) – Detail

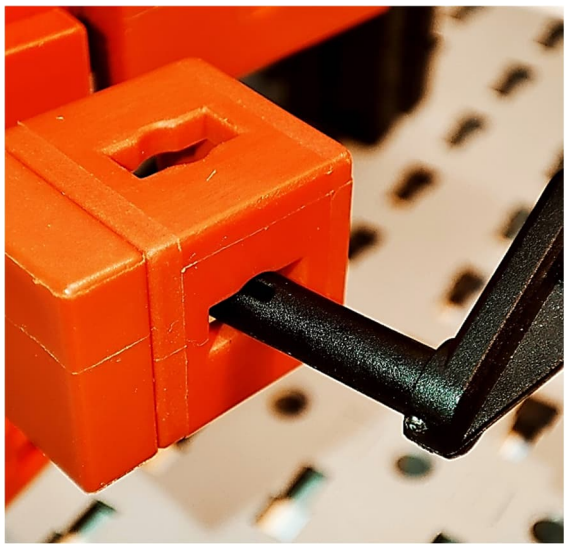


Abb. 13: Variante c) – Detail

Variante d) mit dem kleinen Seilhaken (38225) läuft ebenfalls ziemlich spielfrei und ist die erste Variante, bei der es mir gelungen ist, die Drehbewegung an drei Abtriebswellen zu übertragen (Abb. 14 bis 17).

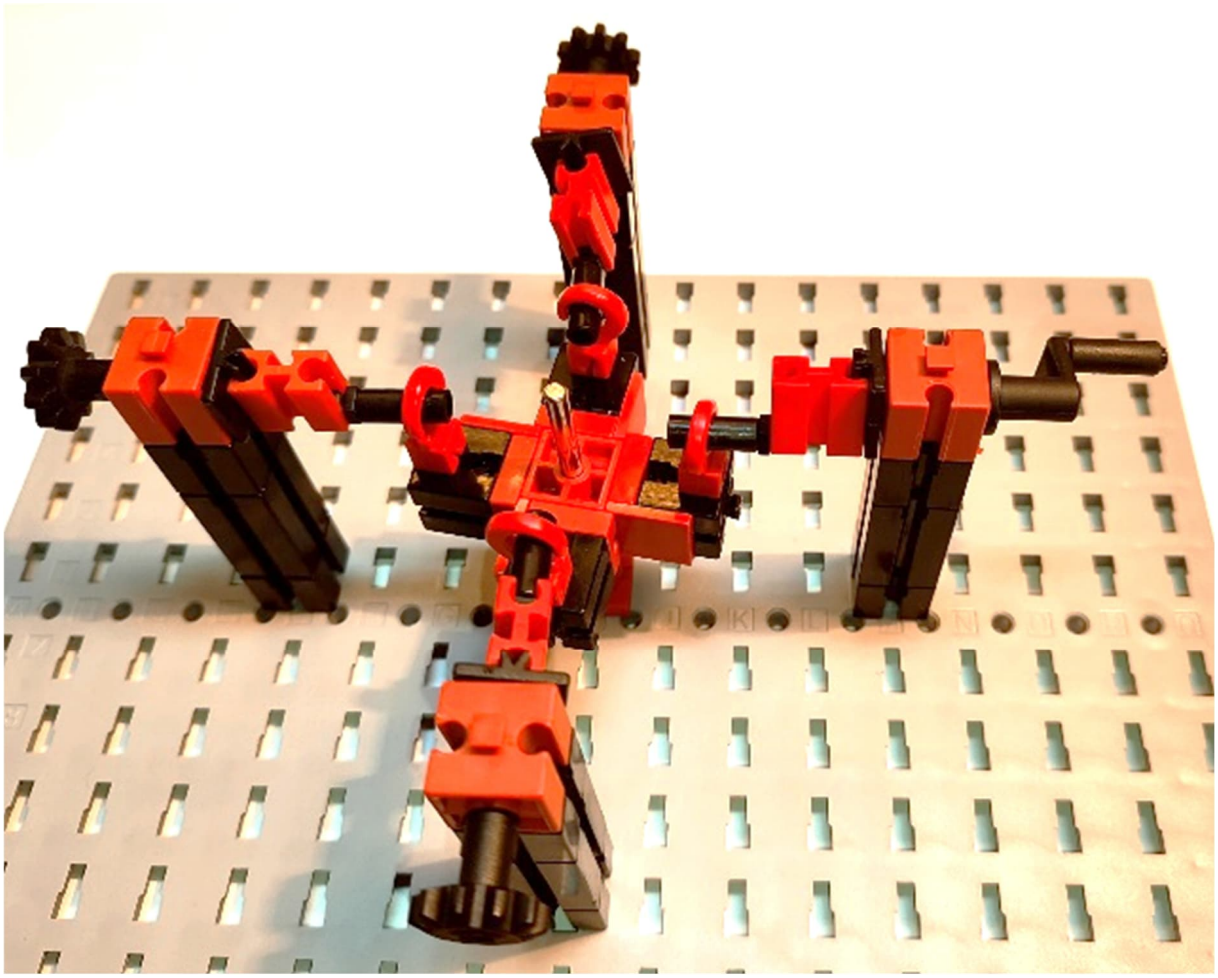


Abb. 14: Variante d)

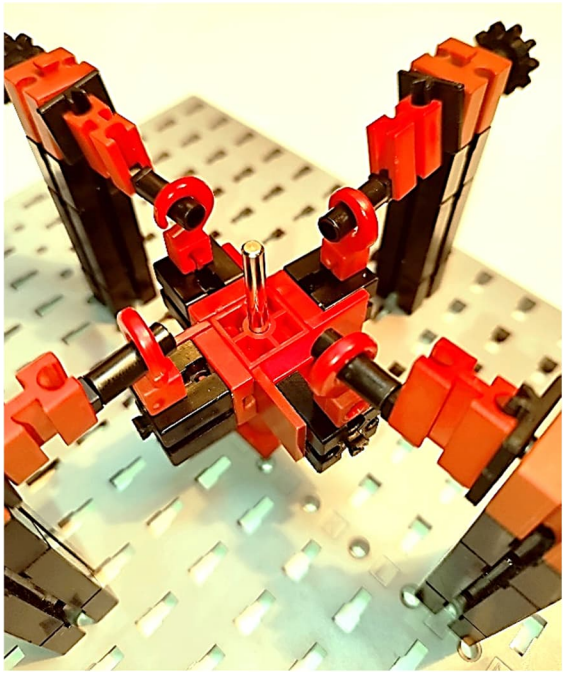


Abb. 15: Variante d) – Detail

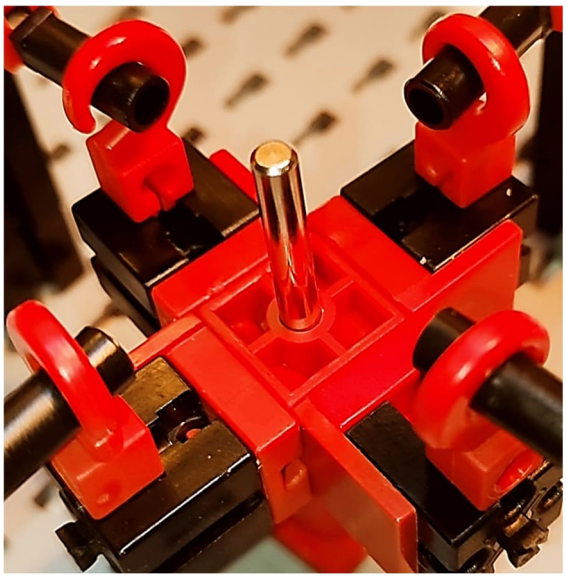


Abb. 16: Variante d) – Detail

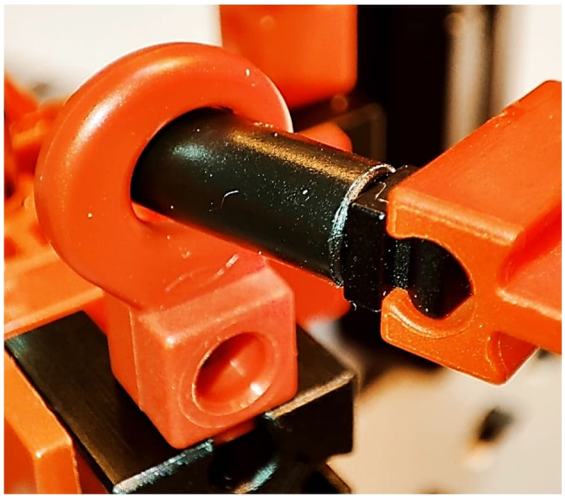


Abb. 17: Variante d) – Detail

Variante e) mit dem Unterteil der Federn, also dem Federboden (31891) und dem Kupplungsstück (38254) läuft auch sehr schön gleichmäßig und kommt dem Original irgendwie am nächsten, weil das Kupplungsstück einen kugelförmigen Kopf hat (Abb. 18 und 19).

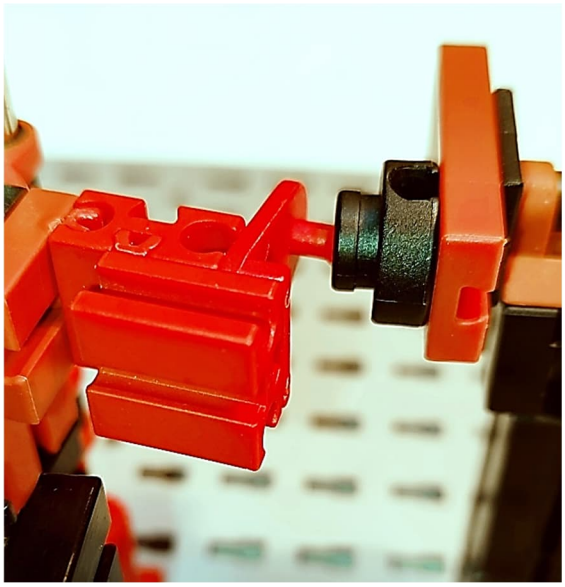


Abb. 18: Variante e) – Detail

Abb. 19: Variante e) – Detail

Variante f) hat sich Thomas Püttmann ausgedacht (Abb. 20), gleich nachdem ich ihn gefragt hatte, ob er dieses Getriebe kennt (das war noch ganz zu Beginn meiner Recherche).

Wie immer bei Thomas besteht dieses Modell aus einer minimalen Anzahl von Bauteilen und funktioniert trotzdem einwandfrei.

Die Funktionsweise der Kupplung ist hier gegenüber dem Original des vorletzten Jahrhunderts leicht modifiziert. Die Gleitpunkte, die sich auf einer kreisähnlichen Bahn befinden, liegen hier auf der inneren Seite. Im Original liegen sie auf der Seite der An- bzw. der Abtriebsachse. Ein Beispielvideo hierzu gibt es unter [2].

Variante g) basiert auf der Modellidee von Thomas Püttmann, läuft ebenfalls sehr ruhig, hat eine durch die Klemmbuchse 5 (37679) etwas stabilere Befestigung, nimmt aber dafür etwas mehr Platz weg (Abb. 21 bis 23).

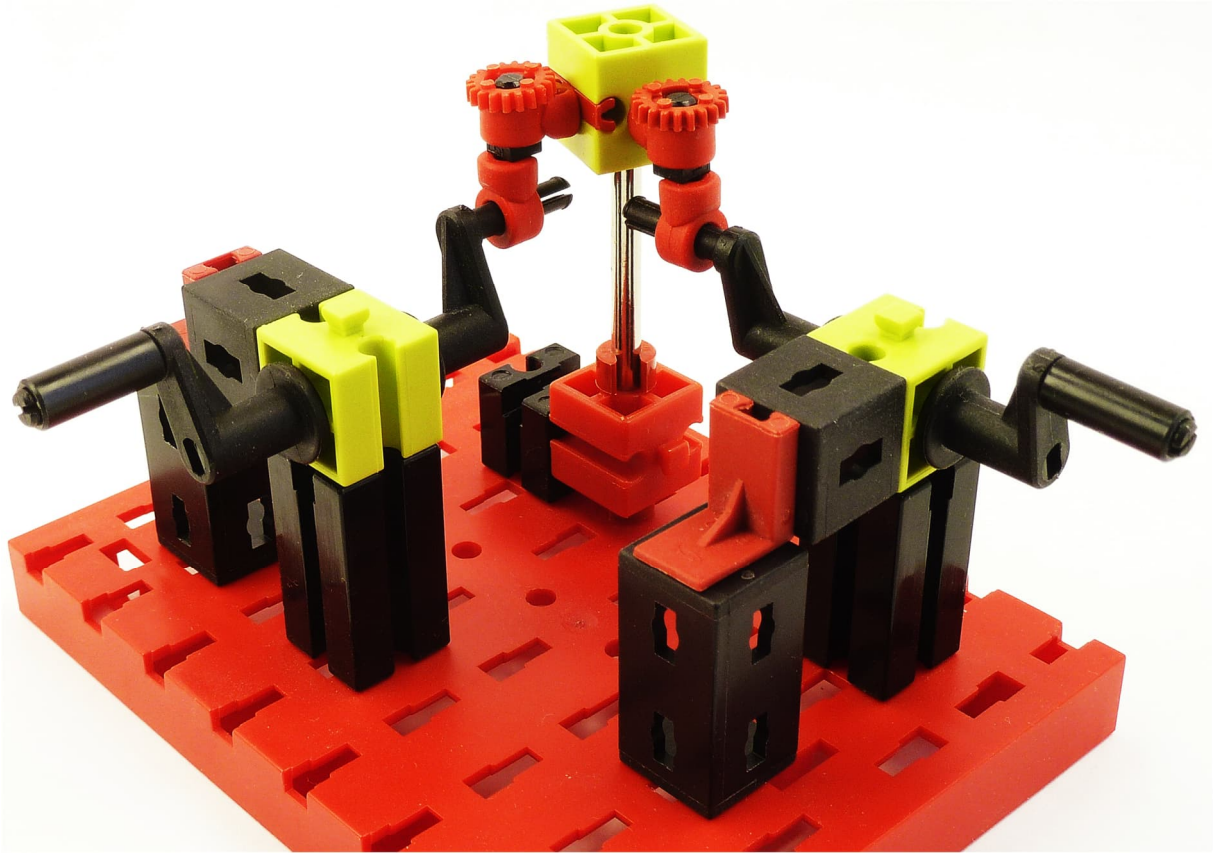


Abb. 20: Variante f) von Thomas Püttmann

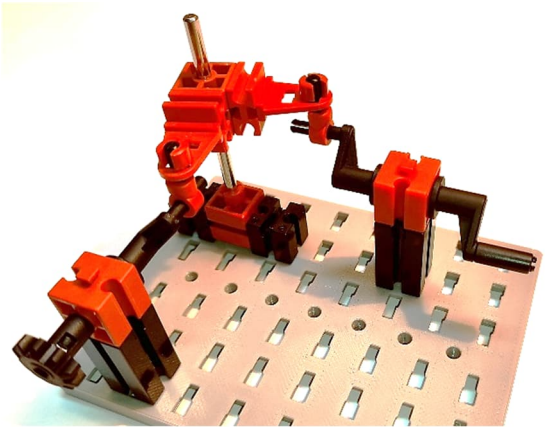


Abb. 21: Variante g)

Die Spielfreiheit (wie ich finde ein schönes Wort bzw. Teekesselchen) der Varianten ist unterschiedlich. Sie ist nicht nur abhängig von einer genauen Justage der Bauteile, sondern auch oft bauart- und oberflächenbedingt.

Besonders spielfrei sind die Varianten a), b) und f). Das sogenannte Umkehrspiel ist hier sehr gering, wobei das Original wahrscheinlich ein kaum messbares Umkehrspiel hat. Das Umkehrspiel kann man in Winkelgraden messen, wenn die Antriebsachse ihre Drehrichtung ändert und die Abtriebsachse ihre Drehrichtung erst zeitlich versetzt ändert.

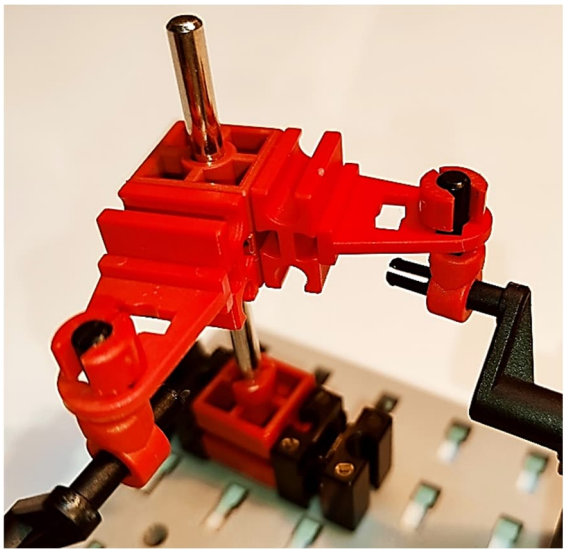


Abb. 22: Variante g) – Detail

Abb. 23: Variante g) – Detail

Beispielvideos für ein paar meiner Varianten gibt es auf [3] zu sehen.

Hier noch ein paar Tipps, wenn es nach dem Zusammenbauen haken sollte:

1. Bei manchen Bauteilen wie z. B. den Rastachsen im Allgemeinen und der Rastachse mit Platte (130593) im Besonderen ist die Oberfläche nicht ganz perfekt entgratet und hakt dann gerne mal.
2. Die Achse der Z-Richtung sollte eine Metallachse sein, weil der Baustein 15 mit Bohrung (32064) auf anderen Achsen eine wesentlich höhere Reibung hat und manchmal sogar klemmt.

Es gibt wahrscheinlich noch viel mehr Möglichkeiten, die geniale Erfindung von Thomas R. Almond aus fischertechnik nachzubauen. Ich habe viele Möglichkeiten ausprobiert, noch andere Teile einzusetzen. Viele Versuche führten ins Leere oder funktionierten nur bei sehr niedrigen Drehzahlen – aber das Experimentieren, mit dieser Art eine Drehbewegung um die Ecke zu führen, macht unglaublich viel Spaß.

Findet eure eigenen Almond Couplings heraus! Es lässt sich noch viel optimieren; auch lassen sich durch 3D-Druck-Teile komplett neue Varianten erschaffen.

Viel Spaß wünsche ich euch dabei und genießt eure Spielfreiheit!

Folgende Bauteile wurden verwendet:

- 4 mm Schlauchanschluss gerade (163204)
- Achsadapter (31422)
- Achse 60 (31032)
- Achse 80 (37384)
- Achse 110 (31031)
- Achsenverschraubung (38843)
- Aufnahmeachse (31124)
- Bauplatte 5 (15 × 30) mit 1 Zapfen und 1 Längsnut (35049)
- Bauplatte 5 (15 × 30) mit 3 Teilnuten (38428)
- Baustein 5 (37237)
- Baustein 7,5 (37468)
- Baustein 15 mit Ansenkung schwarz (32321)
- Baustein 15 mit Bohrung (32064)
- Baustein 15 schwarz mit 1 Zapfen (32881)
- Baustein 30 schwarz (32879)
- Federboden (31891)
- Federnocke (31982)
- Gelenkkurbel (35088)
- Gelenkwürfel-Klaue (31436)
- Klemmbuchse 5 (37679)
- Klemmhülse (Klemmadapter) (35980)
- Kunststoffachse 40 (38414)
- Kunststoffachse 50 (38415)
- Kupplungsstück einfach mit Haken (38254)
- Kupplungsstück zweifach (38253)
- Lagerhülse schwarz (36819)
- Rastachse 30 (35063)
- Rastachse mit Platte (130593)
- Rastadapter (36227)
- Rastritzel Z 10 (35945)
- Riegelscheibe (36334)
- Seilhaken klein (38225)
- Verbindungsstück 15 (31060)
- Winkelstein 10 (38423)
- Winkelstein 7,5° (32071)
- Winkelstein 15° (31981)
- Winkelträger 15, 1 Zapfen schwarz (36922)

- Winkelträger 15, 1 Zapfen rot (127471)
- Winkelträger 15, 2 Zapfen schwarz (36950)
- Zangenmutter (31915)

… und natürlich ein paar unterschiedliche Bauplatten.

Quellen

[1] Google Patents: *US Patent: 304,156 Coupling for shafting*. T. R. Almond, erteilt 26.08.1884.

[2] Thomas Püttmann: *Modified Almond coupling out of fischertechnik.* Auf YouTube, 2023.

[3] Ralf Geerken: YouTube-Kanal.

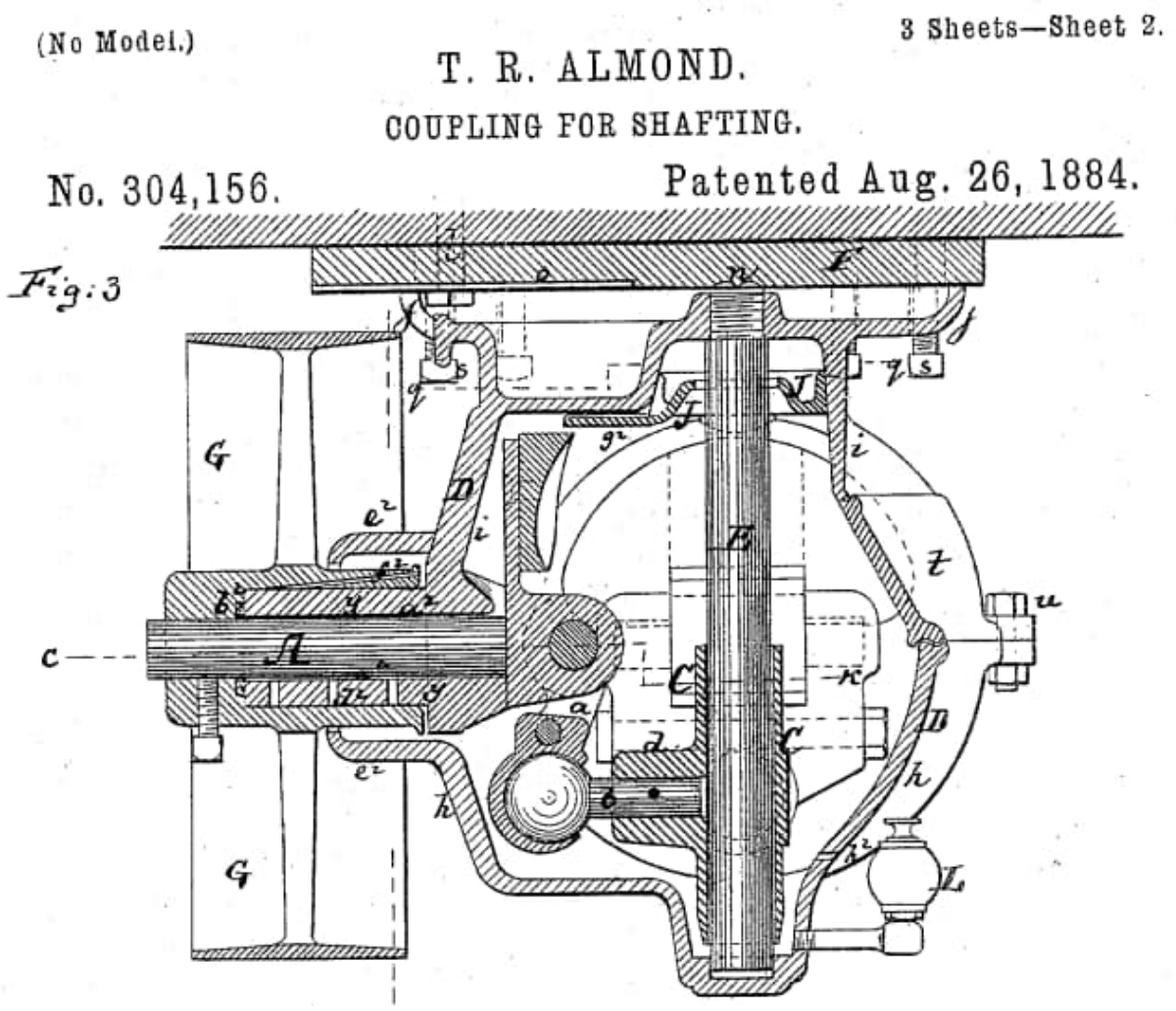


Abb. 24: Abbildung aus dem Patent von T. R. Almond, 1884 [1]

Modell

Schwerlastzugmaschine (Langhauber) – ferngesteuert (2)

Claus Ludwig

In Ausgabe 1/2022 der ft:pedia habe ich einen mit fischertechnik ferngesteuerten Gabelstapler vorgestellt [1]. Ich hatte angekündigt, in loser Folge weitere ferngesteuerte Modelle (LKWs) vorzustellen. Dass es zur Fortsetzung über 1,5 Jahre dauert, hatte ich damals nicht gedacht. Aber der Tod meiner Frau im April 2022 hat bei mir zu einer langen fischertechnik-Pause geführt. Ausstellungen gingen zwar, aber bauen und schreiben nicht. Im September dieses Jahres habe ich wieder angefangen zu bauen, und seit Oktober schreibe ich nun auch wieder. Da es drei verschiedene Bauarten von LKWs gibt und ich auch alle drei Bauarten im Modell mit Fernsteuerung realisiert habe, wird es auch drei weitere Beiträge geben. Alle Modelle sind mit Fernsteuerungen aus dem Modellbaubereich gesteuert.

Abb. 1: Vorder-/Seitenansicht der Schwerlastzugmaschine

Langhauber

LKWs werden nach ihrer Bauart in drei verschiedene Typen unterteilt. Wesentliches Kriterium ist dabei die Position des Motors und damit verbunden die der Vorderachse. Beim Langhauber – der Urversion des LKW – sitzt der Motor komplett vor der Fahrerkabine. Auch die Vorderachse befindet sich deutlich vor der Fahrerkabine, etwa in der Mitte der Motorhaube. Der-

artige LKWs kennen wir aus den Anfängen des LKW-Baus bis in die 50er Jahre – auch in Europa. In weiten Ländern wie z. B. den USA, Kanada oder Australien spielen sie auch heute noch eine wichtige Rolle; in Europa sind sie fast verschwunden. Was einen Grund hat: Entscheidend bei einem LKW ist die Länge und damit die Größe der Ladefläche. Denn wenn die Gesamtlänge des LKW begrenzt ist, kostet jeder cm Motorhaube einen cm Ladefläche. Deshalb gibt es in Europa fast ausschließlich „Frontlenker“, also LKWs ohne Motorhaube. Aber dazu kommen wir in einem der nächsten Beiträge.

Abb. 2: Seitenansicht

Abb. 3: Draufsicht

Fernsteuerung

Grundsätzliches

Der zuletzt vorgestellte Gabelstapler wird mit der fischertechnik-Fernsteuerung gesteuert, mit den bekannten Funktionen und Verkabelungen. Die jetzt vorgestellten Modelle werden alle über Funktionsmodellbau-Fernsteuerungen aus dem Modellbaubereich gesteuert. Damit ergeben sich grundsätzliche Unterschiede in der Verkabelung, den zusätzlich erforderlichen Fahr- und Schaltmodulen und der Stromversorgung.

Der fischertechnik-Empfänger kann direkt an die „normale Stromversorgung“ von 8,4 oder 9 Volt des Modells angeschlossen werden. Und sowohl der Servo als auch die Motoren können (z. B. für den Antrieb des Modells) direkt an den Empfänger angeschlossen werden. Das ist bei Fernsteuerungen aus dem Modellbaubereich anders.

Die Empfänger im Modellbaubereich benötigen für ihre Stromversorgung eine Spannung von ca. 5 Volt. Daher war früher eine zweite Stromversorgung erforderlich. 5 Volt für die Versorgung des Empfängers bzw. der Steuerung und 8,4 bzw. 9 oder 12 Volt für den Antrieb und weitere Verbraucher. Mit dem heute üblichen BEC-System (*Battery Eliminator Circuit*) ist diese zweite Stromversorgung nicht mehr notwendig. Der Empfänger wird über einen BEC-fähigen Fahrregler mit Strom versorgt; dabei spielt die gewählte Spannung für das Modell keine Rolle.

Des Weiteren benötigen alle Verbraucher (Motoren, Kompressoren, Licht etc.) einen eigenen Fahrregler oder ein Schaltmodul, der bzw. das sowohl an den Empfänger als auch an die gewählte Stromversorgung des Modells angeschlossen wird. Der Verbraucher wird also nicht direkt an den Empfänger angeschlossen, sondern an den Fahrregler bzw. das entsprechende Schaltmodul.

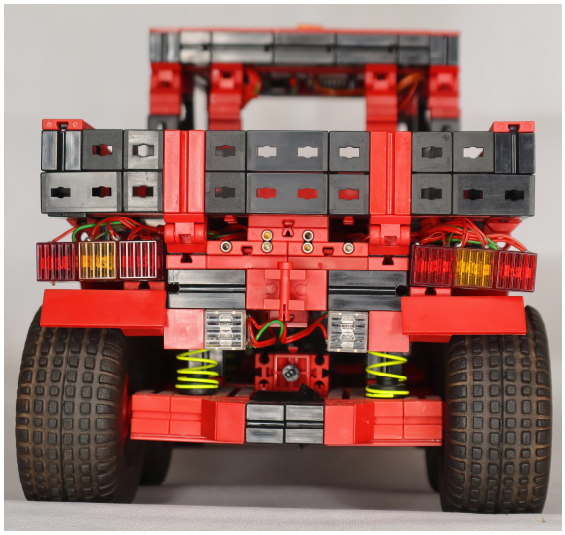


Abb. 4: Rückseite

MultiplexSmart SX9[1]

Dieser Sender (Abb. 5) ist sehr handlich, kompakt und passt – aus meiner Sicht – sehr gut zur fischertechnik. Je nach Empfänger verfügt der Sender über 5 bzw. 9 Kanäle.

Abb. 5: Multiplex-Sender

Der Sender kostet mit RX5 Empfänger als Set 120 €. Er wurde speziell für den Funktionsmodellbaubereich (Fahrzeuge und Schiffe) entwickelt. Die Kreuzknüppel dienen dabei nicht nur zur Ansteuerung der vier proportionalen Funktionen, sondern sind auch mit einer Taste ausgestattet, mit

[1] MULTIPLEX Modellsport GmbH & Co. KG

der eine weitere Funktion geschaltet werden kann. Das gilt auch für drei der vier in der Nähe der Kreuzknüppel befindlichen Schalter. Mit nur einem dieser Schalter kann der Lenk-Servo getrimmt werden.

Das Fernsteuerungssystem verfügt über zwei Empfängertypen: Der RX5 hat fünf Servo-Steckplätze und ermöglicht eine entsprechende Anzahl von zu steuernden Funktionen. Der RX4/9 (Preis: 53 €) verfügt über vier Servo-Steckplätze (vier proportionale Funktionen) und, anstelle des fünften Servo-Steckplatzes, einem Anschluss für sogenannte „MULTIswitch-FLEXX-Bausteine (Preis ca. 12 €). Hier können ein oder mehrere derartige Bausteine angeschlossen werden, über die jeweils drei weitere Funktionen gesteuert werden können. Die Bausteine sind über eine zusätzlich zu erwerbende Software programmierbar (33 €). Jeder Baustein erhält einen eigenen Anschluss an die zentrale Stromversorgung des Modells.

