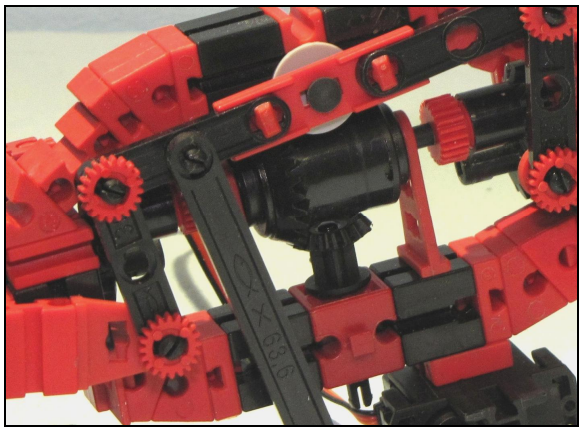
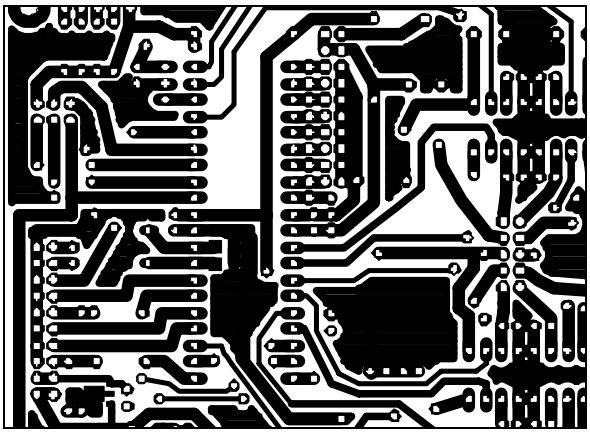
ft:pedia

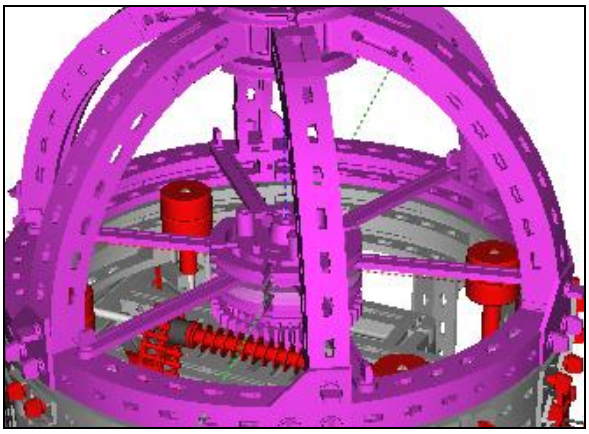
Heft 1/2014

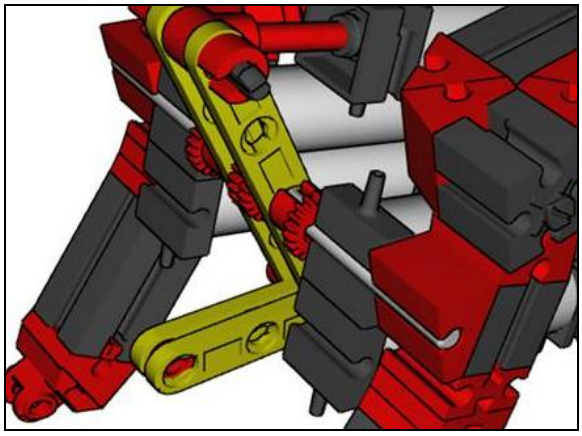


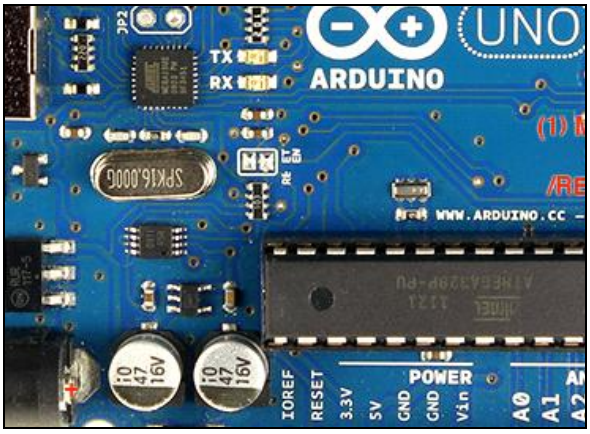


```
    digitalWrite  (ledpin,LOW); //testled au
      }
}

void       getinputs()            {// Input
  digitalWrite  (loadin,HIGH);
  digitalWrite  (clock,LOW);
  digitalWrite  (clock,HIGH);
  digitalWrite  (loadin,LOW);       // Regis
  for (int i = 8; i>0; i--){
    digitalWrite  (clock,HIGH);     // Eingar
     e[i] =digitalRead    (datain);// Bit le
    digitalWrite  (clock,LOW);      // 1. Zyk
```







Herausgegeben von
Dirk Fox und Stefan Falk

ISSN 2192-5879

Editorial

ft in die Schulen!

Dirk Fox, Stefan Falk

Wer bereits länger mit fischertechnik baut, konstruiert und experimentiert, kann sich kaum vorstellen, dass die didaktische Stärke dieses Baukastensystems einer näheren Erörterung bedarf. Dennoch steht außer Zweifel, dass fischertechnik-Kästen nicht nur in Kellern und auf Dachböden, sondern auch in Schulschränken einstauben – teuer erworben, wenig bespielt und bald vergessen.

Woran liegt das bloß? Ist es die unvermeidliche Lernkurve, die bei fischertechnik erst nach einiger Übung das „freie Spiel" erlaubt? Ist es die Dominanz der Lego-Klötzchen, die fischertechnik an den Rand drängen? Oder sind es die Modell-fixierten Anleitungen vieler aktueller ft-Kästen, die die Universalität des ft-Systems nicht überzeugend vermitteln?

Auf dem FanClub-Tag 2013 wurde ein junger Besucher gefragt, was ihm an fischertechnik so gefalle. Die spontane Antwort: „Dass man halt immer fast alles bauen kann, was man will. Dass immer alles möglich ist." Diese Antwort bringt exakt auf den Punkt, was die Faszination von fischertechnik bei Kindern ausmacht: Es sind die schier unbegrenzten Möglichkeiten, etwas Eigenes zu erschaffen.

Denn anders als beim Lernen nach Lehrplan hilft fischertechnik, selbst gestellte Aufgaben zu lösen – und vermittelt damit nicht nur Erfolgserlebnisse, sondern Grundlagen der Physik (Statik, Mechanik, Pneumatik, Optik, Elektronik) und Anwendungen der Mathematik. Auf einmal bekommen (auch in der Schule vermittelte) Kenntnisse einen tieferen Sinn: Sie helfen bei einer konkreten Problemlösung. Besser lässt sich der Wert von Wissen wohl kaum vermitteln.

Daher brauchen wir ft an den Schulen – und Freiräume, in denen Kinder sich spielend für Technik begeistern. Um diese Begeisterung zu befeuern, benötigen wir keine Modellaufbauanleitungen, sondern Bilder interessanter Modellbeispiele wie in der Fotogalerie der ft-Community, die zu eigenen Modellideen anregen und zeigen, was mit den universell kombinierbaren fischertechnik-Komponenten realisiert werden kann. Dafür brauchen wir fischertechnik-Baukästen für Schulen, die Bauteile in einer sorgfältig ausgewogenen Zusammenstellung enthalten. Davon müssen ausreichend viele verfügbar sein, damit die Konstruktionsmöglichkeiten nicht zu schnell an Grenzen stoßen – sonst kann die anfängliche Begeisterung leicht in Frustration umschlagen.

Und schließlich brauchen wir Funktionsmodelle für Lehrkräfte, die sich an den Lehrplänen des Physik- und NWT-Unterrichts orientieren, und sich als Demonstratoren eignen – Teleskope, Dampfmaschinen, Elektromotoren. Keine simplen Modelle, sondern herausfordernde Konstruktionen, die auch Lehrer so begeistern, dass sie selbst zu den Klötzchen greifen.

In diesem Sinne – ran an die Kästen!

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?
Maker Faire 2014	05.-06.07.2014	Hannover
FanClub-Tag	27.07.2014	Waldachtal
Inspiration Modellbau	20.-21.09.2014	Mainz
fischertechnik Convention 2014	27.09.2014	Erbes-Büdesheim

Hinweise

Die Inspiration Modellbau sucht gezielt ft-Aussteller!

Impressum

http://www.ftcommunity.de/ftpedia

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

Autoren: Marco Ahlers (funmca), Jörg und Erik Busch (ft-familie), Stefan Falk (steffalk), Dirk Fox (Dirk Fox), Andreas Gail, Jens Lemkamp (datjens), Harald Steinhaus (Harald), Andreas Tacke (TST), Dirk Uffmann (uffi).

Nachrichtentechnik

Morsetelegraf

Dirk Fox

Tatsächlich gab es einmal eine Zeit ohne Telefon, Funk, Film, Fernsehen und Internet. Und das ist gar nicht so lange her… Bis eine der entscheidenden Erfindungen der Nachrichtentechnik, der Morsetelegraf, die Welt veränderte.

Geschichte

In einer von Handys, Internet, Fernsehen und SMS durchdrungenen Welt ist es kaum vorstellbar, dass noch vor 200 Jahren Boten die einzige Möglichkeit waren, um Nachrichten zu übermitteln – abgesehen von Buschtrommeln und Rauchsignalen.

1792, drei Jahre nach der französischen Revolution, beschloss die französische Nationalversammlung den Aufbau der ersten Telegrafenlinie unter Verwendung des von *Claude Chappe* (1763-1805) entwickelten „Flügeltelegrafen“: Damit konnten einzelne Zeichen optisch zwischen durchschnittlich 11 km voneinander entfernten Türmen übermittelt werden. Der Flügeltelegraf bestand aus einem beweglich montierten Querbalken mit zwei weiteren, an dessen Enden befestigten kürzeren Balken, die über Seilzüge in verschiedene Stellungen gebracht werden konnten. Damit gelang es, ein einzelnes Zeichen innerhalb von zwei Minuten zwischen Paris und Lille, also etwa 270 km weit zu übertragen [1]. Abb. 1 zeigt das von Chappe verwendete ‚Winkeralphabet’. Tatsächlich konnten mit dem Flügeltelegraphen insgesamt 196 verschiedene Zeichen dargestellt werden. Durch die Verwendung von Codebüchern wurden die Nachrichten vor unberechtigtem Mitlesen geschützt. Immerhin drei Zeichen konnten so pro Minute übermittelt werden [2].

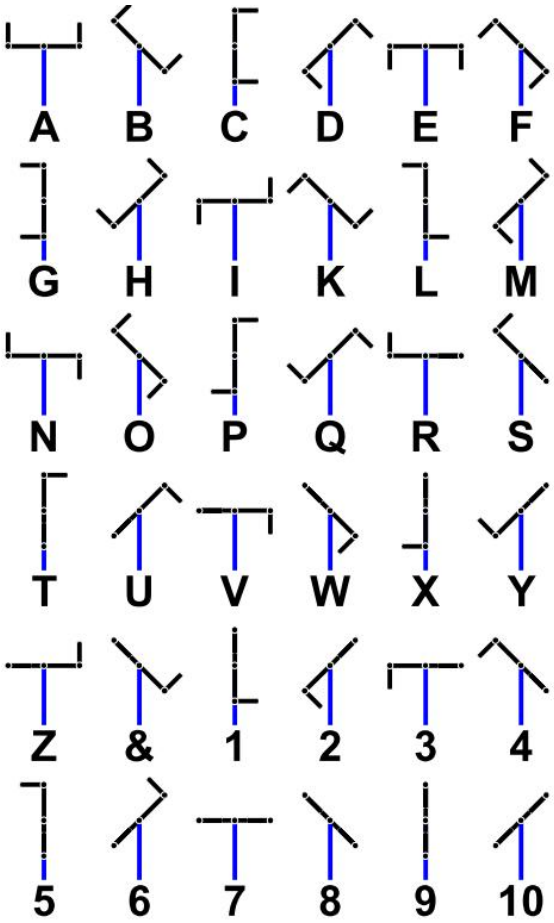


Abb. 1: Winkeralphabet von Claude Chappe (Quelle: Wikipedia)

Ein ähnliches System wurde 1832 in Preußen eingeführt. Die preußischen Telegrafen hatten sechs Flügel (Abb. 2), die jeweils vier verschiedene Positionen einnehmen konnten. Damit ließen sich 4.096 unterschiedliche Zeichen darstellen, neben Buchstaben also auch „Wortzeichen“.

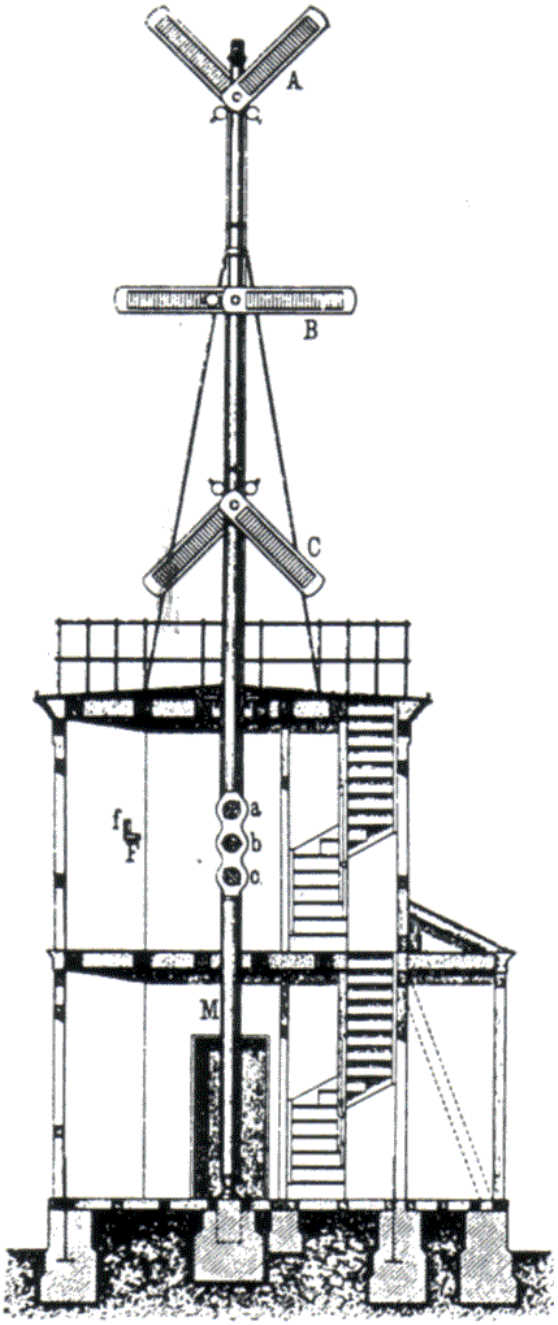


Abb. 2: Preußischer Telegraf
(Quelle: Wikipedia)

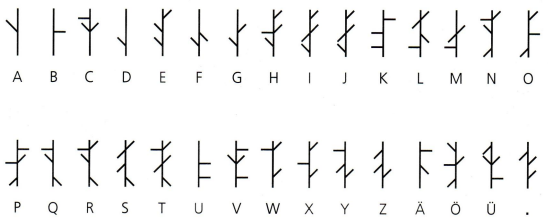


Abb. 3: Alphabet des preußischen Telegrafen
(Quelle: Wikipedia)

Ein fischertechnik-Modell des preußischen Winktelegrafen findet ihr im Anhang des lesenswerten (leider nur noch antiquarisch erhältlichen) fischertechnik-Buchs „Das Ei des Kolumbus“ [3].

Die wesentlichen Nachteile der optischen Telegrafen waren die relativ geringe Bandbreite (Anzahl der übermittelten Zeichen pro Zeiteinheit) und die Beschränkung auf Tageslicht und gute Sicht. Zudem war der Betrieb personalintensiv, denn in jeder Station musste eine Person die nächste Station beobachten und eine zweite den Telegrafen bedienen.

Daher arbeiteten viele Forscher an der Entwicklung alternativer Techniken zur Nachrichtenübermittlung. Besonders viel versprechend erschien der elektrische Strom, der seit der Entwicklung der Volta’schen Säule durch *Alessandro Graf von Volta* (1745-1827) um das Jahr 1800 zuverlässig erzeugt werden konnte.

So konstruierte der deutsche Physiker *Thomas von Soemmering* (1755-1830) im Jahr 1809 einen „elektrolytischen Telegrafen“, der mit 35 Leitungen arbeitete – für jedes Zeichen des Alphabets und für jede Ziffer eine (Abb. 4).

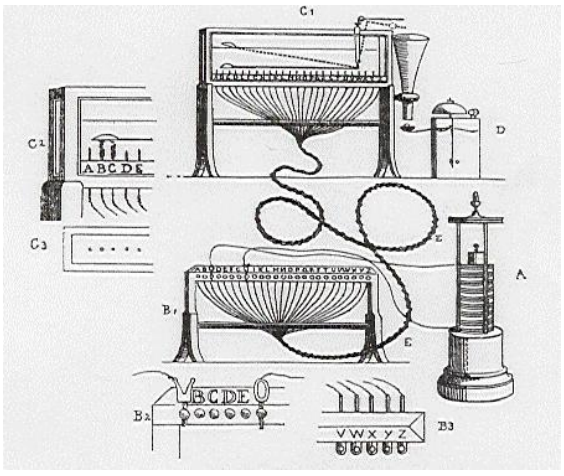


Abb. 4: Telegraf von Soemmering

Legte der Sender eine Spannung an zwei Leitungen, stiegen auf der Empfängerseite, an der die Leitungen in einer Salzlösung endeten, von dem mit dem Minuspol ver-

bundenen Kontakt deutlich sichtbar Bläschen auf. Ein quer liegender ‚Löffel', der von den aufsteigenden Bläschen angehoben wurde, schlug eine Glocke an und verkündete so den Eingang einer neuen Nachricht (Abb. 5).

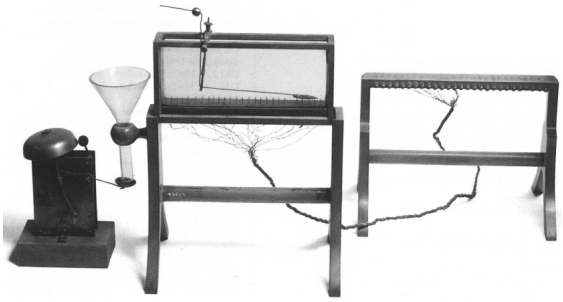


Abb. 5: Nachbau des Soemmering'schen Telegrafen (Deutsches Museum in München) [4]

Der erste Telegraf, der den Elektromagnetismus nutzte, wurde von *Carl Friedrich Gauß* (1777-1855) und *Wilhelm Eduard Weber* (1804-1891) im Jahr 1833 zwischen der Sternwarte und dem physikalischen Laboratorium in Göttingen errichtet: unterschiedlich starke Ströme lenkten beim Empfänger eine Magnetnadel aus; aufzeichnen konnte man die Zeichen jedoch nicht.

Abb. 6: Nadeltelegraf von Charles Wheatstone und William Forthergate Cooke (1837)

Mitte der 1830er Jahre wurden weitere Nadeltelegrafen entwickelt, unter anderem von *Paul Ludwig Schilling von Cannstatt* (1786-1837), dem Physiker *Carl August von Steinheil* (1801-1870) sowie dem Physiker *Charles Wheatstone* (1802-1875) gemeinsam mit *William Fothergill Cooke* (1806-1879), deren Fünf-Nadel-Telegraf (Abb. 6) erstmals 1838 entlang einer 21 km langen britischen Bahnlinie erfolgreich eingesetzt wurde.

Die entscheidende Erfindung und die Einführung der Telegraphie blieben jedoch einem Außenseiter vorbehalten: dem amerikanischen Kunstmaler *Samuel Finley Breese Morse* (1791-1872). 1825 hatte er die *National Academy of Design* mitbegründet und wurde anschließend ihr Präsident. 1835 erhielt er einen Ruf als Professor für Zeichenkunst an die Universität New York.

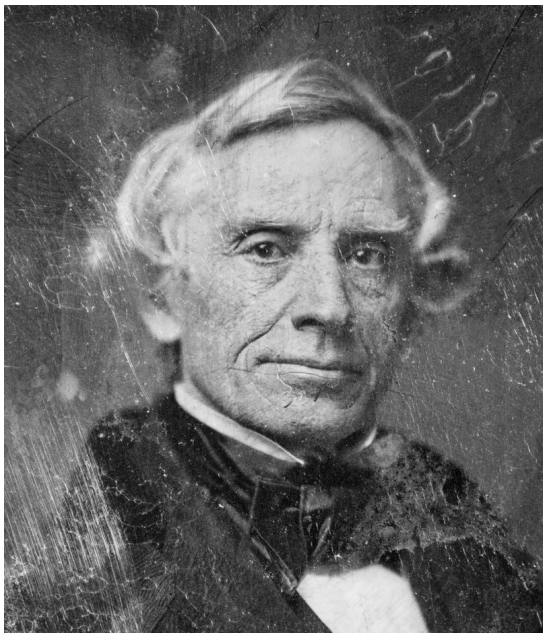


Abb. 7: Samuel Finley Breese Morse (1791-1872)

Nach der Rückkehr von einer Europareise im Jahr 1832 begann er, mit einem elektrischen Telegrafen zu experimentieren. Unter Verwendung einer alten Staffelei, eines Elektromagneten und des Federantriebs einer alten Uhr entwickelten Morse und sein Assistent *Alfred Levis Vail* ein

Gerät, das Zeichen (als eine Folge von kurzzeitigen Stromimpulsen) übermitteln und empfangen konnte. Dieser am 04.09. 1837 erstmals erfolgreich getestete Morseapparat übermittelte – als Ziffernfolge kodierte – Nachrichten, indem auf eine Schiene gesteckte schmale und breite Kupferplättchen über einen Kurbelantrieb gleichmäßig an einem Hebel entlang geführt wurden. Das Kippen des Hebels schloss oder unterbrach einen Stromkreis zum Empfänger (Abb. 8, 9).

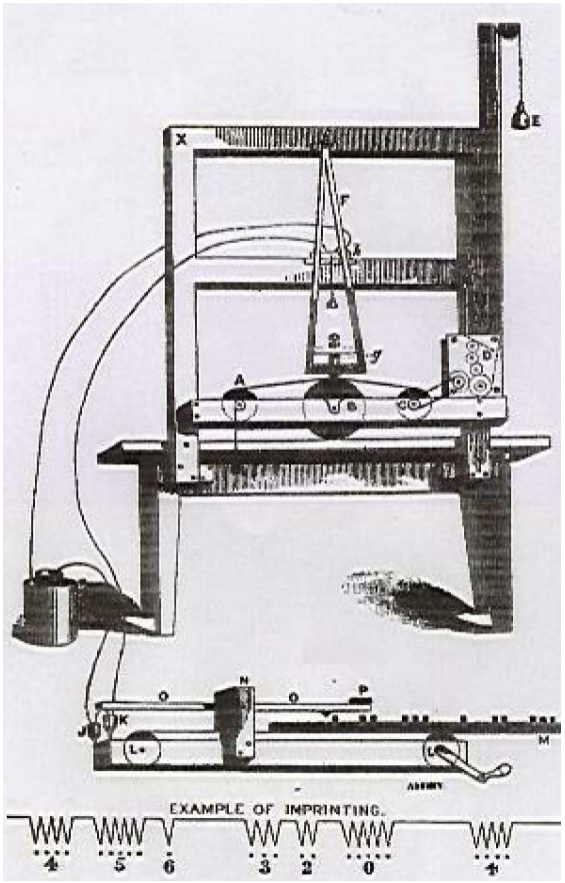


Abb. 8: Erster Morseapparat [7]

Wurde der Stromkreis geschlossen, lenkte beim Empfänger ein angeschlossener Elektromagnet ein ‚Schreibpendel' mit daran befestigtem Stift aus, der auf einem darunter entlang gezogenen Papierstreifen einen V-förmigen ‚Zacken' malte. Anschließend wurde er von einer Feder wieder in Ruhestellung gezogen [5].

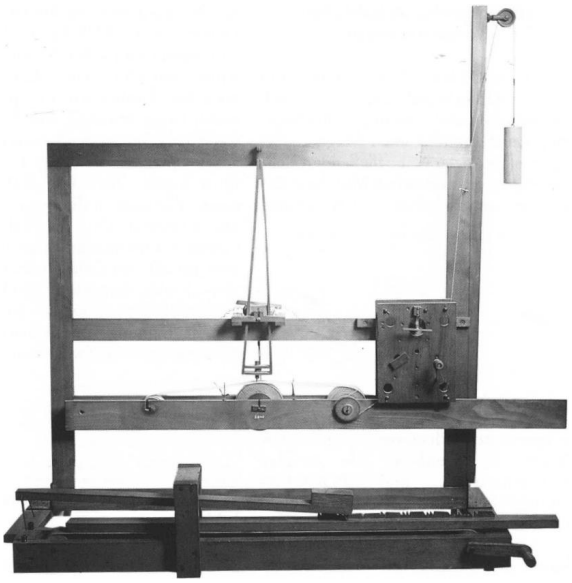


Abb. 9: Rekonstruktion des originalen Morseapparats (Deutsches Museum München)

Morses ursprünglichen simplen „Zackencode", der die Ziffern 0-9 durch in kurzem Abstand hintereinander gesteckte herausragende Kupferplättchen codierte, ersetzte Vail durch einen Zeichencode, der auch Buchstaben umfasste und heute als *American Morse Code* bekannt ist – den Vorläufer des 1865 standardisierten *Internationalen Morsecodes*.