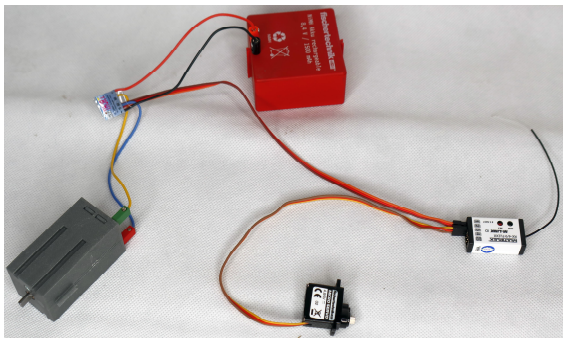


Abb. 6: Beispielhafte Verkabelung von Lenkservo und Motorfahrregler

Fahrregler, Schaltmodule, Y-Kabel

Wie schon im vorherigen Abschnitt erläutert brauchen wir für alle Funktionen neben den eigentlichen Motoren, Leuchten etc. (Akteure) auch Fahrregler und Schaltmodule, die als Bindeglied zwischen Empfänger und Akteur dienen. Hier hat sich bei mir sehr früh die Firma CTI[2] herauskristallisiert. Hier gibt es für jede fischertechnik-Anforderung den entsprechenden Fahrregler in sehr kompakter Form; neuerdings auch einen Fahrregler mit automatischem Bremslicht- und Rückfahrscheinwerfer, was die Verkabelung vereinfacht. Auch die Schaltmodule lassen aus meiner Sicht keine Wünsche offen. Schaltmodule für Blinker, Warnblinker, kleine Motoren, Brems- und Rückfahrscheinwerfer – und es gibt ein Modul, mit dem die Anzahl der proportionalen Kanäle verdoppelt werden kann. Dieses Modul kommt bei meinem nächsten Modell zum Einsatz und wird dort beschrieben.

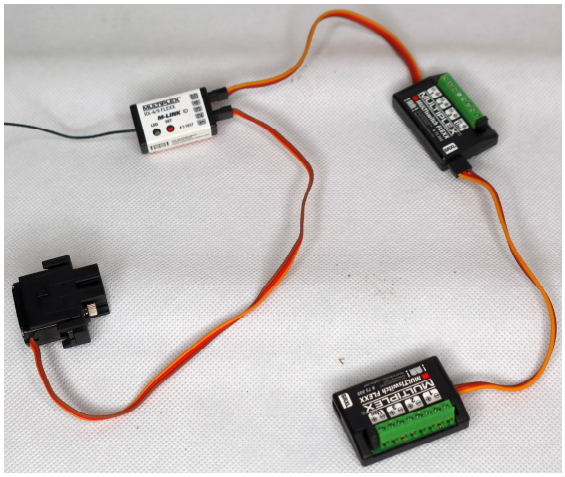


Abb. 7: Empfänger RX4/9 mit zwei MULTIswitch-Bausteinen und Servo. Die Bausteine werden einzeln an die Stromversorgung angeschlossen

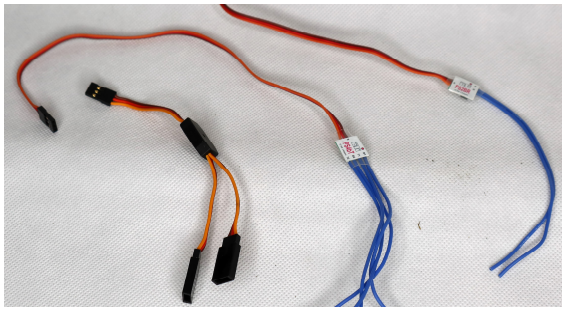


Abb. 8: Y-Kabel

Immer wieder gern von mir verwendet wird das Y-Kabel. Es dient dazu, das vom Empfänger gesendete Signal nicht nur an eine, sondern an zwei Stellen weiterzulei-

2 CTI-Modellbau, Helmut Marschall.

ten. Klassischer Einsatz ist z. B. die Ansteuerung eines Zusatzmoduls für das automatische Bremslicht und die Rückfahrscheinwerfer. Ein Kabel geht an den Fahrregler, das zweite zum Zusatzmodul.

Die Schwerlastzugmaschine

Die Schwerlastzugmaschine ist, wie schon gesagt, als Langhauber gebaut. Daher gibt es eine lange Motorhaube, unter der etwa mittig die Vorderachse sitzt. Die Ladefläche ist kurz gehalten und fest montiert, wie es bei derartigen LKWs üblich ist.

Alles, was beim Original beweglich ist, ist auch beim Modell beweglich: Die Motorhaube lässt sich nach vorne kippen und der Empfänger und der Fahrregler sind zu sehen, die Türen können geöffnet werden und die Seitenwände der Pritsche lassen sich abklappen.

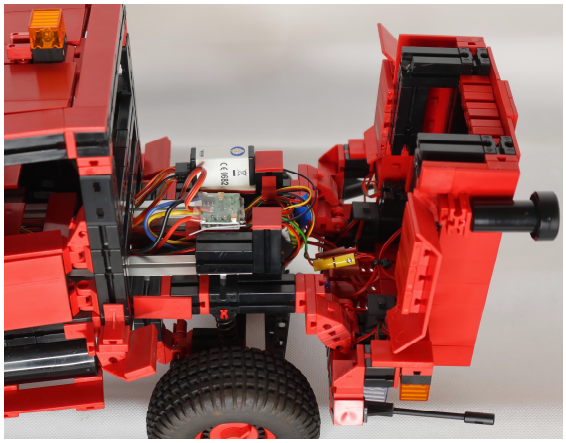


Abb. 9: Unter der Motorhaube befinden sich der Empfänger und der Fahrregler für die Antriebsmotoren

Zusätzlich kann das Dach aufgeklappt werden; hier sitzt noch eine ältere Steuerung für die Blinker. In Abb. 10 auch gut zu sehen ist die Elektronik hinter den Sitzen, welche die Bremsleuchten und die Rückfahrscheinwerfer steuert. Heute würde man beide Elektronikbausteine durch jeweils einen fingernagelgroßen Chip ersetzen beziehungsweise einen Fahrregler mit integrierter Steuerung für die Brems- und Rückfahrscheinwerfer verwenden.

Auch die Ladefläche kann geöffnet werden, um an den fischertechnik-Akku zu gelangen (Abb. 11).

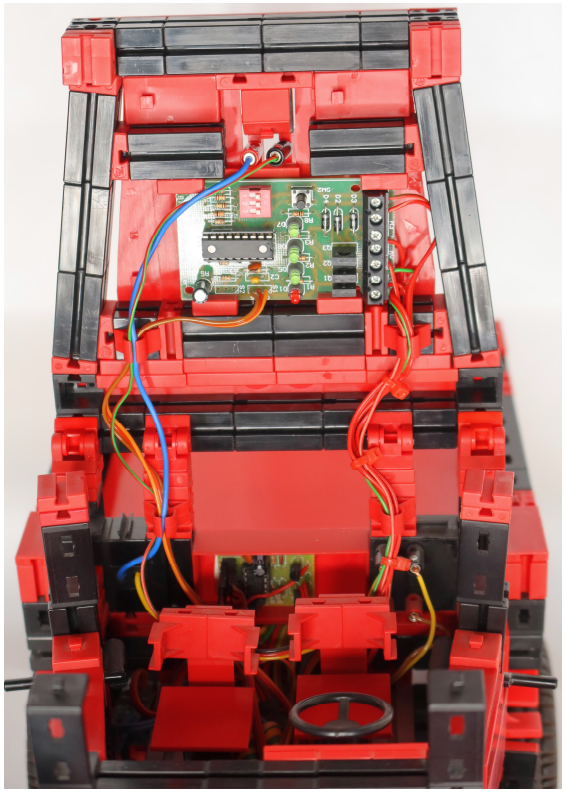


Abb. 10: Aufgeklapptes Dach mit alter Blinkelektronik sowie Elektronik für Brems- und Rückfahrleuchten hinter den Vordersitzen

Abb. 12: Vorderansicht

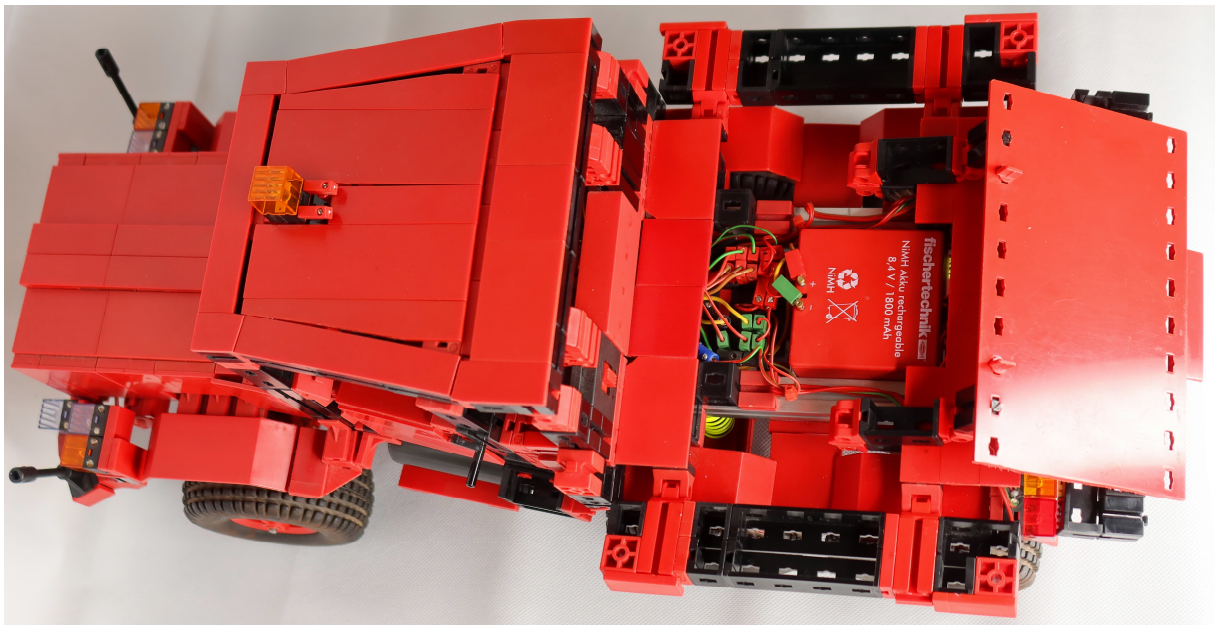


Abb. 11: Aufgeklappte Ladefläche mit fischertechnik-Akku

Der Antrieb erfolgt über zwei Powermotoren 20:1 (grau) und den aktuellen Differentialgetrieben. Anstelle der Rastachsen aus Kunststoff wurden Metallachsen von Andreas Tacke[3] verwendet.

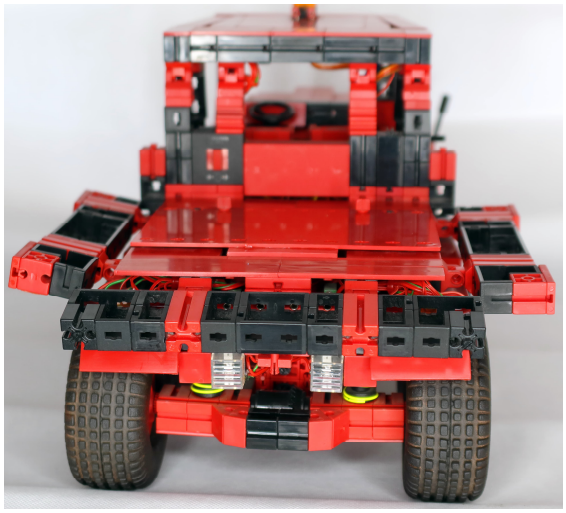


Abb. 13: Rückansicht mit abgeklappten Seitenwänden

Für die Lenkung habe ich einen größeren (vorhandenen) Servo verwendet, der direkt über der Vorderachse sitzt. Die Einbaurichtung spielt dabei keine Rolle, da über die Servoumkehrfunktion des Senders die Ausschlagrichtung umprogrammiert werden kann. Des Weiteren sind alle Achsen federnd gelagert.

Neben den Fahr- und Lenkfunktionen ist der LKWs mit Fahrlicht, Fernlichtzusatzscheinwerfern, Blinkern, einer gelben Rundumleuchte, Bremslicht und Rückfahrscheinwerfern ausgestattet.

Referenzen

[1] Claus Ludwig: *Gabelstapler – ferngesteuert (1)*. ft:pedia 1/2022, S. 31–36.

[3] Andreas Tacke (TST), Spezialteile für fischertechnik

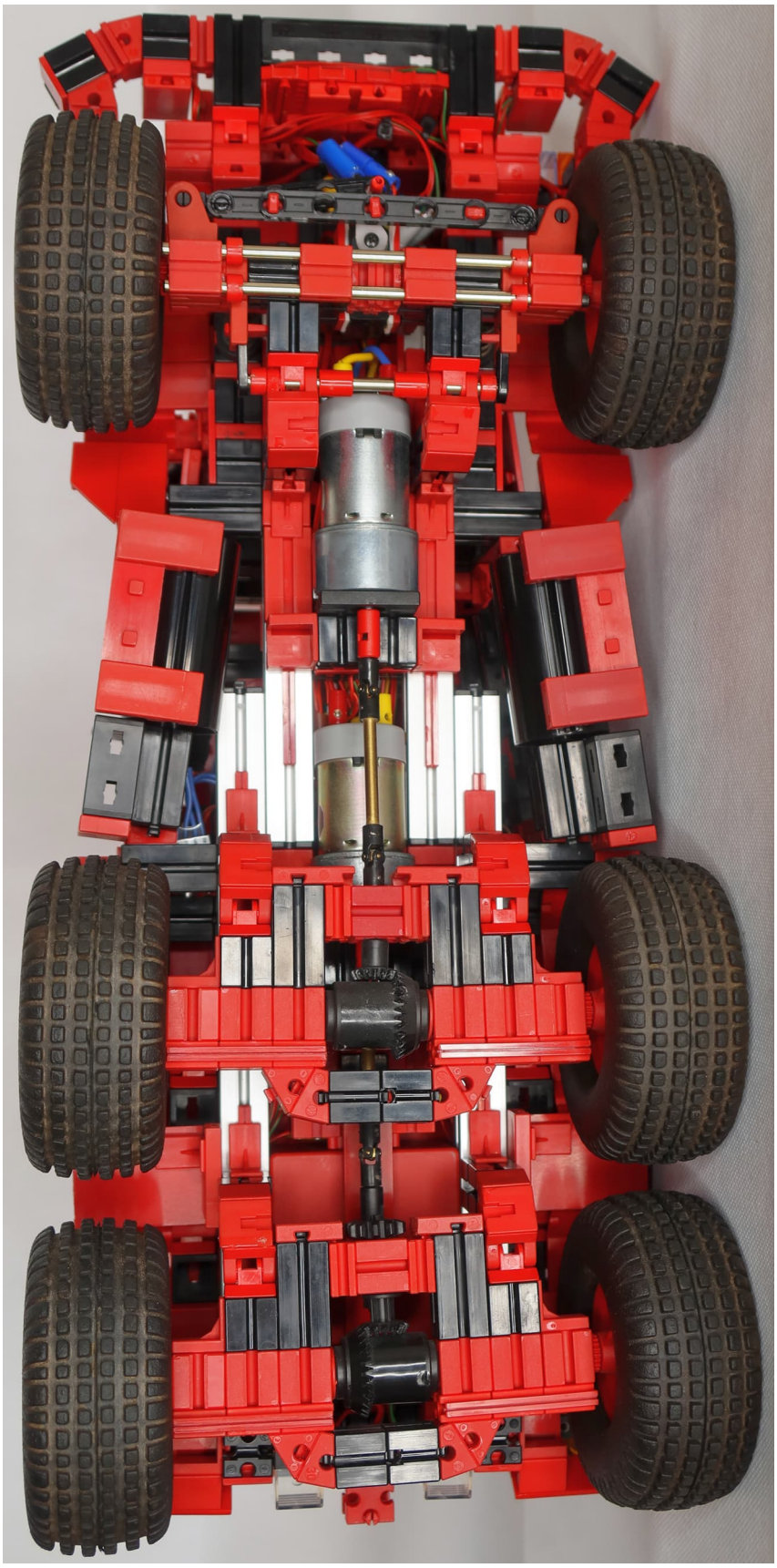


Abb. 14: Detailansicht von unten

Elektronik

Kontaktlose Schalter (Teil 2)

Arnoud van Delden

Im vorigen Teil dieser Beitragsreihe [1] habe ich einen berührungslosen Berührungssensor vorgestellt, dessen Ausgang für digitale Anwendungen ohne störendes Schaltrauschen verwendet werden kann. Diese Sensoren benötigen eine Versorgungsspannung von 5 Volt. Um sie für den Einsatz mit klassischen Elektronikmodulen und dem TXT-Controller geeignet(er) zu machen, habe ich mit der Versorgungsspannung und dem Signalausgangspegel dieser Sensoren experimentiert. Und weil ich auf eine schöne Alternative zu dem beim letzten Mal besprochenen Berührungsschalter gestoßen bin, habe ich eine kleine Platine und verschiedene 3D-druckbare Gehäuse entwickelt, die es ermöglichen, die Sensoren bedenkenlos mit 9 Volt mit den Silberlingen und anderen Elektronikmodulen von fischertechnik zu betreiben.

Einführung

Die im ersten Teil der Beitragsreihe vorgestellten Sensorplatinen [1] sind mit einem TTP223 „Human Body Detector“-Chip bestückt. Der TTP223-ASB-Chip ist ein 16-poliges SMD-Bauteil, aber die online verfügbaren Sensorplatinen verwenden in der Regel eine Variante dieses Chips mit nur 6 Pins (TTP223E-HA6) und etwas weniger Konfigurationsmöglichkeiten.

Standardmäßig verhält sich der Sensor wie ein Taster, aber durch Schließen der Lötbrücke B erhält man eine Ein-/Aus-Schaltfunktion (T-Flipflop). Abb. 1 zeigt zwei DIY-Varianten des kleinen Sensorbricks.

Um eine einwandfreie Berührungserkennung zu gewährleisten, wurde die transparente Oberfläche so dünn wie möglich gehalten. Die kleine Sensorplatine ist normalerweise rot, wurde aber für die Montage oben weiß gespritzt, um einen transparenten Brick mit einem etwas neutraleren Aussehen zu erhalten.

Versorgungsspannung

Im Fazit von Beitragsteil 1 habe ich festgehalten, dass eine Voraussetzung für den Einsatz der beschriebenen Sensoren das Vorhandensein einer 5-Volt-Versorgungsspannung ist [1]. Wie bereits festgestellt arbeiten viele dieser Sensorplatinen mit dieser für Mikrocontroller üblichen Versorgungsspannung. Die HW-763-Sensorplatinen können ohne Modifikationen oder selbstgebaute Randelektronik nicht mit einer höheren Spannung betrieben werden.

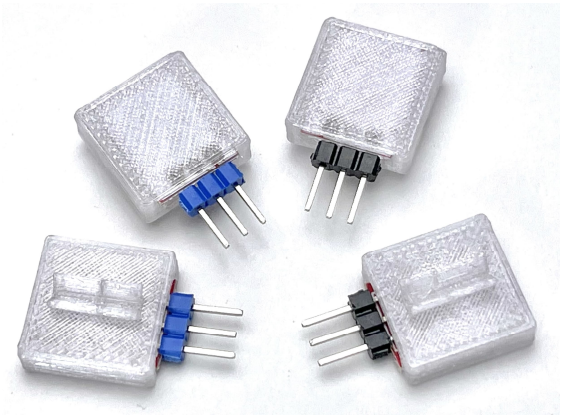


Abb. 1: Berührungssensoren mit Schalt- (blau) oder Drucktastenfunktion (schwarz)

Die Betriebsspannung der fischertechnik-Elektronikmodule und -Controller liegt jedoch traditionell bei 9 Volt. Bei den klassischen Elektronikmodulen, die vom Modul GB Gleichrichter (30811) gespeist

werden, lag die Arbeitsspannung sogar noch höher (ca. 11 Volt). Bevor ich untersuchte, ob und mit welchen Modifikationen die Sensoren vielleicht auch mit einer Versorgungsspannung von 9 Volt funktionieren könnten, kam mir die Vorsorge für eine 5-Volt-Versorgungsspannung wieder sehr gelegen.

Meine bisherigen Selbstbauversuche in dieser Hinsicht reichten bereits von einem sehr einfachen Steckmodul auf dem Gleichrichtermodul bis hin zu einem Modell mit mehreren Ausgangsspannungen, Spannungs- und Strommessung und einer einstellbaren automatischen Sicherung (siehe Abb. 2) [2].

Abb. 2: Verschiedene DIY-5-Volt-Stromversorgungen für weitere Sensor-Experimente

Für zukünftige Experimente mit weiteren DIY-Silberlingen habe ich kürzlich zwei Modelle hinzugefügt, bei denen die Sensoren einfach über die üblichen dreipoligen Servokabel mit den flachen schwarzen (sogenannten Dupont-) Steckern versorgt werden können [3]. Diese Stromversorgungsmodule erleichtern das Experimentieren mit den 5-Volt-Sensoren. Abgebildet ist auch ein Modell mit 2,5-mm-Buchsen für die fischertechnik-Stecker und ein Modell mit Anschlussmöglichkeiten für acht Sensoren, die mit Anschlussschienen für die einpoligen farbigen Experimentierdrähte ausgestattet sind.

Mit Silberlingen verwendbar?

Die kleinen Sensorplatinen haben ein verwertbares Ausgangssignal, dessen Spannungspegel von einem Mikrocontroller oder TXT-Controller leicht ausgelesen werden kann. Dabei ist es sogar unerheblich, ob der Sensorausgang bei Erkennung einen hohen Spannungspegel am Ausgang liefert (Schaltverhalten mit positiver Logik), oder stattdessen im Ruhezustand (Schaltverhalten mit negativer Logik). Schließlich lassen sich Spannungsschwellen und Schaltverhalten einfach in der Controller-Software einstellen. Soll das Ausgangssignal jedoch als Eingangssignal für einen Elektronikbaustein aus diskreter Elektronik dienen, ist die direkte Kompatibilität nicht immer offensichtlich. Aus diesem Grund war ich im ersten Teil dieses Beitrags angenehm überrascht, dass die besprochenen HW-763-Sensormodule mit TTP223-Chip, auch wenn sie mit 5 Volt gespeist werden, direkt an den verschiedenen klassischen fischertechnik-Elektronikmodulen (den Silberlingen) verwendet werden können.

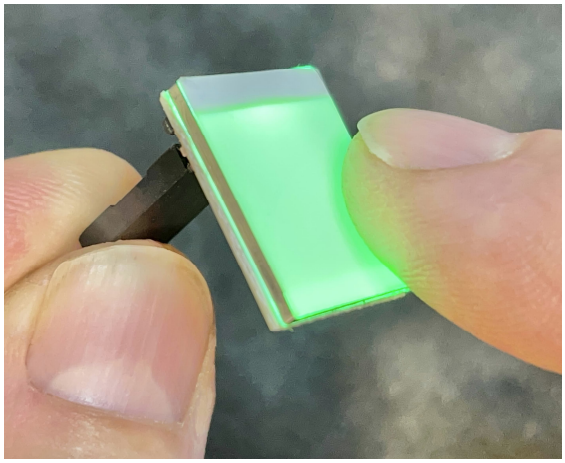


Abb. 3: Touch-Schalter mit großer leuchtender LED-Fläche

Kurze Zeit später stieß ich auf eine Variante der Sensorplatine, bei der das LED-Licht den gesamten Berührungsbereich beleuchtet (Abb. 3).

Es stellte sich heraus, dass dieser Touch-Schalter der „HTTM-Serie“ in verschiedenen LED-Farben (rot, grün, blau, gelb und

eine Art „Regenbogen“-Variante) erhältlich ist. Während die zuvor besprochene HW-763 Sensorplatine eine LED an der Unterseite der Platine hatte, um das gewünschte visuelle Feedback zu geben, machte diese schöne HTTM-Variante das Schalten einfach und deutlich erkennbarer. Zum Beispiel könnte die Sensorfläche dieses Sensors direkt sichtbar und durch eine Öffnung im Deckel zugänglich bleiben. Ein zusätzlicher Vorteil wäre, dass es dann auch für den Benutzer klar ersichtlich wäre, wo er den Finger ansetzen muss, um den Sensor zu aktivieren.

Dieser HTTM-Touch-Schalter ist etwas teurer und durch seine weiße Milchglas-LED-Abdeckung fast doppelt so groß (20 x 16 mm) wie der andere ‚Touch Switch Sensor‘. Ein kleiner Unterschied besteht darin, dass das Schaltverhalten durch eine eingelötete Steckbrücke (0 Ω-SMD-Widerstand) bereits als Kippschalter vorkonfiguriert ist. Eine Konfigurationsmöglichkeit zur Beeinflussung des Einschaltverhaltens mittels einer lötbaren Steckbrücke fehlt.

Nach diesen nicht unüberwindlichen kleinen Hindernissen stellte sich aber leider auch ein viel größerer Unterschied heraus. Im Gegensatz zum kleineren Sensor, der einen sogenannten „Open-Collector“-Ausgang hat [4], hat dieser Sensor einen Ausgang mit einer sehr niedrigen Impedanz. Dadurch ist der Signalausgang elektronisch nicht mit der herkömmlichen Transistor-Dioden-Logik der Silberlinge kompatibel. Da die Silberlinge mit negativer Logik arbeiten, entspricht ein „aktives“ Signal dem Ziehen des entsprechenden Eingangs auf Nullpotential. Der Ausgang dieses Berührungsschalters ist dazu nicht in der Lage, da die LED am Ausgang gegen Masse geschaltet wird.

Da ein Spannungsregler eingebaut ist, beträgt der Spannungspegel des Ausgangs nur 3,3 Volt. Auch hier wird die Erfassung eines solchen Ausgangssignals des Sensors an einem analogen Eingang des TXT-Controllers oder eines anderen Mikrocontrollers kaum Probleme bereiten. Aber wenn dieser Signalpegel in einer Schaltung mit Silberlingen mit dem Elektronik-Grundbaustein G (30813) erfasst werden soll, stoßen wir aufgrund des niedrigen Spannungspegels auf ein anderes Problem.

Detektionsschwelle

Die Erkennung, ob ein Signalpegel eine bestimmte Spannungsschwelle überschreitet, kann dem Grundbaustein überlassen werden. Eine nette Sache ist, dass am Ausgang danach sowohl das normale als auch das komplementäre (invertierte) Sensorsignal an den Ausgängen A1 und A2 dieses Moduls verfügbar sind. Die einzige Verbindung, die hergestellt werden muss, ist die Verbindung des Sensorausgangs mit E1.

Der fischertechnik Grundbaustein ist ein Differenzverstärker: Am Eingang E2 liefert ein Spannungsteiler standardmäßig etwas weniger als die Hälfte der Versorgungsspannung. Diese Spannung bildet die Erkennungsschwelle für den Signalpegel an E1. Bei einer Versorgungsspannung von z. B. 9 Volt werden daher Signale an E1 nicht erkannt, die unterhalb der Detektionsschwelle von 4,5 Volt liegen.

Die kleinen roten HW-763-Sensorplatinen haben ein aktives Ausgangssignal von 4 Volt und wären daher schon mit dem Grundbaustein nicht ohne weiteres erfassbar. Bei der Überprüfung stellt sich heraus, dass dies tatsächlich der Fall ist. Da ich auch Varianten meines Stromversorgungsmoduls habe, die 7,5 Volt statt 9 Volt liefern, konnte ich auch recht einfach überprüfen, dass mit den Silberlingen bei dieser reduzierten Betriebsspannung eine Erkennung stattfindet.

Für die oben erwähnten HTTM-Touch-Schalter mit einer Ausgangsspannung von nur 3,3 Volt ist das aber immer noch keine Lösung. Die Versorgungsspannung der Silberlinge müsste auf mindestens 6,5 Volt reduziert werden, und eine so niedrige

Versorgungsspannung könnte Auswirkungen auf die Schaltstabilität und die Helligkeit der Kontrollleuchte im Grundbaustein haben.

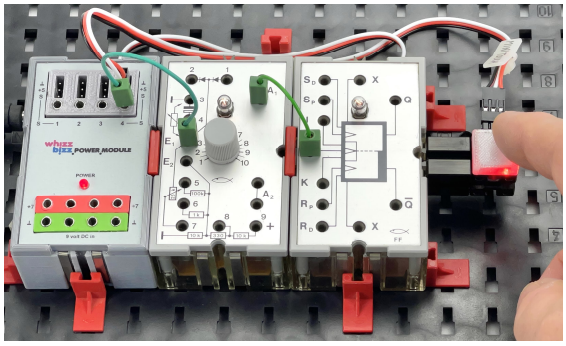


Abb. 4: Bei 7 Volt Versorgungsspannung reicht die Erkennung von 4 Volt gerade noch aus...

Nach einigem Tüfteln bin ich auf eine (meines Wissens bisher nicht dokumentierte) Möglichkeit gestoßen, die Erkennungsschwelle des Differenzverstärkers des Grundbausteins zu senken, ohne die Versorgungsspannung zu verringern. Wenn man die Eingangsspannung an E2 durch Einfügen eines 10 kΩ-Widerstands zwischen Bus 3 (oder einer anderen 0-Volt-Spannung) und Bus 6 verringert, kann die ursprüngliche Versorgungsspannung bei 9-10 Volt bleiben. Der Spannungsteiler wird durch diesen Parallelwiderstand so eingestellt, dass an E1 bereits Eingangsspannungen erkannt werden können, die nur 1/3 der Versorgungsspannung betragen. Auf diese Weise können also auch Sensoren mit kleinen Ausgangspegeln mit den bekannten Silberlingen verwendet werden. Abb. 5 zeigt das Prinzipschaltbild, bei dem ein Widerstand *R* parallel zum unteren Teil des Spannungsteilers geschaltet ist und die Erkennungsschwellenspannung an E2 absenkt.

Es gibt also durchaus Lösungen, um 5-Volt-Sensoren mit relativ geringem Ausgangssignal mit den Silberlingen zu verwenden. Effizienter wäre es jedoch, wenn sie direkt an 9 bis 12 Volt betrieben werden könnten und das Ausgangssignal in jedem Fall die gewünschte Schaltverträglichkeit hätte.