Die Kodierung der Buchstaben erfolgt in „Punkten" und „Strichen" – ein binärer Code. Der Aufbau des Zeichencodes orientierte sich an der Häufigkeit eines Buchstabens in der englischen Sprache. Abb. 10 zeigt den Aufbau des Codes: Jeder „Zweig" des „Codebaums" steht für einen Punkt (.) oder einen Strich (–): So wird der Buchstabe „e" durch einen einzelnen Punkt kodiert; der Code für den Buchstaben „f" lautet: Punkt-Punkt-Strich-Punkt.

Nachdem Morse am 28.09.1837 für das verbesserte Modell des Telegrafen, das statt einer Zackenlinie die heute bekannten Punkte und Striche zeichnete, einen Patentantrag gestellt hatte, führte er den Morsetelegrafen zusammen mit Vail am 06.01. 1838 erstmals öffentlich vor.

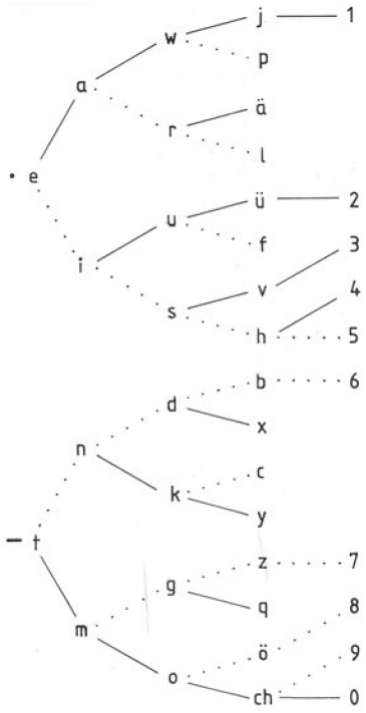


Abb. 10: Morsecode (aus: Experimentierbuch Elektromechanik [6])

Am 20.06.1840 wurde Morse für seinen Telegraphen das US-Patent Nr. 1647 erteilt [7]. Abb. 11 und 12 zeigen den patentierten Zeichengeber und das Schreibgerät beim Empfänger.

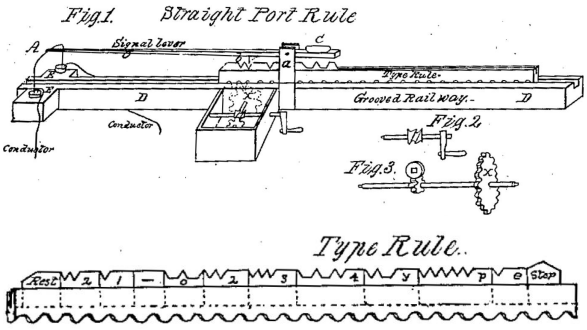


Abb. 11: Zeichengeber-Schiene mit Code-Plättchen und Kurbelantrieb [7]

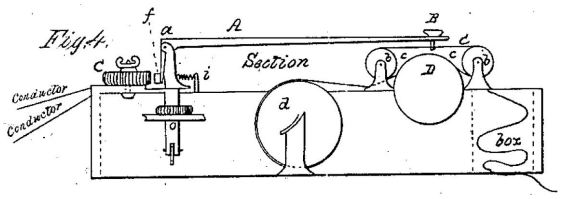


Abb. 12: Morsezeichen-Schreiber [7]

Die erste telegraphische Nachricht mit dem Text „What hath God wrought“ (Was hat Gott geschaffen?) sandte Morse am 24.05. 1844 über eine vom amerikanischen Kongress finanzierte, 41 Meilen (ca. 60 km) lange Leitung vom Capitol in Washington zur Bahnstation in Baltimore – und löste damit eine Erfolgsgeschichte aus [8].

Innerhalb von wenigen Jahren wurden zahllose Städte mit Telegraphenleitungen verbunden, und am 17.08.1858 überquerte die erste Morse-Nachricht in einem Seekabel den Atlantik. Am 04.08.1866 wurden Amerika und Europa mit einem stabileren, 4.200 km langen und 5.000 Tonnen schweren Kabel verbunden, das die „Great Eastern“, der damals größte Dampfer der Welt, verlegt hatte. Zwischengeschaltete Verstärker reduzierten die Dämpfung.

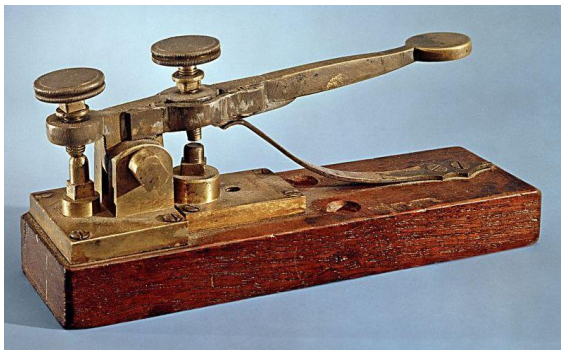


Abb. 13: Morse-Taste (Experimentalphysik-Museum der Universität Innsbruck)

Derweil entwickelte Morse seinen Telegrafen weiter; so ersetzte er die Zeichengeber-Schiene schon bald durch eine Morse-Taste (Abb. 13). Und auf der Empfängerseite wurden die Schreiber (Abb. 14) zunehmend von Lautsprechern (oder Kopfhörern) verdrängt, die das simultane Decodieren der Morsezeichen ermöglichten. Geschulte Telegrafisten konnten auf diese Weise bis zu 80 Zeichen pro Minute übermitteln – fast das 30fache von Chappes Flügeltelegrafen.

Mit der Entdeckung der elektromagnetischen Wellen durch *Heinrich Hertz* (1886) und der Nachrichenübertragung per Funk durch *Ferdinand Braun* (1898) und *Guglielmo Marconi* (1899) erreichte die Verbreitung von Morses Erfindung

schließlich Anfang des 20. Jahrhunderts ihren Höhepunkt: nun konnten Telegrafen auch in der Schifffahrt genutzt werden. Das erlebte Morse allerdings nicht mehr.

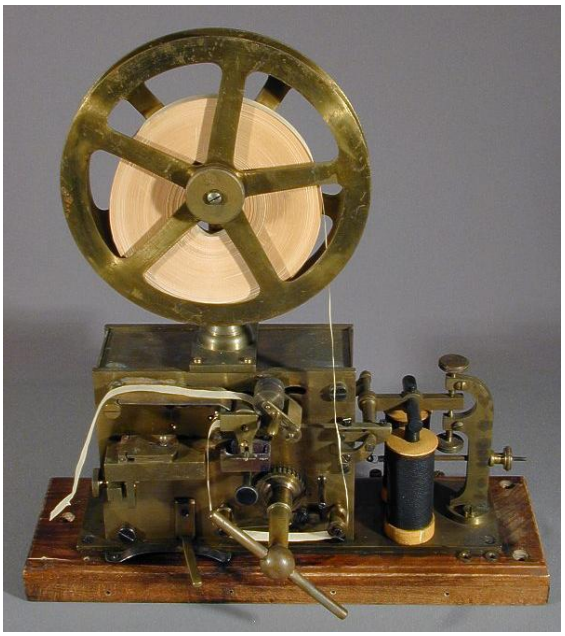


Abb. 14: Morsetelegraf von 1865 (Experimentalphysik-Museum der Universität Innsbruck)

ft-Modellklassiker

Modelle von Morse-Geräten finden sich in mehreren Anleitungen von fischertechnik-Kästen. Das jüngste Modell enthält der Kasten „Technical Revolutions“ (508776) von 2010. Die Beschreibung im Begleitheft füllt allerdings nicht einmal eine Seite.

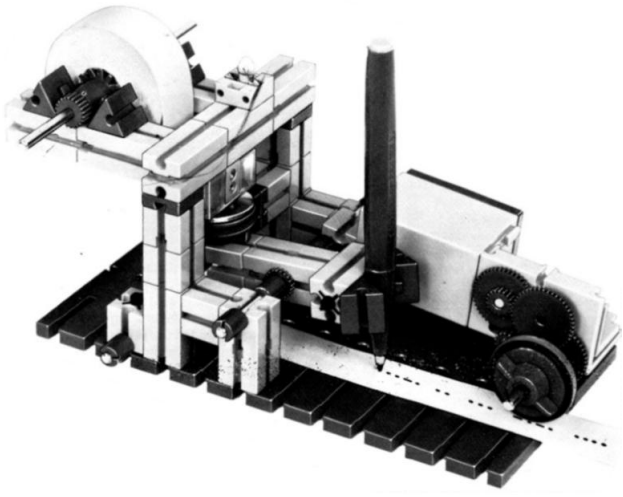


Abb. 15: Morse-Apparat aus: em2 [9]

Zudem ist das Modell äußerst simpel (Morse-Taster mit Lämpchen). Ausgefeilter und dem ursprünglichen Morsetelegrafen aus Abb. 14 ähnlicher war das Modell aus dem Kasten „em2“ (39151) von 1975: Die empfangenen Morsezeichen wurden dabei auf einen Papierstreifen geschrieben (Abb. 15) [9].

Im selben Jahr erschien die Bauanleitung eines um eine akustische Zeichengabe erweiterten Morsegeräts als Club-Modell 1975-1, elektronisch gesteuert von „Silberlingen“: Gleichrichter, zwei Relais, Grundbaustein und zwei Lautsprecher (Abb. 16).

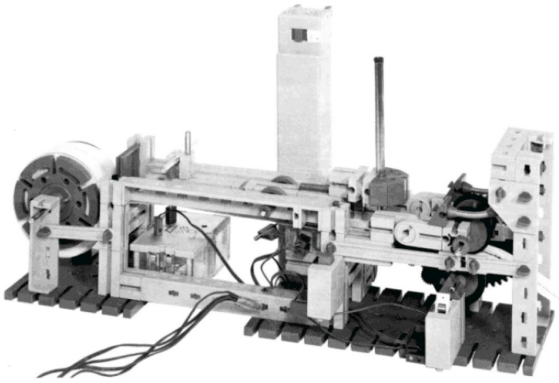


Abb. 16: Club-Modell Morsegerät [10]

Ein ähnliches Telegrafenmodell wie in der em2-Anleitung findet sich im *Experimentierbuch Elektromechanik* [11] des Kastens „Elektromechanik“ von 1981 (39145), abgebildet sogar auf der Titelseite.

Neue Modellideen

Die Dekodierung der Morsezeichen ist allerdings bei allen diesen Modellen für den untrainierten Empfänger eine mühsame Angelegenheit – wohl deshalb haben sich die Modelle selbst bei hart gesottenen fischertechnik-Fans nicht gegen SMS und E-Mail durchsetzen können.

Da wir dem Zeitalter der elektromechanischen Nachrichtenübermittlung inzwischen lange entwachsen sind, spricht wohl wenig dagegen, sich bei der Dekodierung der empfangenen Morsezeichen der Hilfe eines TX Controllers zu versichern.

Wie wäre es daher mit einem automatischen Morsezeichen-Empfänger, der empfangene Morsezeichen automatisch erkennt, dekodiert und auf dem Display anzeigt? In einem „Lernmodus“ könnte er sich sogar automatisch auf den Tastrhythmus des Senders einstellen, um die Länge der Pausen zwischen den Signalen und einzelnen Zeichen zu erlernen.

Und natürlich könnte der TX auch als Trainer fungieren und eine vorgegebene oder zufällige Zeichenfolge anzeigen, die dann in Morsezeichen über eine Morse-Taste eingegeben werden muss – und etwaige Fehler und Fehlerhäufigkeiten bestimmen und anzeigen.

Sicher kommen euch auch noch ein paar Ideen – und vielleicht wird ja ein Modell für die ft:c oder die nächste Convention daraus?

Quellen

[1] Wikipedia: *Optische Telegrafie*.

[2] Wolfgang Back, Erich H. Heimann: *Das Ei des Kolumbus*. Engelbert-Verlag, Tumlingen, 1977.

[3] Erich H. Heimann: *Das Ei des Kolumbus. Anhang mit fischertechnik-Modellen*. Fischer-Werke, Tumlingen, 1977.

[4] Volker Aschoff: *Telgraphie vor 150 Jahren*. In: Kultur & Technik, 4/1987, S. 260-264.

[5] Susanne Päch und Herbert W. Franke: *Samuel Morse und die Telegraphie*. Film, 15 min., aus der Sendereihe ‚Meilensteine der Naturwissenschaft und Technik’ des Schulfernsehens der ARD, 1991.

[6] fischertechnik: *Die Morse-Schrift*. In: Experimentierbuch Elektromechanik, Fischer-Werke, Tumlingen 1981, S. 86.

[7] Samuel F. B. Morse: *Telegraph Signs*. US Patent Nr. 1647, 20.06.1840.

[8] Dennis Karwatka: *Samuel Finley Breese Morse*. In: Technology’s Past, Prakken Pub., Ann Arbor 1996, S. 25-27.

[9] fischertechnik: *Ein Schreibgerät für Morsezeichen*. In: Anleitungsbuch Elektromechanik (em2), Fischer-Werke, Tumlingen 1975, S. 29-31.

[10] fischertechnik: *Bauanleitung Morsegerät*. Club-Modell 1975-1, 1975.

[11] fischertechnik: *Morsetelegraf*. In: Experimentierbuch Elektromechanik, Fischer-Werke, Tumlingen 1981, S. 74-77.

Fahrzeugtechnik

Bergbau-Radlader

Erik und Jörg Busch

Der Bergbau-Radlader ist unser erstes Modell, das wir auf der ft-Convention ausgestellt haben. Die Herausforderung bei Baufahrzeugen mit Knicklenkung ist die Ansteuerung der Pneumatik-Zylinder für die Lenkung, wenn diese mit der ft-Fernsteuerung proportional angesteuert werden sollen. Kernstück des Radladers ist deshalb ein mechanischer Regler, der eine gefühlvolle Lenkung mit Pneumatik ermöglicht.

Nach dem Besuch der ft-Convention 2012 stand für Erik (damals sechs Jahre alt) fest, dass wir im nächsten Jahr Aussteller sein werden. Eine Idee wurde auch schnell gefunden, das „Dübel"-Bergwerk. Ein Radlader und ein Seilbagger fördern verschiedene Dübel und FischerTip aus einem Bergwerk. Eine Förderanlage trennt anschließend die Dübel vom FischerTip. Und als letzter Schritt werden in einer zweiten Förderanlage die guten Fischer-Dübel mittels Farbsensor erkannt und in einen Original-Karton sortiert.

Trotz vieler Besuche auf diversen Baustellen, Kiesgruben und dem Bergwerk in Oberkochen haben wir weder einen Seilbagger noch einen Bergbau-Radlader im Original gesehen. Erste Eindrücke der Fahrzeuge haben wir auf YouTube gesammelt. Hier gibt es genügend Filme der beiden Fahrzeuge in Aktion. Einem Sechsjährigen vermittelt das zwar einen ersten Eindruck, für die Vorstellung der riesigen Dimensionen dieser Fahrzeuge muss man jedoch einfach die Original-Fahrzeuge sehen.

Deshalb planten wir dann einen Besuch auf der großen Baumaschinen-Messe in München, der BAUMA. Die Besichtigung der Giganten der Baufahrzeuge begeisterte die ganze Familie. Auch durch die Verwendung einer Radladerschaufel als Rutsche bleiben die riesigen Dimensionen im Gedächtnis. Eine Sitzprobe in den Fahrerhäusern gehörte natürlich auch dazu. Am Stand der Firma Schopf haben wir dann unser Vorbild für den Bergbauradlader gefunden. Die Mitarbeiter nahmen sich viel Zeit, uns ihren Radlader und seine Technik näher zu erklären. Und natürlich durfte Erik am Steuer des Radladers Platz nehmen (Abb. 3 links).

Das Original

Abb. 1: Schopf-Bergbau-Radlader SFL 60 XLP [1]

Der SFL 60 XLP (Abb. 1 und Abb. 2) ist mit einer Nutzlast von 6 t einer der kleineren Radlader, bei einem Gesamtgewicht von 19,8 t. Der größte Schopf-Radlader besitzt eine Nutzlast von 18 t. Dafür ist der „kleine" SFL 60 XLP nur 1,38 m hoch und damit für niedrigste Schachthöhen im Erz- und Kohlebergbau geeignet.

In der niedrigen Ausführung sind der gesamte Radlader und insbesondere auch die Kabine nur so hoch wie die Reifen. Der

Fahrer sitzt quer zur Fahrtrichtung. Da die Kabine seitlich die Karosserie überragt, kann er am Fahrzeug entlang schauen, für die Übersicht auf der anderen Seite gibt es eine Kamera.

Eine Besonderheit der Schopf-Radlader ist das Drehknickgelenk. Neben dem Knicken ermöglicht es ein Verdrehen um ±8 Grad. Durch diese Verdreh-Möglichkeit ist ein stabiler und einfacher Fahrwerksaufbau möglich; eine aufwändige Pendelachse ist nicht erforderlich.

Abb. 2: Radlader in Aktion [2]

Die Radlader werden zum Teil erst im Bergwerk zusammengebaut, je nach Größe der Aufzugskörbe des Bergwerks. In besonders extremen Fällen werden sogar auch die Rahmen erst unten im Bergwerk zusammengeschweißt.

Abb. 3: Erik am Steuer der Radlader

Der Testfahrer

Das Modell des Radladers sollte mit der Fernsteuerung gut steuerbar sein und auch für den Spielbetrieb ausreichend robust ausgelegt werden. Bei vielen Testfahrten im Testgelände „Wohnzimmer“ wurden die verschiedenen Bauzustände durch Erik bis zum Ausfall getestet. Durch diesen unermüdlichen Einsatz des Testfahrers konnte die Stabilität des Radladers immer weiter optimiert werden. Die großen Fahrten in der großen Halle der ft-Convention (Abb. 3) außerhalb der Besichtigungszeiten hielt das Modell trotz Hindernissen wie Kabelkanälen problemlos aus [3].

Abb. 4: Radlader vor dem Bergwerk bei der ft-Convention

Das Modell

Die Schaufelsteuerung und die Knicklenkung, im Original hydraulisch betätigt, werden im Modell (Abb. 4 und Abb. 5) pneumatisch gesteuert. Die proportionale Ansteuerung der Lenkung über die Pneumatik-Zylinder ist das Kernelement dieses Radladers. Proportionale Ansteuerung bedeutet, dass sich entsprechend der Stellung des Fernsteuerhebels ein definierter Lenkeinschlag ergibt.

Eine proportionale Steuerung basiert immer auf einem Abgleich des aktuellen Ist-Knickwinkels mit dem gewünschten Soll-Knickwinkel. Die Steuerung, d.h. der Abgleich von Soll- und Ist-Knickwinkel, lässt sich natürlich auf viele Arten elektronisch oder mit einer elektronischen Steuerung (z. B. TX) lösen.

In unserem Modell wird diese Steuerung elektromechanisch gelöst. Über Schalter wird jeweils ein Magnetventil aktiviert und damit ein Pneumatik-Zylinder angesteuert. Anstelle von Schalter und Magnetventil könnte man natürlich auch direkt ein leicht schaltbares Pneumatik-Ventil verwenden. Im aktuellen Fischertechnik-Sortiment gibt

es jedoch nur noch die Möglichkeit mit Schalter und Magnetventil.

Die mechanische Regelung ist im hinteren Teil des Radladers untergebracht. Aus Platzgründen wird ein linearer gerader Fahrweg verwendet. Mit der entsprechend großen Länge erreicht man dadurch eine hohe Stellgenauigkeit. Doch davon später mehr.

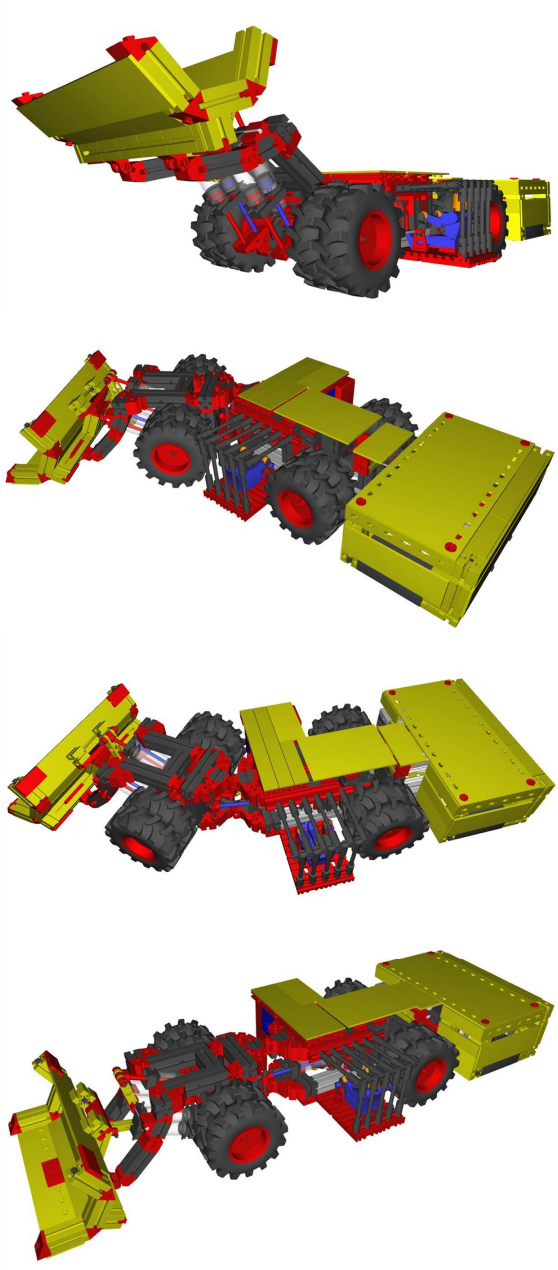


Abb. 5: Radlader als ft-Designer-Modell

Der Fahrantrieb erfolgt über drei Differentiale auf alle vier Räder. Und das Soundmodul für das Dieselmotor-Geräusch darf natürlich auch nicht fehlen.

Der Aufbau

Der Grundrahmen besteht aus zwei 270 mm langen Alu-Profilen (Abb. 6). Daran anschließend befindet sich das sehr stabil aufgebaute Gelenk. Im hinteren Teil, d. h. im Bild rechts, sieht man die Achsen für die Linear-Führung der Ist-Position.

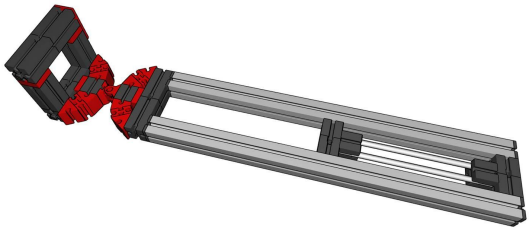


Abb. 6: Grundrahmen

Der obere Teil des Gelenks wird identisch dem unteren Teil aufgebaut (Abb. 7). Die doppelten vertikalen Stützen sind für die Halterung des Motors. Hinter dem Gelenk ist genügend Platz für die später beschriebenen Hydraulikzylinder der Lenkung.

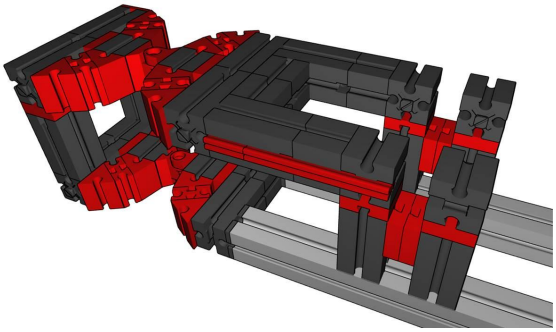


Abb. 7: Gelenk

Im nächsten Schritt werden nun der Motor und der zentrale Antrieb mit Mittendifferential ergänzt (Abb. 8). Eine Antriebskette überträgt das Drehmoment von dem Zahnrad Z10 der Motorwelle auf das Zahnrad Z20 des Differentials. Als Kette wurden, abweichend von der Darstellung, normale Kettenglieder ohne Vierkant an der Seite aus Platzgründen verwendet (im ft-Designer waren diese Elemente nicht vorhanden). Ein Kardan-Gelenk überträgt

das Drehmoment dann über das Knickgelenk.

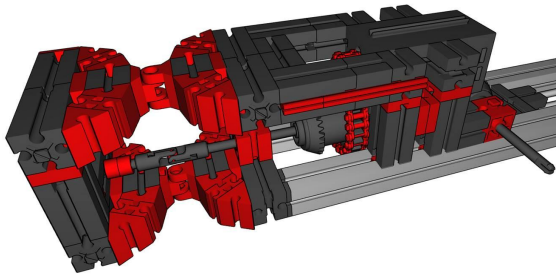


Abb. 8: Motor mit Kettenantrieb

Der vollständige Antrieb mit allen drei Differentialen und den breiten Reifen ist aus Abb. 9 ersichtlich. Das Differential zwischen den Hinterrädern ist aus Platzgründen ohne Stirnrad.

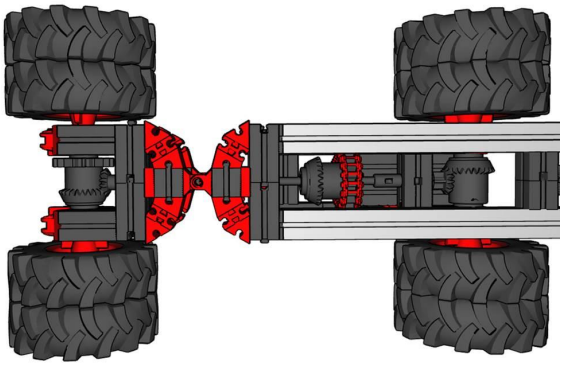


Abb. 9: Antrieb mit drei Differentialen

Die Hebevorrichtung der Schaufel (Abb. 10) wird pro Seite durch jeweils zwei Pneumatik-Zylinder angesteuert. Damit sind die Hebekräfte auch für gut gefüllte Schaufeln ausreichend. Die beiden Pneumatik-Zylinder sind mit Verbindern gekoppelt und liegen mit ihren Achsen in der unteren roten Bauplatte.

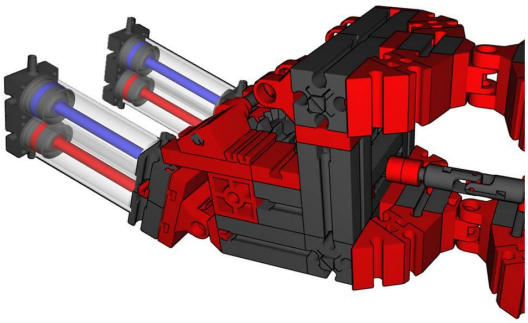


Abb. 10: Hebezylinder

Der obere der Hebe-Zylinder ist über die Achse mit dem Schaufel-Arm verbunden (Abb. 11). Der Kipp-Mechanismus ist eine Z-Kinematik, die mit einem „roten“ Pneumatik-Zylinder mit Rückstellfeder angesteuert wird. Die Feder stellt sicher, dass ohne Betriebsdruck die Schaufel in jeder Lage des Arms sicher nach vorne kippt. Ohne diese Feder müsste man beide Seiten des Pneumatik-Zylinders ansteuern. Dazu würde ein zusätzliches Magnetventil benötigt.

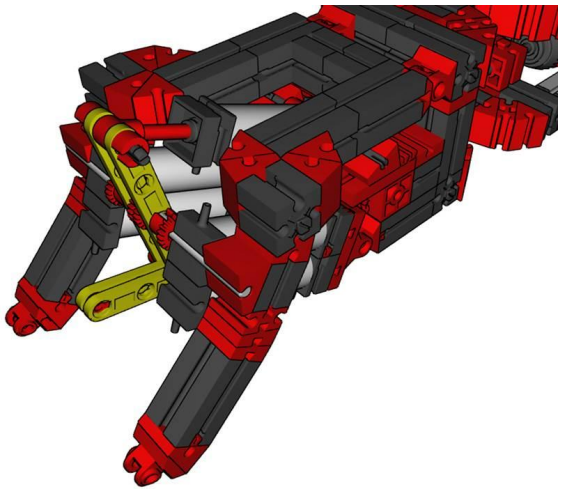


Abb. 11: Kipp-Mechanismus

Die komplette Mechanik mit der Schaufel ist in Abb. 12 dargestellt.

Bei den doppelten Hebezylindern gibt es noch eine kleine Besonderheit. Jeweils einer der beiden Zylinder ist ein „roter“ Pneumatik-Zylinder mit Rückstellfeder.

Für die normale Abwärtsbewegung ist die Schwerkraft der Schaufel absolut ausreichend. Da aber für die Kipp-Bewegung der Schaufel auch ein Pneumatik-Zylinder mit Feder eingebaut ist, ergibt sich ein Aufkanten über die vordere Schaufelkante. Die Ladung fällt dadurch leicht wieder aus der Schaufel. Mit den Rückstellfedern in den Hebezylindern wird die Schaufel immer plan auf den Boden gedrückt und ermöglicht so eine gute Aufnahme der Ladung.

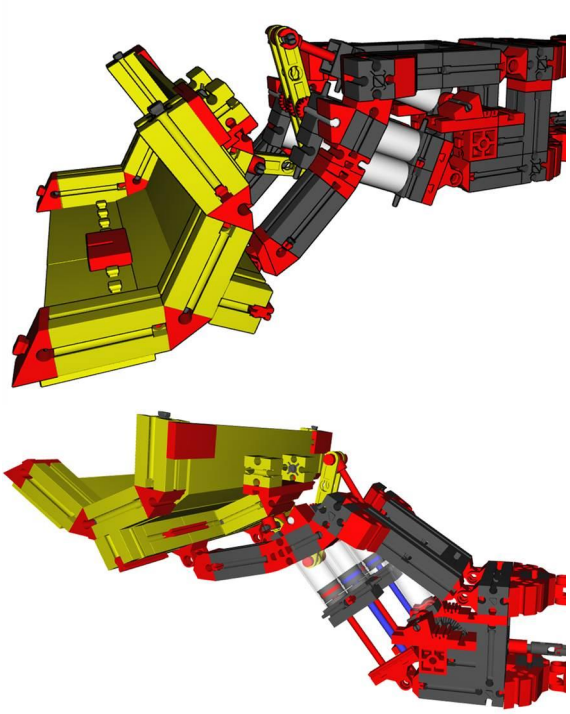


Abb. 12: Schaufel-Mechanismus komplett

Nach Antrieb und Schaufel-Mechanik kommen wir nun zum Kernstück des Radladers, der Knicklenkung. Die beiden Hydraulik-Zylinder sind links und rechts im Rahmen eingebaut (Abb. 13).

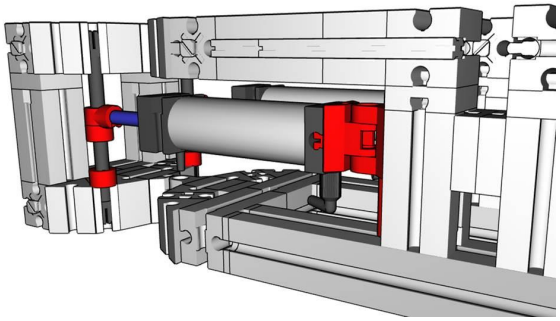


Abb. 13: Zylinder der Knicklenkung

Die Winkelstellung des Knickgelenks wird durch eine lange Dreifachstrebe (Abb. 14) unter dem Grundrahmen nach hinten zu der Linear-Führung übertragen. Diese Strebe muss keine Lenkkräfte, sondern nur die Position übertragen, d. h. es ergeben sich nur minimale Reibungskräfte. Die Schubstange ist über einen Streben-Adapter mit dem Baustein BS30 verbunden, der auf den Achsen der Linearführung verschiebbar ist.

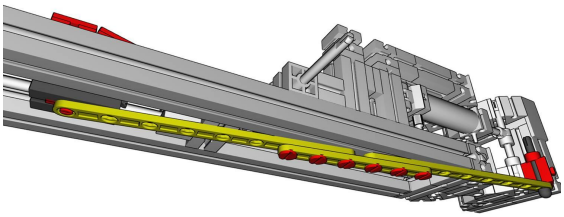


Abb. 14: Schubstange des Knickgelenks zur hinteren Linearführung

Auf dem unteren Baustein BS30 der Linearführung sind zwei Bauplatten mit drei Nuten (38428) mit Hilfe zweier Federnocken angebracht. Auf diesen sind dann, für jede Schalterseite, ein Winkelstein 15° und ein Baustein 7,5 angebracht – sozusagen als Rampe mit Verlängerung, jeweils abgedeckt mit einer Bauplatte 15x15. Damit passt die Bauhöhe ins 15-mm-ft-Raster und die Schalter gleiten weich darüber (Abb. 15).

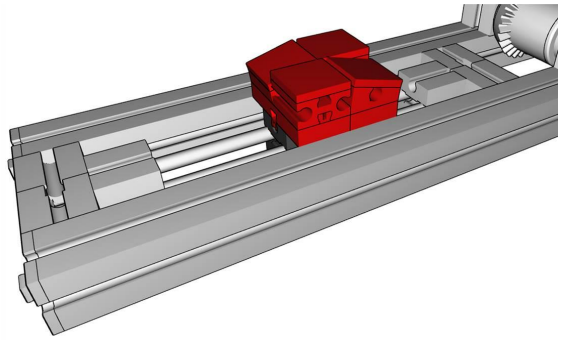


Abb. 15: Schaltflächen der Ist-Position

Über den Schaltflächen für die Ist-Position des Knickgelenks kommt nun eine zweite Linearführung. Auf dieser gleitet ein Baustein BS30 mit zwei Schaltern (Abb. 16).

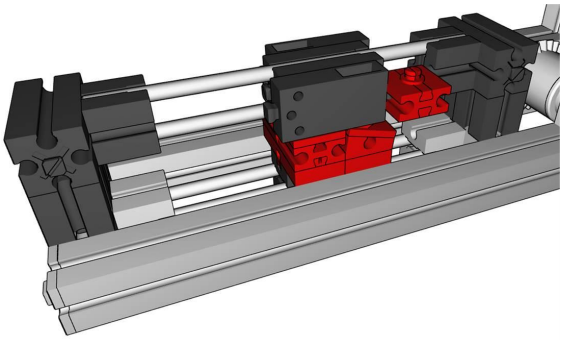


Abb. 16: Soll-Position-Schiebestück mit Schaltern

Diese Schalter werden mit den Magnetventilen elektrisch verbunden. Die Magnetventile wiederum steuern die beiden Pneumatik-Zylinder der Lenkung an.

Die Schalter sind so auf dem Baustein BS30 eingestellt, dass immer nur ein Schalter geschlossen ist.

Die Schalter werden von dem Servo der Fernsteuerung über ein Viergelenk auf der Linearführung verschoben (Abb. 17 und Abb. 18).

Die Schaltereinheit hätte, wenn sie direkt mit dem Servohebel verbunden wäre, d. h. ohne die Viergelenk-Kinematik, einen viel geringeren Verschiebeweg. Die Spiele in der gesamten Kinematik hätten dadurch einen zu großen Einfluss auf die Lenkung, die Lenkung hätte nur ca. 2–3 Stufen zwischen Mittelstellung und Endanschlag.

Durch die Viergelenk-Kinematik ergibt sich ein großer Verschiebeweg, der Einfluss der Spiele wird wesentlich geringer und man erhält eine sehr feinfühlige Lenkung.

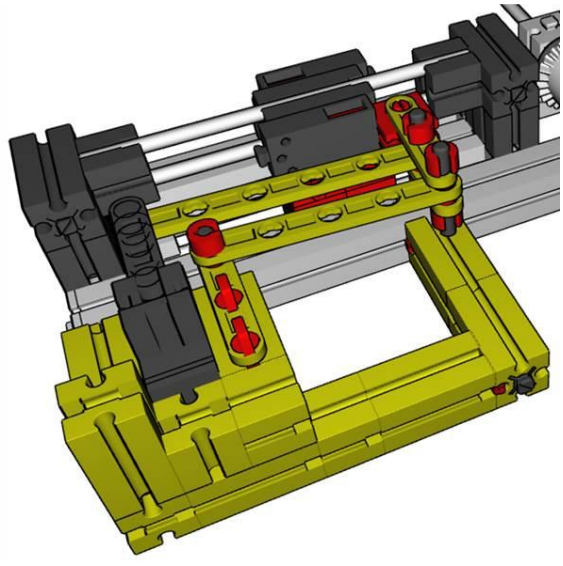


Abb. 17: Servo mit Gelenk-Mechanismus für die Soll-Vorgabe

Weicht die Soll-Position – die Schalter auf dem oberen BS30 – von der Ist-Position – die schrägen Ebenen auf dem unteren BS30 – ab, wird ein Schalter geschlossen, das Magnetventil angesteuert und der Pneumatik-Zylinder der Lenkung korrigiert den Lenkeinschlag so lange, bis wieder Soll- und Ist-Position übereinstimmen. Die Auslegung der Ansteuerung wird im nächsten Kapitel noch näher erläutert.

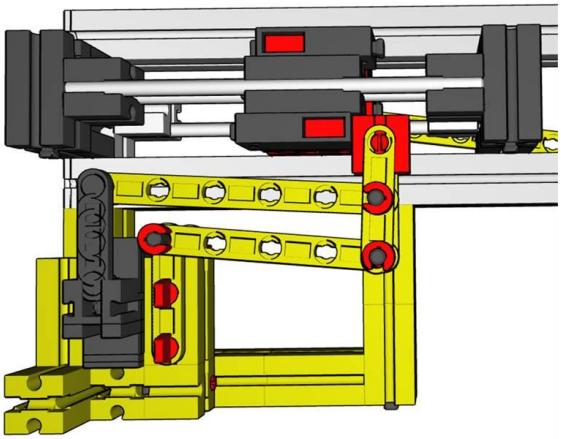


Abb. 18: Servo-Mechanismus von oben

In dem hinteren „Motorraum“ des Radladers ist auf der rechten Seite noch der Fernsteuerungsempfänger untergebracht (Abb. 19). Auf der linken Seite sind die vier Magnetventile und das Sound-Modul (Darstellung nur ähnlich) angeordnet.

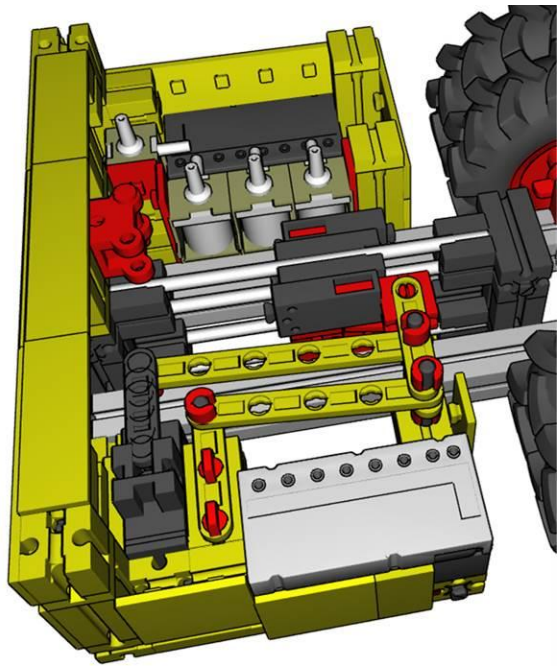


Abb. 19: Motorraum mit Fernsteuerungs-Empfänger, Magnetventilen und Sound-Modul

Die folgende Übersicht zeigt noch die elektrischen und pneumatischen Verbindungen für die Knicklenkung des Modells.

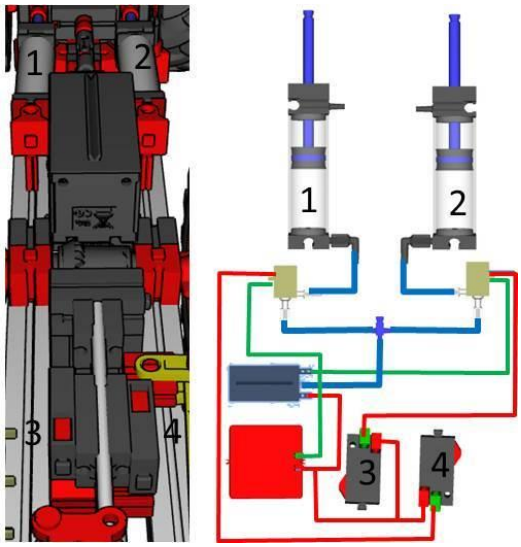


Abb. 20: Elektrische und pneumatische Verbindungen

Die Kinematik und Regelung

Zum Verständnis der Regelung ist in Abb. 21 ein Prinzipbild des „Reglers“ mit den zwei Schaltern dargestellt. Zwischen dem Schalten des einen oder anderen Schalters befindet sich eine sogenannte Totzone. Diese ist abhängig von der Kinematik und dem Spiel in der Kinematik.

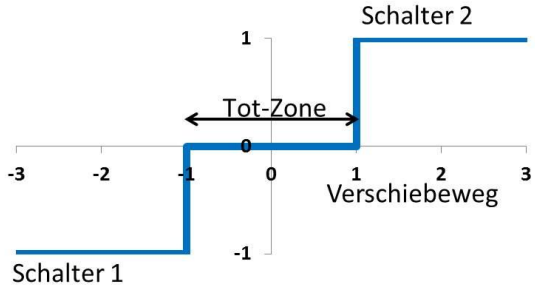


Abb. 21: Prinzipbild des Reglers

Damit sich die Totzone möglichst wenig auf die Lenkung auswirkt, sollte der Verfahrweg der Schalteinheit möglichst groß sein. Der Hebelarm des Servos ist dafür leider zu kurz. Deshalb wurde die abgebildete Hebel-Kinematik (Abb. 22) verwendet.

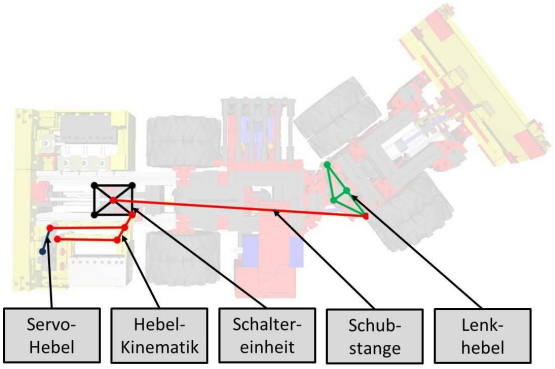


Abb. 22: Aufbau der Kinematik

Der Verfahrweg der Schalteinheit ist durch die Hebel-Kinematik wesentlich größer als bei direkter Ansteuerung und auch nahezu linear über der Servo-Auslenkung (Abb. 23).

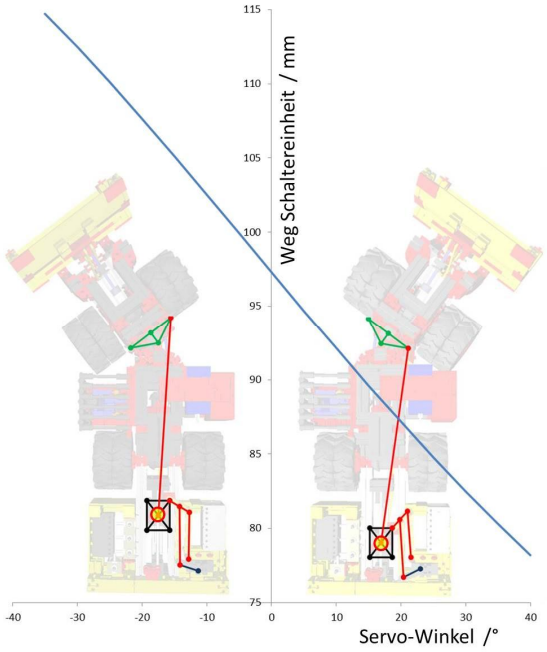


Abb. 23: Weg der Schaltereinheit

Im Original ist der Knickwinkel des Radladers auf beiden Seiten identisch. Im Modell besteht die Möglichkeit, den Lenkausschlag nach links, d.h. zum Fahrer hin, zu vergrößern. Zur rechten Seite beschränkt die Lage des Akkus den Lenkeinschlag.

Durch Verschieben des Anlenkpunktes für die Schubstange am Knickgelenk kann man das Lenkverhalten einfach anpassen.

Liegt der Anlenkpunkt in Längsrichtung gesehen auf Höhe des Knickgelenks, ergeben sich symmetrische Lenkausschläge. Durch Verschieben des Anlenkpunktes nach vorne (Abb. 24) ergeben sich größere Lenkausschläge nach links. Der Abstand des Anlenkpunktes von der Drehachse des Knickgelenks beeinflusst die absoluten Lenkausschläge.

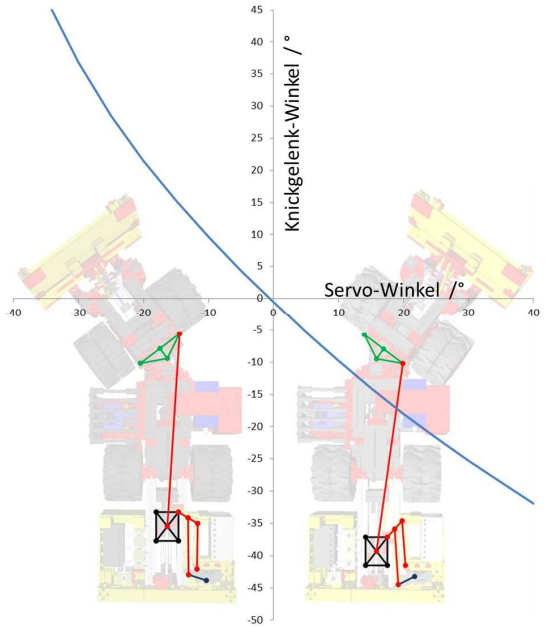


Abb. 24: Verlauf des Knickwinkels

Durch die Wahl des Anlenkpunktes lässt sich also die Lenkcharakteristik nach Wunsch einstellen.

Der Knickwinkel zur linken Seite wird durch den möglichen Weg des Pneumatik-Zylinders begrenzt.

Mit dieser Auslegung lässt sich der Radlader schon sehr gefühlvoll steuern. Wichtig sind geringe Spiele in der Kinematik.

Die notwendige Totzone des Reglers kann man noch weiter verkleinern, wenn man die Bewegung der Lenkung noch etwas dämpft. Die Dämpfung lässt sich z. B. durch die Gegenkammern in mindestens einem der beiden Lenk-Pneumatik-Zylinder erreichen. Dafür kann man die „unbenutzte“ Seite der Lenk-Pneumatik-Zylinder mit einer Drossel, z. B. einem Pneumatik-Handschaltventil, verbinden. Damit lässt sich die Bewegung dämpfen und die Totzone des Reglers, also der Abstand der beiden Schalter, weiter verringern.

Da die Regelung auch ohne zusätzliche Dämpfung sehr gut funktioniert, haben wir darauf verzichtet. Dadurch entfällt eine zusätzliche Einstellung.

Auf der ft-Convention funktionierte unser Radlader trotz „Dauereinsatz“ zwei Tage lang ohne jede Korrektur.

Resümee

Unser Ziel, ein Familienprojekt für die ft-Convention umzusetzen, hat sich erfüllt.

Wichtig waren dafür natürlich die Unternehmungen, um zunächst die Informationen über die Fahrzeuge zu bekommen. Und auch nur durch das unermüdliche Testen von Erik und dem anschließenden Weiterentwickeln konnte gemeinsam so ein stabiles Modell entstehen. Auch unsere „Mädels“ unterstützten uns mit dem Bau des Bergwerks aus Pappmaché und FischerTip. Und gemeinsam erlebten wir dann ein unvergessliches Wochenende bei der ft-Convention. Die Begeisterung von Erik wurde am Ende der ft-Convention deutlich: „Nächstes Jahr sind wir wieder dabei“.

Quellen

[1] Firmenprospekt SCHOPF Maschinenbau GmbH: *Underground Mining Equipment*, Ausgabe 04/2013.

[2] SCHOPF Maschinenbau GmbH: *SFL60XLP – Untertage-Fahrlader*.

[3] Bilderpool ft community: *Ferngesteuerter Schopf-Bergbau-Radlader*.

Tipps & Tricks

ft-Spezialteile made by TST (Teil 7)

Andreas Tacke

In einer lockeren Reihe stellt TST einige von ihm entwickelte Spezialteile vor, die so manche Lücke beim Bauen mit fischertechnik schließen. Im heutigen Beitrag geht es um Strom, bzw. eine zweipolige Verteilerleiste……

Wer kennt das nicht: Da hat man ein Modell gebaut, und zum Schluss steht mal wieder die Verkabelung an. Oft gibt es eine zentrale Stelle, an der alle Kabel zusammenlaufen und die beiden Pole Plus und Minus verteilt werden müssen. Da braucht es z. B. eine Leitung für das Interface oder die Blinkelektronik oder andere Verbraucher, die dauerhaft Strom benötigen. Schnell entsteht durch das zusammenstecken von Steckern ein bizarr aussehendes Gebilde (Abb. 1).

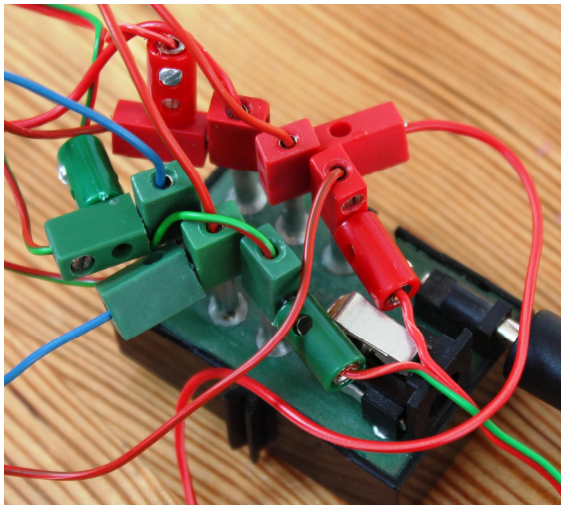


Abb. 1: Typischer „Kabelverhau"…

Früher gab es von fischertechnik Verteilerplatten, die für solch einen Zweck genau geeignet sind. Die verbreitetsten sind die einpoligen Verteilerplatten 31327 und 31328 in grün bzw. rot (Abb. 2, 3; auch [1, 2]). Es gab auch in sehr alten Kästen mal mehrpolige, aber die sind sehr selten.

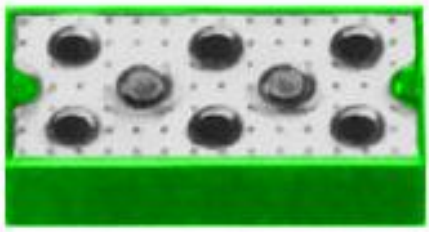


Abb. 2: Verteilerplatte 31327

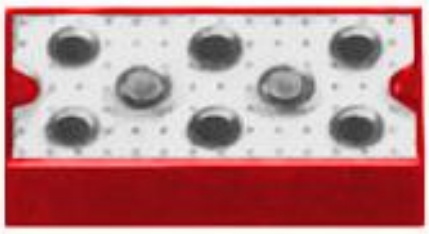


Abb. 3: Verteilerplatte 31328

Leider sind diese Verteilerplatten nicht mehr lieferbar, und ich selbst besaß nie eine solche. Da ich in meinen Modellen aber nicht so einen „Steckerwust" haben wollte, machte ich mir Gedanken, wie sich das Problem lösen lässt.