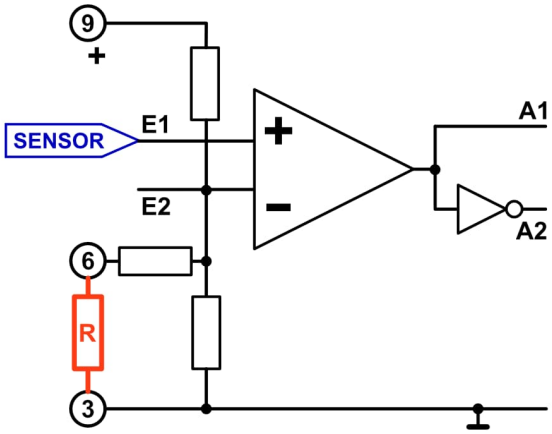


Abb. 5: Prinzipielle Regelung zur Absenkung der Vergleichsschwelle

Platine

Ich wollte nun eine Universalplatine entwickeln, die beide Modelle von Sensorplatinen aufnehmen kann. Die Sensoren sollten über ein dreipoliges Kabel mit Flachstecker mit 5 Volt versorgt werden könnten, sodass vorgefertigte Kabel verwendet werden könnten.

Ein nettes Extra wäre es, wenn man die Sensoren zusätzlich mit den herkömmlichen fischertechnik-Steckern anschließen könnte. Dafür sollten die Sensoren mit 9 Volt arbeiten und ein entsprechendes Ausgangssignal haben. Sie könnten dann mit den Silberlingen und dem TXT-Controller verwendet werden, ohne dass die Ausgangssignale in einem Grundbaustein gepuffert oder wie oben beschrieben mit einem Differenzverstärker verstärkt werden müssten.

Es ist eigentlich schade, dass in der Praxis das Schaltverhalten des Sensors nur einmal (nämlich vor der Endmontage auf der Platine) mit den Lötjumpern ausgewählt werden muss. Viel schöner wäre es, wenn dies später nach Bedarf konfiguriert werden könnte. Aus diesem Grund habe ich auf der neuen Platine für jeden Sensor einen Jumper vorgesehen, mit dem das Schaltverhalten nach Entfernen der Abdeckung geändert werden kann.

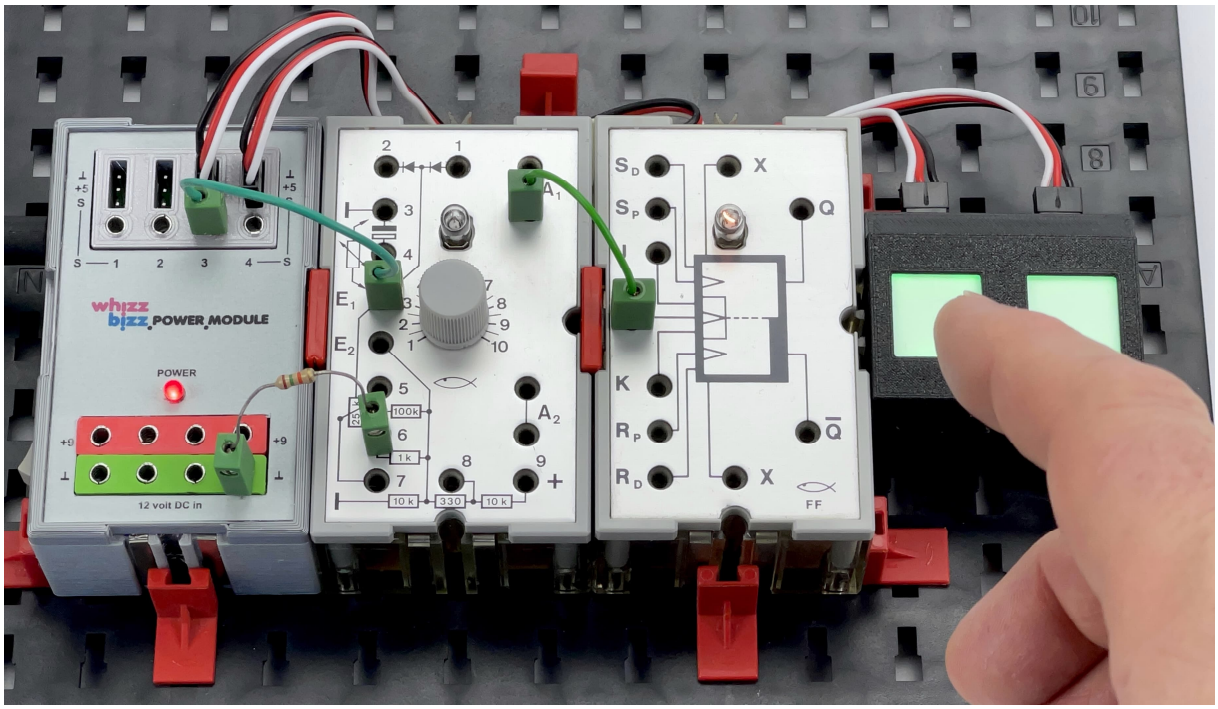


Abb. 6: Durch Einfügen eines Widerstandes können niedrigere Signalpegel an E1 erfasst werden

Das kleine rote Sensormodell HW-763 hat zusätzlich eine Lötbrücke, die, wenn sie geschlossen ist, das Signalverhalten des Ausgangs umkehrt. Beim Anlegen der Versorgungsspannung ist die Spannung des Ausgangssignals dann sofort annähernd gleich der Versorgungsspannung, beim Erkennen fällt sie auf Null ab. Dies scheint ideal für die Verwendung mit Silberlingen, die mit negativer Logik arbeiten.

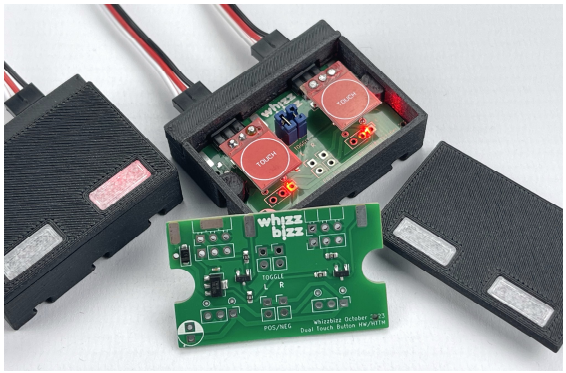


Abb. 7: Leiterplatte mit zwei HW-763-Sensoren

Doch leider hat diese Jumperstellung keinen Einfluss auf das Verhalten der Anzeige-LED. Tatsächlich folgt sie immer noch der positiven Logik, was diesen Modus in der Praxis leider wenig nützlich erscheinen lässt. Obwohl ich im Platinenentwurf extra einen Jumper dafür vorgesehen hatte, habe ich diese Option fallen gelassen. Schließlich verfügt der HTTM-Sensor ohnehin nicht über diese Option.

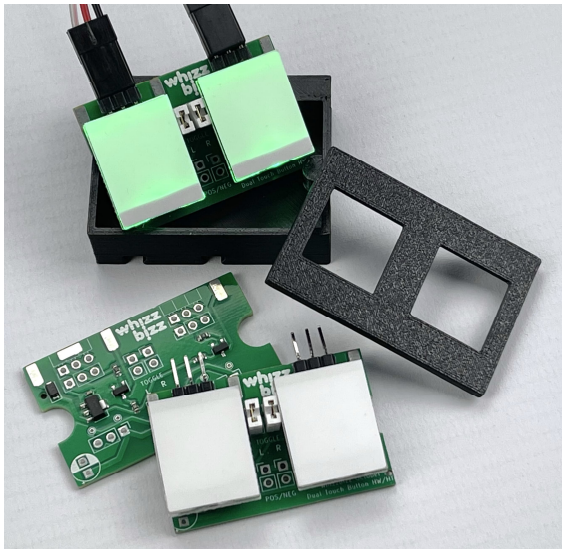


Abb. 8: Leiterplatte mit zwei beleuchteten Sensoren

Einfacher war es bei der 9-Volt-Version ohnehin, die Logikpegel des Ausgangs zu

invertieren. Das Schaltverhalten des Ausgangs entspricht dann der negativen Logik, sodass er mit den Silberlingen ohne Invertierung des Ausgangssignals verwendet werden kann.

Beim Zusammenbau der Sensorplatinen (zwei pro Leiterplatte) habe ich eine Seite des ‚TOG'-Jumpers [5] mit einem kleinen Draht an die Leiterplatte angeschlossen. Auf der HW-763 Sensorplatine ist dies als Lötbrücke B gekennzeichnet. Auf der HTTM-Platine mit der größeren LED-Lichtfläche ist dies nur die eine Lötbrücke. Der 0 Ω-SMD-Widerstand, der standardmäßig das „Toggle"-Verhalten aktiviert, muss dazu allerdings erst entfernt werden.

Reihenfolge der Anschlüsse

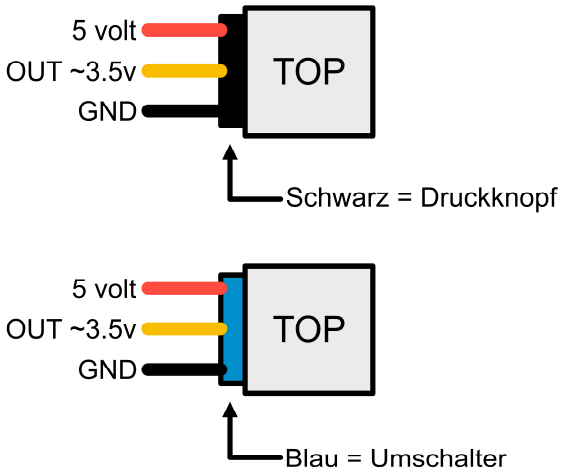


Abb. 9: Anschlussplan des transparenten Sensor Bricks

Die endmontierten Berührungsschalter (Abb. 10) können auf der Sensorseite mit handelsüblichen Servokabeln angeschlossen werden. Beachtet jedoch, dass die Pin-Reihenfolge der kleinen HW-763-Sensorbausteine in Abb. 9 abweicht. Sehr praktisch sind daher Kabel mit drei losen Stiftsteckern auf der einen Seite und 2,5 mm fischertechnik-Steckern auf der anderen.

Das in Abb. 11 gezeigte Verbindungskabel habe ich selbst angefertigt, aber es ist natürlich auch möglich, ein fertig gekauftes Kabel zu halbieren und mit fischertechnik-Steckern zu verschrauben. Auf diese Weise erhält man für wenig Geld zwei brauchbare Anschlusskabel.

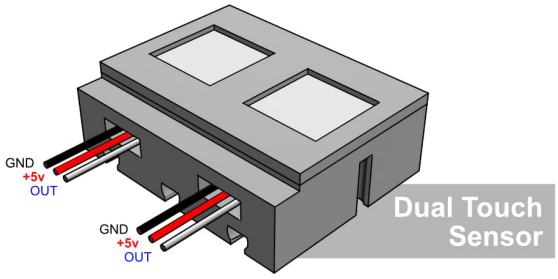


Abb. 10: Anschlussschema des doppelten Tastsensors

Mit drei separaten Stiften ist es einfach, die gewünschte Anschlussreihenfolge für jeden Sensor zu wählen, aber diese Kabel sind (soweit ich weiß) nicht als fertige Produkte erhältlich. Glücklicherweise lässt sich aber die Anschlussreihenfolge der drei Stifte in den handelsüblichen 3-poligen Servokabeln (der Pluspol ist der mittlere Leiter) recht einfach ändern: Spannt den Stecker vorsichtig in einen Schraubstock ein. Hebt mit einem kleinen Schraubendreher vorsichtig die kleine Kunststofflasche des zu entfernenden Stifts an und zieht den Steckerstift am Draht aus dem Stecker (Abb. 12). Drückt gegebenenfalls die Kunststofflippe noch einmal kurz bündig mit dem Steckergehäuse. Setzt dann die Stifte in der gewünschten Reihenfolge wieder ein.

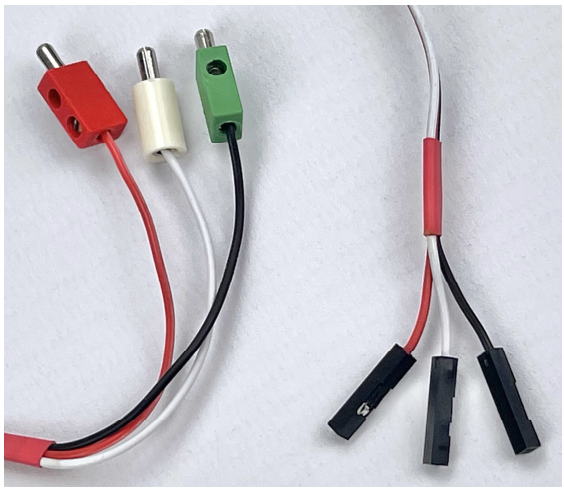


Abb. 11: Universelles ‚Dupont-zu-fischertechnik'-Kabel

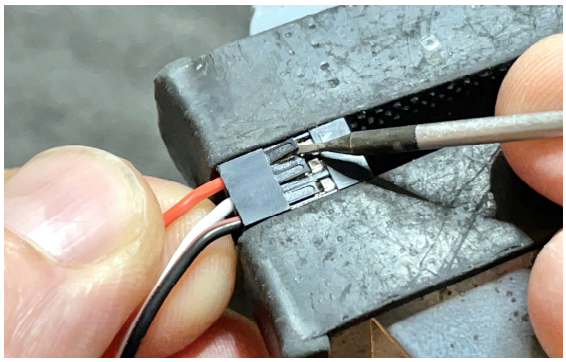


Abb. 12: Änderung der Reihenfolge der Stifte im Stecker

Der nächste Schritt bestand darin, die Gehäuse für die verschiedenen Anschlussmethoden zu entwerfen. Außerdem wurden für die verschiedenen Sensortypen auch unterschiedliche Deckel benötigt.

Verschiedene Gehäuse

Abb. 7 und 13 zeigen die Variante, bei der ein geschlossenes, 3D-gedrucktes Gehäuse jeweils zwei Sensoren beherbergt. Die (rote) LED, die bei Erkennung aufleuchtet, befindet sich an der Unterseite der kleinen Sensorplatine. Eingebaut in einen halbtransparenten Hüllbaustein oder ein spritzwassergeschütztes geschlossenes Gehäuse mit aus transparentem Material gedrucktem Diffusor ist dieses LED-Licht ausreichend sichtbar.

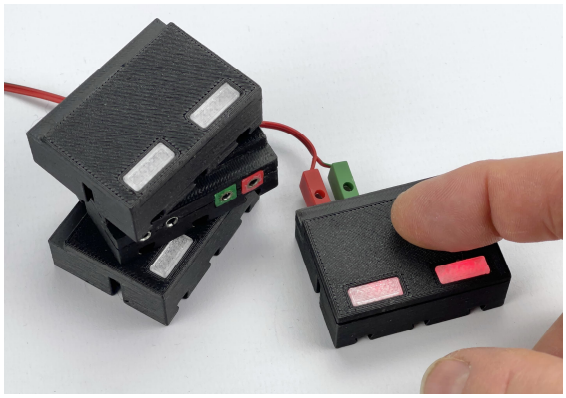


Abb. 13: Das 9-Volt-Sensorgehäuse mit fischertechnik-Buchsen

Beim Praxistest (auf der Südconvention) stellte sich jedoch heraus, dass man intuitiv manchmal den Finger auf den Diffusor legt, statt etwas höher – wo sich die Erfassungsfläche unter der Abdeckung befindet. Glücklicherweise erwies sich der Sensor in den meisten Fällen als empfindlich genug, um trotzdem eine Berührung zu erkennen.

Für die HTTM-Variante habe ich ein Gehäuse entworfen und gedruckt, bei dem die Sensorfläche jedes Sensors durch den Deckel sichtbar bleibt. Wie bei der geschlossenen Variante habe ich zusätzlich mit einer doppelten Variante experimentiert, die direkt vier Sensoren beherbergt. Die Idee ist, diese in Zukunft für digitale Experimente zu verwenden, bei denen dann vier Bits pro Modul „programmiert“ werden können. Von beiden Varianten habe ich eine Version gebaut, die mit 5 Volt arbeitet und mit einem dreipoligen Kabel angeschlossen werden kann, und eine, welche die bekannten 2,5-mm-fischertechnik-Buchsen führt und mit 9 Volt operiert (Abb. 8 und 14).

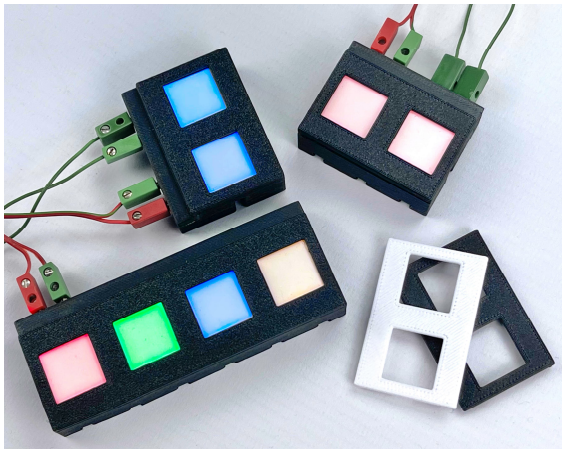


Abb. 14: Sensoren mit LED-Oberfläche bei 9 Volt

Fazit

Ich habe mit verschiedenen Versionen der TTP223-Detektorplatinen experimentiert. Die kleine HW-763-Platine kann sogar in einem 5 mm dicken transparenten Baustein untergebracht werden; der Sensor ist empfindlich genug, um durch den dünnen Deckel eines geschlossenen Gehäuses ausgelöst zu werden. Bei der ‚HTTM-Series‘-Version der Sensorplatine, bei der die gesamte Berührungsfläche farbig leuchten

kann, sollten die Sensoren natürlich durch Löcher im Gehäusedeckel zugänglich sein.

Die Lichtleistung und Sichtbarkeit des gelb leuchtenden Sensors ist am geringsten, der grün leuchtende Sensor ist meiner Meinung nach am deutlichsten sichtbar.

Um die Sensorplatinen möglichst dicht unter dem Deckel zu montieren, habe ich mir eine kleine Montagehilfe auf dem 3D-Drucker ausgedruckt, die beim Löten die genaue Montagehöhe bestimmt.

Eine fertig bestückte Sensorplatine, die auch direkt den dreipoligen Stecker aufnimmt, lässt sich leicht in das Gehäuse einschieben und wird beim Schließen des Deckels schlüssig festgeklemmt.

Als sehr arbeitsintensiv erwies sich die Herstellung der Varianten mit fischertechnik-Buchsen. Wegen des geringen Einbauraums hinter den Sensoren im Gehäuse konnten die Buchsen nur über eine lose, nach der Montage zu verklebende Rückseite mit innenliegenden Drähten angeschlossen werden. Als Steckverbinder habe ich die Bündhulsen-Alternativen (PTN2-10) verwendet [6]. Da die Anfertigung eines Anschlusskabels (Abb. 11) relativ einfach ist, ist nach diesen Schwierigkeiten bei der Montage der kleine dreipolige Flachstecker auf der Sensorseite eindeutig vorzuziehen.

Sowohl die 9-Volt- als auch die 5-Volt-Version des Sensors können mit Mikrocontrollern oder dem TXT-Controller von fischertechnik betrieben werden. Die 5-Volt-Variante wird mit einer dreipoligen Standard-Servokabel angeschlossen. Der Ausgang der HW-763-Variante kann von den Silberlingen direkt als Eingangssignal verwendet werden. Die 5-Volt-HTTM-Variante kann jedoch nur über einen Grundbaustein in der oben beschriebenen Weise genutzt werden. Vielleicht lohnt es, eine modifizierte Version der Platine zu entwickeln, bei der die Versorgungsspannung immer über den Spannungsregler läuft, und die eine bessere elektronische Kompatibilität mit den Eingängen der Silberlinge gewährleistet. Die Wahl zwischen 5 und 9 Volt Versorgungsspannung wird dann ganz trivial.

Die 9-Volt-Versionen der beiden Sensorgehäuse stellen die interne Versorgungsspannung für die Sensoren (5 Volt) selbst zur Verfügung und sind zudem mit einem Ausgang ausgestattet, der so viel Strom aufnehmen kann, dass er direkt an den Eingängen der Silberlinge verwendet werden kann. Damit eignet sich der Sensorausgang z. B. hervorragend zur Ansteuerung des Flip-Flop (30815), des Mono-Flop (30816) oder als Eingang für die Module OR/NOR (36481) bzw. AND/NAND (36482). Er kann auch das Relaismodul mit eingebautem Verstärker (36392) direkt ansteuern. Damit kann eine Reihe dieser Berührungssensoren beispielsweise als Biteingänge für eine digitale Schaltung aus Silberlingen eingesetzt werden.

Am nützlichsten ist es, wenn die Sensoren so modifiziert werden können, dass sie direkt mit 9 Volt betrieben werden können und der Signalausgang so weit wie möglich mit dem der Silberlinge elektronisch kompatibel ist. Dies war auch der Ausgangspunkt, um eine Reihe anderer Sensoren bei meiner fortgesetzten Suche nach berührungslosen Schaltern einsetzbar zu machen (siehe Teil 3 in dieser Ausgabe der ft:pedia).

Alle meine Erkenntnisse können auf meiner Website nachgelesen werden [7]. Dort gibt es auch Links zu einem Video, das ich über meine Erfahrungen mit den Tastsensoren erstellt habe, und zu den 3D-Druckdateien für alle, die die Gehäuse und Abdeckungen selbst drucken möchten.

Quellen

[1] Arnoud van Delden: *Kontaktlose Schalter*. ft:pedia 3/2023, S. 49–53.

[2] Arnoud van Delden: *Eine zukunftssichere Stromversorgung*. ft:pedia 2/2022, S. 73–86.

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

[4] Wikipedia: Open-Collector-Ausgang

[5] TTP223, Datenblatt, V1.0, 04.07.2008.

[6] Arnoud van Delden: *Alternative Verbindungslösungen für Stecker und Buchsen (Teil 2)*. ft:pedia 3/2023, S. 54–62.

[7] Projektseite WhizzBizz.

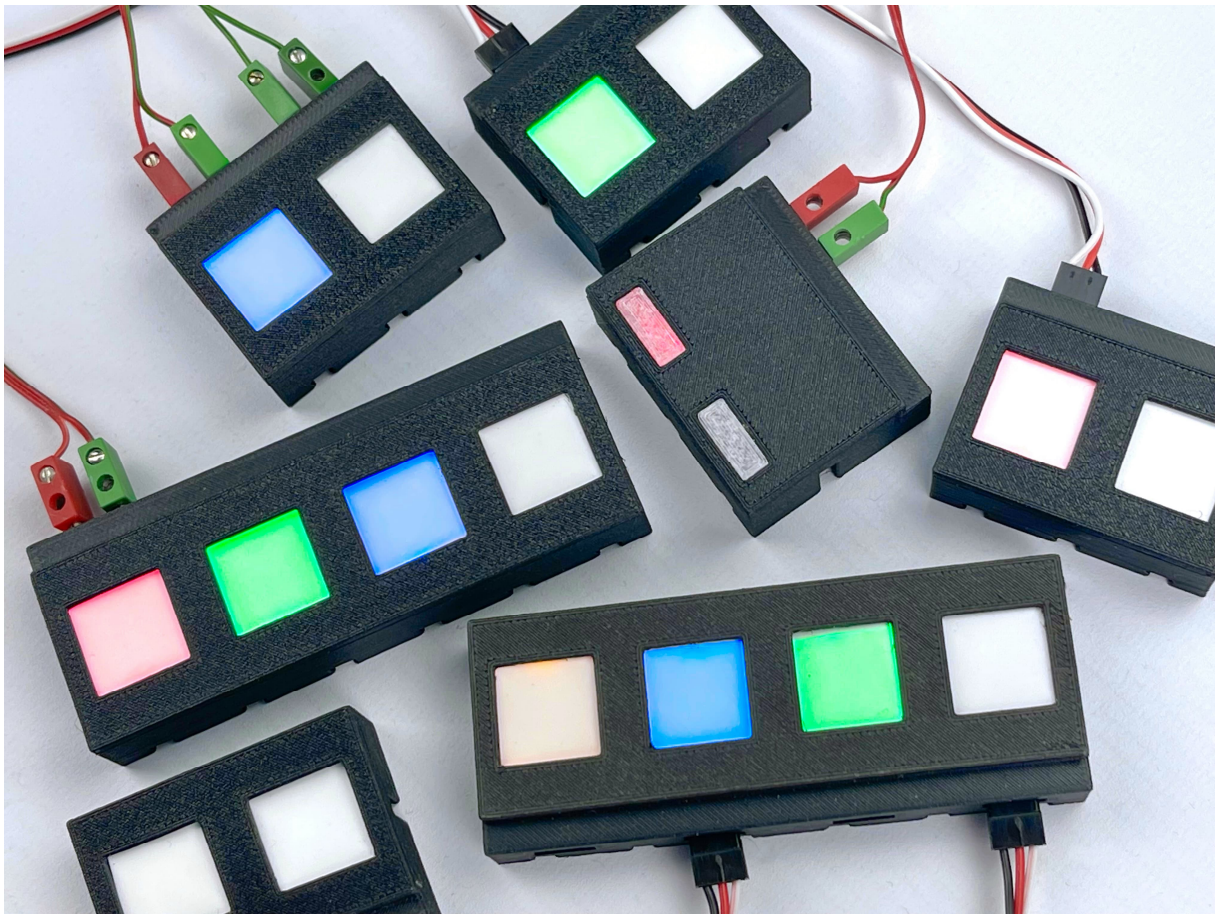


Abb. 15: Eine Sammlung von nützlichen Berührungssensoren

Elektronik

Kontaktlose Schalter (Teil 3)

Arnoud van Delden

In den vorangegangenen Teilen dieser Beitragsreihe [1, 2] wurden verschiedene berührungslose Schalter vorgestellt. Die Prämisse war, dass diese auch in Schaltungen verwendet werden können, die mit traditionellen Silberlingen aufgebaut sind. In der Zwischenzeit habe ich mit der Versorgungsspannung und dem Signalausgangspegel einiger anderer Sensoren experimentiert, wie z. B. IR- (Hinternis-), LDR- (Helligkeits-), Hall- (Hall-Effekt-) und PIR- (Bewegungs-) Sensoren. Die Tatsache, dass einige dieser Sensoren mit relativ geringen Modifikationen direkt mit 9 Volt betrieben werden können, macht sie grundsätzlich für den Einsatz mit dem TXT-Controller geeignet. Einige Ausgänge können sogar direkt an den klassischen fischertechnik-Elektronikmodulen und den Silberlingen angeschlossen oder können zu diesem Zweck leicht elektronisch kompatibel gemacht werden.

Verschiedene Sensoren

Ich habe mit einigen Sensormodulen experimentiert, die man für wenig Geld online kaufen kann und die bereits mit einem Differenzverstärker oder Schmitt-Trigger[1] ausgestattet sind. Das Ausgangssignal dieser Platinen kann direkt und störungsfrei in synchronen Schaltungen verwendet werden. Selbst in den hier vorgestellten Anwendungen, bei denen Silberlinge zum Einsatz kommen, ist es nicht notwendig, das Schaltsignal zu „entprellen“.

Drei der hier besprochenen Sensoren konnten in einem kleinen (22,5 x 15 x 45 mm) Gehäuse untergebracht werden, von dem ich verschiedene Versionen für den Anschluss mit einem flachen 3-poligen (Dupont-)Stecker und mit Buchsen für die Verwendung der bekannten fischertechnik-Stecker entworfen habe. Ich experimentierte auch mit verschiedenen Deckeltypen. Später ersetzte ich den geschlossenen Deckel durch einen transparenten Diffusor, der als Fenster für die Detektions-LED auf der Platine dient. Wenn das Gehäuse mit rot bedruckt ist, passt das optisch gut zu den Elektronikmodulen aus dem fischertechnik-Programm.

Abb. 1: LDR-, Hall-, IR- und PIR-Sensoren

9-Volt-IR-Sensor

Ein sehr nützlicher Sensor, mit dem ich seit einiger Zeit experimentiere, ist der IR-Hindernissensor (HC-SR501). Wie die meisten anderen Sensorplatinen erwartet auch diese in der Grundversion eine Versor-

[1] Ein Schmitt-Trigger sorgt für eindeutige steile Signalflanken um den Schaltpegel.

gungsspannung von 5 Volt. Die kleine Sensorplatine hat auf einer Seite bereits einen 3-poligen Stecker angelötet. Leider ist die Pin-Reihenfolge dieses Steckers nicht genormt. Während bei 3-poligen Servokabeln das + sinnvollerweise in der Mitte liegt (um Verpolung und Beschädigung beim Umstecken zu vermeiden), liegt auf dieser Sensorplatine der 0-Volt-Anschluss (GND) in der Mitte.

Wie man den Sensor in die 9-Volt-Domäne von fischertechnik bringen kann, habe ich in einem früheren Beitrag vorgestellt [3]. Da der auf der Platine verwendete LM393-Differenzverstärker prinzipiell bereits eine Versorgungsspannung von bis zu 36 Volt verträgt, ist der Betrieb an 9 Volt theoretisch nicht einmal ein sehr großes Problem. Allerdings befinden sich auf der Platine zwei sehr helle rote low current SMD-LEDs, die mit einem Vorwiderstand von 1 kΩ versehen sind. Eine LED leuchtet auf, sobald die Versorgungsspannung angeschlossen wird, die andere, wenn sie erkannt wird. Logischerweise brennen diese LEDs bei 9 Volt sehr hell und dürften auch nicht lange halten.

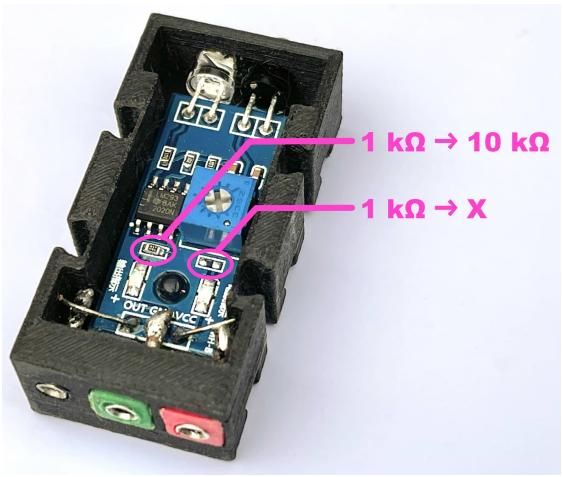


Abb. 2: Modifikationen, um den Sensor für 9-Volt-Betriebsspannung geeignet zu machen

Ich habe das Problem gelöst, indem ich die beiden Vorwiderstände entfernt und die Erkennungs-LED mit einem neuen 10 kΩ SMD-Vorwiderstand versehen habe. Im Prinzip reichen mindestens 4,7 kΩ, aber so haben wir noch etwas Spielraum für bis zu 12 Volt (für die Nutzung an Silberlingen). Die LEDs brennen auch bei sehr geringen Strömen hell genug. Das Einlöten des neuen Widerstandes mit der SMD-Bauform 0603 ist eine Präzisionsarbeit, kann aber auch ohne spezielle Heißluft-Lötgeräte erfolgen.

Da der Sensor nach dieser Modifikation auch mit 9 Volt einwandfrei funktioniert, kann er sowohl mit dem TXT-Controller als auch mit den Silberlingen problemlos verwendet werden. Der Ausgang kann direkt an die Eingänge der Silberlinge angeschlossen werden, da der verwendete LM393 mit seinem Offenen-Kollektor-Ausgang[2] keinerlei Probleme mit der negativen Logik und der konventionellen Elektronik dieser klassischen Elektronikmodule hat. Der Sensorausgang kann daher auch direkt in Schaltungen verwendet werden, die mit Silberlingen aufgebaut sind.

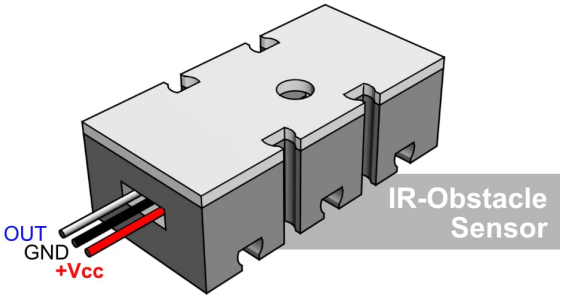


Abb. 3: Anschlussreihenfolge bei Verwendung eines 3-poligen Flachsteckers

Bei einer Spannungsversorgung von 9 Volt entspricht das Ausgangssignal praktisch der Versorgungsspannung. Diese fällt auf 0 Volt ab, wenn ein Objekt innerhalb des Erfassungsbereichs erkannt wird, der durch ein Loch im Deckel eingestellt werden kann. Das Schaltverhalten entspricht damit der negativen Logik der Silberlinge.

2 Siehe Wikipedia, Open-Kollektor-Ausgang.

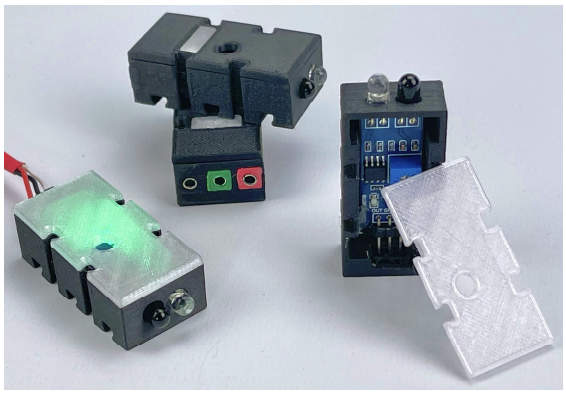


Abb. 4: IR-Hindernissensoren mit verschiedenen Abdeckungen und Anschlussmöglichkeiten

Hall-Effekt-Sensor

Wie bei dem oben beschriebenen Sensor finden wir auch hier den LM393 als Differenzverstärker. Dadurch ist der Schaltimpuls des Ausgangssignals schön steil und kann wegen des fehlenden Schaltrauschens auch direkt als digitaler Takt verwendet werden. Wie bereits erwähnt ist dies eine sehr gute Eigenschaft all dieser berührungslosen Schalter.

Der Hall-Effekt-Sensor[3] verwendet den Hall-Sensor 3144, der grundsätzlich mit einer Versorgungsspannung zwischen 4,5 und 24 Volt funktioniert. Daher kann er wie der oben vorgestellte IR-Hindernissensor ohne Probleme mit 9 Volt betrieben werden. Die Sensorplatine hat nur eine (grüne) LED, die bei Erkennung aufleuchtet, so dass es nicht notwendig ist, wie beim IR-Hindernissensor eine permanent leuchtende LED zu deaktivieren. Der 4,7 kΩ-Vorwiderstand der Erkennungs-LED muss bei Verwendung von 9 Volt nicht geändert werden. Dieser Sensor muss also eigentlich überhaupt nicht verändert werden und kann zudem mit nur minimalen Änderungen am Gehäuse in dem für den IR-Hindernissensor entworfenen und 3D-gedruckten Gehäuse untergebracht werden. Lediglich der Deckel unterscheidet sich, da hier ein relativ hochstehendes 10-Hub-Potentiometer zum Einstellen dieses Sensors angebracht ist.

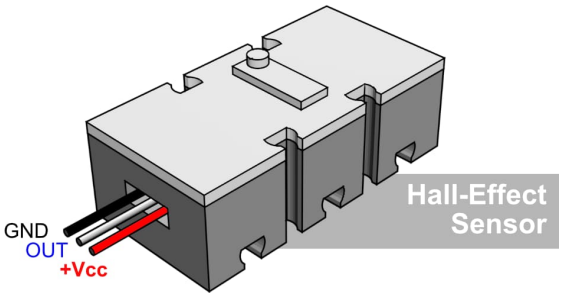


Abb. 5: Anschlussreihenfolge der Sensoren bei Verwendung eines 3-poligen Flachsteckers

Beim Anschluss mit einem 3-poligen Servokabel muss jedoch darauf geachtet werden, dass der Signalausgang am mittleren Pin herauskommt (Abb. 5). Bei einer Spannungsversorgung von 9 Volt beträgt das Ausgangssignal im Ruhezustand etwa 7,4 Volt. Dieses fällt auf 0 Volt ab, wenn ein Magnetfeld erkannt wird, zum Beispiel wenn ein Magnet am Sensor vorbeigeführt wird. Der Sensor hat also, wie alle anderen hier besprochenen 9-Volt-Sensoren, ein invertiertes Schaltverhalten, das der negativen Logik der Silberlinge entspricht. Der Ausgang des Sensors kann direkt als Eingangssignal für diese klassischen Elektronikmodule verwendet werden.

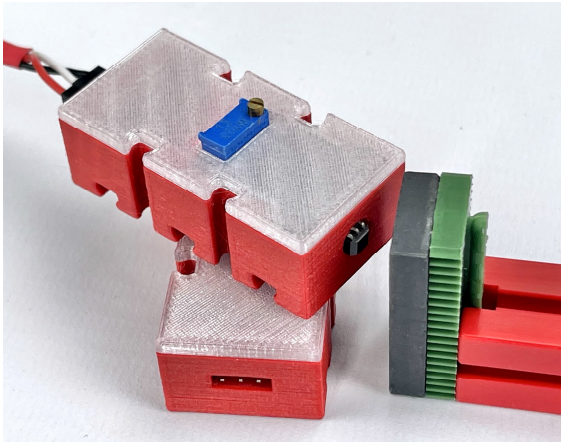


Abb. 6: Der Hall-Effekt-Sensor

[3] Siehe Wikipedia, Hall-Effekt.

LDR-Modul

Die lichtempfindliche Widerstandsplatine ist ebenfalls mit einem LM393-Differenzverstärker ausgestattet und passt in das gleiche Gehäuse wie die anderen oben beschriebenen Sensorplatinen.

Bei diesem Sensor müssen die gleichen Änderungen an den Lastwiderständen der LEDs vorgenommen werden, bevor er mit 9 Volt verwendet werden kann: Der Vorwiderstand der permanent leuchtenden LED kann komplett entfernt werden. Der 1 kΩ Vorwiderstand muss durch einen 10 kΩ SMD-Vorwiderstand ersetzt werden: Nicht nur, um ein Durchbrennen der LED zu verhindern, sondern in diesem Fall auch, damit der Open-Kollektor-Ausgang des Differenzverstärkers sich besser für die Verwendung an den Eingängen der Silberlinge eignet.

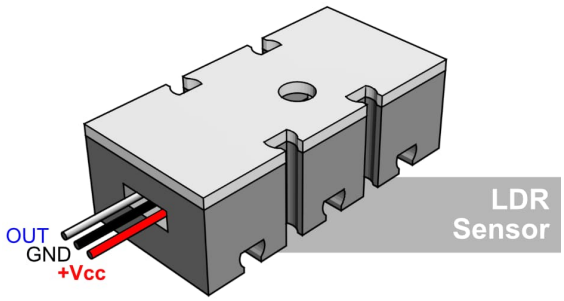


Abb. 7: Anschlussreihenfolge der Sensoren bei Verwendung eines 3-poligen Flachsteckers

Beim Anschluss mit einem 3-poligen Servokabel muss bei diesem Sensor darauf geachtet werden, dass der Nullleiter (GND) auf dem mittleren Pin herauskommt (Abb. 7). Bei der Versorgung mit 9 Volt entspricht das Ausgangssignal praktisch der Versorgungsspannung, solange genügend Licht auf den Sensor fällt. Der Signalpegel fällt beim Dimmen auf 0 Volt ab. Damit hat der Sensor, wie alle anderen hier besprochenen 9-Volt-Sensoren, ein für die meisten Anwendungen invertiertes Schaltverhalten, das der negativen Logik der Silberlinge entspricht. Auch für diesen Sensor wird kein vorgeschalteter Differenzverstärker benötigt, so dass er direkt als Eingangssignal für die verschiedenen klassischen Silberlinge verwendet werden kann.

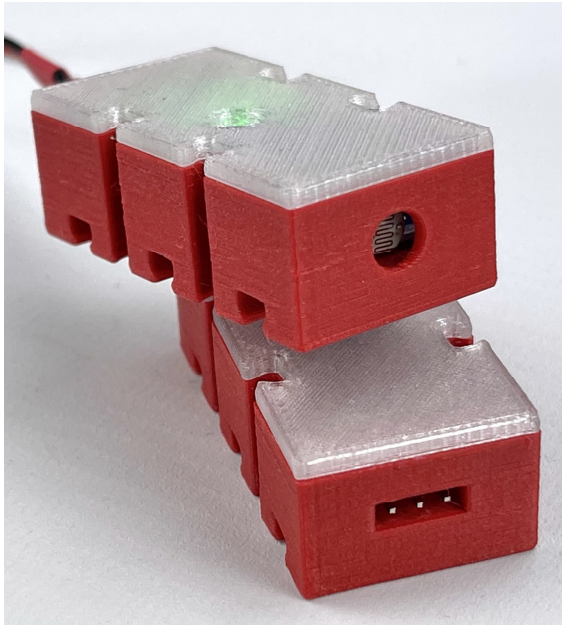


Abb. 8: LDR-Sensor

Bewegungsmelder

Zum Schluss werfen wir noch einen Blick auf den Infrarot-PIR-Bewegungssensor HC-SR501. Ein Passiv-Infrarot-Sensor (PIR-Sensor[4]) ist ein elektronischer Sensor, der Infrarotlicht (IR) misst, das von Objekten im Sichtfeld des Sensors ausgesendet wird. Der Sensor kann auf diese Weise Bewegungen erkennen. Eine naheliegende Anwendung ist z. B., bei einer Ausstellung ein Modell automatisch auslösen zu lassen, sobald sich ein potentieller Zuschauer nähert. Das Modul selbst verfügt bereits über einen Timer, mit dem man in einem solchen Fall die Dauer der Auslösung einstellen kann. Ein wertvolles Mono-Flop-Modul (30816) oder eine Warteschleife bzw. ein Zähler im Falle einer Softwaresteuerung muss hier also nicht vorgesehen werden.

4 Siehe Wikipedia, Bewegungsmelder.

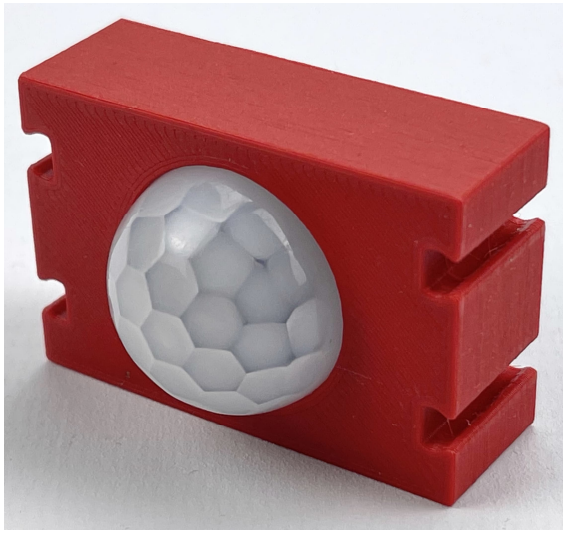


Abb. 9: Infrarot-Bewegungsmelder HC-SR501

Die Abmessungen dieses Sensors unterscheiden sich von den oben genannten Sensoren, sodass ich ein eigenes Gehäuse entwickeln musste. Dieses habe ich recht einfach gehalten, sodass die Anschlussdaten und Einstellmöglichkeiten auf einer Karte in der offenen Rückseite untergebracht werden können (Abb. 10). Die Empfindlichkeit (Erfassungsabstand) und die Zeit, für die das Ausgangssignal nach der Erfassung aktiv bleibt, sind über Öffnungen in der Seite des Gehäuses mit einem kleinen Schraubendreher einstellbar.

Abb. 10: Die Anschlusshinweise auf der Rückseite im Inneren des Gehäuses

Durch Umstecken eines Jumpers kann man zwischen einem Erkennungsimpuls bei wiederholter (innerhalb der Timeout-Zeit) Erkennung einer Bewegung und separaten Impulsmeldungen wählen. Für die meisten Anwendungen scheint mir der „Retrigger"-Modus nahe zu liegen, daher habe ich den Jumper zunächst an seinem Platz gelassen.

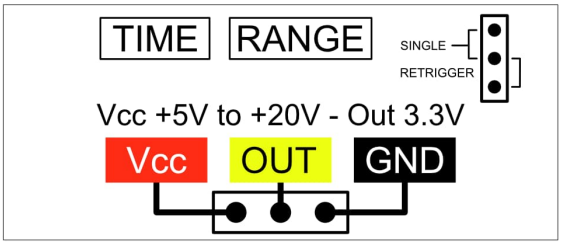


Abb. 11: Schaltplan des PIR-Bewegungsmelders

Dieser Sensor muss nicht für den Betrieb mit 9 Volt modifiziert werden. Der Spannungsregler kann eine Versorgungsspannung von bis zu 20 Volt verarbeiten. Intern arbeitet er jedoch nur mit 3,3 Volt; dies ist auch der Spannungspegel des Ausgangssignals. Geschaltet über den Grundbaustein (30813) mit reduzierter Vergleichsspannung (Widerstand von 10 kΩ zwischen Bus 6 und Masse) an E2 [2], eignet sich auch dieser Sensor, wie alle anderen in dieser Serie besprochenen berührungslosen Schalter, hervorragend für den Einsatz in Silberling-Schaltungen (Abb. 13).

Abschließende Bemerkungen

Vom IR-Hindernissensor habe ich einige Exemplare gebaut, die mit den herkömmlichen fischertechnik-Kabeln mit 2,5-mm-Steckern angeschlossen werden können. Von den anderen Sensoren habe ich einfach Exemplare gebaut, die auf der Sensorseite mit einem Standard-Servokabel mit 3-poligem Flachstecker angeschlossen werden können. Allerdings ist die Anschlussreihenfolge der drei Stifte in diesem Stecker für jeden Sensor unterschiedlich. Zum Glück lassen sich die Drähte im Stecker eines vorgefertigten Servokabels leicht umstecken. In Teil 2 dieser Beitragsreihe (in dieser Ausgabe der ft:pedia) wurden die Möglichkeiten dazu bereits besprochen.

Ich habe mit einem geschlossenen Deckel mit transparentem Diffusor experimentiert

und mich schließlich für einen transparenten Deckel entschieden, um das Licht der Sensor-LED sichtbar zu machen. Der einzige Sensor, bei dem dies störend sein könnte, ist das LDR-Modul. Da die Empfindlichkeit dieses Moduls aber einstellbar ist, war es in der Praxis noch nicht notwendig, eine Blende auf der Innenseite des Deckels oder hinter dem LDR zu anzubringen, um einfallendes Störlicht durch den Deckel zu verhindern.

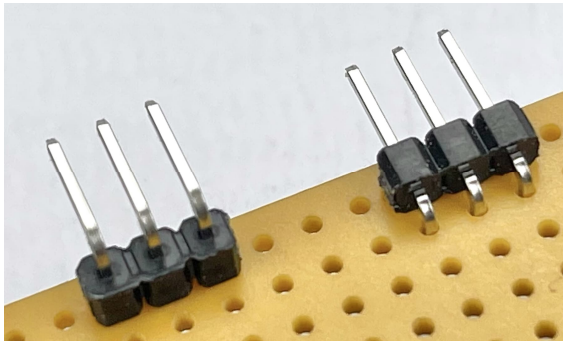


Abb. 12: Zwei Arten von rechtwinkligen Steckern. Nur der rechte kann in der Mitte von 15 mm Höhe zentriert werden

Am besten ist es, wenn die Sensoren in einem möglichst niedrigen Gehäuse untergebracht werden können und die Öffnung für den Anschluss schön zentriert ist. Bei den im niedrigen Gehäuse (Höhe 15 mm) untergebrachten Sensorplatinen mussten daher die abgewinkelten Stecker ersetzt werden. Die meisten Sensormodule werden nämlich leider mit geraden oder abgewinkelten 3-poligen Steckern ausgeliefert, die hoch über die Sensorplatine hinausragen, wie z. B. der Stecker links in Abb. 12. Diese habe ich entfernt und durch die niedrigeren so genannten ‚reverse bending'-Stecker (horizontaler Stecker rechts in Abb. 12) ersetzt, um Einbauhöhe im Gehäuse zu sparen. Hier hilft das Heißluftlöten; man kann aber auch erst den schwarzen Kunststoff wegschneiden und dann die einzelnen Stifte mit einem normalen Lötkolben auslöten.

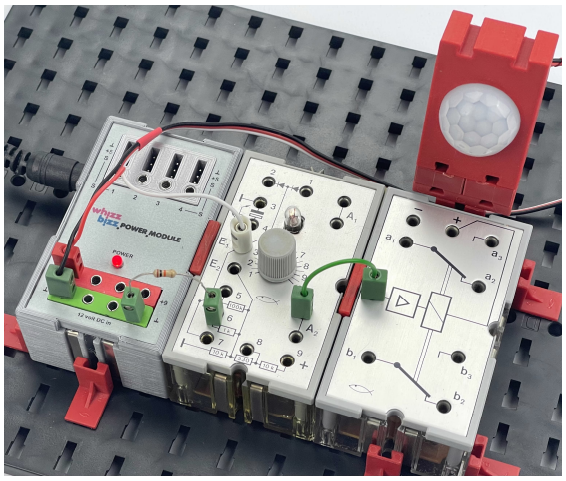


Abb. 13: Erkennung des 3,3-Volt-Ausgangs durch den Grundbaustein mit 10 kΩ-Widerstand zwischen Bus 6 und Masse

Alle in diesem Beitrag besprochenen Sensoren können direkt mit 9 Volt versorgt werden. Aber auch bei einer Versorgungsspannung von nur 5 Volt, zum Beispiel mit einer der im Teil 1 der Beitragsserie besprochenen Spannungsversorgungen [1], ist es für die Sensoren mit eingebautem Differenzverstärker (LM393) möglich, sie in mit den Silberlingen aufgebauten Schaltungen zu verwenden.

IR-, LDR- und Hall-Sensor eignen sich ideal als End- oder Durchgangssensoren in Fabriksimulationen oder Aufzugsmodellen. Ein großer Vorteil ist, dass bei der Arbeit mit Silberlingen oder Elektronikmodulen keine zusätzlichen Differenzverstärker wie der Grundbaustein für die herkömmliche Lichtschranke benötigt werden.

Der PIR-Sensor scheint für ein Modell gut geeignet, das durch menschliche oder tierische Bewegungen gesteuert werden soll. Ein TXT-Controller hat offensichtlich kein Problem, die relativ niedrige Signalspannung dieses Moduls zu erfassen. Um ihn aber als Steuersignal für die Silberlinge zu verwenden, wird ein Grundbaustein benötigt. Ein möglicher Schaltplan dafür wurde bereits im vorherigen Abschnitt vorgestellt.

Weitere Informationen über die verschiedenen Sensoren und meine Experimente findet ihr auf meiner Website [4]. Die Entwürfe der gezeigten Sensorgehäuse und Deckel können von jedem, der einen 3D-Drucker besitzt, ausgedruckt werden. Dort gibt es auch einen Link zu einem Video, das ich darüber erstellt habe. Ich bin offen für Tipps über weitere Anwendungen der Sensoren oder Fragen und jederzeit bereit, diese Sensoren zu bauen oder bei Fragen zum Selbstbau zu helfen.

Quellen

[1] Arnoud van Delden: *Kontaktlose Schalter*. ft:pedia 3/2023, S. 49–53.

[2] Arnoud van Delden: *Kontaktloste Schalter (Teil 2)*, in dieser Ausgabe der ft:pedia.

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

[4] Projektseite WhizzBizz.

Computing

ftDuino32, der große Bruder des ftDuino

Till Harbaum

Der ftDuino [1] hat sich über die Jahre seinen festen Platz im fischertechnik-Universum gesichert. Aber die Arduino-Welt steht nicht still und auch am ftDuino zieht der Fortschritt nicht vorbei. Mit dem ftDuino32 habe ich einmal ausprobiert, wie ein technisch fortgeschrittener ftDuino aussehen könnte.

Der klassische ftDuino bildet nach wie vor das Fundament. Er basiert auf dem Arduino Leonardo, einem der ersten und einfachsten Arduinos, und es sind nicht viele Bauteile nötig, um ihn fischertechnik-kompatibel zu machen. In der neuesten Version geht es selbst in der Variante mit eigebautem Display sehr entspannt im Gehäuse zu (Abb. 2 links). Er wird nach wie vor gebaut und hat vor allem in Schulen seinen Platz gefunden.

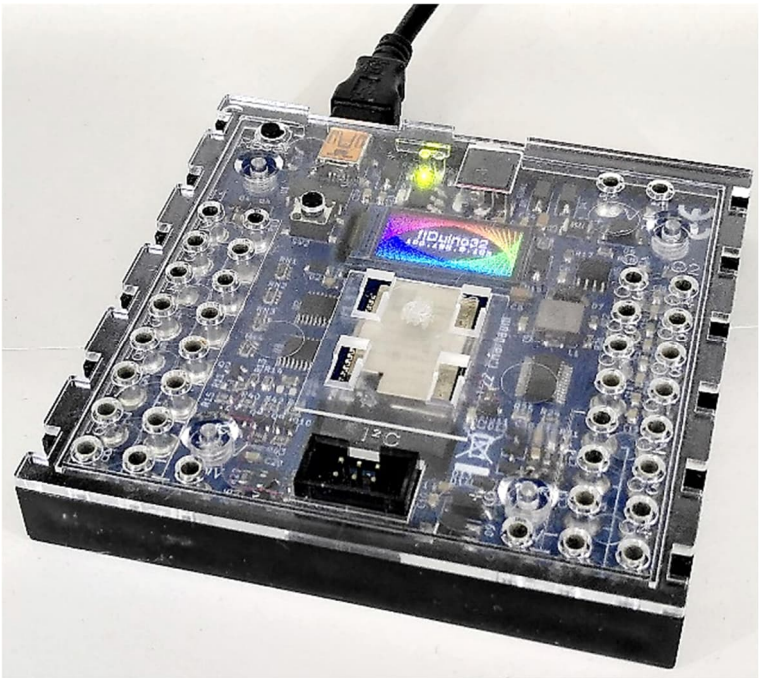


Abb. 1: ftDuino32

Für einen Nachfolger lag es natürlich nahe, auf einen deutlich leistungsfähigeren Controller zu wechseln. Ein offensichtlicher Kandidat dafür ist der ESP32, der dem ftDuino32 seinen Namen gibt. Mit dieser Idee bin ich nicht alleine; zum Beispiel basiert auch der ftSwarm-Controller [2] auf dem ESP32.

Atmega32u4 gegen ESP32

Im ftDuino werkelt ein Atmega32u4. Dieser mit 16 MHz getaktete 8-Bit-Mikrocontroller verfügt neben 32 KB Flash-Speicher und 2,5 KB RAM über etwas Peripherie und vor allem über eine in den Chip integrierte USB-Schnittstelle. Trotz diesen schlichten inneren Werten eignet sich der ftDuino auch für komplexere Projekt, da er z. B. kein eigenes Betriebssystem benötigt. Das hat zugleich den Vorteil, dass der ftDuino unglaublich schnell reagieren kann. Reaktionszeiten im Mikrosekundenbereich sind seine Spezialität und einfache, aber anspruchsvolle Steuer- und Regelaufgaben in der fischertechnik-Welt lassen sich mit dem ftDuino elegant realisieren.

Der ESP32 kommt im direkten Vergleich geradezu riesig daher.

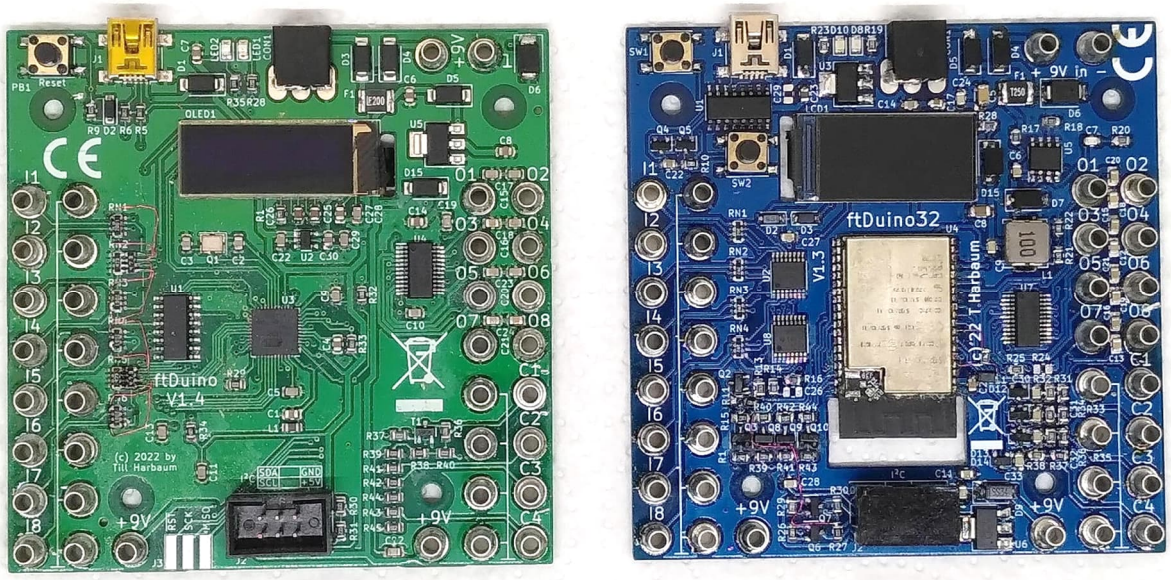


Abb. 2: ftDuino (links) vs. ftDuino32 (rechts)

Flash- und RAM-Speicher reichen je nach Variante in den Megabytes-Bereich. Dazu gesellen sich eine mit 240 MHz getaktete 32 Bit-Dual-Core-CPU sowie WLAN und Bluetooth. Daraus resultiert, dass der ESP32 sich nicht nur über die Arduino-IDE nutzen lässt, sondern auch über viele Ressourcen-hungrige Umgebungen direkt auf dem ESP32 wie zum Beispiel das anfängerfreundliche Micropython [3] oder Lua [4].

Aber natürlich sind vor allem die Drahtlos-Fähigkeit des ESP32 für den Betrieb im fischertechnik-Zimmer interessant. Viele Beispiele für den ESP32 zeigen, wie man einen kleinen Web-Server auf dem Gerät baut und dann z. B. per Smartphone drauf zugreift. Mit etwas mehr Aufwand lässt sich der ESP32 aber auch mit Amazons Alexa koppeln oder in andere Heimautomations-systeme integrieren.

Im fischertechnik-Umfeld ist zudem die Bluetooth-Fähigkeit spannend. fischer-technik selbst nutzt Bluetooth für den BT-Smart-Controller, die Bluetooth-Fern-bedienung und den Early-Coding Robbie. Da deren Kommunikationsprotokolle bekannt sind bzw. sich mit wenig Aufwand analysieren lassen, kann man den ESP32 mit den fischertechnik-Lösungen verbin-den. Dabei kann der ESP32 sogar beide Rollen übernehmen, sich also entweder gegenüber der PC-Software als fischer-technik-Gerät ausgeben oder aber selbst die Rolle der Software übernehmen und das fischertechnik-Gerät steuern. Dem ftDuino eröffnen sich mit dem ESP32 also unzählige neue Möglichkeiten.

Aber wie kommt der ESP32 in den ftDuino? Auf den ersten Blick scheint das Prozessor-Upgrade einfach. ESP32-Platinen aus dem Arduino-Umfeld sind nicht sehr aufwändig und so sollte sich der ESP32 doch mit wenig Aufwand in einen ftDuino verpflanzen lassen…

Die erste große Hürde ist die Antenne. Für sein Funkmodul besitzt der ESP32 eine Antenne, die man üblicherweise am Geräte-rand „ins Freie“ ragen lässt. Im ftDuino-Gehäuse ist dafür leider kein Platz, so dass der ESP32 in die Mitte der Platine kam und für seine Antenne einen großzügigen Ausschnitt auf der Platine erhielt, um die Funkkommunikation nicht durch die Kupferleiterbahnen zu stören.

Als zweite Hürde stellte sich die Strom-versorgung heraus. Während sich der kleine

Atmega mit wenigen Milliampere begnügt, kann sich der ESP32 bei vollem Funkbetrieb auch mal 250 mA, in Spitzen sogar 400 mA genehmigen. Damit gerät man an die Grenzen dessen, was ein kleiner Längsregler zu leisten vermag. Regelt man die fischertechnik-üblichen 9 V auf die vom ESP32 bevorzugten 3,3 V runter, dann müssen die verbleibenden 5,7 V im Regler verschwinden. Bei 250 mA ist das mehr als ein Watt, das als Wärme am Regler frei wird. Die energie- und damit abwärmesparende Variante in Form eines Schaltreglers ist zwar weder übermäßig teuer noch aufwändig. Aber der Platzbedarf mit Dioden, Spulen und Regelchip steigt spürbar. Dazu kommt, dass der ESP32 selbst kein USB-Protokoll spricht, sondern einen – wieder weder teuren noch aufwändigen – USB-UART-Brückenbaustein benötigt, um USB zu verstehen. Auch das erfordert wieder ein paar Quadratzentimeter auf der Platine. Der gerade noch so großzügig erscheinende Platinenplatz ist damit schon deutlich knapper, und fischertechnik-Anschlüsse hat unser ftDuino32 damit noch nicht. Deren Ansteuerung übernehmen aus dem ftDuino entliehene Schaltungen, wobei die 3,3 V des ESP32 etwas Extra-Aufwand gegenüber den 5 V des ftDuino erfordern.

Der Platinenplatz reichte am Ende gerade so. Aber einen Wunsch hatte ich noch: Statt des kleinen Schwarz-Weiß-OLED des ftDuino wäre ein etwas größeres farbiges TFT-LCD doch fein. Das Display-Upgrade ließ sich auch noch in das inzwischen recht enge Gehäuse quetschen. Aber damit ist auch langsam Schluss. Ein zusätzlicher Taster musste noch untergebracht werden, denn der ESP32 benötigt zwei Tasten, um in jedem erdenklichen Zustand wieder sicher in die Grundkonfiguration versetzt zu werden. Das Resultat ist in Abb. 3 in der 3D-Darstellung des CAD-Programmes zu sehen.