Gesucht war ein Verteiler, der zweipolig ausgeführt ist und sich perfekt mit den fischertechnik-Bauteilen kombinieren lässt.

Nach dem einen oder anderen Prototypen kam ich zu dem Entschluss, mir aus POM ein Gehäuse zu fräsen und dort zwei Messingwellen als Stromleiter zu integrieren.

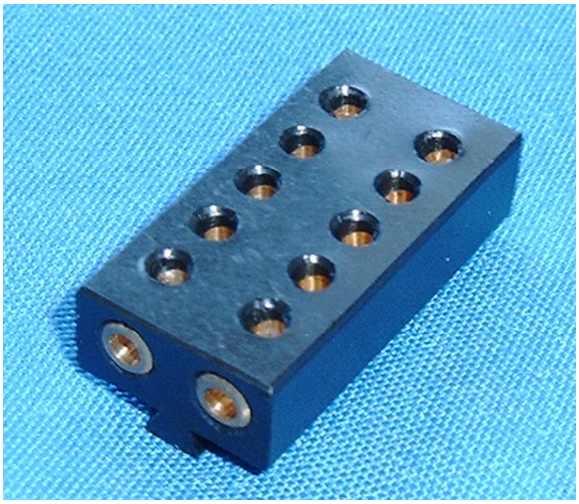


Abb. 4: Verteilerleiste zweipolig

In dieser Leiste haben nun insgesamt 14 Stecker Platz, z. B. sieben für den Plus- und sieben für den Minuspol.

Somit hat man einen zentralen Platz, an dem alles verteilt wird, und das Ganze ist dann auch noch sehr übersichtlich.

Um die Sache abzurunden habe ich das Teil noch in 3D konstruiert (Abb. 5) – es sieht da echt schick aus, finde ich.

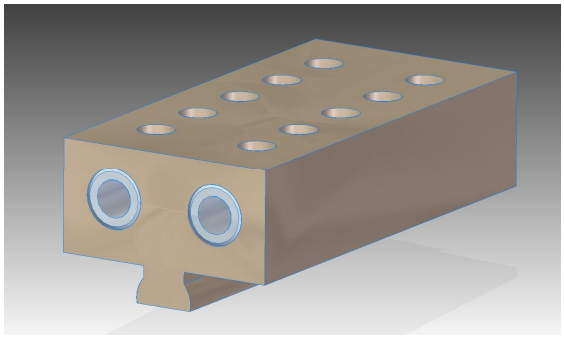


Abb. 5: Verteilerleiste zweipolig im 3D CAD

Hier zeigt sich mal wieder, wie leicht zu einem vorliegenden Problem mit ein paar kreativen Gedanken ein Spezialteil entsteht, welches perfekt zu fischertechnik passt…

Referenzen

[1] Stefan Falk: *Perlentauchen (Teil 3)*. fischertechnik-Basiswissen, ft:pedia 1/2013, S. 22-31.

[2] Dirk Fox: *Verkabelung*. Tipps & Tricks, ft:pedia 2/2013, S. 13-17.

Tipps & Tricks

Kaulquappen (Teil 4)

Harald Steinhaus

Wir rekapitulieren: Kaulquappen sind Entwürfe, die noch etwas heranreifen müssen, bis sie zu Fröschen werden. Davon muss man viele küssen (als technische Problemlösung ausprobieren). Das alles in der Hoffnung, dass ein paar davon zu Prinzen werden und nicht gar zu viele bitter schmeckende Kröten darunter sind.

Nach den Teilen eins bis drei der Kaulquappen-Serie [1] geht die Reihe jetzt dort weiter, wo Teil 3 mit den Worten „An der Lenkung wird aber noch gearbeitet“ endete: mit eben diesen.

Bekannt ist die Achsschenkellenkung nach Ackermann, bei der die Spurstange mit den anderen Teilen ein Lenktrapez bildet. Das Modell in Abb. 1 und 2 zeigt dazu einen Entwurf, der noch seine Probleme hat: das Rad gleitet sehr leicht vom Achszapfen.

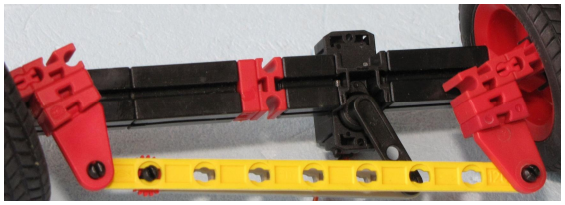


Abb. 1: Erster Entwurf Achsschenkellenkung

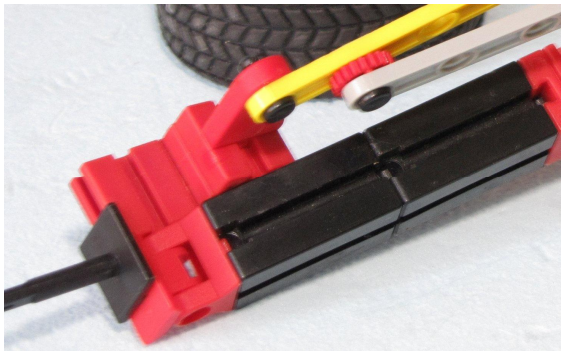


Abb. 2: Radaufhängung

Das Rad sitzt besser, wenn man es umdreht, was aber dem Ziel zuwider läuft, den Drehpunkt möglichst in die Mitte der Radaufstandsfläche zu bekommen. Außerdem wird die Rastachse mit Platte (130593) nur dadurch in Position gehalten, dass die Felge unten auf dem Gelenkstein aufsitzt und beim Fahren daran entlang schleift.

Diese Probleme hat der Entwurf in Abb. 3 nicht: das Gewicht wird ohne Tricks getragen. Die K-Achse mit Vierkant (78237) verhindert zusammen mit der Nabenmutter, dass irgendwelche Teile eigene Wege finden. Voilà, wir haben einen Prinzen!

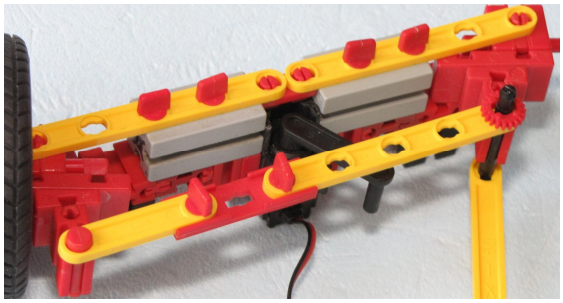


Abb. 3: Verbesserter Entwurf

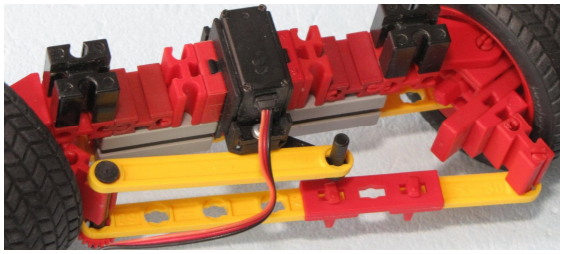


Abb. 4: Der erste Prinz

Und wenn wir gerade dabei sind: hier kommt der nächste. Ein BS30/Loch passt aufrecht in die Felge hinein, die Achse ist wieder eine K-Achse mit Vierkant, nur diesmal ist sie starr im BS30 eingeklemmt und das Rad dreht mittels Freilaufnabe. Am Drehpunkt der Achsschenkel geht es sehr eng zu, und die Reifen 65 würden schleifen. Also kommen Reifen 80 drauf; die Achse ist sowieso fürs Grobe geeignet (Abb. 5, 6).

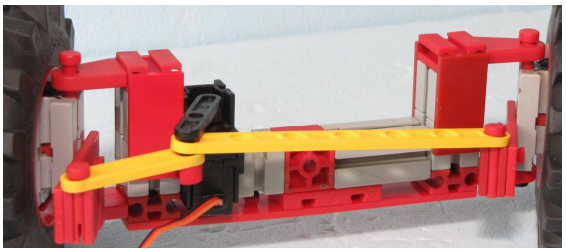


Abb. 5: … und der nächste Prinz

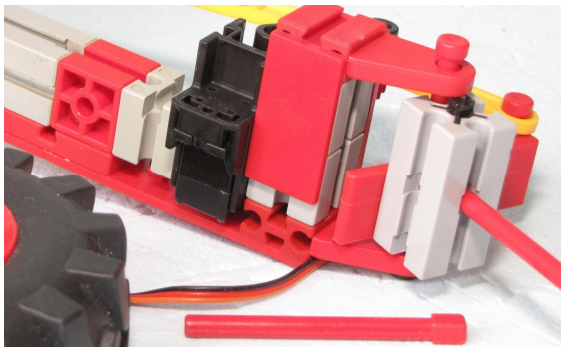


Abb. 6: Servo-Halterung und Radaufhängung

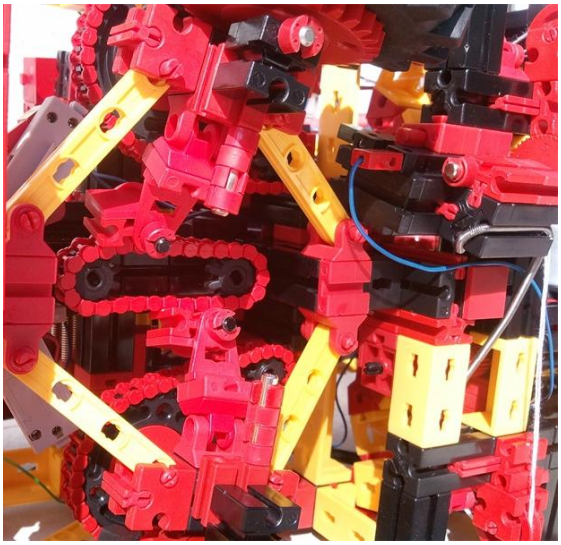


Abb. 7: Lenkung Teleskop-Mobilkran (Stefan Falk)

Mit der angetriebenen Vorderachse des Teleskopmobilkrans von Stefan Falk wurde die Tür zu einer neuen Bauart von Lenkung aufgetan (Abb. 7).

Er verwendet eine längs im Fahrzeug umlaufende Kette, die zur Linken und zur Rechten die Hebel mitnimmt, mittels derer der Lenkeinschlag der Vorderräder bestimmt wird. Die Bauart kommt gänzlich ohne Spurstange aus. Von hier aus ging es weiter mit dem Gedanken „wenn der Stefan mittels Kette die eine Seite vor und die andere zurück bewegen kann, dann geht das doch auch mit einem Hebel in der Mitte“. Und so entstand die Vorderachse des Büssing 6x4, deren Rad-Lagerung die gleichen Probleme hat wie das Modell in Abb. 1 und 2 – kein Wunder; es ist ja auch die gleiche Lagerung.

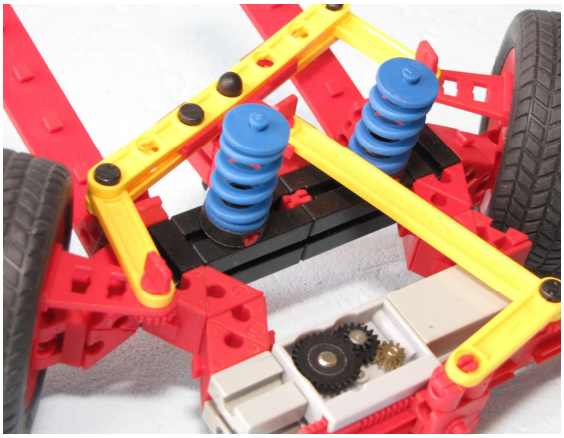


Abb. 8: Vorderachse des Büssing 6x4

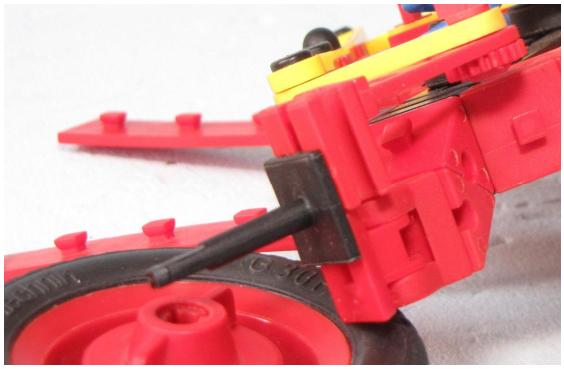


Abb. 9: Radaufhängung des Büssing 6x4

Der nächste Anlauf räumt damit auf, und führt eine um 7,5° geneigte Drehachse ein: dadurch kommt der Drehpunkt des Rades

näher an die Mitte der Aufstandfläche heran (es fehlt aber schon noch etwas). Der Lenkwürfel (31843) musste hierzu etwas gemoddet [2] werden (Sacklöcher durch eine 4-mm-Bohrung verbinden). Die Geometrie der Hebel ist aber sehr hakelig. Ein ft-Servo bringt die nötigen Kräfte nicht auf.

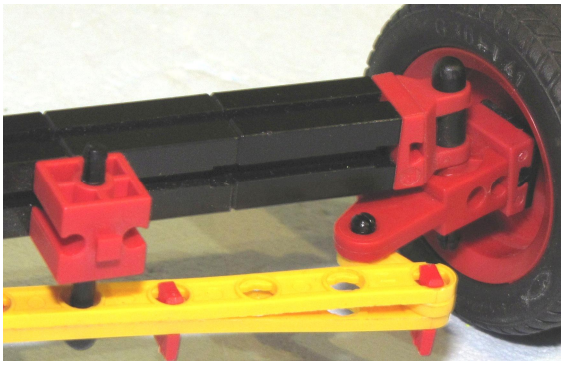


Abb. 10: „ich muss noch weitersuchen...“

Also weiter basteln... und siehe da: Die Variante in Abb. 11 hat eine saubere Geometrie, um 7,5° geneigte Drehachsen, mit den alten Reifen 60 und Freilaufnaben bleiben auch die Räder da, wo sie hin gehören, und eine Federung gibt es obendrein. Das Bild zeigt keinen Servo, aber das sollte leicht zu machen sein.

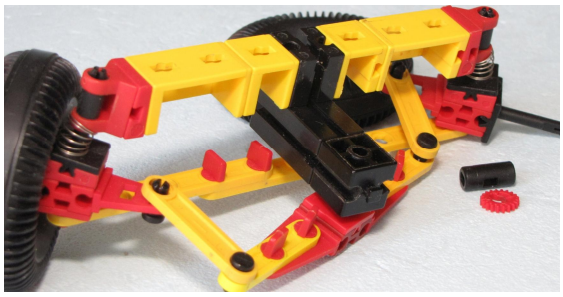


Abb. 11: „So soll es sein, so kann es bleiben“

Zu guter Letzt gibt es noch einen Prinzen: eine Lenkung mit schwenkendem Hebel, mit Frontantrieb. Diese Vorderachse wurde im Land Rover von 2013 verwendet und ist daher für regelmäßige Besucher von ft-Convention und Bilderpool keine Überraschung.

Referenzen

[1] Harald Steinhaus: *Kaulquappen (Teil 1-3)*. Tipps & Tricks, ft:pedia 1/2011, S. 22-27; ft:pedia 2/2011, S. 9-13; ft:pedia 3/2012, S. 24-26.

[2] Harald Steinhaus: *Neue ft-Teile selbst gemacht: Teile-Modding*. Tipps & Tricks, ft:pedia 3/2011, S. 20-24.

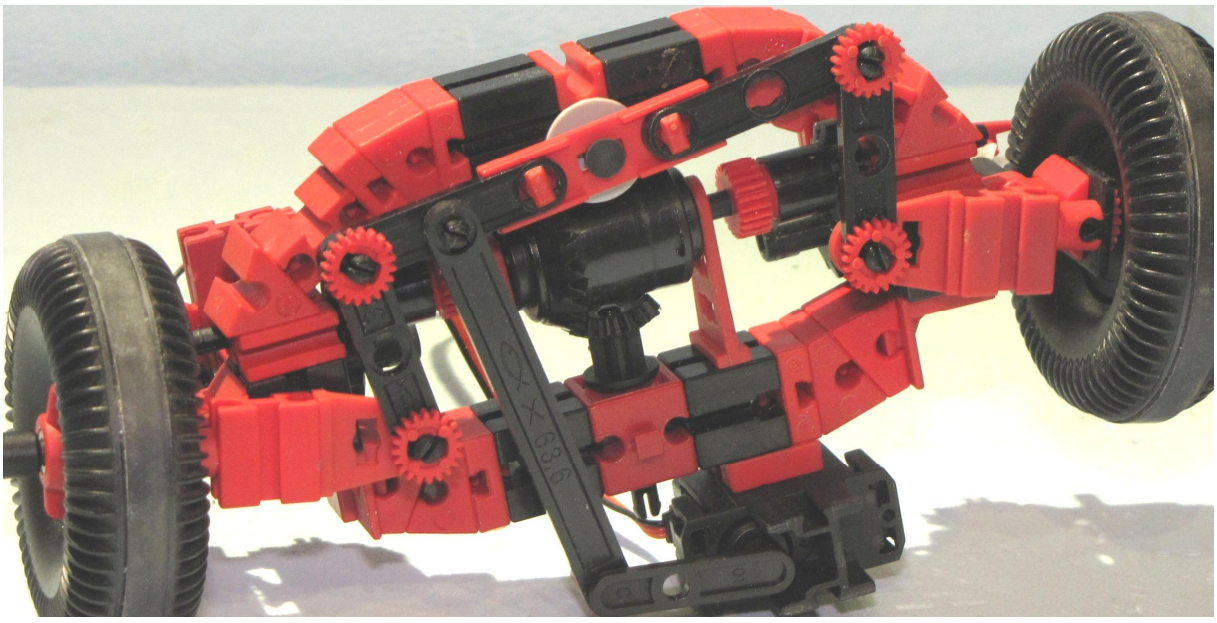


Abb. 12: Lenkung des Land Rover (2013)

Computing

Parallel-Interface durch Arduino gesteuert (1)

Jens Lemkamp

Man schrieb das Jahr 1984, als wir mit dem C64, Schneider CPC, später auch Amigas oder Atari STs und IBM-PCs die ersten Computing-Modelle steuern konnten. Eine neue Welt, in der wir durch selbst geschriebene Programme in der Programmiersprache BASIC Roboter-Modelle zum Leben erweckten. „Bit“ und „Byte“ waren damals noch Fremdworte. Dieses Projekt zeigt, wie man die alten Interfaces mit moderner kostengünstiger Steuer-Elektronik wieder nutzbar machen kann – auch noch „autonom“, also fast wie mit einem TX Conroller.

Worum geht es?

Wir möchten mit möglichst wenig handwerklicher Arbeit und geringem Hardwareaufwand unsere ft-Modelle kostengünstig steuern. Mit einfachen, schnellen Programmen, und das Ganze autonom, das heißt dass der PC oder Mac nicht die ganze Zeit mitlaufen muss.

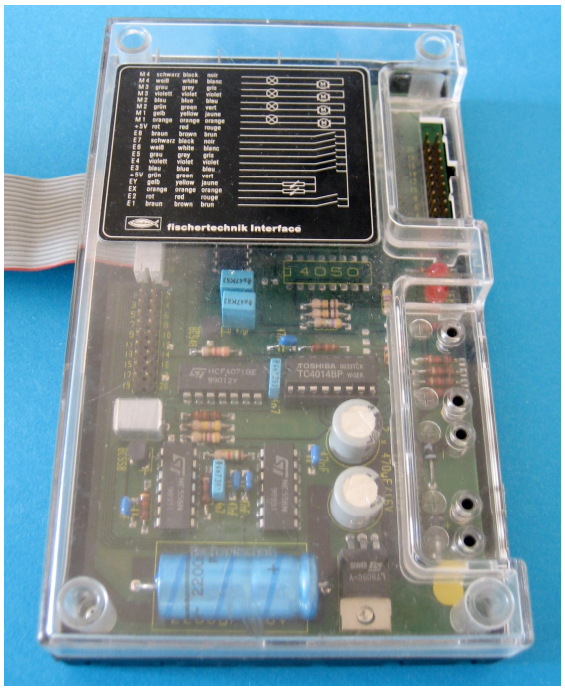


Abb. 1: Parallel-Interface

Programmiert wird in einer vereinfachten Version von C unter Windows, MAC-OS oder Linux. Gesteuert wird mit dem recht bekannten Arduino, einem kleinen Einplatinen-Computer, der insbesondere für Computer-Anfänger und Medienkünstler von einem netten italienischem Professor (den ich sogar einmal selbst kennen lernen durfte) als Studentenprojekt entwickelt wurde.

Die technischen Daten des Interfaces:

- vier Motorausgänge (Linkslauf, Rechtslauf, Stopp)
- acht Digitaleingänge für Taster, Relais etc.
- legendäres Notaus-System (bei Programmabsturz oder Abschalten des Arduino-Controllers werden alle Ausgänge nach ca. 0,5 s automatisch deaktiviert)

Wir nutzen sechs analoge Eingänge des Arduinos zur Abfrage von Potentiometern, Fotowiderständen, Heißleitern oder Kaltleitern (sprich NTC und PTC).

Das System bleibt dabei offen – wir können es jederzeit mit I²C-Bus-fähiger Hardware erweitern, um theoretisch beliebig viele Ein- und Ausgänge, Speicher, Uhrenbausteine, Anzeigeelemente, Gyros-/Kompass-Sensoren usw. anzuschließen.

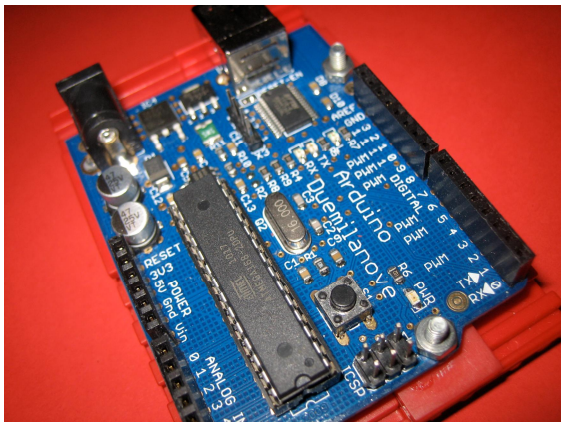


Abb. 2: Arduino Duemilanove

In der Grundversion können wir nach kurzer Zeit die vier Ausgänge steuern, acht Taster und die sechs hochauflösenden Analog-Eingänge abfragen.

Für das Anfertigen der Hardware benötigt ein mittelmäßig versierter Bastler nur ca. 30 bis 60 Minuten, geübte Löter schaffen es *sehr* viel schneller.

Man nehme:

- Ein fischertechnik-Interface, z. B. 30562 (C64 / VC20 / PC; Abb. 3),
- ein Arduino-Board, z. B. „Uno" oder „Duelimanove" (Abb. 2),
- ein paar Kleinteile (gibt's in jedem Elektronik-Versand, Stückliste folgt mit der Bauanleitung später im Artikel).

Werkzeug (Minimalausstattung):

- Elektronik-Lötkolben (keine Angst – es ist nichts Wildes damit zu tun),
- ein kleiner (Elektronik-)Seitenschneider
- evtl. eine Spitzzange oder Pinzette und
- etwas Lötzinn.

Vor rund 30 Jahren (eine irre lange Zeit, oder?) hatte man das Problem, dass die Computer kaum genormte Schnittstellen zur Außenwelt boten – USB war noch längst nicht erfunden. Hier sei es exemplarisch anhand des legendären C64 beschrieben: der Hersteller Commodore hatte für den Home-Computer (so hießen die Geräte damals) einen so genannten *User Port* als Schnittstelle für die Außenwelt vorgesehen – acht Bit standen zur Verfügung, sowie einige wenige zusätzliche Steuerleitungen. Darüber sollten nun vier Motoren mit jeweils zwei Drehrichtungen, acht Taster und zwei analoge Eingänge „bedient" werden. Wie geht das? Mit einigen Tricks war das tatsächlich möglich.

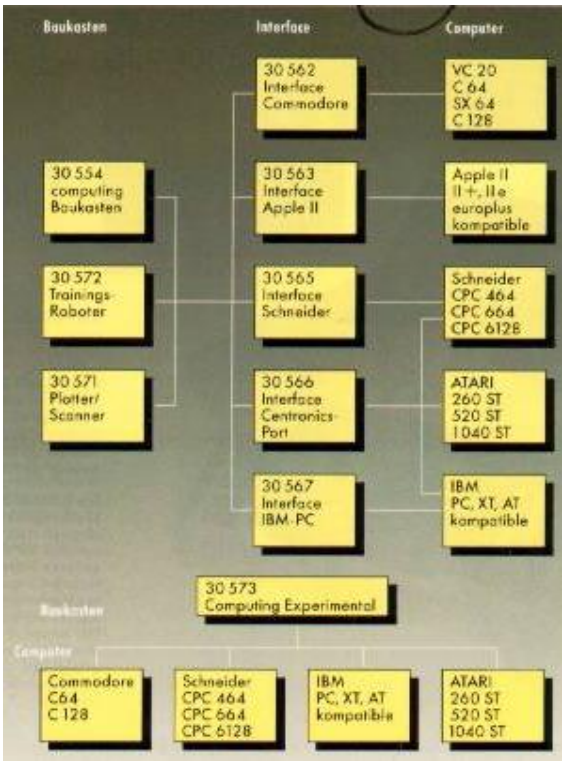


Abb. 3: Systemübersicht 1987 [1]

Um einen Motor in zwei Richtungen zu steuern, benötigt man zwei Bits, welche insgesamt vier verschiedene Zustände annehmen können:

0 0 – Motor aus

0 1 – Motor Rechtslauf

1 0 – Motor Linkslauf

1 1 – Motor aus

Der vermeintlich doppelte Zustand „11" ist sinnvoll; später benötigen wir ihn noch für eine andere Funktion.

Für vier Motoren benötigen wir also je zwei Bits: 4 x 2 Bit = 8 Bit = 1 Byte. Für acht Digitaleingänge benötigen wir auch

genau ein Byte. Wie bekommt man nun diese ganzen „Leitungen“ aus dem Computer bzw. dort hinein?

Mit Schieberegistern! Man taktet die einzelnen Bits nacheinander ins Interface und holt auch die Daten genauso da heraus.

So sieht das fischertechnik-Interface aus den 1980ern (grob) von innen aus:

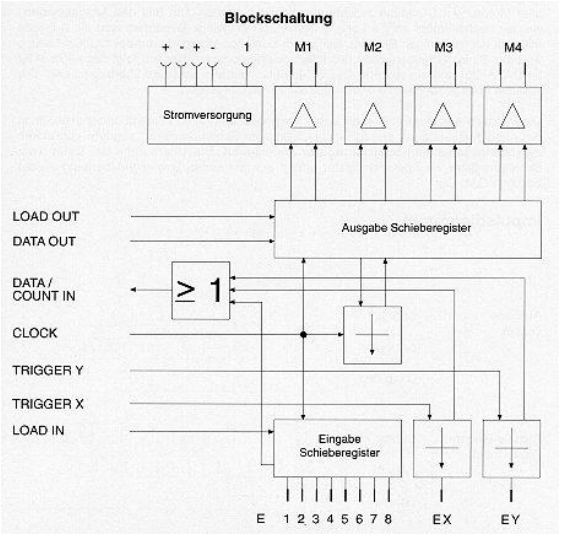


Abb. 4: Blockdiagramm [1]

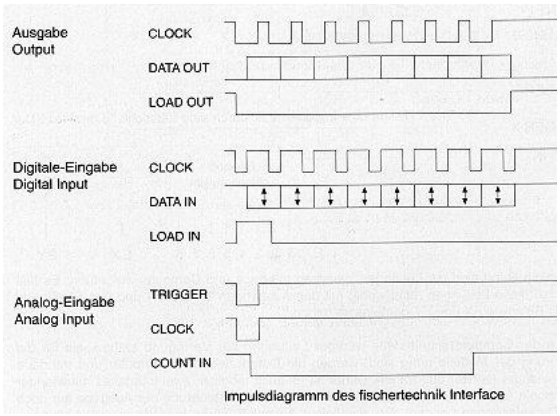


Abb. 5: Impulsdiagramm [1]

Und wie funktioniert nun so ein Schieberegister? Wir geben nacheinander (also seriell) Bit für Bit unsere gewünschte Ausgabeinformation in das Register. Dazu gibt es einen Eingang für die Bits (DATAOUT), einen Takteingang (CLOCK) und einen Übernahmeeingang (LOADOUT).

IN und OUT ist jeweils aus Computer-Richtung gedacht. DATAOUT ist also z. B. ein Ausgang des Computers bzw. Eingang des Schieberegisters.

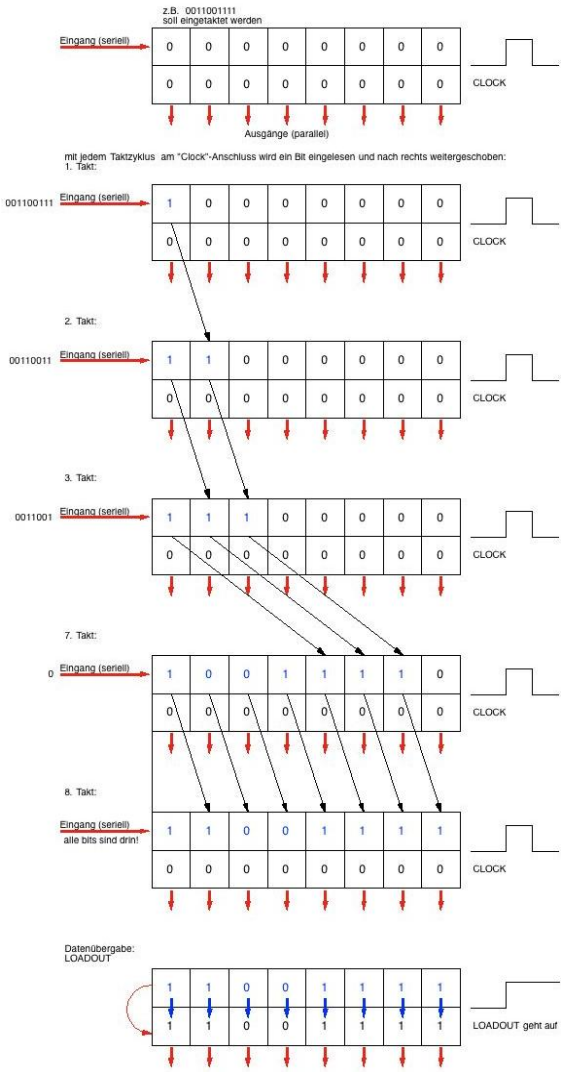


Abb. 6: Impulsdiagramm Schieberegister

Zunächst schieben wir also unsere acht Bit in das Register. Man beginnt ganz rechts mit Motor 1 (M1), dann kommen M2, M3 und zum Schluss M4. Nach jedem angelegten Bit geht CLOCK einmal kurz von 0 auf 1 und dann wieder auf 0. Nach acht Taktzyklen ist das ganze Byte im Schieberegister und wir setzen LOADOUT auf 1 – damit wird das Byte an die Motorstufe übergeben. Je nach Bitmuster drehen ein oder mehrere unserer Motoren nun in die (hoffentlich) richtige Richtung.

Die Ausgangssituation ist wie folgt:

- CLOCK = 1 (oder auch +5 V)
- LOADOUT = 1
- DATAOUT = 0 (0 V)

Und nun geht's los. Wir wollen jetzt ein komplettes Byte 'reinschieben, sagen wir „0 0 0 0 0 0 0 1" (Motor 1 Rechtslauf, alle anderen Motoren bleiben aus).

Dazu setzen wir zunächst LOADOUT auf 0. Immer wenn CLOCK nun von 0 auf 1 wechselt, wird „geschoben" – d. h. wir bringen unser Byte nun Bit für Bit ins Schieberegister und schalten zum Schluss als Übergabekommando noch LOADOUT auf 1. Und schon dreht sich unser Motor 1.

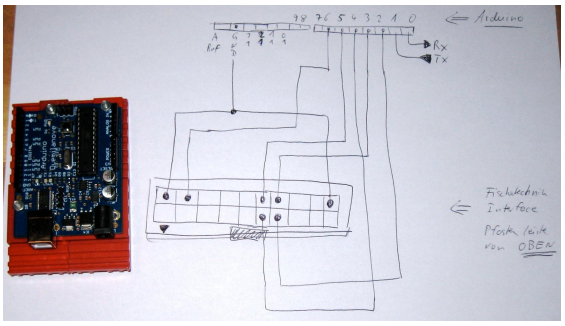


Abb. 7: Schaltskizze

Die Eingänge werden mit einem ähnlichen Verfahren eingelesen:

- Impuls auf LOADIN, um die Eingänge ins Schieberegister zu „laden"
- CLOCK setzen wir auf 1, und schon steht der Wert 0 oder 1 am DATAIN und wir lesen ihn in den Arduino. War zum Zeitpunkt des LOADIN der Taster am E1 (Eingang 1) des Interfaces gedrückt, bekommen wir eine 1, sonst eine 0.
- CLOCK geht auf 0 und dann wieder auf 1 – und Schwupps, E2 liegt am DATAIN.
- Nach sieben CLOCKs haben wir alle Eingänge abgefragt.

Hardware

Der Adapter zur Verbindung des Arduino mit dem Pfostenstecker des Interfaces kann auf verschiedene Arten nach folgender Belegung angefertigt werden:

Interface-Anschluss	Pin des Pfosten-steckers von oben (Steckseite)	Arduino PIN-Nr.
CLOCK	□□□□□X□□□□ □□□□□□□□□□	5
DATAOUT	□□□□□□□□□□ □□□□□X□□□□	4
LOADOUT	□□□□□□□□□□ □□□□□□X□□□	2
DATAIN	□X□□□□□□□□ □□□□□□□□□□	6
LOADIN	□□□□□□X□□□ □□□□□□□□□□	3
GND	X□□□□□□□□X □□□□□□□□□□	GND

Tabelle 1: Anschlussbelegung des Adapters

Der Adapter nach Abb. 8 ist schnell zusammengelötet und erfordert wenig Materialeinsatz, ist aber auch „labil". Schrumpfschlauch und Heißkleber können da helfen.

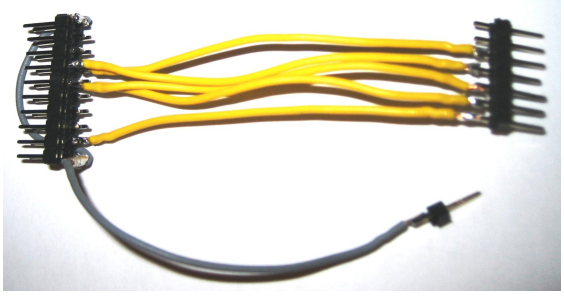


Abb. 8: Adapter (Typ „einfach")

Für 3 bis 10 € gibt es die so genannten „Screw-Shields", z. B. im großen Auktionshaus im Netz oder bei diversen Arduino-Zubehör-Distributoren. Sie erlauben zudem den komfortablen Zugriff auf alle weiteren Arduino-Anschlüsse.

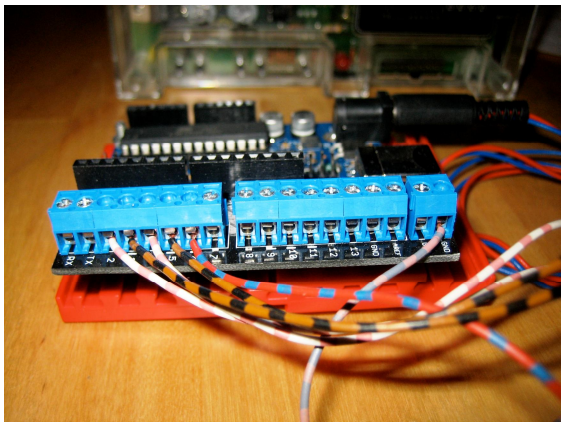


Abb. 9: Adapter (Typ „Screw-Shield“)

Die Interface-Seite könnte dann wie beim Adapter-Typ „einfach“ gelöst werden oder evtl. noch mit der Platinen-Stiftleisten-Variante wie in Abb. 12.

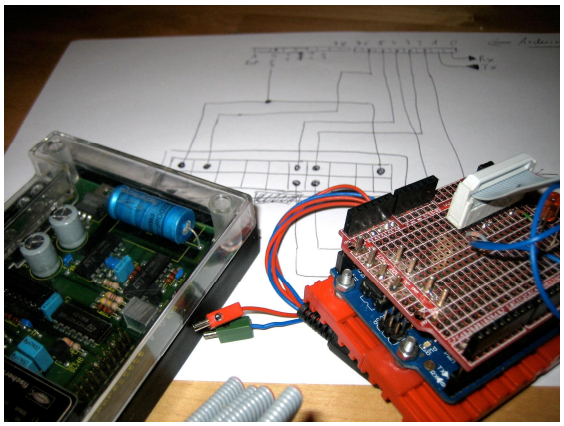


Abb. 10: Die „Luxusversion“ des Aufbaus

Abb. 10 zeigt meine Luxus-Version (rote Platine, unten/Mitte) „Protoshield“. Diese Platine kann auf das Arduino-Board gesteckt werden, und dort ist dann der 20polige Pfostenverbinder für das Interface fest aufgelötet.

```
int clock = 5;
int dataout = 4;
int loadout = 2;
int datain = 6;
int loadin = 3;
int outbyte = B00000000;  //M4 M3 M2 M1  Motoroutputbyte
                          //je 2 Bits beschreiebn einen Motor

void setup(){
  pinMode(clock, OUTPUT);
  pinMode(dataout, OUTPUT);
  pinMode(loadout, OUTPUT);
  pinMode(datain, INPUT);
  pinMode(clock, OUTPUT);

  digitalWrite(clock, LOW);
  digitalWrite(dataout, LOW);
  digitalWrite(loadout, LOW);

  shiftOut(dataout, clock, MSBFIRST, 0);   // alle (Motor) Ausgänge auf 0
}
void loop(){
  outbyte = B00000001;  // M4 M3 M2 M1 ist die Reihenfolge der Motoren
                        // hier M1 soll eingeschaltet werden
  digitalWrite (loadout, LOW);
  shiftOut(dataout, clock, MSBFIRST, outbyte); //ins Schieberegister shiften
  digitalWrite (loadout, HIGH);  // einschreiben / übertragen an OutputStufe im Interface
}
```

Abb. 11: Arduino-Sketch-Ausgabe

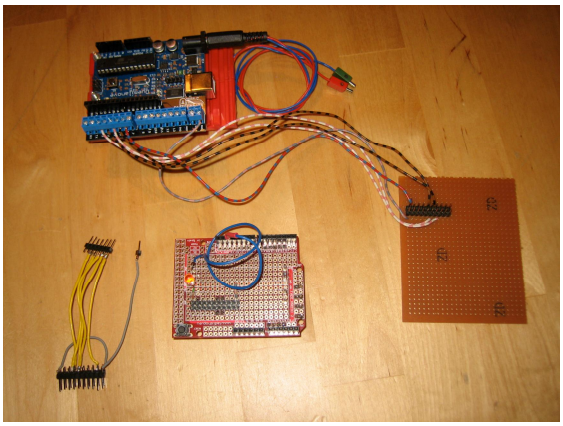


Abb. 12: Adapter-Type „Platine“

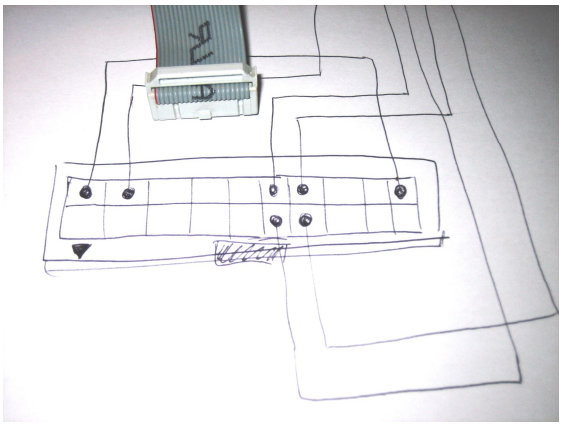


Abb. 13: Pinzuordnung mit Interface

```
int ledpin = 13;
int clock = 5;
int dataout = 4;
int loadout = 2;
int datain = 6;
int loadin = 3;
// Speicherarray für Eingänge:
int e[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};// E1 = e[1]  - e[0] bleibt ungenutzt!

void setup(){
  pinMode(ledpin, OUTPUT);
  pinMode(clock, OUTPUT);
  pinMode(dataout, OUTPUT);
  pinMode(loadout, OUTPUT);
  pinMode(datain, INPUT);
  pinMode(loadin, OUTPUT);

  digitalWrite(clock, LOW);
  digitalWrite(dataout, LOW);
  digitalWrite(loadout, LOW);
  digitalWrite(datain, HIGH);
  digitalWrite(loadin, LOW);
}

void loop(){
     getinputs();// routine zum Einlesen der Eingänge aufrufen

  if (e[1] == 1) {
    digitalWrite  (ledpin,HIGH); //testled an, wenn E1 gedrückt wurde
    }
  if (e[2] == 1) {
    digitalWrite  (ledpin,LOW); //testled aus, wenn E2 gedrückt wurde
      }
}

void       getinputs()           {// Input Status in e[x] Variablen legen
  digitalWrite  (loadin,HIGH);
  digitalWrite  (clock,LOW);
  digitalWrite  (clock,HIGH);
  digitalWrite  (loadin,LOW);      // Registereingang geholt
  for (int i = 8; i>0; i--){
    digitalWrite  (clock,HIGH);    // Eingang holen
     e[i] =digitalRead    (datain);// Bit lesen & speichern
    digitalWrite  (clock,LOW);     // 1. Zyklus bzw. 1. Bit ist übertragen (E8)
    }
}
```

Abb. 14: Arduino-Sketch: Einlesen der Eingänge

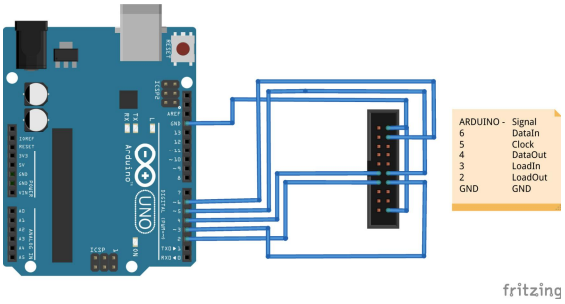


Abb. 15: Schaltbild

Software

Ich setze hier voraus, dass ihr euch selbst in die Programmierung und die grundsätzliche Vorgehensweise zur Programmierung eines Arduinos einarbeitet (oder bereits eingearbeitet habt).

Hier zwei Beispielprogramme:

Zunächst werden die Anschluss-Pins mit Konstanten benannt, und zwar so wie die Funktionen des jeweiligen Pins auch bei fischertechnik im Blockschaltbild benannt sind.

Durch die Funktion *setup()* wird jeder Pin entweder als Eingang oder Ausgang deklariert und anschließend zunächst der Ausgang genullt: Eine 0 wird mittels ShiftOut in den Ausgang des Interfaces geschoben. Alle Motoren sind damit aus.

Im Hauptprogramm *loop()* starten wir einen Motor (M1). Der läuft dann, bis das Programm beendet wird.

Das Einlesen der Eingänge habe ich diskret programmiert. Die Funktion *getinputs()* überträgt den Status der acht Eingänge in ein Variablenarray *e[1]* bis *e[8]*. Das Element *e[0]* wird *nicht* genutzt.

Versuchsaufbau zum Testen: Zwei Taster an E1 und E2 können die LED auf dem Arduino-Board ein- (E1) oder ausschalten (E2).

Fortsetzung folgt

Aufbauend auf diesem Kurzeinstieg sind weitere Beiträge in einer kleinen Serie geplant. Darin werden die folgenden Themen behandelt:

- Abfrage analoger Eingangsgrößen wie Widerstände, Sensoren, Spannungen
- Aufbau einer Tastaturmatrix mit ft-Tastern
- Beispielprogramme und Bau einiger Modelle des Ur-Computing-Kastens 30554 (1984)
- Anbindung der 1980er-Jahre-Elektronik (Schwellwertschalter und Leistungsstufe mit Lautsprecher)

Quellen

[1] Ulrich Müller: *ft-Computing-Website*

[2] Arduino Homepage

[3] Open Source Hardware-Projekt

Bezugsquellen

Elektronik-Teile:

[4] Conrad Electronic SE

[5] reichelt elektronik GmbH & Co. KG

[6] Pollin Electronic GmbH

Arduino:

[7] Watterott electronic GmbH

Computing

Arduino mit dem TX verbinden

Marco Ahlers

Wer die Grenzen des Robo TX Controllers sprengen möchte, kann auf den Nachfolger TXT warten, zu Lego Mindstorms wechseln, sich grämen, ein eigenes Mikrocontroller-Board bestücken [1] – oder einen Arduino zu Hilfe nehmen.

Hintergrund

Der Robo TX-Controller bietet eine leistungsfähige Plattform für fischertechnik Computing- und Robotics-Projekte. Die Möglichkeit jedoch, den Arduino für ausgelagerte Kalkulationen zu verwenden, hat mich fasziniert. So könnte man bereits im Arduino Kompasswerte mittels Kalmanfilter von Spitzen befreien oder Sensorwerte zu weiteren Berechnungen nutzen und dem TX-Controller nur noch die entscheidenden Ergebnisse mitteilen.

Des Weiteren bietet die Arduino-Programmiersprache eine an C angelehnte freiere Programmierung, als dies mit Robo Pro möglich ist. Außerdem sind interessante Erweiterungen möglich, wie z. B.:

- **SD-Karten:** Da man am Arduino SD-Karten betreiben kann, lassen sich auch offline Daten, wie z. B. Messwerte, die ein mobiler Roboter unterwegs aufgezeichnet hat, dauerhaft speichern und weiterverarbeiten.
- **Kamera:** Am Arduino lassen sich auch Kameras anschließen um Bilder aufzunehmen oder mit einem zusätzlichen Raspberry PI auch Bilder auszuwerten und zu verarbeiten.
- **Text to Speech (TTS):** Mit TTS-Modulen kann man seinen fischertechnik-Roboter zum Sprechen bringen.

Sensorenwerte vom Arduino abfragen

Konkret war mein Ziel, an den Arduino angeschlossene Sensoren auszuwerten und dann jeweils die Variablen nur bestimmter Sensoren im TX-Controller auszulesen.

Der I²C-Anschluss des TX Controllers bietet dabei eine gute und einfache Möglichkeit, den Arduino anzubinden. Dabei sind einige Dinge zu beachten, die ich im Folgenden beschreibe.

Die Verbindung

Eine serielle I²C-Verbindung besteht aus vier Anschlüssen:

- der 5 V-Versorgung,
- dem Masseanschluss (Erde),
- einer Taktleitung (SCL) und
- dem Datenbus (SDA).

Jetzt kann man sich näher mit I²C beschäftigen, muss man aber nicht. Entscheidend ist, dass man über das I²C-Protokoll mehrere Geräte miteinander verbinden kann und diese jeweils von einem Hauptcontroller (dem Master, z.B. einem TX-Controller) direkt anspricht und Signale bzw. Werte sendet und empfängt.

Sind mehrere Personen in einem Raum, kann man, kennt man die Namen, jeden direkt rufen, ansprechen und Aufgaben

verteilen. So in etwa funktioniert dies auch bei I²C. Dazu besitzt jedes angeschlossene Gerät eine eigene 7-bit-Adresse. In der ft:pedia wurde der I²C-Anschluss bereits näher beleuchtet, daher gehe ich hier nicht weiter darauf ein [2].

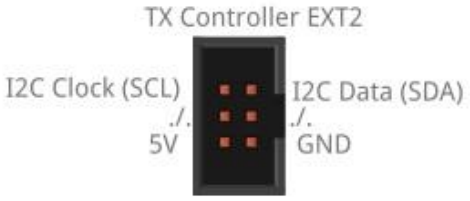


Abb. 1: I²C (EXT2) Belegung am TX-Controller

Der Arduino hat eine eigene Energieversorgung, was ja auch gut ist, spart man so Energie beim TX-Controller. Wie stellt man aber nun die Verbindung ganz praktisch her? Die beiden 5 *V*-Anschlüsse des TX-Controllers und des mit eigenem Strom versorgten Arduino miteinander zu verbinden ist keine gute Idee. Der Masseanschluss jedoch kann (und muss sogar) gemeinsam benutzt werden. Den 5 *V*-Ausgang würde man nur benötigen, wenn man z. B. einen per I²C an den TX-Controller direkt angeschlossenen Sensor oder den Arduino selbst so mit Strom versorgen möchte.

Man verbindet also die mit SCL, SDA und GND belegten Pins des EXT2-Anschlusses am TX-Controller mit denen des Arduino: Fertig, Anschluss erledigt. Ich habe dazu eine I²C-Buchse genutzt, dort das Extension-Kabel des TX-Controllers eingesteckt und an die Buchsenkontakte die nötigen Drähte angelötet. Der GND-Draht wird dann mit dem GND des Arduino verbunden, ebenso SDA und SCL (siehe Abb. 2). Dies macht man am besten mit einem *Breadboard*. Auf der folgenden Seite (Abb. 3) seht ihr die komplette Schaltung inklusive dreier Infrarotdistanzsensoren.