Was die fischertechnik-Grundfunktionen angeht, läuft auf dem ESP32 einiges anders. Der ftDuino, so simpel er ist, ist unglaublich schnell und kann sich problemlos nebenbei um alle Ein- und Ausgänge kümmern. Der ESP32 dagegen ist viel häufiger „abgelenkt“: WLAN und Bluetooth erfordern viele nebenläufige Tätigkeiten, und der ESP32 muss sich immer mal wieder um die Funkschnittstellen kümmern, auch wenn er eigentlich gerade ein sauberes analoges PWM-Signal für einen Motorausgang erzeugen sollte. Aus diesem Grund werden im ftDuino32 Motortreiber eingesetzt, die einige der zeitkritischen Dinge wie die Erzeugung der analogen PWM-Signale direkt im Chip unterstützen und auf die permanente Mitarbeit des ESP32 nicht angewiesen sind. Passende Bibliotheken für diese Chips existierten bereits, und was fehlte, wurde entsprechend hinzugefügt.

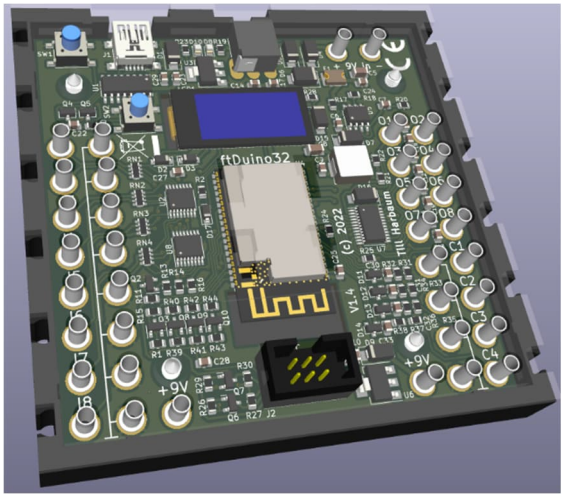


Abb. 3: CAD-Ansicht des ftDuino32

Neben der elektrischen Funktion ist die Software die zweite große Baustelle auf dem Weg zum ftDuino32. Der ESP32 lässt sich tatsächlich mit der Arduino-IDE programmieren und verhält sich dabei aus Programmierersicht dem Arduino sehr ähnlich. Aber spätestens bei komplexen Internet-Anwendungen wird es für Einsteiger schnell schwierig. Hier drängt sich Micropython geradezu auf. Daher habe ich eine Anpassung von Micropython an den ftDuino32 vorgenommen. Micropython bietet bereits ab Werk Unterstützung für viele gängige Hardware, und die meisten

Funktionen des ESP32 lassen sich sofort nutzen. Alle Ein- und Ausgänge des ftDuino32 lassen sich aus Micropython ebenso ansprechen wie die Bluetooth- und WLAN-Funktionen. Einige Kleinigkeiten fehlten mir aber. So verfügt der ESP32 über sogenannte Pulse-Counter (PCNT), mit denen er diverse Impuls-Sensoren auswerten kann. In der Micropython-Welt scheint der Bedarf dafür gering zu sein, so dass Micropython die PCNT-Funktion nicht unterstützt. Nun erzeugen aber die TXT-Encodermotoren genau solche Impuls-Signale, und es wäre natürlich toll, wenn sich diese unter Micropython mit der passenden Hardware bequem auswerten ließen. Ich habe also eine entsprechende Erweiterung für Micropython geschrieben.

Eine andere Änderung betraf das Funkmodul. Einige fischertechnik-Geräte und -Apps sind sehr wählerisch, was ihre akzeptierten Kommunikationspartner angeht, und sie verbinden sich nur mit Gegenseiten, die eine Kennung der Firma LNT tragen. Die nötigen Schnittstellen, um die Kennung des ESP32-Funkmoduls entsprechend anzupassen, mussten ebenfalls in Micropython nachgerüstet werden. Die angepasste Version sowie die nötigen Patches gibt es unter [5] zum Download. Man kann aber auch mit geringen Einschränkungen eine der Standardeditionen von Micropython für den ESP32 nutzen [6].

Wie die Arduino-IDE für Arduino gibt es auch für Micropython sogenannte IDEs, also PC-Software, die eine bequeme

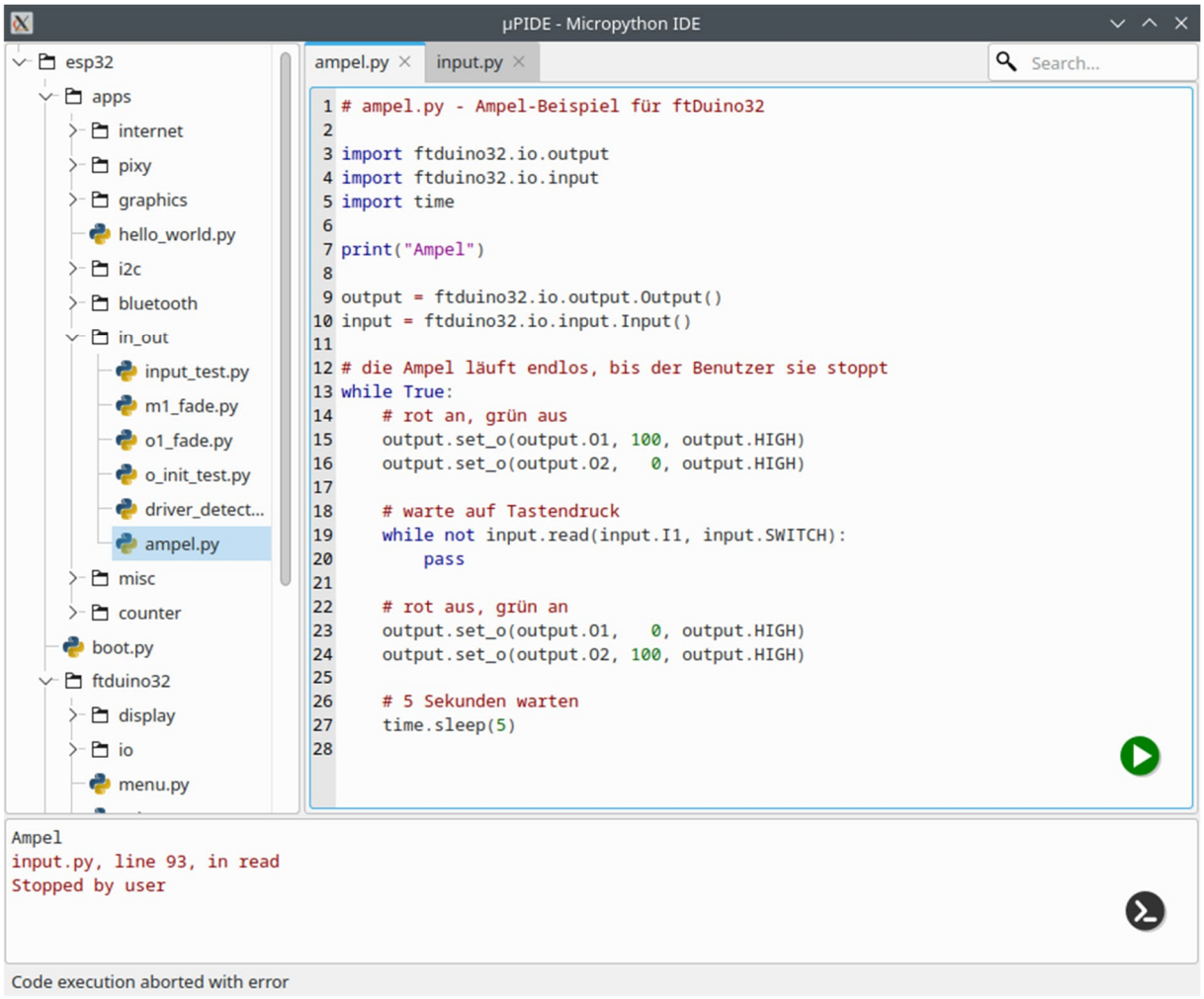


Abb. 4: µPIDE

Programmierung ermöglichen und alle Funktionen zum Umgang mit einem Micropython-Gerät unter einer Haube vereinen. Beispiele für solche IDEs sind Thonny [7], uPyCraft [8] oder PyCharm [9]. Für alle finden sich unzählige Tutorials im Internet, die einen Einstieg mit Micropython-Geräten beschreiben und sich auch auf den ftDuino32 anwenden lassen. Direkt für den ftDuino32 wurde die µPIDE-IDE [10] entwickelt. Allen IDEs gemeinsam ist, dass sie einen Zugriff auf die auf dem ftDuino32 gespeicherten Programmdaten erlauben. Es können beliebige Python-Programme angelegt, verändert und ausgeführt werden. Dabei bleiben alle Daten auf dem ftDuino32, und es können sogar mehrere Programme parallel installiert und am Gerät aus einem grafischen Menü ausgewählt werden. Im Gegensatz zum ftDuino hat man also beim ftDuino32 immer alle Daten auf dem Gerät dabei, was gerade im Schul- oder Ausbildungseinsatz praktisch ist, da keine Daten auf dem PC verbleiben und man mit dem Gerät jederzeit mit allen Daten an einem anderen PC weiterarbeiten kann.

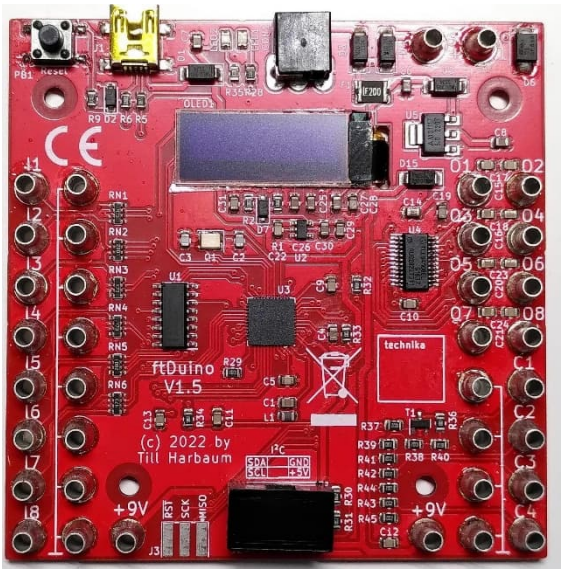


Abb. 5: ftDuino in der technika-Version

Die Hardware ist seit einiger Zeit fast fertig; einige kosmetische Änderungen werden noch kommen, aber im Wesentlichen bleibt das Gerät, wie es ist. Die bisherigen Daten stehen auf Github frei zur Verfügung, sodass sich jeder an einem Nachbau versuchen kann. Ob und wann das Gerät in Serie geht, ist noch unklar; ein Nachbau mit Schülern z. B. im Rahmen der Karlsruher Technik-Initiative [11] ist aber fest eingeplant.

Spannend wird es, wenn fischertechnik mit dem neuen RX-Controller auf den Markt kommt. Den knappen Informationen zu diesem Gerät nach ist es dem ftDuino32 sehr ähnlich. Es wird interessant sein zu sehen, wie fischertechnik selbst ein Micropython-Gerät umsetzt und wie sich die beiden Geräte im Vergleich schlagen.

Quellen

[1] Till Harbaum: ftDuino-Homepage.

[2] Stefan Fuss, Christian Bergschneider: *ftSwarm.* Auf Github

[3] Micropython

[4] Espressif Systems: *Lua-RTOS for ESP32*.

[5] Till Harbaum: *ftDuino32.* Auf Github.

[6] *Getting started with MicroPython on the ESP32* auf microphyton.org.

[7] Thonny-IDE

[8] uPyCraft-IDE auf gitbooks.io.

[9] JetBrains: *PyCharm* (IDE).

[10] Till Harbaum: *µPIDE - Micropython IDE*. Auf Github.

[11] technika | Karlsruher Technik-Initiative

Computing

NFC für TXT

Axel Chobe

Wenn man im Internet nach ‚NFC für fischertechnik' sucht, stößt man nur auf den Hinweis, dass es so etwas für die Lernfabrik 4.0 gibt. Dazu stellt fischertechnik sogar eigene Werkstücke bereit (174622 bis 174627). Das hat mich inspiriert, ein NFC-System für meine Zwecke zu erstellen. Allerdings musste ich einen Umweg nehmen und einen Arduino-Nano für die entsprechenden NFC-Leseeinheit einsetzen. Für die Auswertung am TXT habe ich zwei Lösungsansätze gefunden: Zum einen über drei Eingänge, womit es möglich ist, bis zu 7 verschiedene NFC-Tags auszuwerten. Die zweite Lösung funktioniert mittels PWM an nur einem Ausgang. Dabei wird jedem NFC-Tag ein unterschiedliches PWM-Signal zugewiesen, welches im TXT dann als entsprechender Spannungswert ausgelesen wird.

Begriffsbestimmung

NFC steht für *Near Field Communication* (Nahfeldkommunikation), da diese Funktechnik Daten über eine kurze Distanz von wenigen Zentimetern überträgt. NFC ist eine RFID-Technologie *(radio-frequency identification),* geht aber insofern weiter, da hierbei der NFC-Tag auch beschrieben werden kann.

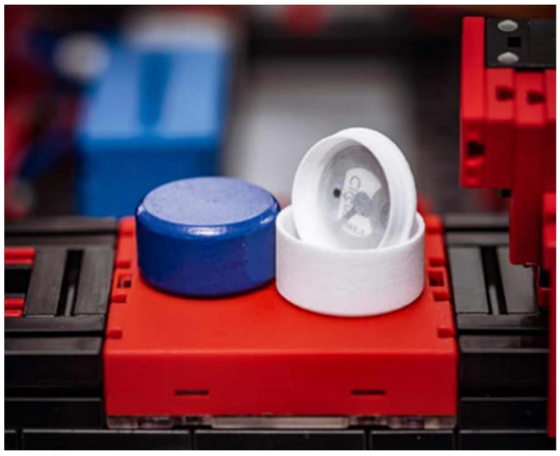


Abb. 1: NFC-Tag in einem fischertechnik-Modell

Es gibt NFC-Tags in vielen unterschiedlichen Formen wie Aufkleber, Anhänger, Karten oder Armbänder (Abb. 1).

Sie enthalten einen Mikrochip, der eine kleine Datenmenge, wie z. B. eine URL oder eine digitale Visitenkarte, und eine eindeutige Kennung speichern kann. Heutzutage ist es möglich, einen NFC-Tag mit dem Handy auszulesen.

Hardware

Als Hardware-Komponenten habe ich einen Arduino Nano, eine NFC-Leseeinheit RC522, einen PWM2Volt-Wandler und selbstklebende RFID-Tags verwendet (Abb. 2-5).

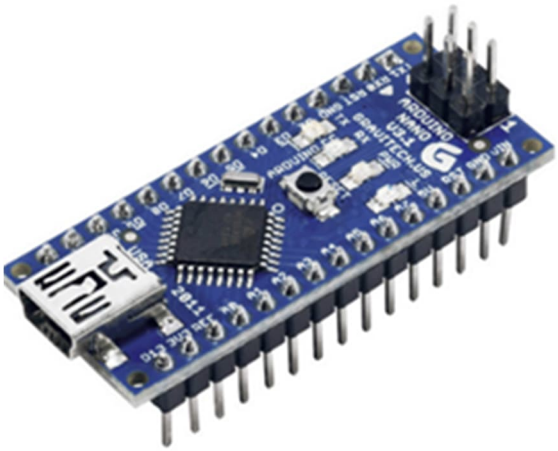


Abb. 2: Arduino Nano, ca. 7 €

Abb. 3: NFC-Leseeinheit RFID-RC522, ca. 7 € (wird meist mit 2 Tags verkauft).

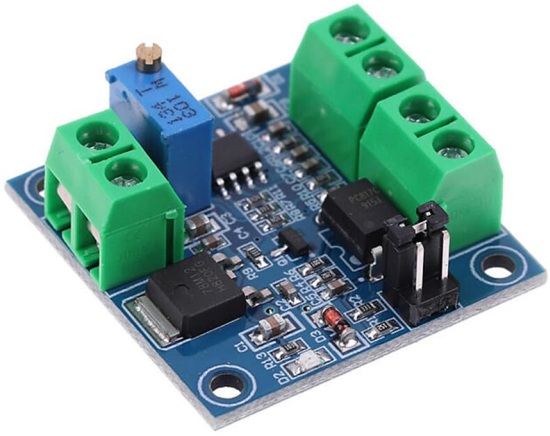


Abb. 4: PWM2Volt-Wandler, ca. 4 €. Wandelt das PWM-Signal in eine Spannung um.

Abb. 5: BERRYBASE 125452 RFID/NFC Tags, selbstklebend, farbig, 6 Stück, ca. 4 €.

Der Anschluss am Arduino erfolgt so:

RFID-RC522	Arduino Nano
SDA	D10
SCK	D13
MOSI	D11
MISO	D12
GND	GND
RST	D9
3,3V	3V3

Tab. 1: Anschlussbelegung

Variante 1: Auswertung über drei Eingänge

Zuerst wurde der Arduino mit dem RFID-RC522 laut Tabelle verbunden. An den Ausgängen D5 bis D7 sind zur Kontrolle noch drei LEDs angeschlossen, die dem Tag entsprechend leuchten.

Für die Arduino-Software wird dann das Programm aus Abb. 6 benutzt. Hierbei besteht auch noch die Möglichkeit, sich die ID auf der Konsole anzeigen zu lassen.

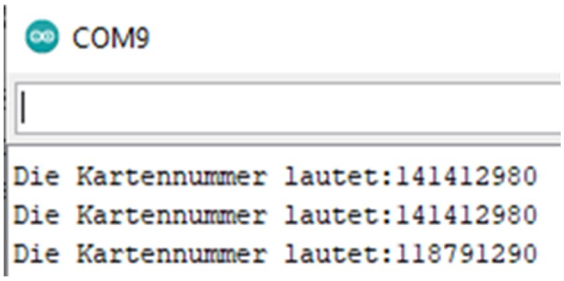


Abb. 6: Konsolenausgabe

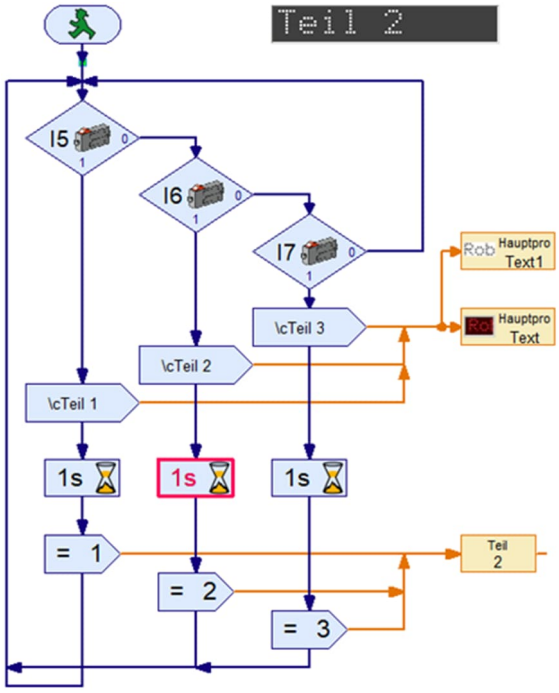


Abb. 7: ROBO Pro-Programm zu Variante 1

Die Arduino-Ausgänge D5 bis D7 und GND wurden direkt mit dem TXT verbunden. Die Eingangsart ist auf 10 V festgelegt. Mit dem ROBO Pro-Programm in Abb. 7 kann nun das Werkstück erkannt werden, um dann weitere Aktionen durchzuführen. Abb. 8 zeigt den Schaltungsaufbau.

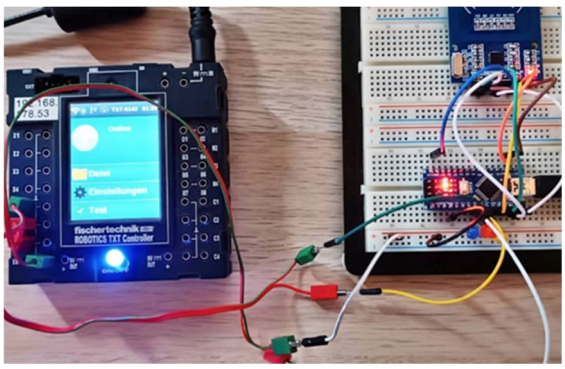


Abb. 8: Der Schaltungsaufbau

```
#include <MFRC522.h>
#define SS_PIN 10 // SDA an Pin 10
#define RST_PIN 9 // RST an Pin 9
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup()
{
Serial.begin(9600);
SPI.begin();
mfrc522.PCD_Init();
pinMode (6, OUTPUT);
pinMode (5, OUTPUT);
pinMode (7, OUTPUT);
}
void loop()
{
if ( ! mfrc522.PICC_IsNewCardPresent())
{
return; // gehe weiter...
}
if ( ! mfrc522.PICC_ReadCardSerial())
{
return; //
}
long code=0;
for (byte i = 0; i < mfrc522.uid.size; i++)
{
code=((code+mfrc522.uid.uidByte[i])*10);
}
Serial.print("Die Kartennummer lautet:");
Serial.println(code);
if (code==118791290)
{
digitalWrite (6, HIGH);
delay (2000);
digitalWrite (6, LOW); }
if (code==158791290)
{
digitalWrite (5, HIGH);
delay (2000);
digitalWrite (5, LOW);}
if (code==141412980)
{
digitalWrite (7, HIGH);
delay (2000);
digitalWrite (7, LOW);
}
}
```

Abb. 9: Arduino-Sketch zu Variante 1

Nach dem letzten Test geht es um das Zusammenbauen. Als Gehäusegrundlage habe ich die Vorlage [1] benutzt (Abb. 10).

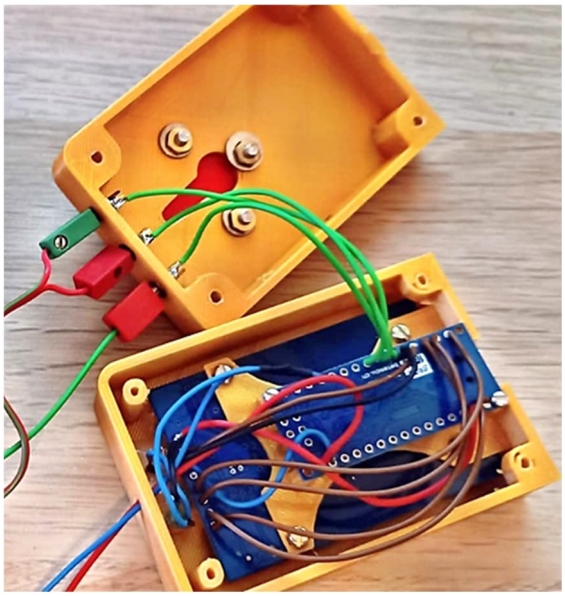


Abb. 10: 3D-Druck-Gehäuse

Für die Befestigung des Gehäuses an fischertechnik-Teilen wurde eine Bauplatte 30 × 30 4Z (38259) mit Senkkopfschrauben angebracht (Abb. 11).

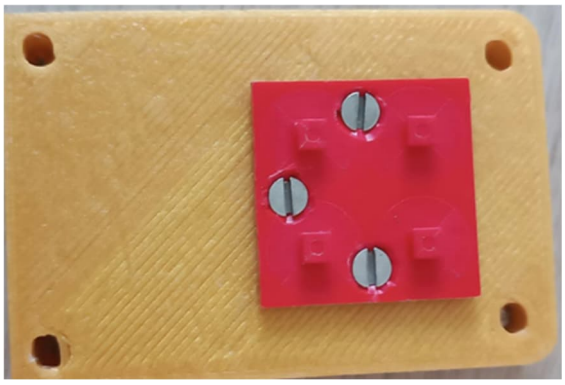


Abb. 11: Verbindung zur fischertechnik-Welt

Während die Kabel für die Stromversorgung direkt herausgeführt werden, sind die drei Ausgänge als Buchsen ausgeführt. Dafür habe ich passende Hohlnieten eingesetzt und von hinten auf eine Unterlegscheibe gelötet (Abb. 12).

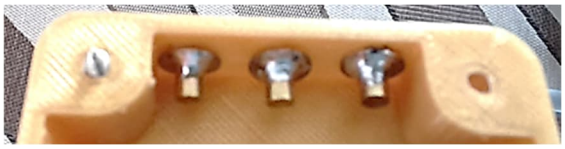


Abb. 12: Eingebaute Hohlnieten

Um die Funktion real zu testen habe ich eine V-Rohrhülse 30 × 20 (35409) mit den entsprechenden Deckeln benutzt. Von unten passt genau ein selbstklebender NFC-Tag darauf. Oben wurde die Beschriftung des Werkstückes und an der Seite die Identifizierungsnummer des Tags aufgeklebt (Abb. 13 und 14).

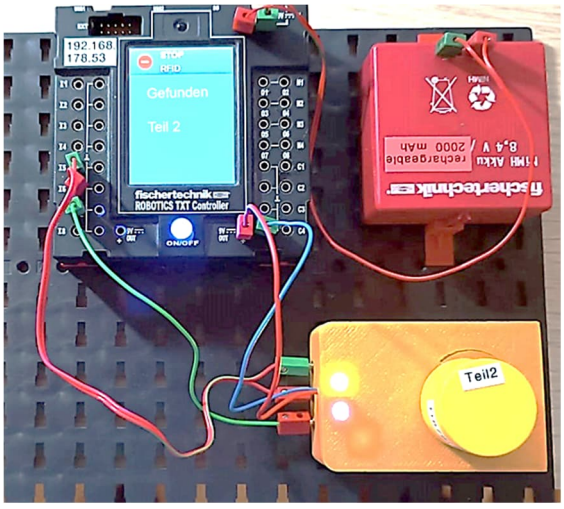


Abb. 13: Der komplette Testaufbau

Abb. 14: Beschriftungen und NFC-Tag

Ein Beispielvideo kann unter [2] aufgerufen werden.

Variante 2: Auswertung über Spannungswert

In der zweiten Variante wird das PWM-Signal am Ausgang des Arduino mit der ROBO Pro-Software ausgewertet. Dabei sind zwei Dinge zu beachten: Erstens muss der Arduino-Ausgang für das PWM-Signal geeignet sein (betrifft die rot umrandeten Ausgänge). Zweitens ist der TXT nicht in

der Lage, ein PWM-Signal auszuwerten. Daher kommt ein kleiner Wandler, wie z. B. der LC-LM358-PWM2V zum Einsatz: Er wandelt das digitale PWM-Signal in ein Analogsignal von 0 bis 10 V um.

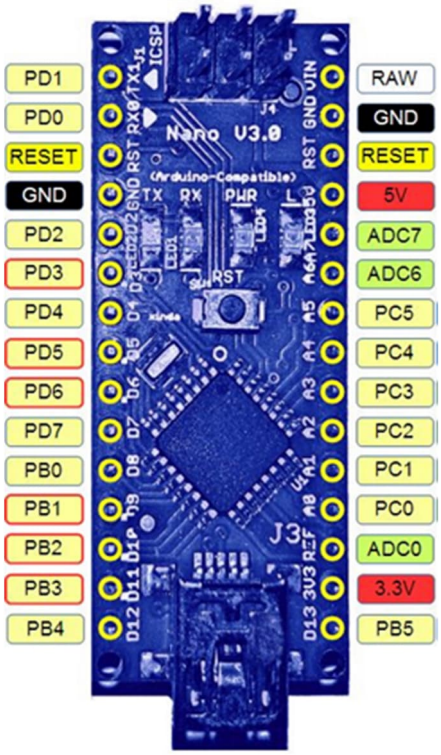


Abb. 15: Pin-Belegung des Arduino Nano

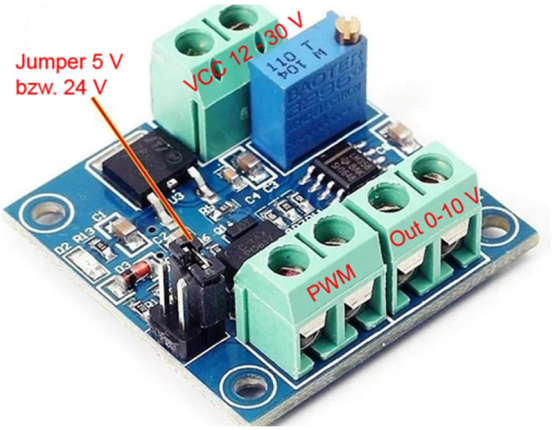


Abb. 16: Wandler LC-LM358-PWM2V

Die Werte für den PWM-Ausgang sollten experimentell ermittelt werden, da viele Faktoren das Ergebnis beeinflussen. Vom Programm wird deshalb neben dem gesuchten Teil auch die Eingangsspannung angezeigt.