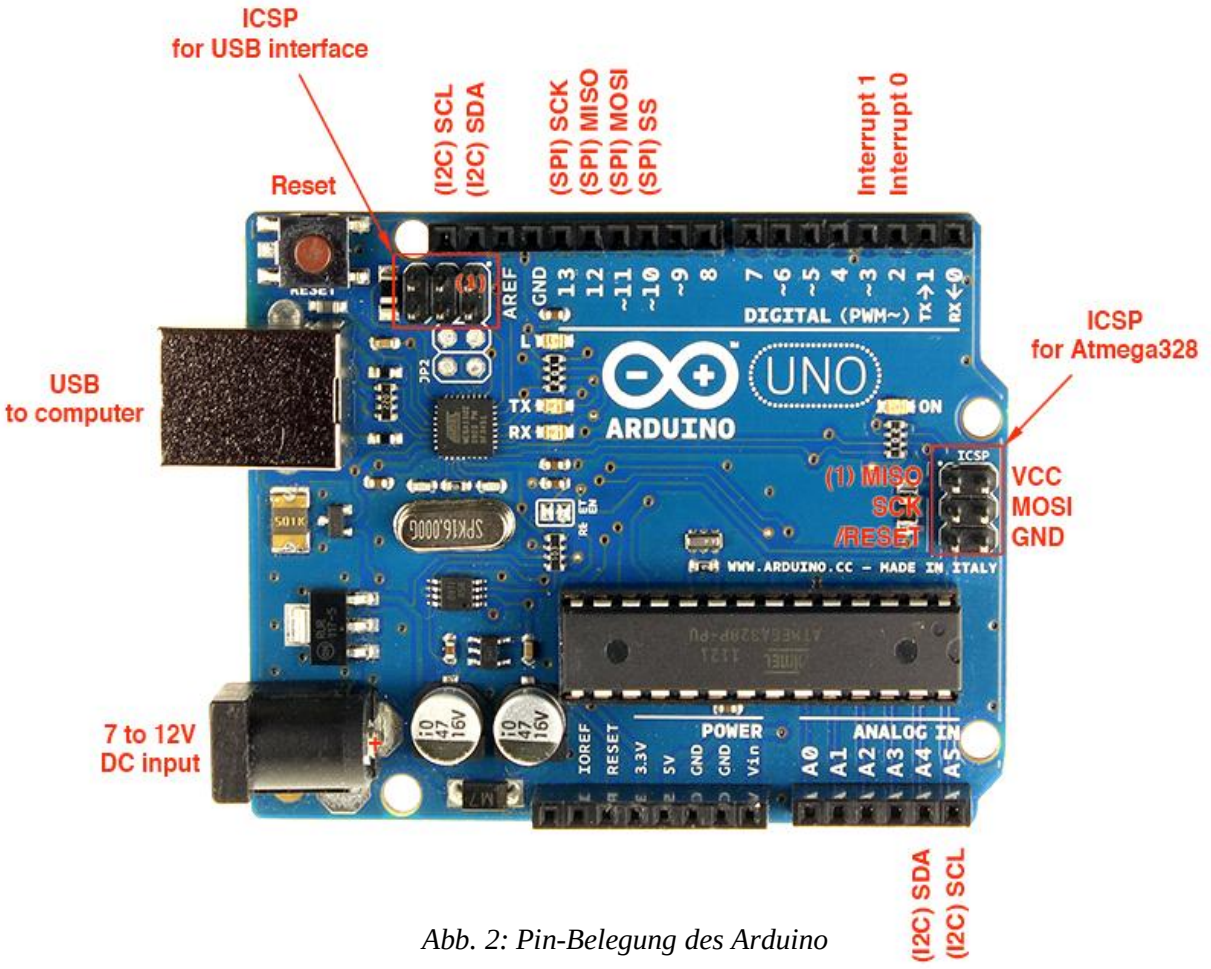


Abb. 2: Pin-Belegung des Arduino

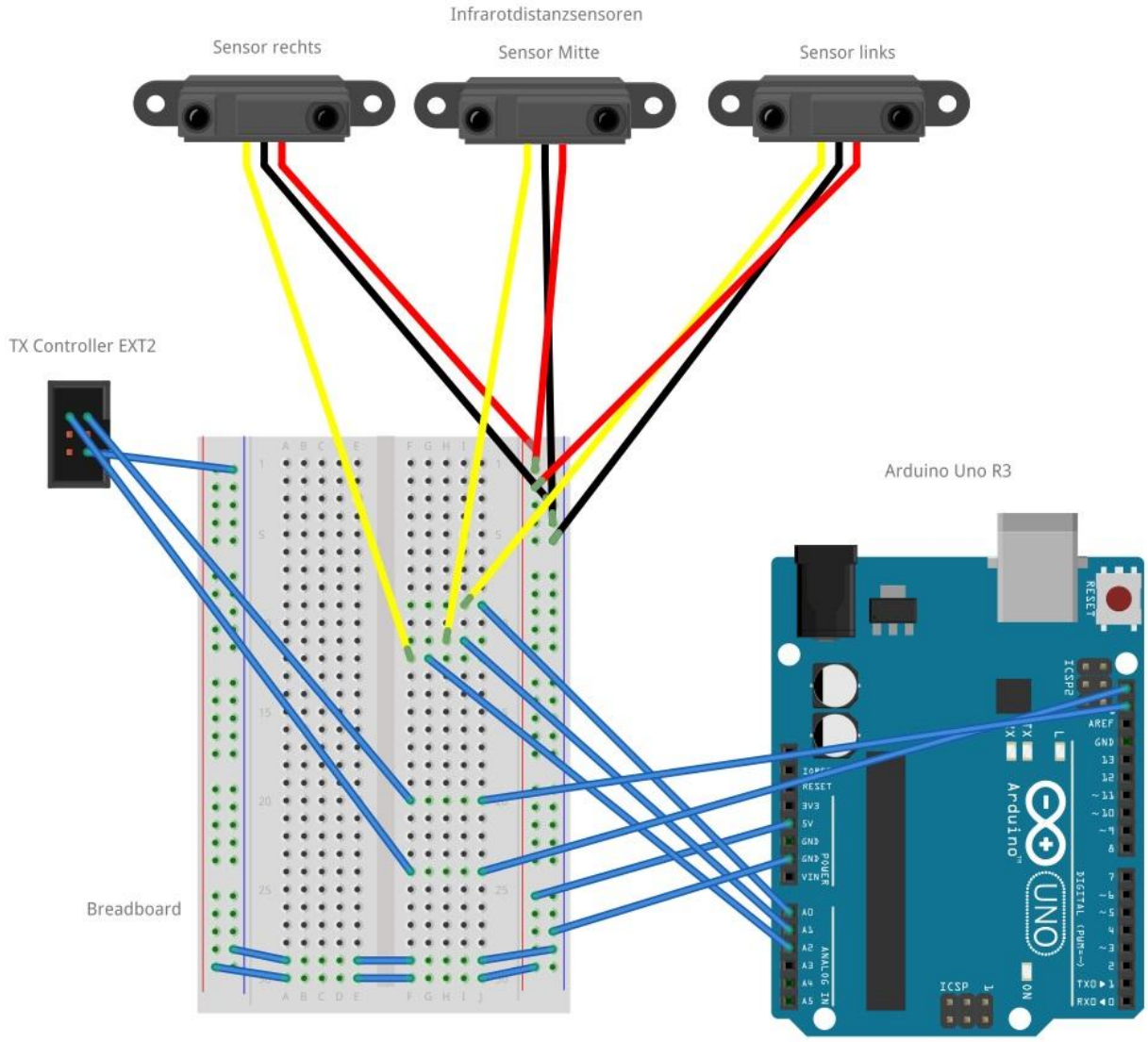


Abb. 3: Schaltplan TX Controller via I2C an Arduino

Diese drei Sensoren werden wir dann einzeln vom TX Controller aus abfragen können. Sie liefern analoge Werte, also gehören sie an die Analog-IN-Pins.

Die drei Sensoren werden jeweils in einen der analogen Arduino Eingänge gesteckt. Das ist ganz wichtig, man weiß also ganz genau: A0 ist der linke Sensor (vorne links am späteren Fahrzeug), A1 der mittlere und A2 der an der rechten Vorderseite des Fahrzeugs.

Auf der folgenden Seite seht ihr in Abb. 4 den ganzen mobilen Roboter. Auf die vorne sichtbaren Infrarotsensoren habe ich noch zwischen GND und 5 *V* 100 *µF*-Kondensatoren gesteckt, die gleichen ein wenig das Zittern der Messwerte aus. Die Sensoren habe ich mit Heißkleber auf Grundbausteine geklebt. Der Turm ist drehbar und so kann man längere Abstände mit Ultraschallsensoren messen. Ich habe nur zwei, daher dreht der Turm nach links und rechts, um seine Umgebung zu scannen. Die Ultraschallsensoren werden in diesem Beispiel aber nicht genutzt.

Hier die Verdrahtung aus der Nähe (stimmt nicht ganz mit der Grafik oben überein, aus Drahtlängengründen habe ich einige Drähte direkt in den Arduino gesteckt statt alles über das *Breadbord* zu führen) Die sichtbare 9 *V*-Batterie ist die Stromversorgung des Arduino. Man könnte ihn auch vom TX Controller aus mit 5 *V* versorgen:

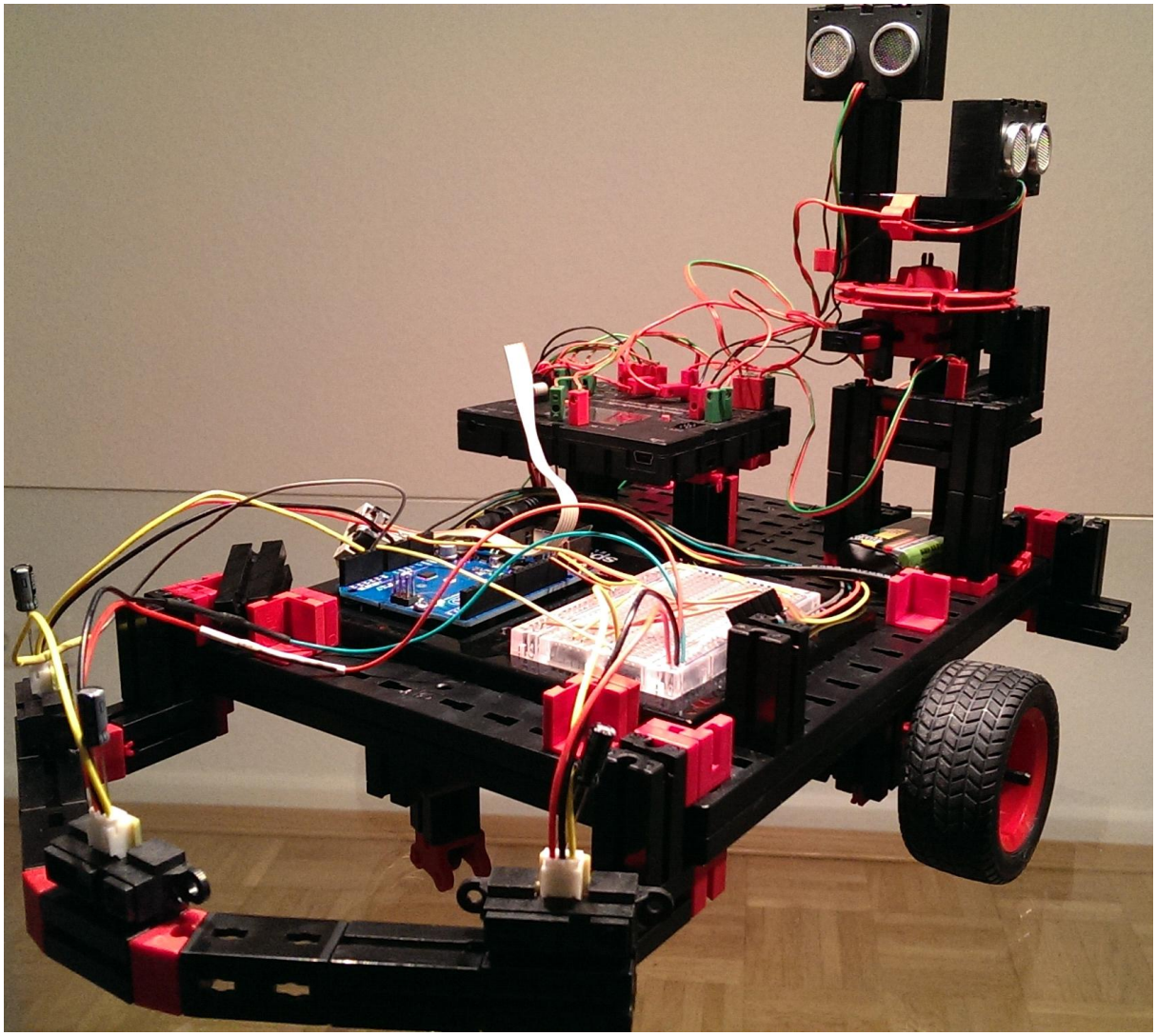


Abb. 4: Beispiel eines mobilen Roboters mit Arduino, TX Controller und drei Infrarotdistanzsensoren

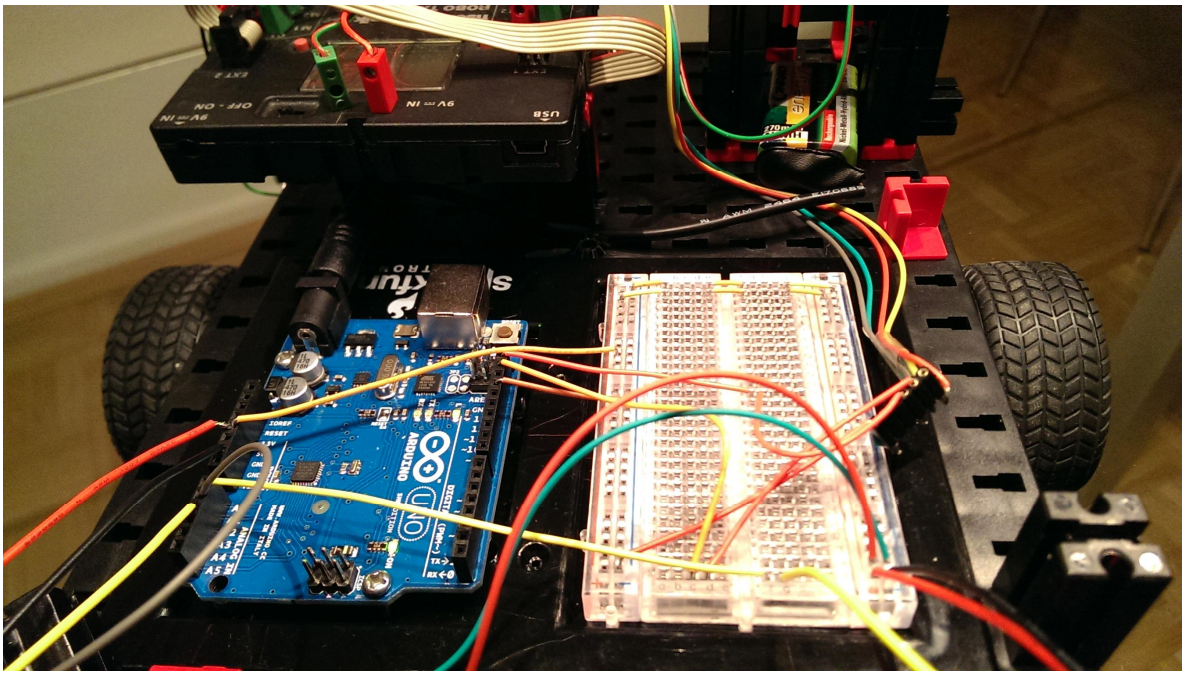


Abb. 5: Die Verdrahtung des Arduinos und des TX Controllers aus der Nähe

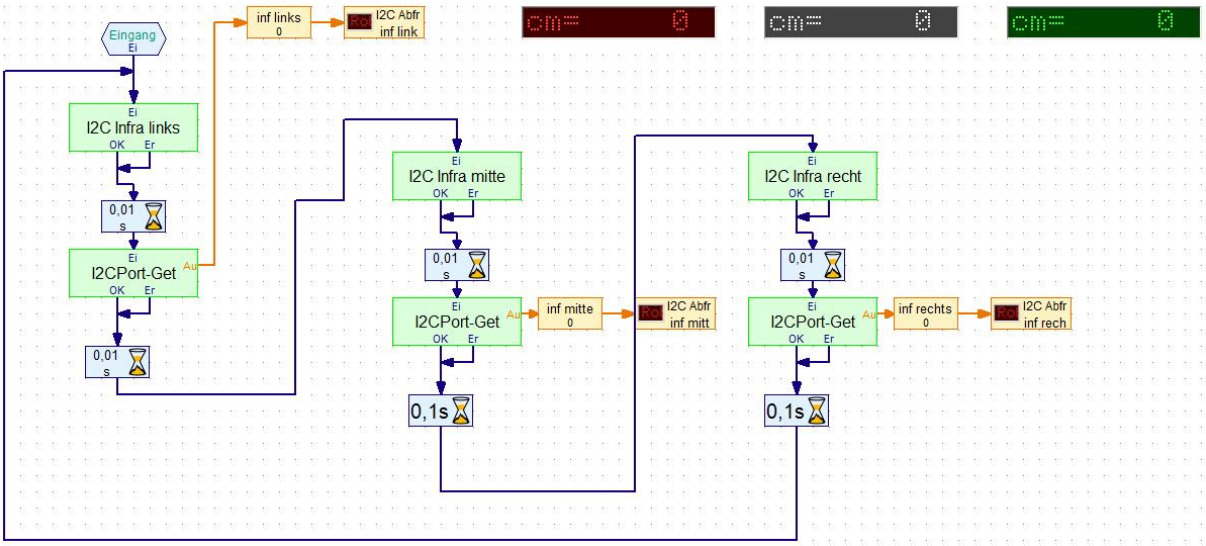


Abb. 6: Programmschleife in Robo Pro: Abfrage der Sensorenwerte mit I²C

Die Software des Masters

Die Aufgabe des Masters (des TX Controllers) ist es, die vom Arduino aufbereiteten Messwerte abzufragen und dann die Motorsteuerung zu regeln. Mehr nicht.

Auf die Motorsteuerung gehe ich nicht weiter ein und werde mich auf die Sensorabfragen konzentrieren. Wichtig ist, die korrekten Sensoren abzufragen und z. B. genau zu wissen, ob ein Hindernis beim linken Sensor erkannt wurde. Dazu wird an den Arduino ein *Request* gesendet mit dem Hinweis, dass man den linken Sensor abfragen möchte. Dies geschieht durch Senden einer Unteradresse.

In Robo Pro gibt es eine I²C-Bibliothek I/O-Port PCF8574, die ich hier verwende. In dieser Bibliothek befinden sich zwei Unterprogramme: ‚I2C Port-Set' (Senden von Befehlen an den I²C-Slave) und ‚I2C Port-Get' (*Request* der eigentlichen Daten vom I²C-Slave). Der Slave ist dabei der Arduino – man kommuniziert also nicht direkt mit den Sensoren.

Die beiden Unterprogramme habe ich etwas angepasst, z. B. die Adresse des Arduinos (0x20) angegeben und die Subadressen in drei Kopien verteilt (0x01, 0x02, 0x03). Abb. 6 zeigt das Hauptprogramm zur Abfrage der Sensoren über I²C vom Arduino.

‚I2C Infra links' ist das ehemalige Unterprogramm ‚I2C Port-Set'; es sendet an 0x20 (den Arduino) eine Unteradresse (0x01). Danach erfolgt mit ‚I2CPort-Get' die Abfrage der Variablen vom Arduino. Da man 0x01 gesendet hat, weiß der Arduino, dass man den Wert des linken Sensors haben möchte und sendet diesen zurück.

Hier der Inhalt der Anfrage (‚I2C Port-Set' bzw. ‚I2C Infra rechts') für den rechten Sensor, Unteradresse 0x03 (Abb. 7):

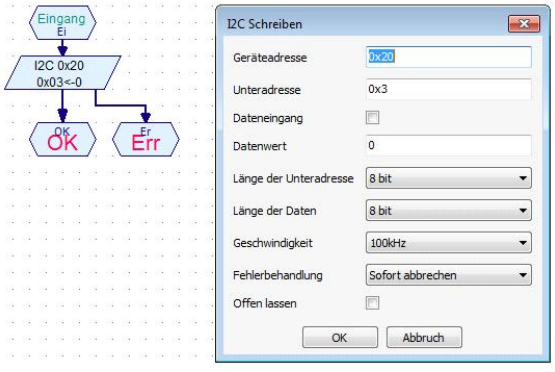


Abb. 7: Inhalt des ‚I2C Port-SET'-Befehls in Robo Pro

Und auf der folgenden Seite in Abb. 8 der zugehörige *Request* (‚I2C Port-Get'), um die eigentlichen Daten zu empfangen.

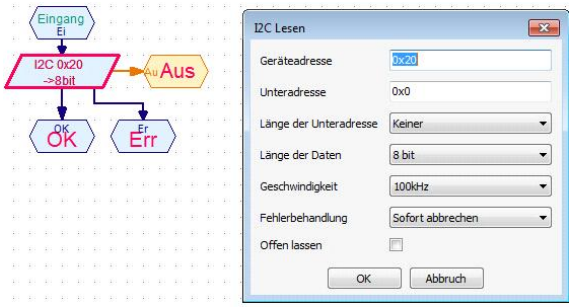


Abb. 8: Inhalt des ‚I2C Port-Get'-Requests in Robo Pro

Also: Zunächst teilt man mit ‚I2C Port-Set' dem Arduino (I²C-Adresse 0x20) die Unteradresse des Sensors (z. B. 0x03) mit. Damit weiß der Arduino, welche Daten man erwartet. Diese werden dann schon mal vom Arduino vorbereitet und bereitgestellt. Mit dem *Request* ‚I2C Port-Get' ruft man die Daten dann ab.

Dazwischen habe ich eine kleine Pause von 0,01 Sekunden gesetzt, um dem Arduino etwas Zeit zu geben, die Daten vorzubereiten.

Das war es. Was dann mit den Sensorwerten in Robo Pro anzufangen ist, kommt auf euer Projekt an.

Die Software des Slave

Arduino stellt eine eigene Programmieroberfläche zur Verfügung. Diese ist nicht grafisch, so wie Robo Pro, sondern sie ist eine klassische Programmierung wie Basic oder C. Sie wurde aus der Programmierumgebung *Processing* abgeleitet. Die Software kann man sich kostenlos auf www.arduino.cc herunterladen.

Ich gehe hier nicht näher darauf ein. Wer sich einen Arduino anschafft, sollte sich zunächst mit den Grundlagen beschäftigen. Ich habe mir dafür ein Lernpaket von „Franzis" gekauft: Es ist gut gemacht, hat ein ordentliches und leicht verständliches Handbuch und erklärt Schritt für Schritt die Programmierung und Handhabung des Arduino.

Im Internet gibt es unzählige Beispielprogramme (*Sketches*) für den Arduino. Auf der Suche nach der Lösung meines hier beschriebenen Zieles bin ich auf einen Vorschlag zur Verbindung eines Lego NXT an den Arduino gestoßen. Dieses Programm habe ich als Grundlage genommen und erweitert – ihr findet es auf der folgenden Seite.

Die ‚Serial.print'-Befehle senden Werte seriell (USB) an ein Terminalprogramm. Die werden nicht unbedingt benötigt, man kann so aber das Geschehen auch aus Sicht des Arduino verfolgen. Ihr könnt diesen Text kopieren und direkt in die Arduino-Programmierumgebung einfügen:

Nach dem Upload des Programms auf den Arduino wird dies automatisch gestartet. Danach könnt ihr das Robo Pro-Programm starten. Und jetzt reden die beiden miteinander und ihr könnt die Werte in Robo Pro ablesen.

Viel Spaß und viel Erfolg bei euren Projekten!

Quellen

[1] Dirk Uffmann: *ft-Modellsteuerung mit selbst entwickeltem Mikrocontroller-Board*, in dieser Ausgabe der ft:pedia 1/2014.

[2] Dirk Fox*: I²C mit TX und Robo Pro – Teil 1: Grundlagen*. ft:pedia 3/2012, S. 32-37.

[3] Dexter Industries: *Interfacing the Arduino and Lego Mindstorms*, 26.02.2013

Arduino Sketch zum Senden und Empfangen vom Master

```
// I2C Slave Send / Receive

// Demonstrates how to connect a TX Controller to an Arduino and Send commands, receive data.
// Mit dieser abgewandelten Form eines Programms für den Lego NXT werden drei Infrarotsensoren
// abgefragt. Dazu habe ich
// noch die Berechnung der Distanz in cm aus den Spannungswerten der Sharp Infrarotdistanzsenoren
// eingefügt.
// Dem Arduino habe ich die Adresse 0x20 gegeben, zur Abfrage werden die Adressen 0x01 linker Sensor,
//0x02 Mitte, 0x03 rechts genutzt.

#include <Wire.h>; // Nutzung der Wire Bibliothek für I2C

byte read_register = 0x00;

void setup() // Vorbereitende Definitionen
{
  Wire.begin(0x20);                // join i2c bus with address 0x20
  Wire.onRequest(requestEvent);  // Sending information back to the TX Controller!
  Wire.onReceive(receiveI2C);    // Receiving information from TX Controller!
  pinMode(A2, INPUT); // Vorbereiten der analogen Pins als Eingang
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
}

void loop()
{
  delay(100);
}
// Hier folgen die Funktionen zur Berechnung der cm aus den rohen Sensorwerten
int read_inflinks() // Variable, die spaeter für den linken Sensor an TX übergeben wird.
{ int val = analogRead(A0);// Abfrage Analog Pin 0
  if (val < 3) return -1; // invalid value
  float ret = (6787.0 /((float)val - 3.0)) - 4.0;
  if(ret > 80.0)// 80 cm ist der maximal messbare Abstand
  return(80.0);
  else
  if(ret < 11) return(0.0);// minimal können 10 cm gemessen werden,
//der TX interessiert sich aber für alles kleiner 11 cm!
  else return(ret);
}
int read_infmitte() // Variable, die spaeter für den mittleren Sensor an TX übergeben wird.
{ int val = analogRead(A1);//Abfrage Analog Pin 1
  if (val < 3) return -1; // invalid value
  float ret = (6787.0 /((float)val - 3.0)) - 4.0;
  if(ret > 80.0) // 80 cm ist der maximal messbare Abstand
  return(80.0);
  else
  if(ret < 11) return(0.0); // minimal können 10 cm gemessen werden,
//der TX interessiert sich aber für alles kleiner 11 cm!
  else return(ret);
}
int read_infrechts() // Variable, die spaeter für den rechten Sensor an TX übergeben wird.
{ int val = analogRead(A2);//Abfrage Analog Pin 3
  if (val < 3) return -1; // invalid value
  float ret = (6787.0 /((float)val - 3.0)) - 4.0;
  if(ret > 80.0)// 80 cm ist der maximal messbare Abstand
  return(80.0);
  else
  if(ret < 11) return(0.0);// minimal können 10 cm gemessen werden,
//der TX interessiert sich aber für alles kleiner 11 cm!
  else return(ret);
}
// When data is received from TX, this function is called.
void receiveI2C(int bytesIn)
{
  read_register = bytesIn;
  while(1 < Wire.available())       // loop through all but the last
  {
    read_register = Wire.read();    // The incoming byte tells use the register we will use.
    Serial.print(read_register);    // Print the incoming byte as a character on the Serial line.
  }
  int x = Wire.read();        // Read the incoming byte
  Serial.println(x);          // Print the incoming byte
}

// Abhängig vom  read_register value werden die jeweiligen Abstände des linken, mittleren
// oder rechten Sensors gesendet.
// This can be expanded to have 256 different registers that can be updated and accesed
// when the master calls for them.

void requestEvent()
```

```
{
  // We're going to send a fixed response.  However,
  // you can use a switch case here and send different responses
  // based on a register system. Dies habe ich geändert.
  if(read_register == 0x01){
        Wire.write(read_inflinks());   // Antwortet mit Variable "read_inflinks" Kalkuliert für linken IFD
  }
  else if(read_register == 0x02){

    Wire.write(read_infmitte());  // Antwortet mit Variable "read_infmitte" Kalkuliert für mittigen IFD
  }
else if(read_register == 0x03){

    Wire.write(read_infrechts());   // Antwortet mit Variable "read_infrechts" Kalkuliert für rechten IFD
  }            // as expected by master

  else if(read_register == 0x05){

    Wire.write(read_infrechts());   // Antwortet mit Variable "read_infrechts" Kalkuliert für rechten IFD
  }            // as expected by master
}
```

Computing

ft-Modellsteuerung mit selbst gebautem Mikrocontroller-Board

Dirk Uffmann

Einige kennen vielleicht diesen Wunsch: Ich habe mehrere fischertechnik-Modelle, die ich mit einem TX-Controller steuern möchte – aber mir fehlt das Budget für die nötige Anzahl dieser Bausteine. Außerdem möchte ich die Eingangssignale in Echtzeit verarbeiten, z. B. zum Auslösen von Interrupt-Service-Routinen, die das Hauptprogramm unterbrechen und mit denen die Zeit zwischen zwei Signaländerungen sehr genau gemessen werden kann. Und ich möchte die I/O- Pins der Steuerung flexibler nutzen, z. B. auch als Output zur Erzeugung von Pulsen für Lichtschranken oder zum Schalten von LEDs. Wenn ihr ähnliche Wünsche habt, dann zeigt euch dieser Beitrag einen Weg, mit dem ihr euch diese erfüllen könnt.