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10 // SDA an Pin 10
#define RST_PIN 9 // RST an Pin 9
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup()
{
Serial.begin(9600);
SPI.begin();
mfrc522.PCD_Init();
pinMode (8, OUTPUT);
}
void loop()
{
if ( ! mfrc522.PICC_IsNewCardPresent())
{
return;
}
if ( ! mfrc522.PICC_ReadCardSerial())
{
return;
}
long code=0;
for (byte i = 0; i < mfrc522.uid.size; i++)
{
code=((code+mfrc522.uid.uidByte[i])*10);
}
Serial.print("Die Kartennummer lautet:");
Serial.println(code);
if (code==158791290) //Teil 1
{
analogWrite(5,35); // ca. 1 Volt
digitalWrite(8, HIGH);
delay (1000);
digitalWrite(8, LOW);
}
if (code==118791290) //Teil 2
{
analogWrite(5,75); // ca. 2 Volt
digitalWrite(8, HIGH);
delay (1000);
digitalWrite(8, LOW);
}
if (code==141412980) // Teil 3
{
analogWrite(5,115); // ca. 3 Volt
digitalWrite(8, HIGH);
delay (1000);
digitalWrite(8, LOW);
}}
```

Abb. 17: Arduino-Programm zu Variante 2

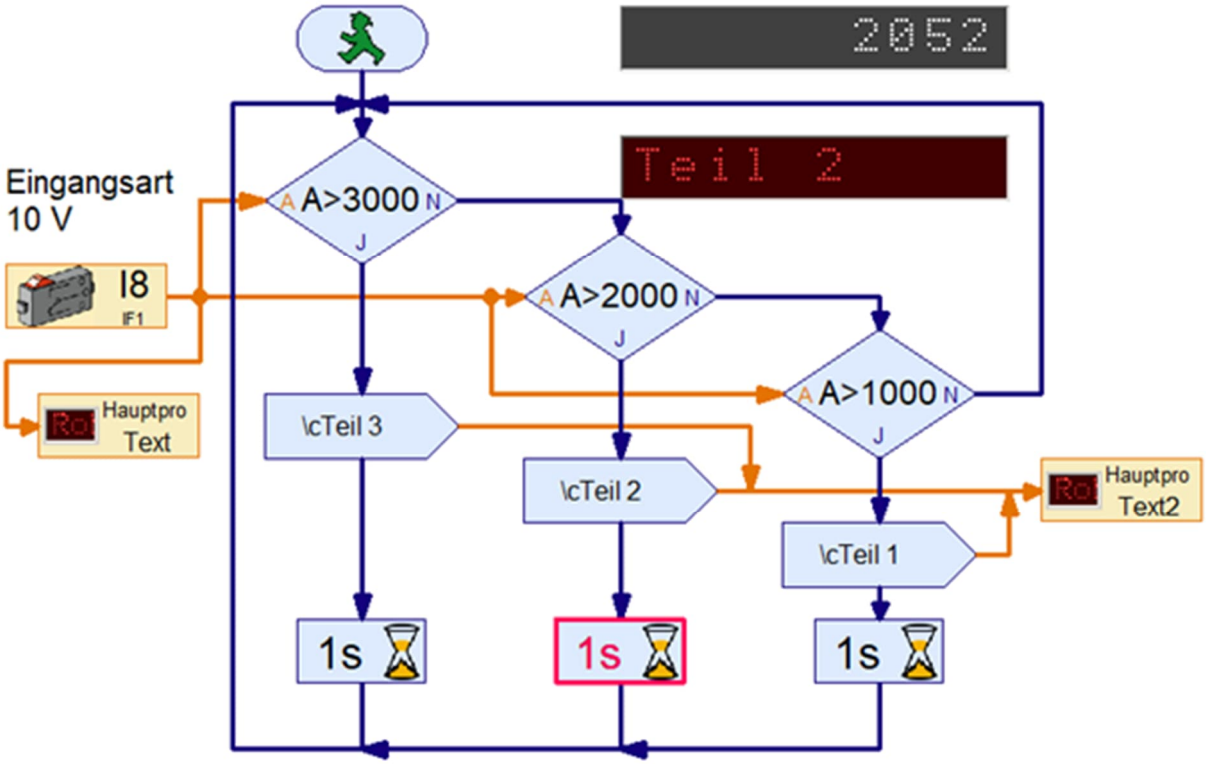


Abb. 18: ROBO Pro-Programm zu Variante 2

Quellen

[1] David Gomà: *Desktop RFID reader 13,56MHz + arduino nano*. Auf Thingiverse, 2016.

[2] Axel Chobe: Beispielvideo

[3] Axel Chobe: Quellcodes für Arduino bei den Downloads zu diesem Artikel.

Elektronik

Fahrtregler (4): Vom Widerstandsdraht bis zur PWM

Peter Krijnen

Ich möchte diese Serie mit der versprochenen vollständigen Eisenbahn-Steuerung beenden. Nachdem es mir mit fischertechnik gelungen war, die Steuerung aus ELEKTOR zu bauen, wollte ich noch einen Schritt weiter gehen: Die Züge sollten nicht mehr manuell bedient, sondern automatisch gesteuert werden.

Der Schaltungsaufbau

Bei der Steuerung, die ich in Teil 3 vorgestellt habe, wurde die Geschwindigkeitsänderung über ein Potentiometer eingestellt. In diesem Fall erfolgt dies durch Relais RB3. In der Ruhestellung wird die Spannung an E1 von G2 über das untere Poti von PB1 abgenommen. Nachdem RB3 angezogen hat, lädt sich der Kondensator C1 weiter auf die mit dem oberen Poti

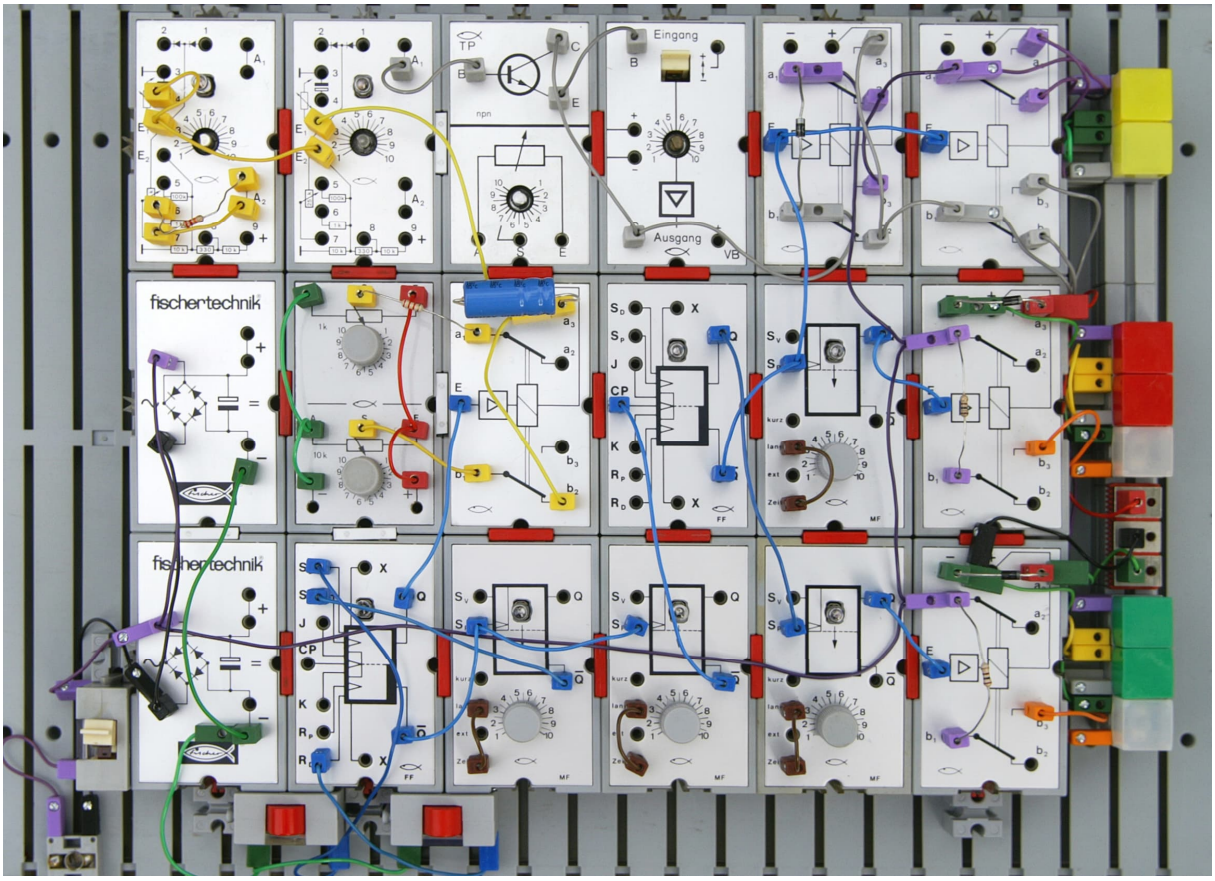


Abb. 52: Die vollständige Steuerung

eingestellte Spannung auf, wodurch die Impulse auf A1 von G2 breiter werden.

Beide Grundbausteine dienen der Drehzahlregelung mittels einer PWM-Steuerung. Die Leistungsstufe wird wiederum durch den Verstärker- und Transistor-Poti-Baustein gebildet. Bei dieser Anordnung wird die Spannung an den Schienen durch Relais RB1 umgekehrt. Relais RB2 sorgt dafür, dass nur eines der beiden Bahnhofsgleise mit Strom versorgt wird. Allerdings werden beide Relais gleichzeitig aktiviert.

Rezept

Wenn ich Bäcker wäre, würde ich zuerst mit dem Rezept beginnen. Man nehme:

- 5 × Relaisbaustein 36392
- 4 × MONO-FLOP 36480
- 2 × FLIP-FLOP 36479
- 2 × Grundbaustein 36391
- 1 × Verstärkerbaustein 36733
- 1 × Transistor-Potentiometer-Baustein 36735
- 1 × Potentiometer Baustein 37158
- 2 × Taster 31332
- 1 × Polwendeschalter 31331
- 2 × Reedkontakt 36120
- 2 × Gleichrichterbaustein 36393

Für die Garnitur:

- farbige Stecker
- farbige Litze
- einige Lämpchen zur Kontrolle

Wenn wir uns das Schaltbild in Abb. 53 ansehen, sehen wir einen START- und einen STOP-Taster.

Beide Reedkontakte liegen parallel zur STOP-Taste. Zusammen mit FF2 und MF2 bilden sie die Basis der Steuerung.

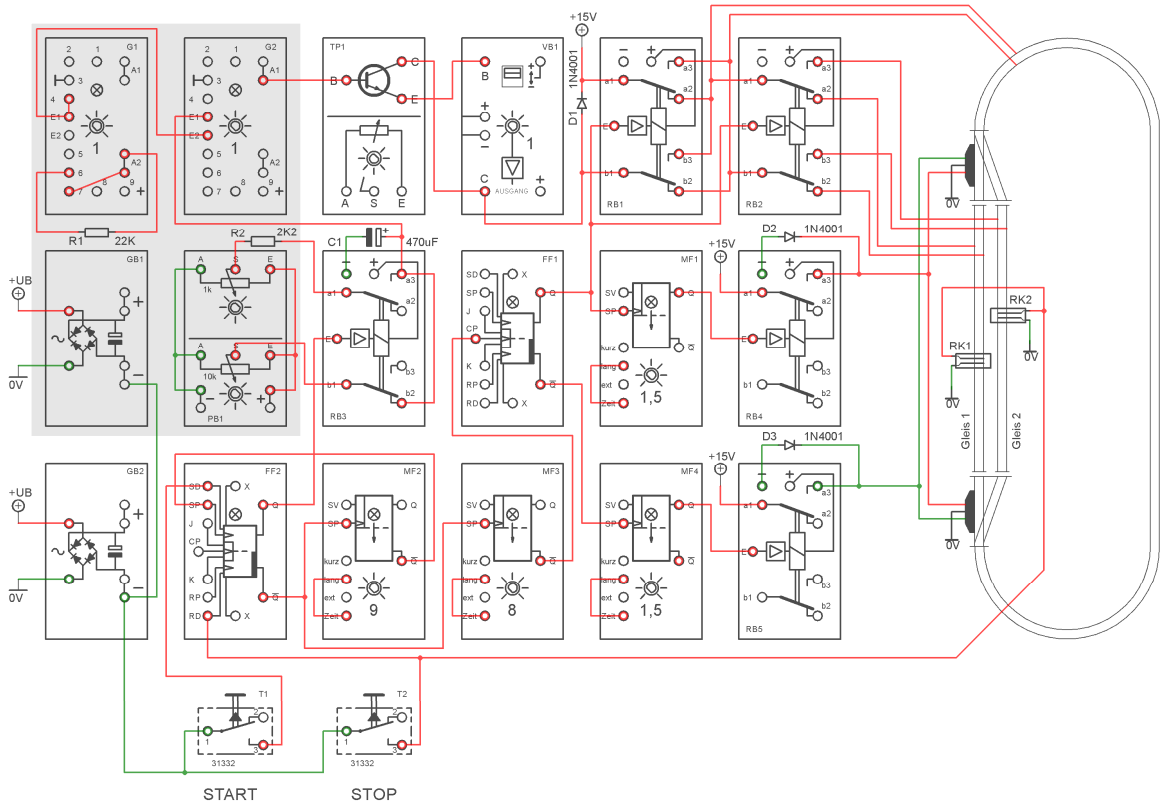


Abb. 53: Schaltbild

Ablauf

Sobald alles richtig angeschlossen ist und beide Züge auf den Bahnhofsgleisen stehen, wird die Steuerung durch Drücken der START-Taste gestartet.

Taste T1 setzt FF2 und Ausgang Q fällt auf 0 V. Dadurch wird RB3 angezogen und schaltet somit den Kondensator C1 mit der höheren Spannung von PB1. Dadurch erhöht sich die Spannung an E1 von G2 und die Impulse an A1 von G2 werden breiter. Die PWM wird vom Kollektor C von TP1 + VB1 entnommen und von RB1 und RB2 an die Gleise weitergeleitet.

Da die beiden Relais RB1 und RB2 noch nicht aktiviert sind, fährt der Zug auf Gleis 1 los. Dieser beginnt seine Runde gegen den Uhrzeigersinn. Wenn er am Ende seiner Runde zum Bahnhofsgleis 1 zurückkehrt, aktiviert er über einen an der Unterseite der Lokomotive angebrachten Magneten den Reedkontakt RK1.

Dies führt zu einem Reset von FF2. Dadurch fällt RB3 ab und MF2 und MF3 werden gesetzt. Der Zug wird langsamer und kommt schließlich zum Stehen.

Die auf MF2 eingestellte Zeit (9) muss länger sein als die Zeit, die beide Züge benötigen, um zum Stehen zu kommen – sonst würden sie wieder rückwärts fahren. Außerdem wollen wir, dass der Zug noch eine Weile stillsteht.

Nach Ablauf der an MF3 eingestellten Zeit (8) wird FF1 auf seinen CP-Eingang gesetzt. RB 1 und RB2 werden aktiviert, wodurch die Spannung umgekehrt und Bahnhofsgleis 2 mit Spannung versorgt wird.

Gleichzeitig mit RB1 und RB2 wird auch MF1 gesetzt. Während der eingestellten Zeit (1,5 s) wird dann RB4 aktiviert, was dafür sorgt, dass beide Weichen gleichzeitig umgelegt werden. Nach Ablauf der mit MF2 und MF3 eingestellten Wartezeiten wird FF2 wieder durch MF2 gesetzt.

Dann zieht RB3 erneut an. Da nun auch die Relais RB1 und RB2 aktiviert sind und beide Weichen umgelegt wurden, fährt der Zug auf Gleis 2 los – diesmal im Uhrzeigersinn – und aktiviert bei Rückkehr den Reedkontakt RK2.

Dies führt wiederum zu einem Reset von FF2. Dadurch fällt RB3 ab und MF2 und MF3 werden gesetzt. Der Zug wird langsamer und kommt schließlich zum Stehen.

Nach Ablauf der Zeit von MF3 wird FF1 zurückgesetzt. Die Relais RB1 und RB2 fallen ab und MF4 wird gesetzt. MF4 sorgt dann über das Relais RB5 dafür, dass die Weichen auf geradeaus gestellt werden.

Sobald auch die Zeit von MF2 vorbei ist, beginnt alles wieder von vorne, bis man die Nase voll hat und den Stecker zieht.

Wichtig zu wissen

Wichtig ist, dass beide Bahnhofsgleise komplett isoliert sein müssen. Andernfalls würden beide Züge gleichzeitig losfahren.

Die zum Beschleunigen und Abbremsen benötigte Zeit wird durch die Ladezeit von C1 bestimmt. Der Wert von C1 muss experimentell ermittelt werden. Der Bremsweg des Zuges bestimmt ihn: Die Bremsung beginnt mit dem Schließen eines der Reed-Kontakte und dauert bis zum Stillstand des Zuges. Die Länge der Bahnhofsgleise muss entsprechend angepasst werden. Es ist natürlich nicht beabsichtigt, dass der Zug auf der Weiche zum Stehen kommt oder darüberfährt. In meinem Fall musste ich die Bahnhofsgleise 1,5 m lang machen. Für C1 habe ich 470, 1000 und 2200 µF verwendet. 1000 µF erwiesen sich als ideal.

Wichtig zu wissen ist, dass die sehr großen Toleranzen der in den MFs verwendeten Bauteile und die mit dem Potentiometer eingestellten Zeiten nicht übereinstimmen. Auch spielt vor allem die Qualität des Potentiometers eine sehr große Rolle. Mit den mir zur Verfügung stehenden MFs war die eingestellte Zeit bei einem der MFs auf

Poti-Stellung 7 auf der Skala viel länger als in Stellung 9 bei einem anderen.

Für MF1 und MF4 muss die eingestellte Zeit möglichst kurz, aber lang genug sein, um sicherzustellen, dass beide Weichen umgelegt werden können.

MF3 darf nur bei stehendem Zug schalten. Andernfalls entgleist er beim Rückwärtsfahren auf einer Weiche. MF2 darf erst wieder schalten, wenn der Zug zum Stillstand gekommen ist, die Weiche umgestellt und die Spannung auf den Schienen umgeschaltet wurden.

Beim Aufbau der Schaltung mit den originalen Silberlingen funktionierte es nicht so, wie ich es mir vorgestellt hatte. Ich hatte die Schaltung bereits mit meinen Nachbau-Modulen (Abb. 55) aufgebaut und das hat funktioniert.

Es stellte sich heraus, dass das Problem das Flip-Flop war, das mir Jeroen Regtien zur Verfügung gestellt hatte. Aus irgendeinem Grund funktionierte es anders als meine eigenen FF. Nachdem ich beide FFs getauscht hatte, funktionierte die Schaltung einwandfrei.

Ein weiteres Problem tritt beim Einschalten der Stromversorgung auf. Nach dem Einschalten muss sich der Kondensator C1 zunächst auf die mit dem 10-kΩ-Poti eingestellte Spannung aufladen. Erst wenn START gedrückt wird, darf C1 weiter auf die mit dem 1-kΩ-Poti eingestellte Spannung aufgeladen werden.

Nach längerer Nichtbenutzung der Schaltung wird C1 sofort auf die maximale Spannung aufgeladen. Das bedeutet, dass RB3 anzieht. Dies kann jedoch nur passieren, wenn Q von FF2 0 V führt. Wenn das passiert, schalte ich sofort die Stromversorgung mit dem Polwendeschalter (31331) aus, warte ein paar Sekunden und schalte sie dann wieder ein. Dann funktioniert die Schaltung ordnungsgemäß. Mir ist aber nicht klar, was wirklich passiert.

Ich denke darüber nach, eine Art POWER-ON-Schaltung zu installieren: Eine Schaltung, die FF2 und damit RB3 nur dann freigibt, wenn C1 auf den Mindestpegel aufgeladen ist. Mal sehen, ob das mit einem dritten Grundbaustein gelingt.

Um zu verhindern, dass die Versorgungsspannung durch die Aktivierung eines Relais oder der Motoren der Lokomotiven gestört wird, habe ich auch für diese Schaltung einen modifizierten Gleichrichterbaustein verwendet. Ausgestattet mit einem 7809-Stabilisator speist er nur G1, G2 und PB1 (grauer Hintergrund in Abb. 53).

Das bedeutete, dass ich diese vier Bausteine vom Rest isolieren musste. Ich habe das mit ein paar Zahnstochern und Stückchen StripStyrene 2,5 × 3,2 mm gemacht (Marke Evergreen Nr. 176). Die Zahnstocher sind etwas zu dick, passen aber trotzdem gut. Ich musste sie kürzen und schmaler machen. Abb. 54 zeigt, wie das aussieht.

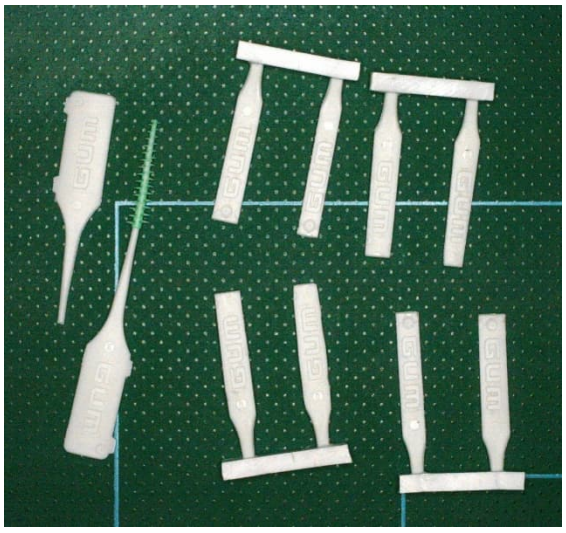


Abb. 54: Selbstgebauter Isolierstecker

Die restlichen Bausteine werden vom zweiten Gleichrichterbaustein GB2 versorgt.

Für meine TRIX-Lokomotive musste ich eine höhere Spannung (15 V) verwenden. Diese 15 V dürfen natürlich nicht an die „+“-Anschlüsse der Bausteine angeschlossen werden.

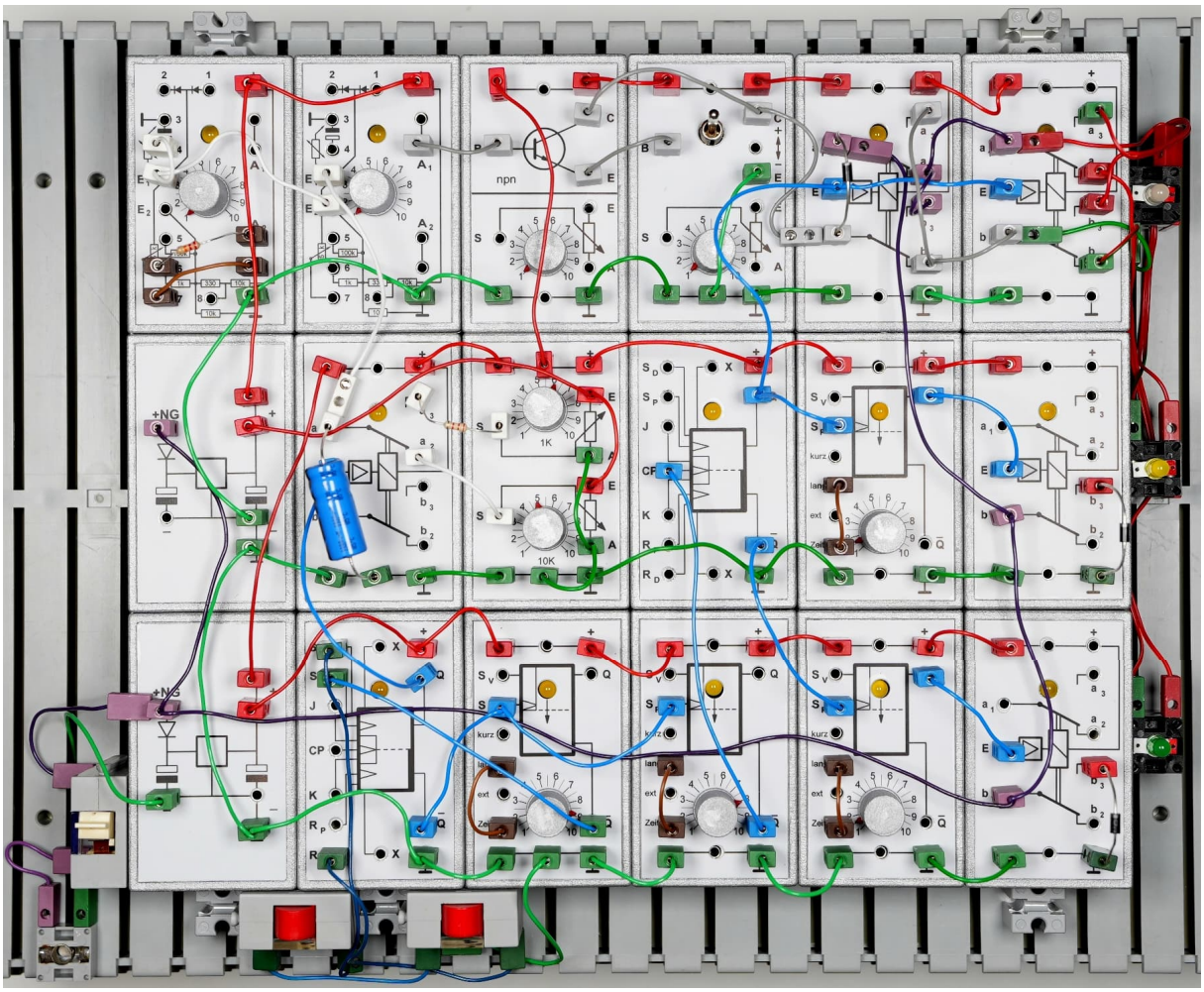


Abb. 55: Nachbau der Steuerung aus Bild 52

Wichtig zu wissen ist auch, dass die im TP und VB verwendeten Transistoren nur 200 mA vertragen. Da beide parallel geschaltet sind, sollte der maximale Strom unter 400 mA liegen. Reicht das nicht aus, empfiehlt sich der Einsatz der Leistungsstufe 36296.

Für die Bau-Spiel-Bahn kann man die Spannung aus den Bausteinen benutzen.

Nun, das war's für dieses Mal. Bevor ich mich aber verabschiede, möchte ich Jeroen Regtien für die Leihgabe einiger Bausteine danken.

Computing

Controlling parallel (universal) and serial (intelligent) interfaces with an Arduino

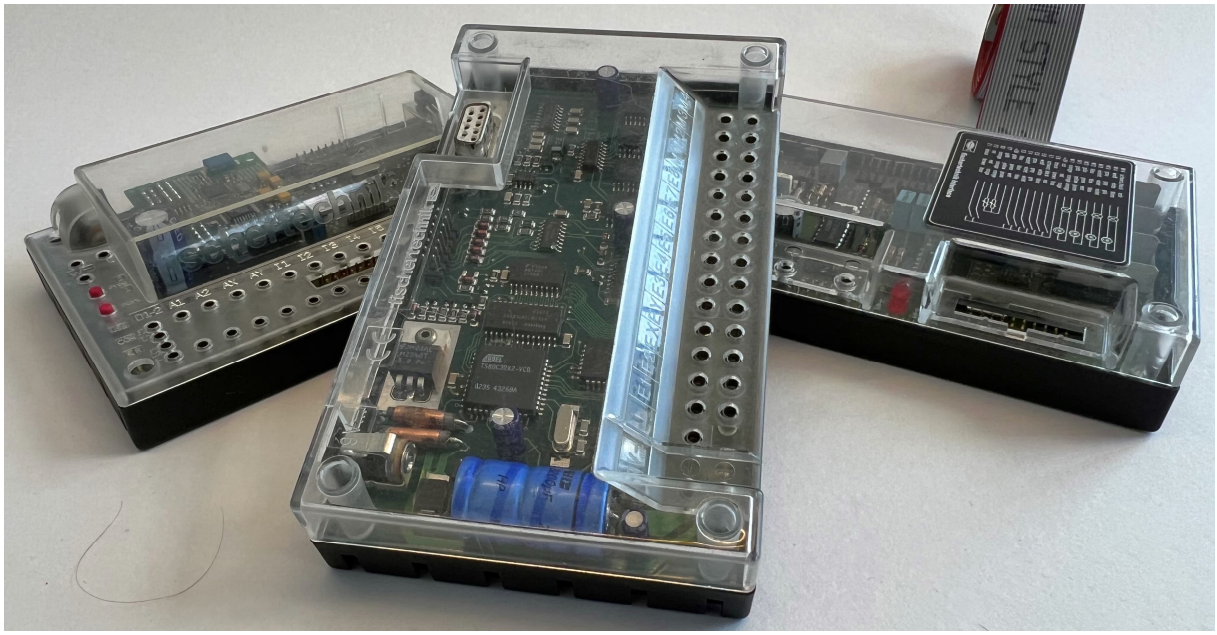
Jeroen Regtien

The oldies still work fine…

Introduction

The first fischertechnik computer interfaces appeared from 1984 onwards, with dedicated versions for various hobby and personal computers available at the time such as IBM PCs, Amigas or Atari's, C64, Apple and the Schneider CPC. A few years later the universal parallel interface (30520) was introduced, with a different adapter plug for the various desktop computers, reducing the number of versions to one. Another few years later the Cornelsen company introduced its CVK version for educational purposes that indicated with LEDs which digital input and output channels were activated, a very handy feature to check the interface's status. These different versions have extensively been described in various ft:pedia articles, specifically by Rene Trapp [1, 2, 3]. The interfaces had to be permanently connected to the parallel printer port of the desktop computer and controlled via the available Pascal, Basic, Lucky Logic or LLWIN software.

In 1997, this type of interface was replaced by the so-called Intelligent Interface (30402) which communicated with the serial RS232 protocol. As long as the controller remained connected to a power source, it could also function offline, but the program and data in memory were erased once powered off.



Pic. 1: From left to right a Robo, serial and parallel interface

This article describes a number of advanced prototypes that control these parallel and serial interfaces via an Arduino Uno, Due and/or Mega. This was done in a number of ways; by connecting the Arduino to the unaltered interface, by integrating an Arduino into the housing of the interface or creating a version that also has a rechargeable battery pack built into the housing. The advantage of this approach is that these legacy interfaces can now also be used remotely from the desktop computer. These prototypes were then initially used to control models such as the Antennen Rotor, the Teach-In Robot (30554) or Training Robot Arm (30572). More applications are to follow as the software portfolio grows.

The goal of this project is to reuse these legacy interfaces by using the widely used Arduino programming environment (IDE). The Centronics (30566), CVK (39319/66843), Universal (30520) and Commodore (30562) parallel interfaces as well as the Intelligent Interface (30402) and Robo Interface (93293 in intelligent interface mode) were successfully tested during this project. A secondary objective was to write a single software library that would shield the parallel and serial interface protocols, register-shifting, and bit manipulation from the programmer. This allows the programmer to focus on the inputs and outputs of the interfaces to control a model. The Arduino IDE C++ language allows for this in an elegant manner.

Why pay attention to these old interfaces? After all, the modern controllers TX, TXT, TXT 4.0 and ftDuino are much more powerful and versatile, and although I also use the ftDuino and TXT4.0, I remain very interested to reuse these readily available and cheap old interfaces for various older and even more modern models. Additionally, it becomes increasingly more difficult to use these interfaces with modern desktop computers and in my experience the Arduino C++ programming environment is a great alternative to LLWIN and RoboPro.

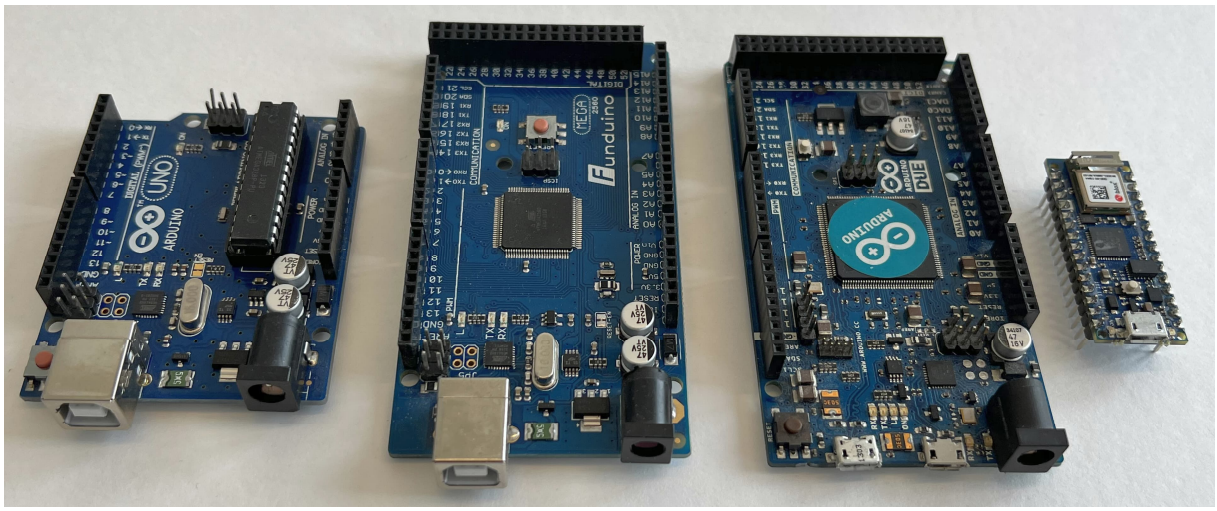
Both for the parallel (universal), serial (intelligent) and Robo interface the software solution to date only addresses single interfaces. The addition of extension interfaces to the first interface is a matter of enhancing the software libraries and will be investigated and implemented at a later date.

Arduino

Arduino is an electronic platform of which all hardware and software details are public (open source). User-friendliness is one of the design criteria. An Arduino environment consists of hardware, a circuit board (printed circuit board) with micro-controller and I/O functionality and software in the form of the very user-friendly Arduino IDE (Integrated Development Environment) environment. One can write the C++ code in the Arduino IDE and upload it to the Arduino with a USB cable from a Windows PC, Mac, or Linux machine. Once loaded, the Arduino can work independently.

The Arduino consortium has made programming micro-controllers and connecting these to various sensors and other electronic components much more accessible to hobbyists. The threshold for use is very low due to the large number of examples and software libraries on the internet allowing a programmer to quickly add functionality into their programs. The Arduino is therefore used in many secondary and tertiary education courses.

In addition to reading sensors to control lights, magnets and motors, the Arduino can also be used for Internet of Things (IOT) applications. You can connect an Arduino to the internet via WiFi, bluetooth, LORA, and therefore also monitor or control the Arduino remotely. By connecting fischertechnik models to an Arduino via the old interfaces, all these possibilities are within reach.



Pic. 2: From left to right the Arduino Uno, Mega (clone), Due and Nano

An Arduino cannot be used to directly control an actuator as there are current and voltage limitations. For the Uno these are 5V and 20mA per IO pin or 200mA for all pins together. This is clearly not enough for standard fischertechnik actuators as standard lights take 80-100mA and the various motors 50-150 mA. So there needs to be an intermediate between the Arduino (output pins) and the actuators. In Arduino speak this is done with a motor shield which contains motor drivers, H-Bridges or relays. Other solutions are the ftDuino [4] with embedded motor drivers and the shields offered by Adafruit as chosen by Fox and Püttmann in their book "Bauen, erleben, begreifen: fischertechnik-Roboter mit Arduino" [5] and Didacta [6], where the Didacta ones can be directly connected using fischtertechnik plugs. In this project the parallel and serial interfaces serve as the motor shield.

There are various Arduino models and new and more advanced versions are released every year. The standard versions are the Uno, the original work horse, then the Mega, a larger version with many additional input and output options, and the Due, an even faster model with more memory followed by the Nano, which is much smaller, so easier to embed, but has slightly fewer input and output options than the Uno. In this project the Arduino Uno, Mega, Due have been used. The use of an Arduino microcontroller in combination with fischertechnik has been covered before in a number of ft:pedia articles, a general description can be found in [7].

The additional options and speed the Arduino Mega or Due offer allow for the connection of more interfaces. Because of the large number of IO pins compared to the Arduino Uno the Mega/Due (with 54 digital, 16 analog, and 4 hardware serial ports) offer the possibility to connect up to four parallel and/or four serial interfaces at the same time.

Parallel Interface

Earlier publications

The inner working of the parallel interfaces will not be described in any detail in this article, this was done before in an excellent book and ft:pedia articles before.

In 1986 J.P.M. Steeman published a book called "Robot Control with your home-computer", a year later also translated in German [8]. This book describes in great detail the inner workings of interfaces, the fischertechnik parallel interfaces, the computing training robot (30572), plotter/scanner (30571) and the computing kit (30554, 39496, 39497), and is as far as I am con-

cerned a foundational book for all hobbyists working with the old fischertechnik models and interfaces.

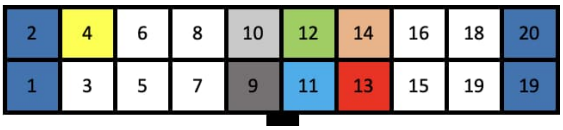
Jens Lemkamp describes in detail in his article [9] how the parallel interfaces that appeared from 1984 onwards work, but also how to connect them to an Arduino. His work made my job much easier, although figuring out which pin in the 20-pin plugs has which function turns out to be a job every time because not all authors use the same numbering convention as fischertechnik itself.

Dirk Uffmann describes [10] how to connect the universal interface to any microcontroller and built a microcontroller into the interface housing that could be controlled with an IR remote control. Jack Goudsmit has presented an implementation on Github [11], referencing the earlier article by Lemkamp [9] of but either the interface he used is different than the ones I have used or there is an issue in the description because I had difficulties getting the parallel interface to work with the connections described. Manfred Amann has also published an implementation, reading the analog values directly with the Arduino, bypassing the interface [12] As mentioned earlier, Rene Trapp describes in a number of more recent articles [1, 2, 3] extensively the inner workings of the various interfaces.

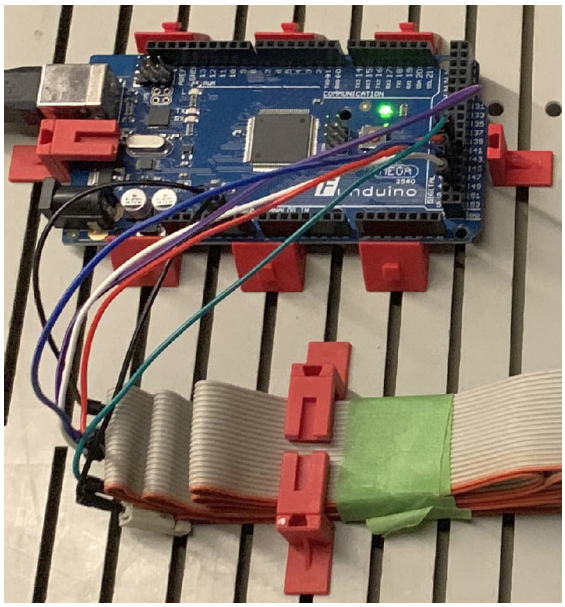
In a second article Dirk Uffmann describes software for a Uno Mega to control the motor outputs of up to two universal parallel interfaces with PWM [13]. This has not been incorporated in this project and could be a future enhancement.

First steps

First, a connection is made between the Arduino and the interface. The correct pins in the 20-pin plug needed to be identified and connected to the Arduino.



Pic. 3a: Female 20-pin plug, top (cable) view or male socket, top view



Pic. 3b: Prototype connection to Arduino Mega

Function	Plug	Arduino-Pin
Data Count In	4	2
Trigger Y	9	4
Trigger X	10	3
Data Out	11	5
Clock	12	6
Load OUT	13	7
Load IN	14	8
GND	1,2,19,20	GND

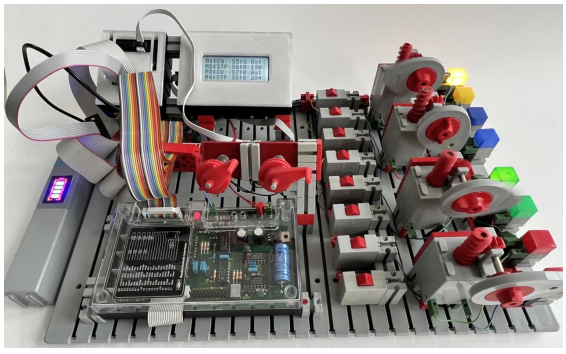
Pic. 3c: Connection pin numbers

The interface communication functions are presented in the above table, with the 20-pin plug number and the programmer defined Arduino pins. These can be any valid Arduino pin as long as the software is adapted accordingly. For the Mega up to four

interfaces can be connected, here the start-pins are 2, 22, 32, or 42 for the first, second, third and fourth interface.

To test the connection an interface test setup was created. I have seen pretty neat and handy PCB solutions with LEDs and micro-buttons on the forum, but I would like the testunit to be in style and use some surplus components. The setup contains 4 motors, eight lamps linked to the motor's outputs (two each), eight switches and two potentiometers. The two lights per motor are connected in such a way (one connection to ground) that both directions of rotation can be indicated.

With this testbed an interface or correct operation can be quickly tested. It turned out that all tested interfaces worked well as far as digital input (switches) and motors were concerned, but the analog input could not be made to work. No matter how much I followed the various instructions in the various articles and fischertechnik documentation, I was unable to get the analog input (Ex and Ey) to work via the interface. I don't think I was the first because many authors have ignored the analogue readout via the interface and connected the analogue signal generators (potmeters, photocells, NTCs) directly to the analogue inputs of the Arduino.



Pic. 4: Parallel Interface testbed with digital and analog input, motor/lamp output and a 1604 LCD display that is controlled by the Arduino

Implementation

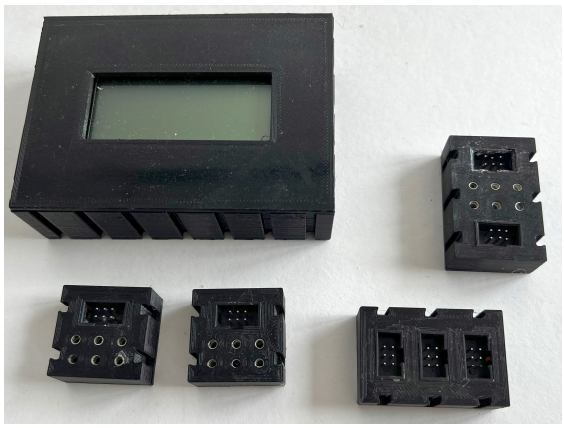
Once the interfaces were tested successfully, three different Arduino-to-Interface implementations were made:

Standalone: The interface remains unchanged, the Arduino is placed in a separate housing and connected to the interface with the existing 20 pin cable.

Integrated: an Arduino Uno is added to the interface housing and the connections between Arduino and interface are hidden.

Integrated with battery: Arduino Uno and Li-Ion battery, battery management system and DC/DC converter are built into the interface housing. The result is a compact module that can independently control motors and process signals from sensors and switches.

To display the output status of the interface as well as digital and analog input signals, a 1604 LCD with a custom 3D printed housing was used or a 1306 OLED. A 1604 display can show four lines of sixteen characters and the 1306 OLED display can be configured in many ways. Both can be connected via I2C to an Arduino.



Pic. 5: 1604 LCD display in housing with I^2C extension and connections

The analog input and LCD connections are standardized using a six-way ribbon cable with associated 2x3 pin sockets/headers. This way, the analog input and LCD displays can be shared between all prototypes.

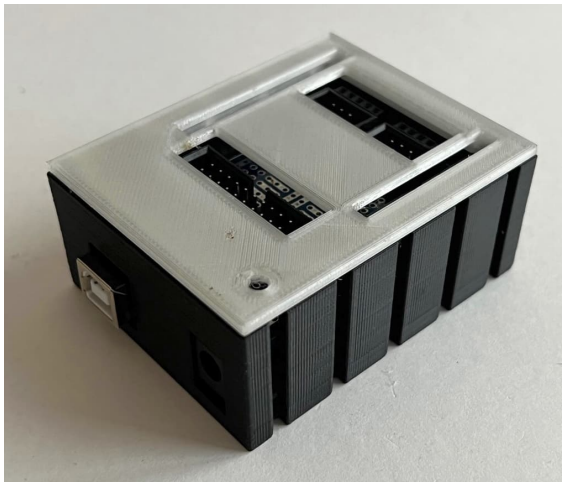
Standalone

The Arduino is equipped with an additional commercially available prototype shield on which the 20-pin male box header for the cable to the interface as well as two 6-pin male box headers are soldered. These connectors are internally connected to digital IO pins and the selected digital and analog Arduino I/O connections as well as 5V and neutral (GND). The advantage of this solution is that the interface can remain in its original state, the disadvantage is a larger number of cables.

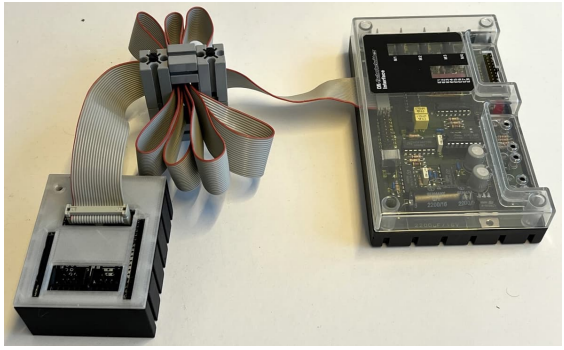
The analog input and LCD connections are standardized using a six-way ribbon cable with associated 2x3 pin connectors. This way, the analog input and I2C driven LCD displays can be shared with all Arduino shields.

Of the six pins, two are reserved for 5V and ground coming from the Arduino, two for I2C pins being SDA (data) and SCL (clock) to connect LCD displays and two input pins for incoming analog-X and analog-Y values.

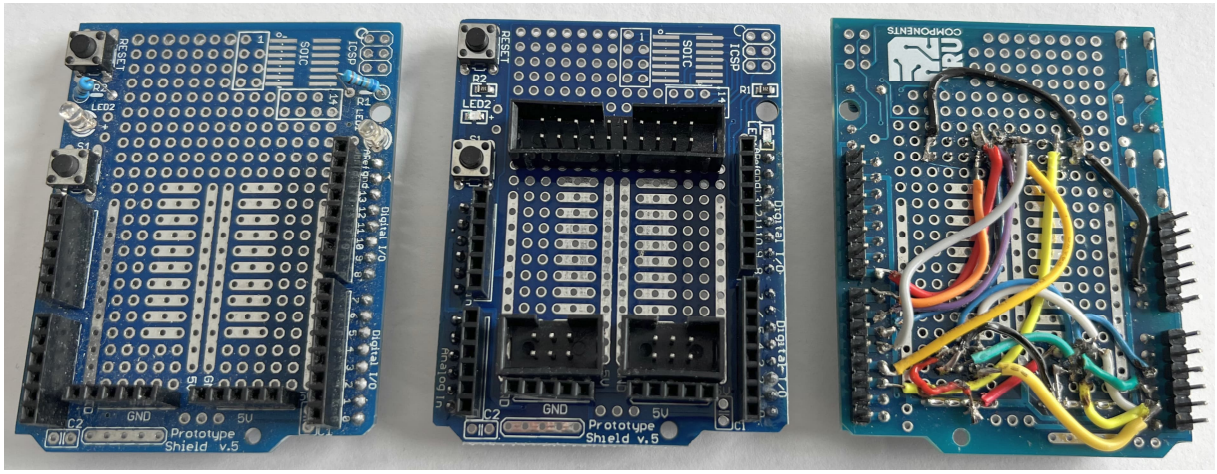
A custom-made 3D printed housing was made with slots for mounting contains both the Arduino Uno and the prototype shield.



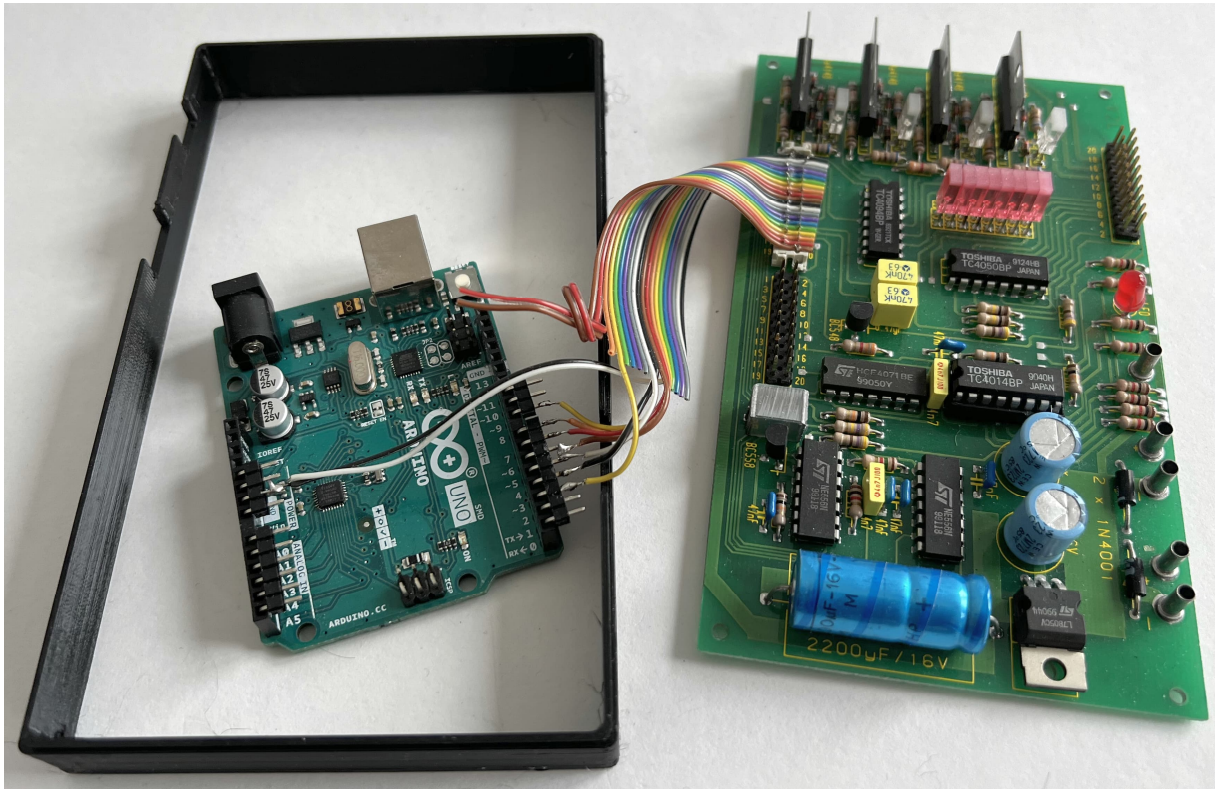
Pic. 7a: The standalone Arduino housing



Pic. 7b: Connection to interface



Pic. 6: The original prototype Arduino Uno shield, with added 20-pin and six pin box headers and the backside with connections



Pic. 8: Interface print with connected Arduino Uno and intermediate bracket

Integrated

In the integrated version the Arduino Uno is mounted under the interface PCB. There was originally insufficient room for this, but by using an intermediate 3D printed rectangular bracing and spacers the height of the unit was elevated to 45mm. To fit the Arduino, a bracket was made for the Arduino Uno. The intermediate layer, spacers and bracket were 3D printed; the files will be made available on Thingiverse at a later date when the prototypes have matured.

Partial holes were added in the black bottom cover to make room for the Arduino 9V DC and USB connectors as well as the 6-pin connector for analog signals and the LCD display.

The Arduino and interface are switched on by applying 9V to the Arduino. The voltage is supplied internally to the interface, but this can also optionally be avoided by disconnecting a plug in the housing. In the latter case, the interface and the Arduino must be supplied separately with power.

Pic. 9: Integrated Interface housing with embedded Arduino Uno

Integrated with battery

In a number of situations, it can also useful when the power supply of the interface and Arduino are integrated. That will be the next step. A battery unit consisting of two 18650 Li-Ion batteries combined with a battery management system (BMS) and a DC/DC converter to supply a constant voltage of 9V to the Arduino and interface.

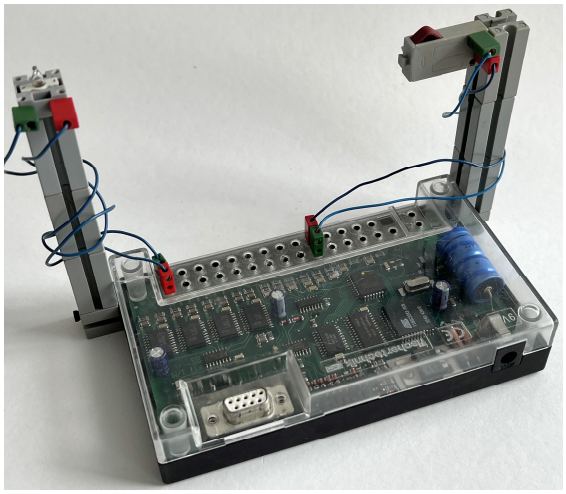
Due to the thickness of the batteries, the intermediate layer has to be raised slightly and the mounting bracket was adjusted. Furthermore, a two-way switch with a zero

position has been added so that one can choose between charging and on/off.

The state of charge of the battery is a parameter that should be visible and this can be done by adding a small 2S capacity display glued in the transparent cover of the interface. The external 9V voltage source for charging is also connected to the Arduino DC bus. Internally, the voltage is conducted to the BMS via the two-way switch.

Intelligent Interface

The Intelligent Interface (30402) can be used in two ways; on-line and off-line. The off-line mode has slower response times than the on-line version as documented in the fischertechnik manual. This article only covers the on-line mode.



Pic. 10: Intelligent (Serial) Interface

The Intelligent Interface is controlled by a host computer using the serial RS232 protocol. This standard was widespread at the time and almost every desktop computer or laptop from the period 1980-2000 had such a Sub D connection. The LLWIN or RoboPro software sent to and received from the interface the coded signals.

There is also extensive documentation about the Knobloch Robo Connect Box allowing a modern Windows PC to control a Robo Controller and Intelligent Interface via USB using RoboPro software.

I had seen a blog from Peter Fürle about controlling an Intelligent Interface via Python [14], which I was able to reproduce successfully, but I was not aware of any previous work about connecting an Intelligent Interface to an Arduino. However, if control with Python via a serial port of the Mac works, then it should also be possible with the serial port of an Arduino, I thought, and so the experiments began.

Communications protocol

The communications protocol is fairly simple. The host sends an instruction consisting of two bytes and then reads the response. The first byte indicates which information is requested, the second byte controls the motors. The first byte can be selected to ask:

- digital inputs,
- digital inputs plus analog X or
- digital input plus analog Y.

A byte consists of 8 bits and for the second byte 2 bits are used per motor. code '10' means clockwise (CW) for motor 1, '01' means counterclockwise (CCW) and '00' stop (STP). For example, if the instruction '10010010' is sent, this means motor 1 clockwise, motor 2 counterclockwise, motor 3 stop and motor 4 clockwise. Code '11' is invalid. If analog data is requested, the interface response is three bytes. The first byte is the status of the digital inputs and the remaining two contain the analog value between 0 and 1024.

Binary	Hex	Dec.	Description
11000001	C1	193	Only I/O state
11000101	C5	197	I/O state and analog value Ex
11001001	C9	201	I/O state and analog value Ey
10010010	Any	Any	2nd Byte example:

Binary	Hex	Dec.	Description
			M1 CW, CCW CCW, M3 STP, M4 CW

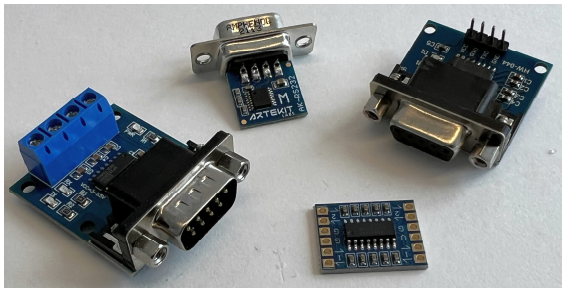