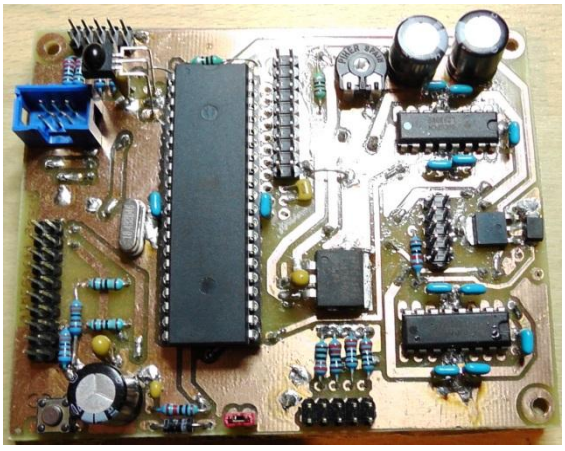


Abb. 1: Bauteilseite des Moduls

Hintergrund

fischertechnik bietet für die elektronische Steuerung von Modellen viele gelungene Produkte. Der TX Controller verfügt über tolle Möglichkeiten und ist für die meisten Anwender sehr gut geeignet. Manchmal wünscht man sich jedoch, näher an der Hardware programmieren zu können oder eine höhere Flexibilität für die Nutzung der Ein- und Ausgänge zur Verfügung zu haben. Beispiele sind das Auslösen von Interrupts durch die Eingangssignale, das Erzeugen von Pulsen oder das direkte Treiben von LEDs. Wer zudem viele Modelle parallel ansteuern und seinen TX Controller nicht immer zwischen den Modellen austauschen will, kommt möglicherweise an seine Budget-Grenzen.

Dieser Beitrag beschreibt den Selbstbau eines einfachen Mikrocontroller-Boards für die automatisierte elektronische Steuerung von fischertechnik-Modellen (Abb. 1).

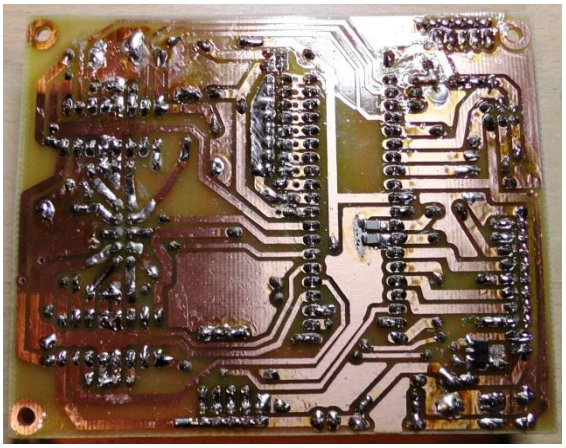


Abb. 2: Unterseite/Lötseite des Moduls

Das Board wurde als zweiseitige Platine der Größe 80 x 100 mm (halbes Europakartenformat) entworfen. Bis auf wenige Ausnahmen wurden bedrahtete Bauteile im

Raster 2,54 mm verwendet. Dadurch sind die Strukturgrößen der Leitbahnen relativ groß und gut für Hobbymittel geeignet (Abb. 2).

Features

Das Board bietet:

- acht digitale Anschlüsse (jeder gepaart mit einer Masse-Leitung), einzeln konfigurierbar als Ein- oder Ausgang mit spezifischen Sonderfunktionen wie Interrupts oder für Kommunikationsschnittstellen
- acht analoge Eingänge (jeder gepaart mit einer Masse-Leitung), alternativ einzeln konfigurierbar als digitaler Ein- oder Ausgang
- vier PWM (*Pulse Width Modulation*) steuerbare Motor-Ausgänge mit zwei L293DNE Motor-Treiber ICs
- Infrarot-Fernbedienungsempfänger SFH5110-38kHz zur Nutzung einer vorhandenen Fernbedienung
- SPI-Kommunikationsschnittstelle (*Serial Peripheral Interface*) zu einem weiteren identischen Board über den 6-Pin ISP-Anschluss (*In System Programmer*); dadurch besteht eine Erweiterungsmöglichkeit auf acht Motoren
- I²C-Kommunikation mit bis zu 127 Bauteilen über ein Bus-Protokoll (Beispiel: Anschluss mehrerer Ultraschallsensoren und eines I²C LC-Displays)
- 8 bit-µController ATMEGA1284P-PU (ca. 6,35 € beim Reichelt Elektronik-Versand) oder ATMEGA644P-PU [1]
- Programmierung in C (GCC: Gnu C-Compiler) oder in Assembler mit AVR Studio, einer komfortablen Entwicklungsumgebung (kostenloser Download)
- Serielle UART-Schnittstelle (*Universal Asynchronous Receiver Transmitter*), vorbereitet für 3,3 V Bauteile (z. B. eine Handy-Kamera MCA-25)
- 18,432 MHz Baud-Raten-Quarz (für die UART-Schnittstelle)

Das Laden des Programmcodes in den internen Flash-Speicher erfolgt mit Hilfe des Programmiergerätes AVRISP MKII (ca. 30-40 €) über USB. Für den Code-Download muss das Board mit Strom versorgt werden.

Pin-Belegung am 20-poligen Flachbandstecker (oben rechts, analog)

PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	+5V	+5V
2	4	6	8	10	12	14	16	18	20
1	3	5	7	9	11	13	15	17	19
GND	GND	GND	GND	GND	GND	GND	GND	+5V	0-5V
									Poti

Pin-Belegung am 20-poligen Flachbandstecker (unten links, digital)

SCL	SDA	SS	RxD	TxD	INT0	INT1	ICP		
PC0	PC1	PB4	PD0	PD1	PD2	PD3	PD6	+3V3	+5V
2	4	6	8	10	12	14	16	18	20
1	3	5	7	9	11	13	15	17	19
GND	GND	GND	GND	GND	GND	GND	GND	+3V3	+5V

Pin-Belegung am 10-poligen Flachbandstecker (Motoren, Versorgung)

	links	rechts		links	rechts	
	PC4	PC2		PB1	PC7	
rückw.	Out2	Out3	+9V	Out2	Out3	rückw.
	2	4	6	8	10	
	1	3	5	7	9	
vorw.	Out1	Out4	GND	Out1	Out4	vorw.
	PC5	PC3		PB0	PC6	
	En1	En2		En1	En2	
	OC1A	OC1B		OC0	OC2	
	PD5	PD4		PB3	PD7	
	unten			oben		

Pin-Belegung am 10-poligen Flachbandstecker für LEDs oben links

GND	GND	GND	GND	GND	
1	3	5	7	9	außen
2	4	6	8	10	innen
100Ω/5V	100Ω/5V	100Ω/5V	100Ω/5V	+5V	

Pin-Belegung am 10-poligen Flachbandstecker für LEDs unten

GND	100Ω/⊥	100Ω/⊥	100Ω/⊥	100Ω/⊥	
1	3	5	7	9	innen
2	4	6	8	10	außen
+5V	+5V	+5V	+5V	+5V	

AVR-Programmierstecker

MISO	SCK	RESET
1	3	5
2	4	6
+5V	MOSI	GND

Abb. 3: Anschlussbelegung auf den Steckerleisten

Die Anschlüsse sind alle mit Pfostensteckerleisten für Flachbandkabel ausgestattet (Belegung siehe Abb. 3). Dadurch spart man ca. 50 % der nötigen fischertechnik-Stecker ein (die ja auch häufig Mangelware sind). Die ft-Stecker sind dann nur an einem Ende des jeweiligen

Kabels erforderlich. Die Breite der 20er-Flachbandkabel für die beiden Eingangskontaktleisten stimmt mit der des alten Universal-Interfaces und des Robo-Pro-Interfaces überein: Die vorhandenen Flachbandkabel können also weiter verwendet werden.

Da die Glühbirnen für die Leuchtsteine immer seltener werden und viele bereits 5 mm-LEDs in den Leuchtsteinen einsetzen, habe ich auf zwei weiteren, 10-poligen Steckerleisten auch noch eine 5 V-Versorgung mit und ohne Vorwiderstände vorgesehen. Dann ist man nicht unbedingt darauf angewiesen, die Vorwiderstände in die Leuchtsteine mit einzubauen.

Um LED-Licht zu schalten, kann man die Pins auf den 20-poligen Pfostenstecker-leisten als Ausgang konfigurieren. Damit lassen sich LEDs direkt ansteuern und mit einem Strom von bis zu 40 mA treiben. Für weiße und blaue LEDs, die zwischen 2,7 und 4 V Spannung benötigen, geht das auch gut ohne Vorwiderstand. Für rote und gelbe LEDs mit geringerer Spannung kann man jeweils zwei in Serie geschaltete LEDs an einem Ausgang ohne Vorwiderstand einsetzen.

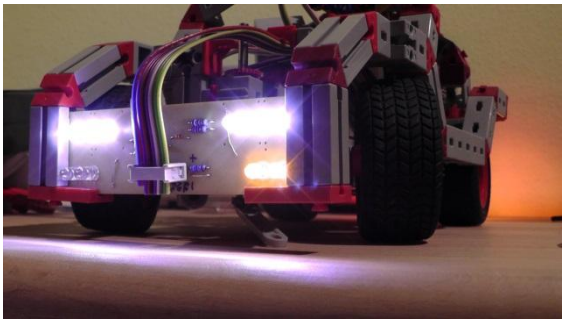


Abb. 4: Fahrzeugbeleuchtung mit LED-Modul

Abb. 4 zeigt ein LED-Modul zur Fahrzeugbeleuchtung mit Scheinwerferlicht (je drei weiße LEDs) und Blinkern (je zwei gelbe LEDs), die direkt mit dem Mikrocontrollerboard angesteuert werden. Der Anschluss erfolgt über Flachbandkabel und Pfostensteckverbinder ohne einen einzigen ft-Stecker. An der Wand hinten rechts im Bild sieht man die Reflexion des rückseitigen LED-Moduls für Rücklicht, Bremslicht und Blinker.

Für mobile Roboter und sonstige Stell- und Positionierungsaufgaben der Motoren bieten sich die beiden Eingänge PD2 und PD3 an, die einen externen Interrupt auslösen, wenn sie entsprechend konfiguriert sind. Mit diesen Eingängen lassen sich nebenbei Impulse von Drehencodern oder Gabellichtschranken zählen (Abb. 5) und man kann die Zeit zwischen zwei Impulsen sehr genau messen, z. B. für eine Geschwindigkeitsregelung.

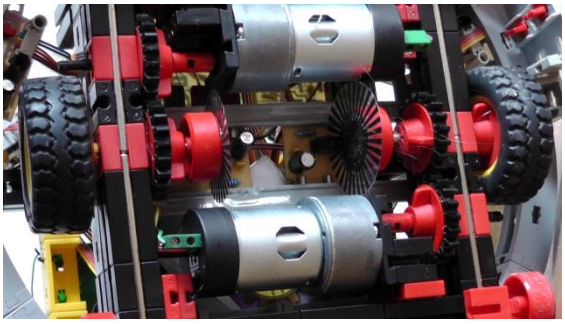


Abb. 5: Beispiel für den Einsatz einer Gabellichtschranke als Drehencoder

Damit bietet dieses Mikrocontrollerboard viele Features, die man vom TX-Controller kennt. Der TX bietet darüber hinaus ein schönes Gehäuse und ein LC-Display. Als Ersatz für das Gehäuse habe ich die Unterseite der Platine mit einer auf 80 x 100 mm zugeschnittenen Pappe, die von Gummibändern an der Platine gehalten wird, abgedeckt. Im Modell wird die Platine dann liegend mit vier Bausteinen V15 Eck (38240) eingefasst (Abb. 6).

Abb. 6: So kann das Mikrocontrollerboard in ein ft-Modell eingebaut werden.

Ein LCD kann man an das Mikrocontroller-Board auch anschließen. So ein LCD ist bei der Codierung des Steuerprogramms für das Modell oft sehr nützlich, um Fehler im Programm zu finden, z. B. indem man die im Programm berechneten Variablen auf dem LCD darstellen lässt. Das LCD ist außerdem nützlich, um sich aktuelle Messwerte von Sensoren etc. ausgeben zu lassen (Abb. 7).

Abb. 7: Beispiel für ein Text-LCD mit 4 Zeilen à 20 Zeichen, ergänzt um eine I²C-Interface-Platine (auf der Unterseite angelötet)

Besonders einfach geht das mit Text-LCDs nach Industriestandard – einem HD44780-Controller. Entweder betreibt man dieses direkt an den I/O-Pins des Mikrocontroller-Boards im 4-bit-Modus, wofür sieben I/O-Pins benötigt werden. Oder, noch besser, man verwendet ein I²C-fähiges LCD (z. B. mit einem PCF8574 als Interface, siehe auch [2]). In diesem Fall benötigt man nur zwei I/O-Pins. Zur Zeit kann man blaue vierzeilige LCD mit weißem Hintergrundlicht zum Stückpreis von ca. 5 € aus China in der „Bucht“ erwerben. Die passenden, direkt auf die LCD-Anschlusspins lötbaren I²C-Anschlussboards gibt es für ca. 2 € pro Stück (Pinbelegung beachten). Ein zweizeiliges grünes LCD gibt es komplett bei Pollin für 6,95 €. Zur Ansteuerung des I²C-Busses und für Text-LCDs stehen im Internet Software-Bibliotheken für Atmel-Mikrocontroller zum freien Download zur Verfügung, z. B. von Peter Fleury. Für die Ansteuerung des hier vorgestellten I²C-LCDs stellt der Autor passenden Code im Downloadpaket auf seiner Website zur Verfügung.

Herstellung der Platine

Nun stellt sich die Frage, wie man sich ein solches selbstgebautes Mikrocontroller-Board herstellen kann. Die Designdaten für das Entwurfsprogramm Eagle von Cadsoft nebst Maskenvorlagen als .eps (*encapsulated postscript* als Grafik in Word o. ä. druckbar, Abb. 8, 9) stelle ich zum Download auf meiner Website zur Verfügung. Das Programm Eagle selbst kann als kostenlose Freeware im Internet bezogen werden (als Light-Version bis zu einer Platinengröße von 80 x 100 mm und zweiseitiger Verdrahtung). Mit Eagle lässt sich mein Entwurf auch verändern oder weiterentwickeln.

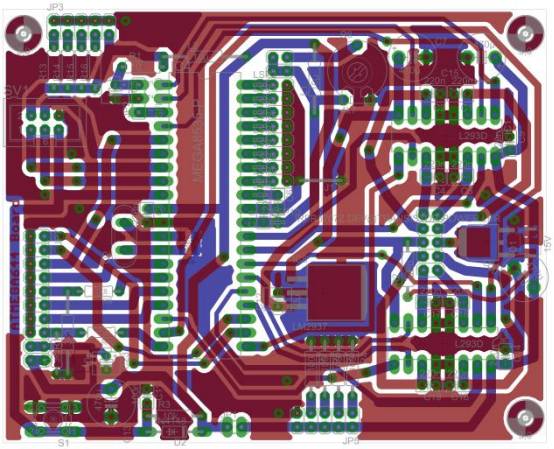


Abb. 8: Ansicht des entworfenen Boards in Eagle (Leiterbahnen auf der Komponentenseite in roter, auf der Lötseite in blauer Farbe)

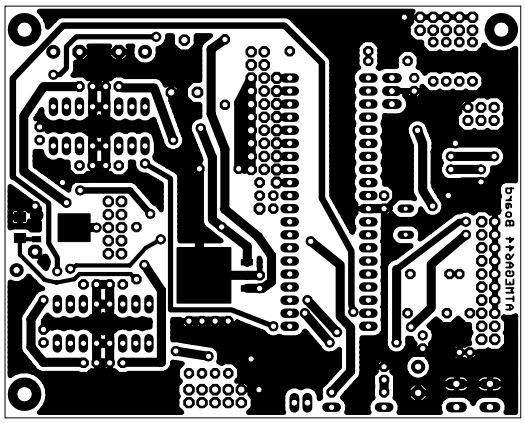


Abb. 9: Maskenvorlage für die Leiterbahnen auf der Komponentenseite. Gespiegelte Ansicht, damit sich der Toner für die Belichtung auf der Seite der Fotoschicht befindet

Der einfachste Weg zur Herstellung von Platinen geht über einen Leiterplattenhersteller für Prototypen wie z. B. bilex (Fertigung in Bulgarien, aber Deutsch sprechender Kontakt und Konto in Deutschland). Dort kann man die Eagle-Designdaten per E-Mail abgeben und sich für ca. 15-20 €/Stück die Leiterplatten belichten, ätzen, bohren und galvanisch durchkontaktieren lassen. Das spart viel Zeit.

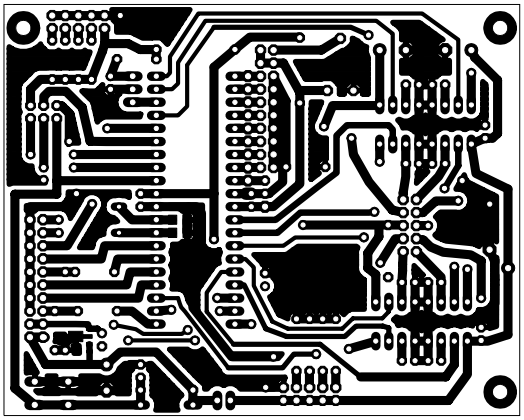


Abb. 10: Maskenvorlage für die Leiterbahnen auf der Lötseite

Alternativ und mit viel mehr persönlichem Zeitaufwand stellt man die Platinen selbst her. Dazu verwendet man meine fertigen Maskenvorlagen, druckt diese auf Overheadfolie aus und belichtet damit doppelseitig fotobeschichtete und kupferkaschierte Platinen (am besten Europakartenformat 100 x 160 mm mit zwei der Boards darauf nebeneinander platziert).

Dazu klebt man die beiden Maskenseiten als eine Art Fototasche zusammen, so dass die Tonerschicht innen liegt und die Strukturen, besonders die Durchkontaktierungen, passgenau übereinander zu liegen kommen (Abb. 11).

Direkt vor dem Belichtungsvorgang muss noch von beiden Seiten der Platine die lichtdichte Schutzfolie abgezogen werden.

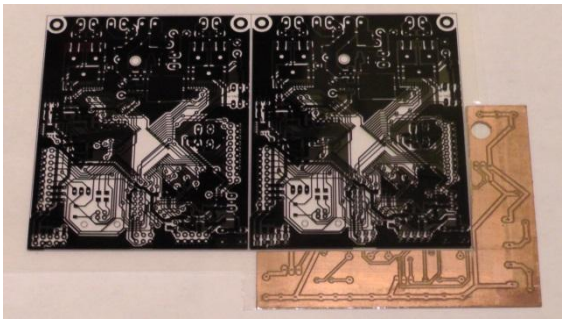


Abb. 11: Beispiel für eine geklebte „Fototasche“ für beidseitige Belichtung einer Europakarte (Format 100 x 160 mm). Als „Rahmen“ dient ein rechtwinklig ausgeschnittenes Platinenstück, an das die beiden Folien mit Klebeband geklebt werden.

Danach kann dann von beiden Seiten belichtet werden. Dazu benutze ich einen UVA-Oberkörperbräuner im Abstand von ca. 60 cm, den ich auf zwei Stühlen über der zu belichtenden Platine in horizontale Position bringe. Dabei drückt eine kleine Glasplatte die Overheadfolie plan auf die Fotoschicht. Anschließend wird die Platine samt Fototasche gewendet und von der anderen Seite belichtet. Dabei ist zu beachten, dass die Platine beim Wenden nicht in der Fototasche verrutscht. Die Belichtungszeit beträgt bei mir ca. 90 s. Dies ist von der Lichtintensität des UV-Lichtes abhängig und jeder sollte erst eine kleine Belichtungs- und Entwicklungsreihe mit kleinen Probestücken oder Streifen des Platinenmaterials machen.

Die belichtete Platine wird dann für etwa 2-3 min in ein Entwicklerbad aus verdünntem Natriumhydroxid (NaOH) gegeben, hin und her bewegt und nach dem Herauslösen der belichteten Bereiche (bei Positivlack) in Wasser gespült. Danach wird die entwickelte Platine in ein erwärmtes Ätzbad aus Natriumpersulfat ($Na_2S_2O_8$; exakter Name: Natriumperoxodisulfat) gegeben. Ideal zum Ätzen wäre dafür eine Ätzküvette, die die Temperatur auf ca. 40-50° C regelt und Luft einbläst. Es geht aber auch ohne Ätzküvette: Man erwärmt die angesetzte Lösung in einem geschlossenen Glas in der Mikrowelle, bis sie deutlich

mehr als handwarm ist. Ist das Glas ausreichend groß für die Platine (evtl. großes Einmachglas), gibt man die Platinen in das Glas, schließt dieses dicht ab (verriegelter Deckel mit Dichtung) und schüttelt es, bis das Kupfer weggeätzt ist (deutlich sichtbare Kontraständerung, die Strukturen auf der hinteren Seite der Platine werden durchscheinend erkennbar).

Oder aber man verwendet eine Ätzschale aus Kunststoff und bewegt die Platine in der warmen Lösung hin und her. Der Ätzvorgang kann dann bis zu 20 min dauern. Anschließend wird die Platine gründlich mit Wasser abgespült (erst in einem großen Wassereimer, dann unter dem Wasserhahn). Nun müssen noch die verbliebenen Fotolackstrukturen entfernt werden. Dazu kann man höher konzentrierte Entwicklerlösung ansetzen und die Platinen darin baden, bis der Fotolack vollständig aufgelöst wurde.

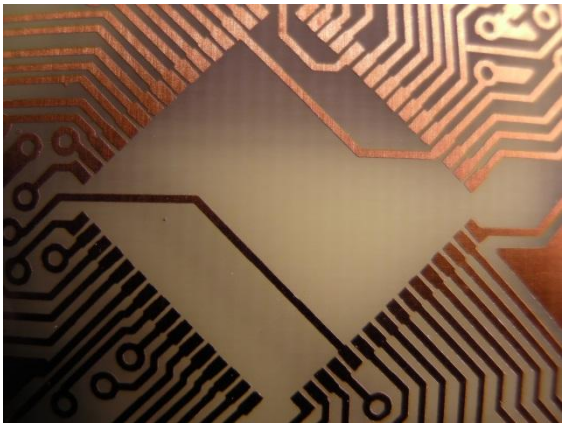


Abb. 12: Platinenstrukturen nach dem Ätzen. Mit dem dargestellten Verfahren können sehr feine Strukturen hergestellt werden. Der Pitch, also der Abstand der SMD Pads von Mitte zu Mitte für dieses TQFP64-Package beträgt 0,8 mm (Board für einen AT90USB1287).

Nun muss die Platine noch gebohrt und durchkontaktiert werden. Zum Bohren benutze ich einen Proxxon Feinbohrschleifer FBS 240/E mit Bohrständer Proxxon MB200. Alternativ eignen sich dafür auch die Geräte des Wettbewerbers Dremel oder anderer. Nach dem Bohren wird die Platine nochmals mit Wasser abgespült und gereinigt und nach dem Trocknen mit Lötlack eingesprüht. Für diejenigen Durchkontaktierungen (leitende Verbindungen zwischen den beiden Platinenseiten), in denen nach dem Bestücken mit Bauteilen kein Draht eines Bauteils sitzt, löte ich kurze Drähte als Durchkontaktierung ein. Nun kann die Platine bestückt und gelötet werden. Dafür kann man als Vorlage die Bestückungspläne aus Eagle benutzen (Abb. 13, 14).