Retrieving the correct digital and analog values took some time. Reading the interfaces on the Mega hardware serial ports after sending the write instruction gave often the same analog X and Y values and it became clear that a short delay of 10ms had to be inserted between subsequent analog retrieval operations. Delays of less than 7 ms still gave the occasional random data. For the digital data a delay of 5ms had to be inserted between the write and read instructions.

TTL - RS232 conversion

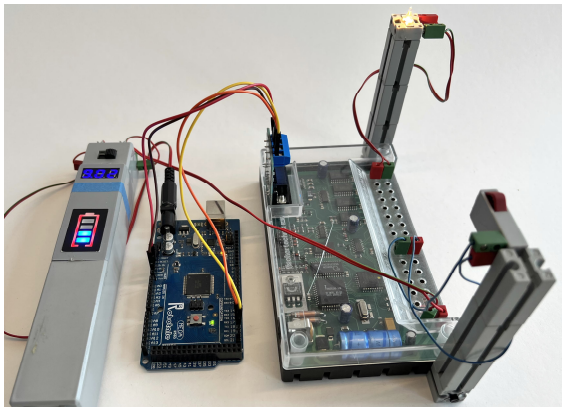
The RS232 connection of the interface cannot be connected directly to the serial connections of an Arduino because an RS232 connection can receive up to 12V DC via the PC (or 9V via the intelligent interface) and an Arduino works with 5 or 3.3V.

There are third party converters based on the MAX3232 chip that are specially designed for this. Using such a converter, an Arduino can be safely connected to a fischertechnik serial interface. Interestingly, the Intelligent Interface has a Max3232 chip on the PCB to go from 5V TTL to RS232. I resisted the temptation to connect directly to the input of this chip because it required a hardware intervention and instead used an extra converter to convert back. This proved a bit of a frustrating exercise because I purchased a number of different MAX3232 converters but the first two didn't work, so I was about to give up when the third and fourth version finally worked.

Normally, in order to communicate, the T(ransmit)X pin of the Arduino must be connected to the R(eceive)X pin of the interface and the RX pin of the Arduino must be connected to the T(ransmit)X pin of the interface, but not all manufacturers adhere to the same conventions and after some trying it turned out that I had to connect the RX to RX and TX to TX to get proper contact. After that it was just a matter of time before I could control motors, after that decoding the digital and analog signals with some bit manipulation in C++ which took a bit of time. The first prototype worked!



Pic. 11: Commercially available RS232-TTL converters; only the bottom two worked for me



Pic. 12: Prototype setup with Arduino Mega using Serial 1, 9V accu for both Arduino and Intelligent Interface with lamp & switch

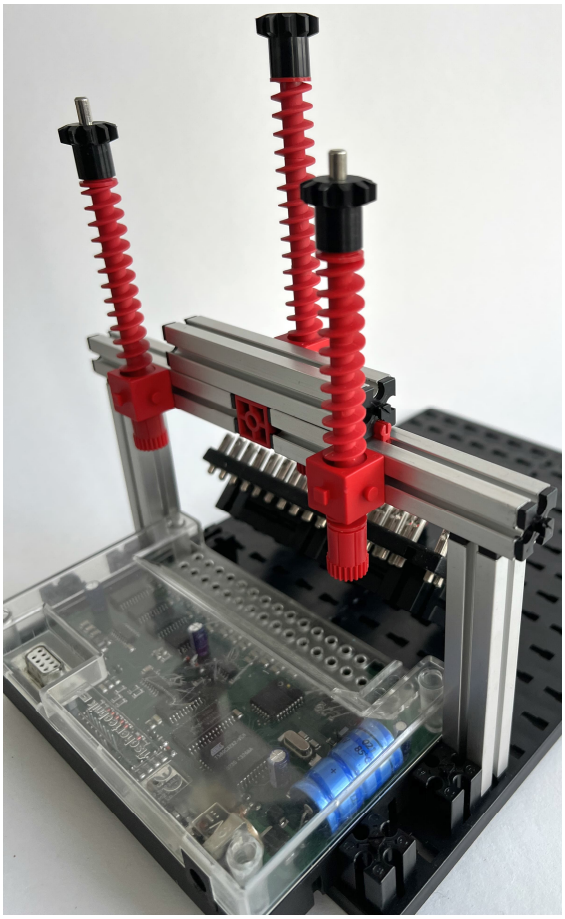
Test Unit

Testing several serial interfaces is a cumbersome job because each time up to 31 fischertechnik plugs have to be taken in and out compared to a single 24-pin plug for the parallel interfaces. So, the idea developed to make a bracket that would use the spring contacts (31306) used in the rotating switch (31311), one of the many innovative components fischertechnik developed and patented. There were unfortunately no fischertechnik components I could find with the correct inter-socket distance, so I designed

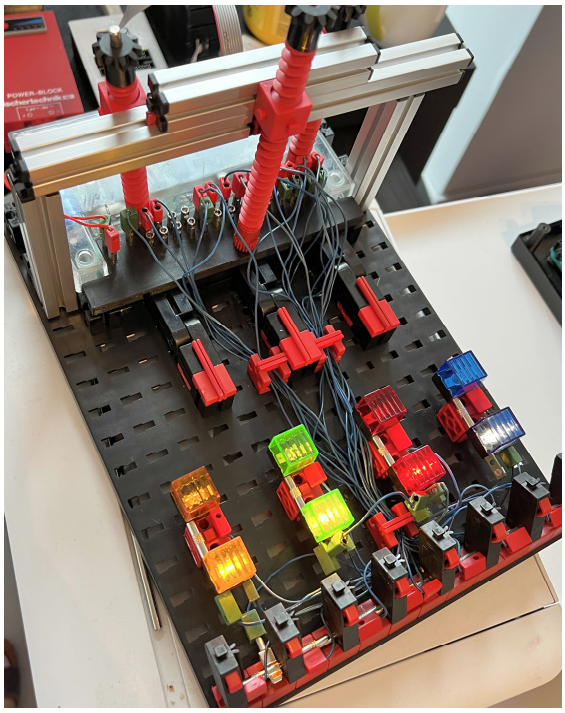
something in OpenScad and 3D printed it. After adding the spring contacts and integrating it into a housing I could test serial interfaces quickly. The concept applies to other interfaces as well, I designed, printed, and tested a TX version as well.

Pic. 13a: 3D printed bracket with spring contacts

The mechanical part of the interface consists of a base plate with hinges connected to the bracket and a vice-type of construction to tighten the bracket on the interface.



Pic. 13b: Mechanical part



Pic. 13c: Interface in operation

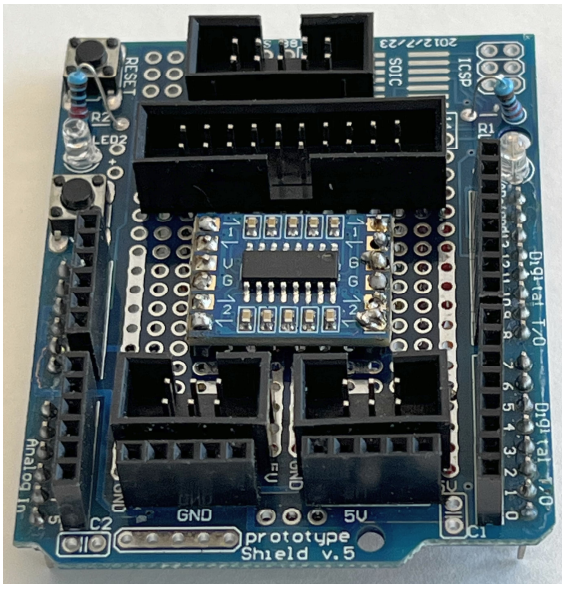
On-line Python control

Because I was initially unable to make contact between the Arduino serial ports and the interface, I connected the Intelligent Interface directly to my iMac with an RS232-USB cable and tried to connect to it from a Python programming environment using the information from Peter Fürle's blog [14]. This worked without any significant problems; hence my problem obviously was in the Arduino TTL - Intelligent Interface RS232 connection and this made me look for other TTL/RS232 converters. The on-line (intelligent) mode of the Robo Interface was also tested and it turned out to be easily accessible via Python as well. Applications of this on-line link via Python to the Mac are subject of further work and experiments.

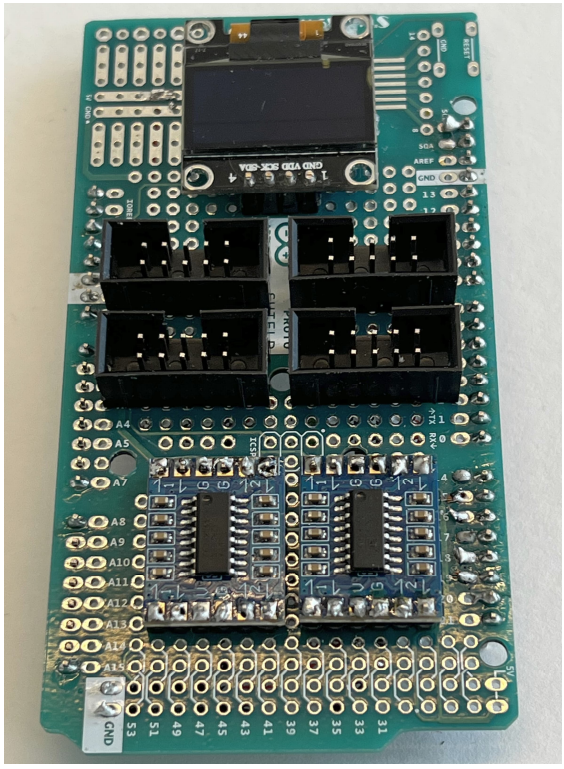
Implementation

Although there are only three connections required for a serial connection (Rx, Tx and Ground) I opted for a 10-pin IDC connection because of the availability of ready-made and cheap Sub D RS232 to 10pin IDC cables. Two prototypes were made, one for an Arduino Uno and one for the Mega.

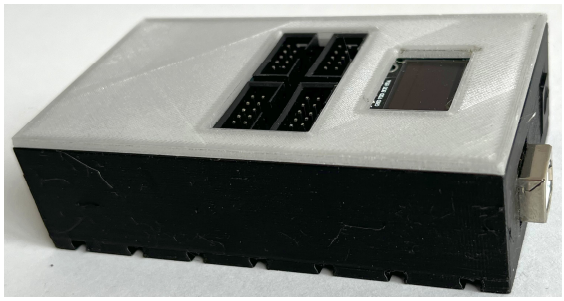
Because the two connection protocols for a parallel and serial interface are different and actually independent, there is no reason why both types of interfaces cannot be controlled by the same Arduino. The parallel interface requires 7 digital I/O ports, the intelligent interface two serial ports (Tx, Rx). For the Uno the earlier made shield for the parallel interfaces was therefore enhanced with a 10-pin socket and a RS232/TTL converter, internally connected to the required Arduino pins. This resulted in a hybrid version to which both a parallel and a serial interface can be connected.



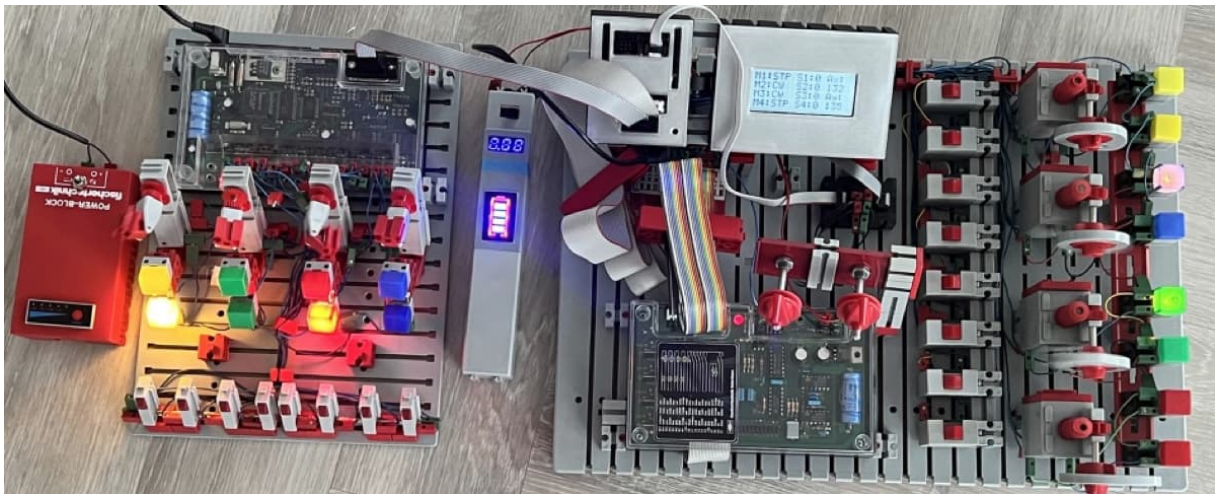
Pic. 14: Arduino prototype shield with headers for I²C and the serial port



Pic. 15a: Prototype shield with added sockets, converters, and OLED display



Pic. 15b: Complete unit in 3D printed housing



Pic. 16: The hybrid prototype shield with a parallel and serial interface connected and controlled using a single Arduino script using the software library

For the Mega four 10-pin sockets, a 1306 OLED display and the two dual channel RS232/TTL PCBs were added to a standard Arduino Mega prototype shield.

Software

With the hardware functioning with prototype C++ programs, a proper structure for the software became important. Because it was possible to connect the Arduino to the parallel (universal) interface, the intelligent interface and the Robo controller, I wanted to standardize the software in such a way that I could control all three types of interfaces in combination with an Arduino Uno, Mega or Due board with a few common software libraries underlying the same application program. The Arduino programmer should be shielded as much as possible from the detailed control protocols for the hardware.

This was possible because the control software for the interfaces was written in the Arduino Integrated Development Environment (IDE), which uses C++ which is a so-called object-oriented programming language. By structuring the code into a number of separate files, using the object class functionality, a concept library of code components could be created that can be reused for many other applications. This means that code does not have to be duplicated for the different interfaces and the programmer can focus on controlling the model that is being built.

More to come

In a future article these software libraries will be described in more detail and applied to several models. In addition, some further hardware prototypes are under development that will be presented. Once the prototype hardware and software are sufficiently matured it will be made available via Thingiverse and/or Github.

References

[1] René Trapp: *V. I. P. – Ein I²C-nach-Computing-Interface-Umsetzer (Teil 1)*. ft:pedia 2/2017, S. 63–73.

[2] René Trapp: *V. I. P. – Ein I²C-nach-Computing-Interface-Umsetzer (Teil 2)*. ft:pedia 3/2017, S. 57–68.

[3] René Trapp: *V. I. P. – Ein I²C-nach-Computing-Interface-Umsetzer (Teil 3)*. ft:pedia 4/2017, S. 36–49.

[4] Till Harbaum: *ftDuino – Open-Source trifft Konstruktions-Baukasten*. ft:pedia 1/2018, S. 85–91.

[5] Dirk Fox, Thomas Püttmann: *fischertechnik-Roboter mit Arduino*. Heidelberg, 2020, ISBN 978-86490-426-4 (printed edition).

[6] www.didacta.hr

[7] David Holtz: *Alternative Controller (1): Der Arduino*. ft:pedia 2/2016, S. 56–59.

[8] J. P. M. Steeman: *Robotik mit dem Homecomputer*. Aachen: Elektor, 1987, ISBN-10: 3921608465.

[9] Jens Lemkamp: *Parallel-Interface durch Arduino gesteuert (1)*. ft:pedia 1/2014, S. 24–30.

[10] Dirk Uffmann: *Nutzung des Universal-Interfaces 30520 als Port-Erweiterung an einem Mikrocontroller*. ft:pedia 2/2014, S. 30–35.

[11] Jack Goudsmit: *Fishduino*

[12] Manfred Amann: *fischertechnik Interface*

[13] Dirk Uffmann: *PWM-Motorsteuerung am fischertechnik-Universal-Interface*. ft:pedia 4/2015, S. 49–54.

[14] Peter Fürle: nc-x.com

Elektronik

Silberlinge: Original oder Nachbau (Teil 13)

Peter Krijnen

Nachdem ich mich in Teil 12 ausschließlich auf den Universalzähler konzentriert habe, werde ich in diesem Teil mit den restlichen Modulen fortfahren.

In diesem Teil behandle ich:

- 1 Leistungsstufe
- 1 IC-Baustein-16
- 2 Relaisbausteine
- 2 IC-Stromversorgungen

Leistungsstufe 3

Die Frage ist: Inwieweit handelt es sich bei der hier vorgestellten Leistungsstufe (Abb. 358) tatsächlich um eine dritte Version?

Betrachtet man die Anzahl der Anschlusspins und deren Bezeichnungen, würde ich gerne davon ausgehen, dass es sich hierbei um die zweite Version handelt. Es sieht genauso aus wie bei Leistungsstufe 36296 [1] (Abb. 306), aber in der unteren Stufe fehlen sowohl Br- als auch E2-Anschlüsse. Beide Stufen sind also gleich. Allerdings wurden die Namen der Anschlüsse aus der ersten Version übernommen [2] (Abb. 327).

Vermutlich wurden diese Anschlüsse für die letzte Version der Leistungsstufe 36296 hinzugefügt, um mehr Möglichkeiten zu erhalten. Anschließend erhielten die Anschlüsse ihre endgültigen Namen. Ob beide Versionen einst als Leistungsstufe 36296 in der Elektronik 30253 enthalten waren, kann ich allerdings nicht herausfinden.

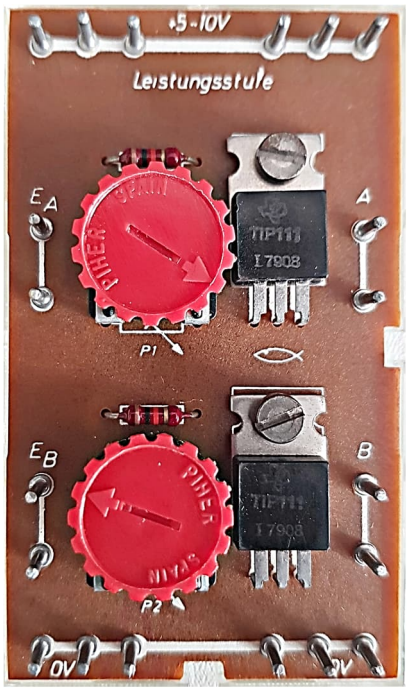


Abb. 358: Leistungsstufe 3

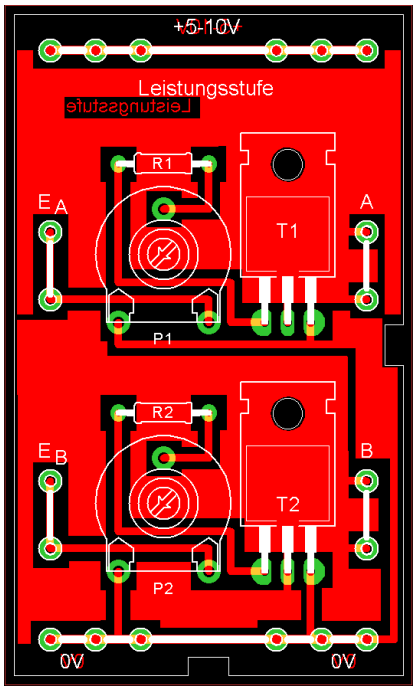


Abb. 359: Leistungsstufe 3, Layout

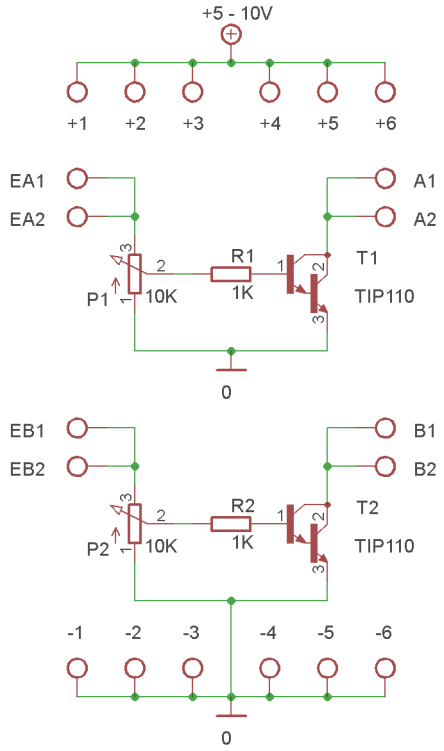


Abb. 360: Leistungsstufe 3, Schaltbild

IC-Baustein 16

Der IC-Baustein 16 ist in allen Punkten identisch mit seiner kleineren Schwester, dem IC-Baustein 14.

Sorry, das ist natürlich nicht ganz richtig. Der IC-Baustein 16 verfügt natürlich über 2 × 2 Anschlüsse mehr. Und der Sockel für den IC hat zwei weitere Pins. Er ist also breiter.

Dies führte dazu, dass weitere Leiterbahnen verlegt werden mussten: Wenn wir uns das Layout in Abb. 362 ansehen, sehen wir, dass drei Leiterbahnen am linken und rechten Rand verlegt wurden. Da bereits wenig Platz vorhanden ist, sind diese Leiterbahnen sehr schmal geworden. Dies gilt umso mehr für die drei Leiterbahnen, die zwischen Pin 8 und R2 verlegt wurden. Wären beide Widerstände um 2,5 mm in Richtung der LEDs verschoben worden, wäre mehr Platz für die Leiterbahnen gewesen.

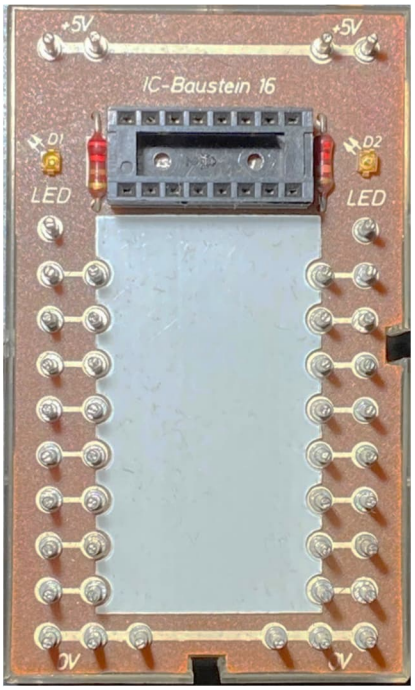


Abb. 361: IC-Baustein 16

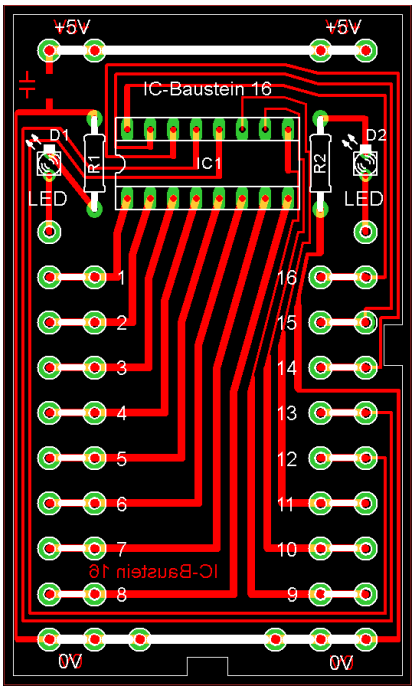


Abb. 362: IC-Baustein 16, Layout

IC-Baustein 14 mit Relais

Beim Betrachten von Abb. 363 wird deutlich, dass ein IC-Baustein nicht nur für einen IC verwendet werden muss. Relais gibt es in allen Formen und Größen, auch mit DIL14- oder DIL16-Gehäuse.

fischertechnik verwendete sogar eine Version im DIL10-Gehäuse. Der IC-Baustein kann jedoch auch als kleines Experimentierfeld genutzt werden.

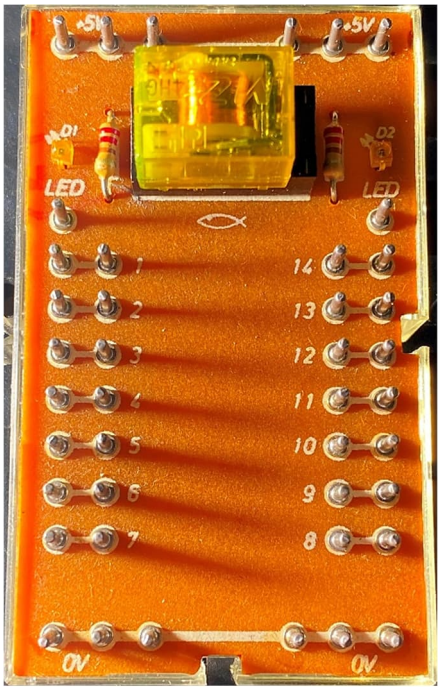


Abb. 363: IC-Baustein 14 mit Relais

Relais Baustein-2

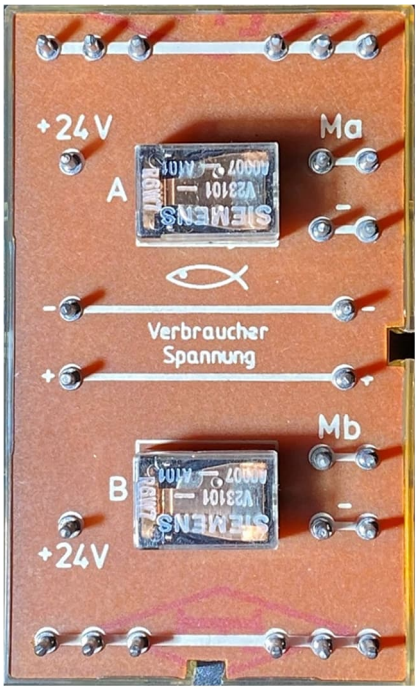


Abb. 364: Relais Baustein-2

fischertechnik wird seit langem auch zur Simulation industrieller Prozesse eingesetzt. Im Gegensatz zu den bei fischertechnik üblichen Versorgungsspannungen 5 V und 9 V kommen in der Industrie 24 V zum Einsatz. Daher hat fischertechnik Motoren im Sortiment, die für 24 V geeignet sind.

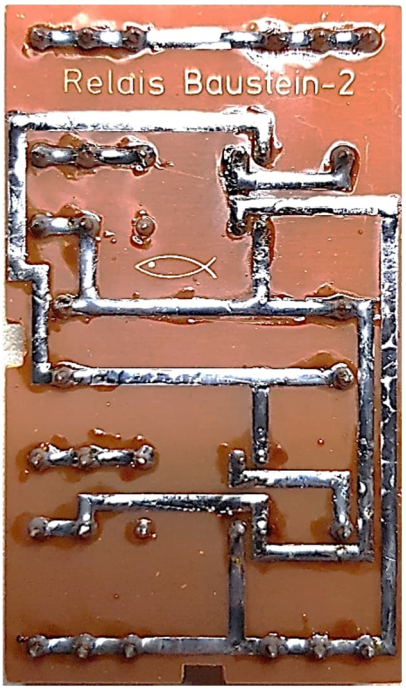


Abb. 365: Relais Baustein-2, Leiterbahnseite

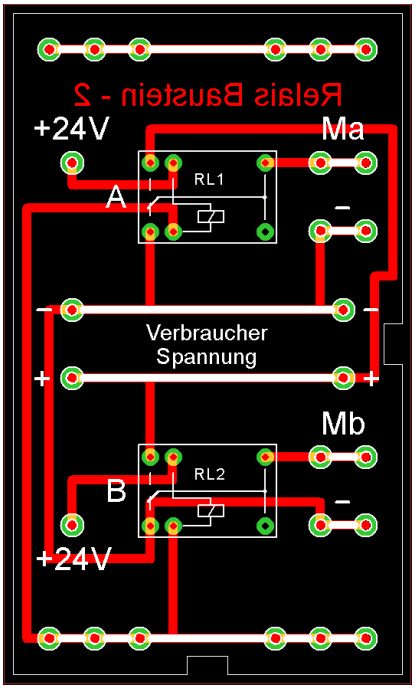


Abb. 366: Relais Baustein-2, Layout

In vielen Prozessen ist eine galvanische Trennung zwischen den verschiedenen Geräten notwendig. Dies kann mit einem Optokoppler erfolgen, aber auch elektromechanisch mittels Relais. Der Relaisbaustein-2 verwendet zwei 24-V-Relais.

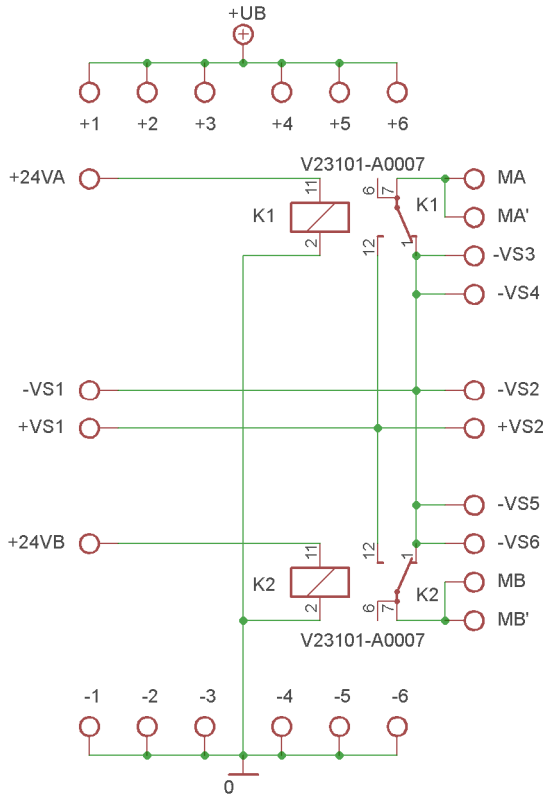


Abb. 367: Relaisbaustein-2, Schaltbild

Relaisbaustein-2 mit Verstärker

Unter den vielen Modulen, die Micha Etz besitzt, befindet sich auch ein modifizierter Relaisbaustein-2. Abb. 368 zeigt die Unterseite des Moduls. Durch das transparente Gehäuse sehen wir für jedes Relais einen Transistor, eine Diode und einen Widerstand. An mehreren Stellen sind die Leitungen unterbrochen und die Lötstelle des Widerstands mit der Basis des Transistors schwebt einfach in der Luft. Das modifizierte Schaltbild ist in Abb. 369 zu bewundern.

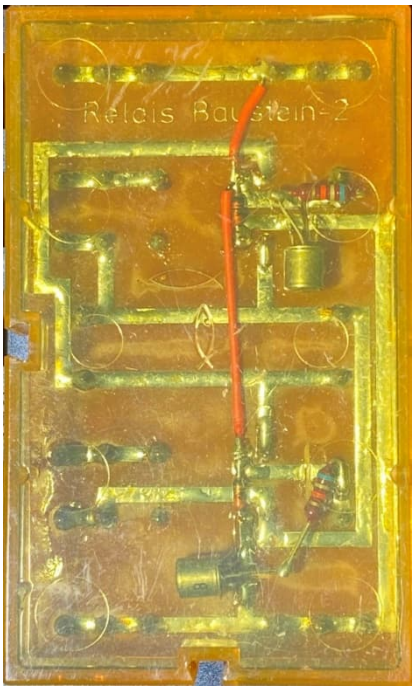


Abb. 368: Relaisbaustein-2 mit eingelöteter Schaltstufe, Leiterbahnseite

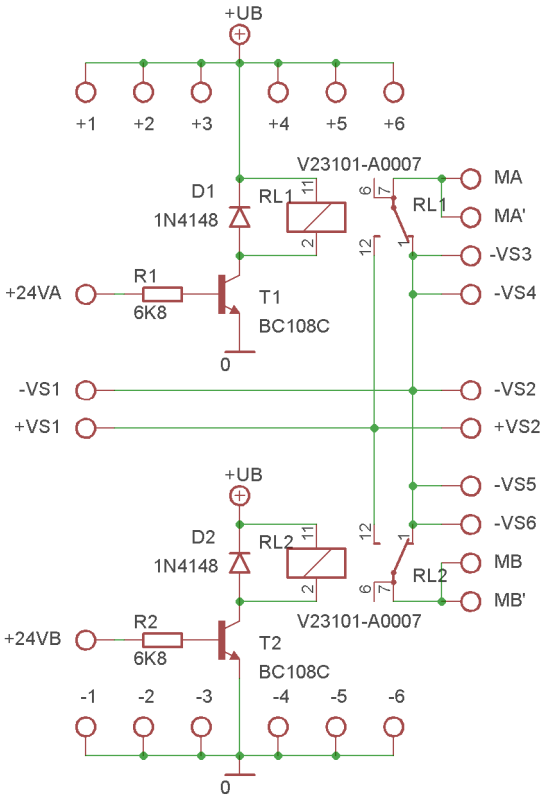


Abb. 369: Relaisbaustein-2 mit eingelöteter Schaltstufe, Schaltbild

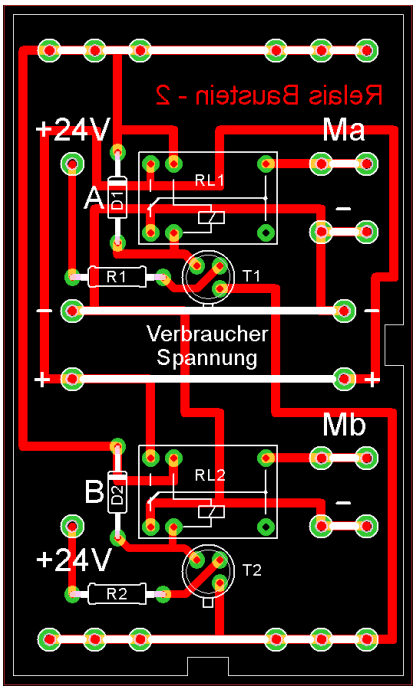


Abb. 370: Relaisbaustein-2 mit Schaltstufe, mögliches Layout

IC-Stromversorgung

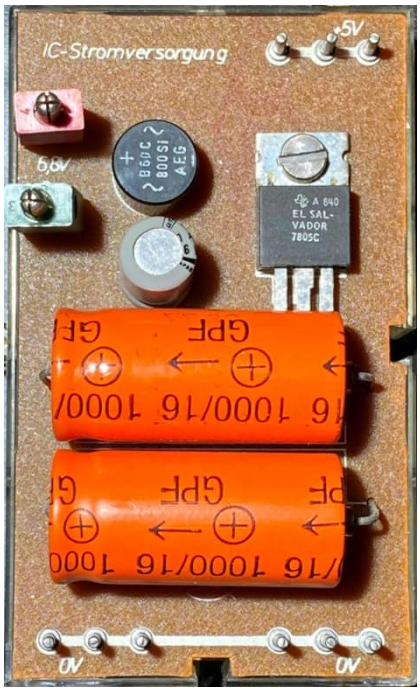


Abb. 371: IC-Stromversorgung 1

Es gibt zwei Stromversorgungen. Mir ist nicht klar, warum diese beiden Module Stromversorgung heißen: Vom Schaltungsaufbau her entsprechen sie der Spannungsversorgung 35732.

Stromversorgung Nummer 1 (Abb. 372) verfügt nur über drei „+“-Pins. Nummer 2 (Abb. 375) hat jedoch 2 × 2 „+“-Pins.

Abb. 372: IC-Stromversorgung 1, Detail der Leiterbahnseite: „Stromversorung“ mit fehlendem „g“

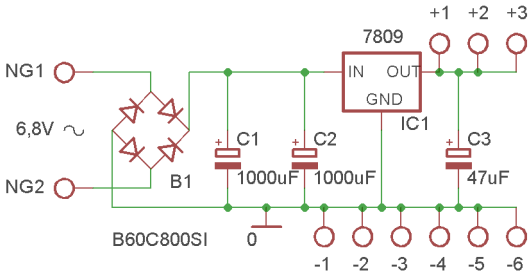


Abb. 373: IC-Stromversorgung 1, Schaltbild

Der große Unterschied liegt jedoch auf der Leiterbahnseite: Nummer 2 verfügt über eine deutlich kleinere Kupferfläche, die als Kühlplatte für den Stabilisator-IC 7805 dient.

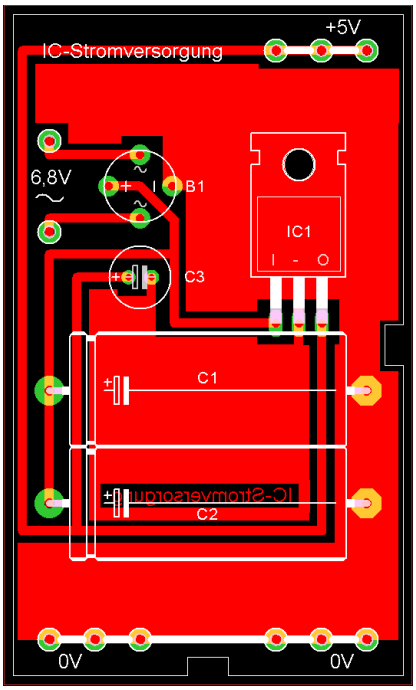


Abb. 374: IC-Stromversorgung 1, Layout

Abb. 375: IC-Stromversorgung 2

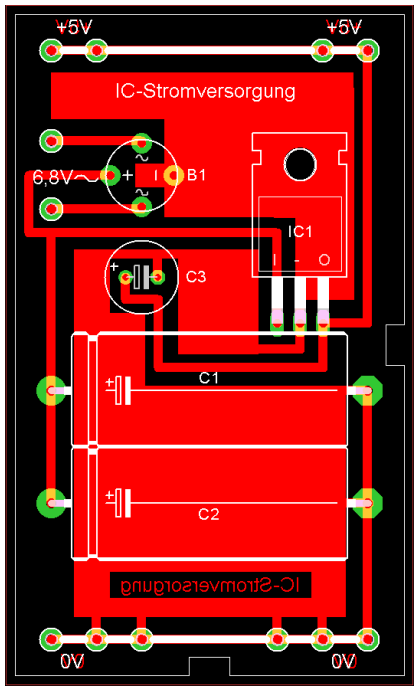


Abb. 376: IC-Stromversorgung 2, Layout

Es war auch meine Absicht, in diesem Teil auf den Taktgeber 2 einzugehen. Leider fehlte mir (und auch Arnoud van Delden) dafür die Zeit. Es stellte sich heraus, dass dieses Modul bei uns so viele Fragen aufgeworfen hat, dass wir mehr Zeit brauchen, um zu verstehen, wie es funktioniert. Der Taktgeber 2 wird daher im nächsten ft:pedia einen gesonderten Beitrag erhalten.

Bildhinweise

Die folgenden Abbildungen stammen von Thomas Brestrich: 358, 365, 373, 375.

Die folgenden Bilder stammen von Micha Etz: 361, 363, 364, 368, 372.

Arnoud van Delden hat Abb. 358 für mich bearbeitet.

Referenzen

[1] Peter Krijnen: *Silberlinge: Original oder Nachbau (Teil 10)*. ft:pedia 1/2023, S. 101–110.

[2] Peter Krijnen: *Silberlinge: Original oder Nachbau (Teil 11)*. ft:pedia 2/2023, S. 57–64.