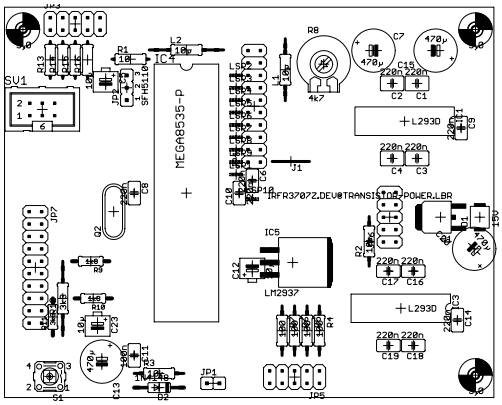


Abb. 13: Vorlage für die Bestückung auf der Komponentenseite (bitte den MEGA 8535P ersetzen durch ATMEGA 1284P-PU oder 644P-PU; für den JP2 den Infrarotempfänger SFH 5110-38 einsetzen)

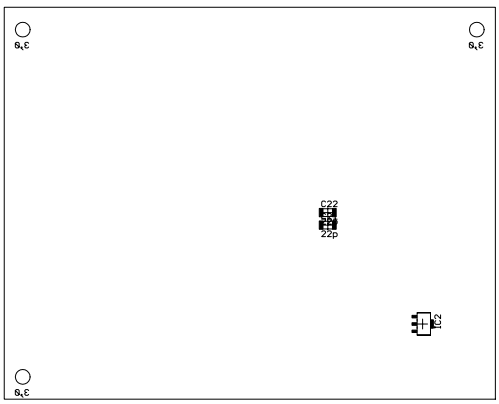


Abb. 14: Vorlage für die Bestückung auf der Lötseite

Beim Löten sollte man darauf achten, diejenigen Bauteile, deren Drähte auch als Durchkontaktierung zur Komponentenseite dienen, auch auf der Oberseite anzulöten. Generell ist wichtig, alle Masseflächen über Durchkontaktierungen auch an Masse anzuschließen. Am besten misst man das

mit einem Multimeter nach. Nach dem Bestücken und Löten kann die Platine in Betrieb genommen und an die externe Versorgungsspannung angeschlossen werden.

Menge	Wert	Device	Bauteile
1		10-XX	S1
1		JP1E	JP1
1		ML6	SV1
3		PINHD-2X5	JP3, JP5, JP6
2		PINHD-2X10	JP4, JP7
4	100	R-EU_0207/7	R13, R14, R15, R16
1		XC6202P332P	IC2
1	18,432 MHz	XTAL/S	Q2
1	1N4148	1N4148DO35-10	D2
2	1k8	R-EU_0204/7	R9, R10
1	3k3	R-EU_0204/7	R11
1	3k3	R-EU_0207/15	R12
1	4k7	POTENTIOMETER_PT-10	R8
3	10µ	CPOL-EUETR1	C5, C12, C23
1	10µ	L-US0207/12	L1
1	10µ	L-US0207/15	L2
1	10	R-EU_0207/7	R1
2	10k	R-EU_0207/10	R2, R3
1	15V	SUPPRESSOR-SMBJ	D1
2	22p	C-EUC1206	C21, C22
4	100	R-EU_0207/10	R4, R5, R6, R7
1	100n	C-EU025-025X050	C11
13	220n	C-EU025-025X050	C1-C4, C6, C8-C10, C14, C16-C19
4	470µ	CPOL-EUE5-8.5	C7, C13, C15, C20
1	IRFR3707Z	IRFR3707Z	Q1
1	J7MM	J7MM	J1
2	L293D	L293D	IC1, IC3
1	LM2937	LM2937	IC5
10	LSP10	LSP10	LSP1-LSP10
1	AT**MEGA1284P-PU**	~~MEGA8535-P~~	IC4
1	**SFH5110-38**	JP2E	JP2

Abb. 15: Bestückungsliste

Erste Schritte sind einfache Prüfungen mit einem Multimeter: liegt die externe Spannung auch innen auf dem Board an? Liefern die Spannungsregler wie beabsichtigt 5 *V* bzw. 3,3 *V*?

Programmierung des Mikrocontrollers

Spätestens jetzt sollte man mal im Datenbuch des verwendeten Mikrocontrollers lesen. Dann sollte man prüfen, ob der Mikrocontroller über den ISP-Stecker angesprochen werden kann und die Fuses programmieren. Für das Board hier habe ich die in Abb. 16 dargestellten Fuse-Einstellungen gewählt.

Nun kann der erste Programmcode auf den Baustein geflasht werden. Vom Hauptprogramm sollte zunächst eine Funktion aufgerufen werden, die die Pins konfiguriert (Abb. 17).

Danach müssen die Timer, der Analog-Digital-Wandler (ADC) und die Interrupts konfiguriert werden (Abb. 18).

```
 * Fuse Extended:  11111100     0xFC
 *                      100     BOD Level 4.3 V
 *
 * Fuse High Byte: 11011001     0xD9
 *                 1            OCD disabled
 *                  1           JTAG disabled
 *                   0          SPI prog enabled
 *                    1         Watch Dog Timer off
 *                     1        EESAVE disabled
 *                      00      Bootsize 4096 words
 *                        1     BootRST not selected
 *
 * Fuse Low Byte:  11110111     0xF7
 *                 1            Clock not divided by 8
 *                  1           no Clock output on PortB
 *                   11         max startup time and delay
 *                     0111     CKSEL = quarz 18.432 MHz
```

Abb. 16: Fuse-Einstellungen für den ATMEGA 644P-PU

```
void ioinit (void)
{
/* description of IO-pins:
 *                  Motor1 Motor2 Motor3 Motor4
 * forward          PC5    PC3    PB0    PC7    =1
 * backward         PC4    PC2    PB1    PC6    =1
 * PWM              PD5    PD4    PB3    PD7
 * OC               1A     1B     0      2
 *
 * Digital I/O
 * 1   2     3     4     5     6     7     8
 * SCL SDA   SS    RxD   TxD   INT0  INT1  ICP
 * PC0 PC1   PB4   PD0   PD1   PD2   PD3   PD6
 *
 * Analog Input or Digital I/O
 * 1   2     3     4     5     6     7     8
 * PA7 PA6   PA5   PA4   PA3   PA2   PA1   PA0
 *
 * PA0: pulse generation for the 3 down-looking reflex light barriers for floor detection
 * PA1: left side floor detection, is high if pulse is reflected by the floor
 * PA2: front floor detection, is high if pulse is reflected by the floor
 * PA3: right side floor detection, is high if pulse is reflected by the floor
 * PA4: push button left side for detection of object collision (low in case of collision)
 * PA5: push button right side for detection of object collision (low in case of collision)
 */

      DDRA = _BV(PA0); /* Pin 0 on port A is output for pulse generation */
      PORTA = _BV(PA4) + _BV(PA5); /* all outputs are low, pull-ups activated on PA4 and PA5 */
      DDRB = 0x0B;  /* output: PB0, PB1, PB3, remaining pins are input */
      PORTB = 0xF4; /* pull-up resistors activated on input, PB0,1,3="low"*/
      DDRC = 0xFE;  /* pins 7,6,5,4,3,2,0 on port C are output, PC1 is input SDA(TWI) */
      PORTC = 0xFE; /* outputs on port C are initialized as "high" */
      DDRD = _BV(DDD1)+_BV(DDD4)+_BV(DDD5)+_BV(DDD6)+_BV(DDD7);/*output: PD1,4,5,6,7, rest input*/
      PORTD = _BV(PORTD0) + _BV(PORTD2) + _BV(PORTD3);
      /* all pull-up resistors activated for input, output PORTs="low" */
}
```

Abb. 17: Code-Beispiel für die Konfiguration der I/O-Pins

```
void init(void)
{

      /* system clock 18.432 MHz is used (external quartz oscillator),
       * 18.432 MHz/256/128/8=70Hz pwm frequency
       * timer 1 is used for PWM control of motor 1 and 2 in 15-bit resolution
       * ICR1 contains the top value for timer 1
       * fast pwm mode 14 is used, i.e. WGM13:10 = "1110" */

      TCCR1A = _BV(COM1A1) + _BV(COM1B1) + _BV(WGM11);
      TCCR1B = _BV(WGM13) + _BV(WGM12) + _BV(CS11); //timer 1 with prescaler 8
      ICR1 = 0x7FFF;
      OCR1A = 0x0000;
      OCR1B = 0x0000;

      TCCR0B = _BV(CS01); //start timer0 with prescaler clock/8
      // activate 8-bit Timer0 overflow interrupt 18.432 MHz / 256 / 8 = 3906 Hz = 256 µs
      // activate 16-bit Timer1 overflow interrupt 18.432 MHz / 256 / 128 / 8 = 70 Hz = 14.2 ms
      TIMSK0 = _BV(TOIE0);
      TIMSK1 = _BV(TOIE1);

      // activate interrupt INT0 (on PD2), INT1 (on PD3), INT2 (on PB2)
      EIMSK = _BV(INT0) + _BV(INT1) + _BV(INT2);

      /* PINA0 is used as analog input
      /* Vref=AVCC decoupled, single ended input ADC0  */
      ADMUX = _BV(REFS0);

      // set ADC enable and interrupt enable, prescaler 18.432 MHz / 128 = 144 kHz
      ADCSRA= _BV(ADEN) + _BV(ADIE) + _BV(ADPS2) + _BV(ADPS1) + _BV(ADPS0);

      // idle mode is set up as sleep mode
      // falling edge on INT1 and INT0 generates an interrupt
      SMCR = _BV(SE);
      EICRA = _BV(ISC21) + _BV(ISC11) + _BV(ISC01);
      sei ();
}
```

Abb. 18: Code-Beispiel für die Konfiguration der Timer, des ADC und der Interrupts

Die Interrupt-Routinen sollte man möglichst kurz und einfach halten, damit sie das Hauptprogramm nicht zu lange unterbrechen (siehe die Code-Beispiele in Abb. 19-21). Diese Code-Beispiele findet ihr zusammen mit einem kompletten Beispielprogramm für einen mobilen Roboter inkl. Ansteuerung von I²C-Modulen und

den Motoren in dem bereits erwähnten Downloadpaket. Für alles weitere findet man Tutorials und Foren z. B. beim Roboternetz oder bei Mikrocontroller.

In der nächsten Ausgabe der ft-pedia könnt ihr lesen, wie man ein oder sogar zwei FT-Universal Interfaces 30520 mit nur fünf I/O-Pins als Erweiterung um vier (bzw. acht) Motoren und acht (bzw. 16) digitale Eingänge an dem hier vorgestellten Board betreiben kann.

```
ISR (TIMER0_OVF_vect)
{
      // timing is adapted for system clock 18.432 MHz
      // with prescaler divide by 8 -> 8 * 256 / (18.432 MHz)
      // = 111 µs or 9 kHz
      // the variable timer_IR is exclusively used
      // in INT2 ISR for IR control.
      // do not change timer_IR in other functions

      timer0 += 1;
      timer_IR += 1;
      IR_flag += 1;
      timer_left +=1;
      timer_right +=1;

      clock_9kHz += 1;    // implementation of a real time clock
      if (clock_9kHz -- 9000)
      {
            seconds += 1;
            clock_9kHz = 0;
            if (seconds == 60)
            {
                  minutes += 1;
                  seconds = 0;
                  if (minutes == 60)
                  {
                        hours += 1;
                        minutes = 0;
                        if (hours == 24) hours = 0;
                  }
            }
      }
}
```

Abb. 19: Code-Beispiel für die Interrupt-Service-Routine für den Timer0-Overflow

```
// ********************************************************
// ISR (INT0_vect) counts the pulses from the left wheel
// fork light barrier. It represents the left wheel encoder.
// The variable t_left can be used for speed regulation.
// ********************************************************
ISR (INT0_vect)
{
      if (left_backward)
      {
            count_left -= 1;

      }
      else
      {
            count_left += 1;
      }

      t_left = timer_left;
      timer_left = 0;
}
```

Abb. 20: Code-Beispiel für die Interrupt-Service-Routine für einen Rad-Encoder

```
// ********************************************************************
// ISR (INT2_vect) evaluates REC80 code transmission via IR remote control.
// The resulting RC command is stored in the volatile variable "command".
// After "command" has been processed (by the main program) you should
// reset "command" to the value 0xFFFF. The timing is adapted for 18.432 MHz
// system clock prescaled by divider 8 for timer0.
// ********************************************************************
ISR (INT2_vect)
{
      if (bitcount==0xFF) // detect start condition
      {
            timer_IR = 0;
            bitcount = 0xFE;
      }
      else if (bitcount==0xFE)
      {
            if (timer_IR>133) // error detected
            {
                  timer_IR = 0;
                  bitcount = 0xFE;
            }
            else if (timer_IR>109) // start condition detected
            {
                  timer_IR = 0;
                  bitcount = 0;
                  RC = 0;
            }
            else if (timer_IR>92) // stop condition detected
            {
                  bitcount = 0xFF;
            }
      }
// Now the timing is synchronized to the relevant negative edge of the coding.
// After ca 1.1 ms (="0") or 2.2 ms (="1") the next relevant edge should occur
// positive edges in between are ignored.

      else if (bitcount < 32)
      {
            if ((timer_IR > 28)) // error detected
            {
                  timer_IR = 0;
                  bitcount = 0xFE; // error detected, reset bitcount to 0xFE
            }
            else if (timer_IR >= 16) // logical 1 detected
            {
                  RC = 2*RC + 1;
                  timer_IR = 0;
                  bitcount += 1;
            }
            else if (timer_IR >= 7) // logical 0 detected
            {
                  RC = 2*RC;
                  timer_IR = 0;
                  bitcount += 1;
            }
      }
      if (bitcount == 16)
      {
            sys_address = RC; // RC system address
            RC = 0;
      }
      else if (bitcount == 32)
      {
            if (sys_address == system) //== system)
            {
                  command = RC; // RC command
            }
            else
            {
                  command = 0xFFFF; // this system was not addressed
            }
            sys_address = 0xFFFF; // reset sys_address
            bitcount = 0xFF; // reset bitcount to 0xFF
      }
}
```

Abb. 21: Code-Beispiel für die Interrupt-Service-Routine für den Infrarot-Empfänger

Nun habt ihr hoffentlich Appetit bekommen, in diese Welt der Mikrocontroller-Steuerungen für fischertechnik-Modelle einzusteigen. Ich wünsche euch viel Erfolg und vor allem Spaß dabei.

Referenzen

[1] Atmel: *Datasheet ATMEGA 644P/1284P*, Rev. 8272E, 04/2013.

[2] Dirk Fox: *I²C mit dem TX – Teil 9: LC-Displays*, in dieser Ausgabe der ft:pedia 1/2014.

Computing

I²C mit dem TX – Teil 9: LC-Displays

Dirk Fox

Das Display des TX eignet sich nur sehr eingeschränkt als Ausgabeeinheit oder zur Kontrolle des Programmablaufs: Der Kontrast ist schwach und das 16 Zeichen breite Display sehr klein (2 x 3 cm) und unbeleuchtet. Zudem schaltet es sich bei einigen Programmen nach kurzer Zeit ab – offenbar ein Designfehler. Dank I²C-Schnittstelle ist jedoch Abhilfe möglich: Für kleines Geld gibt es leistungsfähige LC-Displays, die sich vom TX ansteuern lassen.

Hintergrund

Tatsächlich ist die „Benutzerschnittstelle“ des TX, das Display, in mehrerlei Hinsicht ein Ärgernis: Je nach Lichteinfall ist es kaum ablesbar, da der Kontrast schwach ist und eine Hintergrundbeleuchtung fehlt. Zudem neigt es bei manchen Programmen dazu, nach z. T. sehr kurzer Betriebszeit den Dienst zu quittieren. Eine effektive Unterstützung bei der Fehlersuche im Download-Mode ist daher damit ebenso wenig möglich wie eine verlässliche Anzeige von Mess- oder Zählwerten.

Bis vor wenigen Jahren waren Flüssigkristallanzeigen (*Liquid Crystal Displays*, LCD) relativ kostspielig, daher wurde an LCDs gerne gespart. Inzwischen sind großflächige und mit einer LED-Hintergrundbeleuchtung ausgestattete LCDs zumindest für monochrome Textausgabe zu moderaten Preisen als I²C-Komponenten erhältlich.

1 Das Conrad-LCD

LCDs mit einem Controller für die zeilenweise Textausgabe richten sich meist nach der Spezifikation des HD44780 von Hitachi [1]. Ein ebensolches – zweizeiliges – 5 *V*-Display (1,5 x 5 *cm*, ohne Hintergrundbeleuchtung) gibt es „ready for TX“ von Conrad Electronic (Best.-Nr. 198330, Abb. 1) für gut 37 € (zzgl. Versand) – nicht gerade günstig. Dafür enthält die I²C-Bibliothek von Robo Pro bereits einen Treiber (LCD-PCF8574-HD44780) für dieses Display, und der Wannenstecker des dem TX beiliegenden Extension-Kabels passt exakt in die Anschlussbuchse.

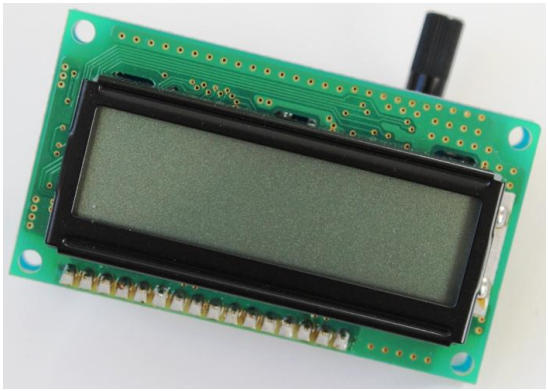


Abb. 1: Conrad-LCD (2 x 16 Zeichen)

Für die Ansteuerung über das I²C-Protokoll kommt der 8-bit-Expander PCF8574 [2] von NXP (aus dem Jahr 1994, damals noch Philips) zum Einsatz, der das serielle I²C-Protokoll in eine parallele Ansteuerung des Displays umsetzt.[1] Der Kontrast des Displays lässt sich manuell über ein Poti einstellen (Abb. 2).

[1] Von diesem IC wird noch an anderer Stelle zu reden sein, denn mit ihm lassen sich die begrenzten I/O-Ports des TX um ein Vielfaches erweitern.

Voreingestellt ist die I²C-Adresse 0x20; sieben weitere Adressen, 0x21 bis 0x27, können über Jumper auf dem Board des PCF8574 ausgewählt werden (Abb. 2). Damit lassen sich bis zu acht LCDs an einem I²C-Bus getrennt adressieren.

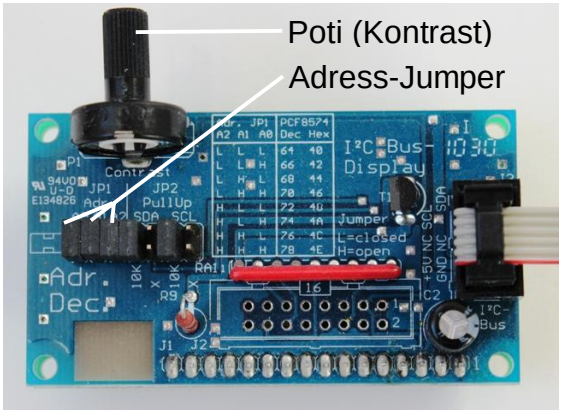


Abb. 2: Poti und Adress-Jumper des Conrad-Displays

Der PCF8574 beherrscht lediglich den *Standard Mode*. Tatsächlich sind aber nicht die 100 *kHz* des I²C-Protokolls der ‚Flaschenhals', sondern die Verarbeitungsgeschwindigkeit des HD44780.

Seit kurzem gibt es ein Nachfolgemodell, das ebenfalls zweizeilige (2 x 16 Zeichen) LC-Display C-Control Pro AVR 32-bit (Conrad Best.-Nr. 192602) mit blauem Display und Hintergrundbeleuchtung – für nur noch 25 € (zzgl. Versand), ebenfalls mit TX-kompatiblem Wannenstecker.

Der I²C-Treiber aus der Robo Pro-Bibliothek funktioniert sicherlich auch mit diesem Board – das habe ich allerdings nicht getestet.

Auf die Ansteuerung dieses Displays unter Robo-Pro will ich hier nicht näher eingehen, da es interessante Alternativen gibt – und die Programmierung der des im folgenden Abschnitt vorgestellten LCD2004 ähnelt.

2 Das LCD2004

Wer bereit ist, sich ein passendes Anschlusskabel zu basteln, erhält für einen niedrigeren Preis ein erheblich größeres Display (vierzeilig mit je 20 Zeichen; 2,5 x 7,5 *cm*) inklusive LED-Backlight: das LCD2004 mit einem PCF8574T (Abb. 3).

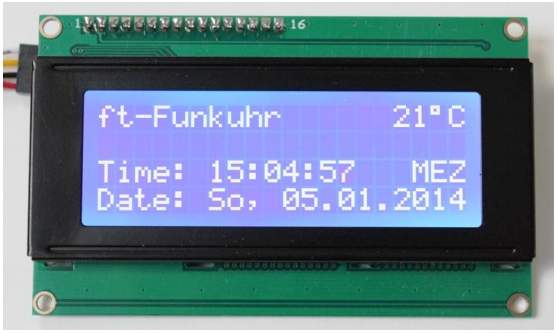


Abb. 3: LCD2004 (4 x 20 Zeichen)

Die Helligkeit der LED-Hintergrundbeleuchtung kann mit einem auf der Platine des PCF8574T angebrachten Poti mit einem kleinen Schraubenzieher manuell eingestellt werden (Abb. 4). Der mit ‚LED' bezeichnete Jumper aktiviert die LED-Hintergrundbeleuchtung. Das Display wird unter anderem angeboten von Sainsmart (15 US$ inkl. Versand) und EXP (20 € zzgl. Versand).

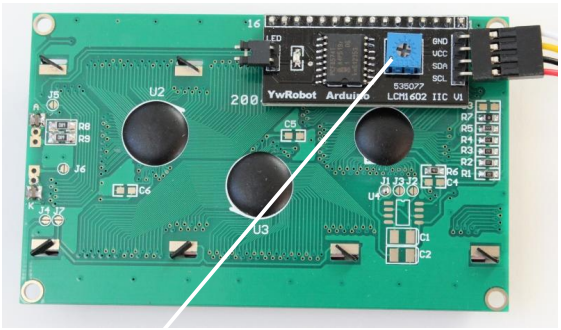


Abb. 4: Poti zur Einstellung der Helligkeit des LED-Backlights

Der zweizeilige „kleine Bruder" des LCD2004, das Board LCD1602 – bei EXP für knapp 14 € (zzgl. Versand) erhältlich – ähnelt technisch dem neuen Conrad-Display, verfügt allerdings wie das LCD2004 nur über die feste I²C-Adresse 0x27 (siehe unten).

Anschluss an den TX

Der PCF8574T erfordert eine Betriebsspannung zwischen 2,5 und 6 *V*. Das Display kann daher direkt am TX betrieben werden. Geliefert wird es mit einem vieradrigen Anschlusskabel. Wer über ein Messer und eine sechspolige Pfostenbuchse verfügt, hat daraus im Handumdrehen ein TX-kompatibles Anschlusskabel gefertigt (siehe ft:pedia 4/2013, S. 28 [3]). Die Kabel haben bereits die „richtige" Belegungsreihenfolge (GND, VCC, SDA, SCL) und müssen daher zwischen Wannenstecker und Display-Kontakten nicht gekreuzt werden. Die Belegung der Kontakte ist außerdem gut lesbar auf dem Board gekennzeichnet.

I²C-Protokoll

Da das LCD via I²C über den schon etwas angejahrten PCF8574 adressiert wird, ist auch hier die Übertragungsgeschwindigkeit auf den *Standard Mode* (100 *kHz*) beschränkt [4].

Die feste I²C-Slave-Adresse des Expanders ist 0x27 (= 0100 111). Anders als beim oben erwähnten Conrad-LCD-Modul ist die Adresse beim LCD2004 fest. Daher können auf einem I²C-Bus ohne Multiplexer nicht mehrere LCD2004 unabhängig voneinander adressiert werden.

Der PCF8574 kennt nur einen Schreib- und einen Lesebefehl. Der Schreibbefehl legt die acht übermittelten Datenbits auf acht Datenleitungen, über die der HD44780 angeschlossen ist. Der Lesebefehl liest die an diesen acht Leitungen anliegenden Potentiale aus und schickt sie als Datenbyte über den I²C-Bus.

Der HD44780U

Die Spezifikation des Hitachi-Controllers HD44780 aus dem Jahr 1999 [1] hat sich zum „De-facto-Standard" für die Ansteuerung von LC-Text- (*Dot Matrix*) Displays entwickelt – sogar eine Wikipedia-Seite ist ihm gewidmet [5].

Der Controller verfügt über ein 8-bit *Data Register* und ein ebenso großes *Instruction Register*. An den Controller können 8-bit-ASCII-kodierte Zeichen zur Ausgabe auf dem Display (*Data*) und eine Auswahl von Befehlen (*Instructions*) übermittelt werden.

Neben den Datenleitungen werden dafür die folgenden vier Steuerleitungen benötigt:

- RS: *Register Selection* – *Data* (= 1), *Instruction* (= 0)
- RW: *Read* (= 1), *Write* (= 0)
- EN: *Enable*, lässt den Controller, das Daten-/*Instruction*-Register auswerten
- BL: *Backlight on* (= 1), *off* (= 0)

Die maximale Ausführungszeit des Controllers für *Instructions* liegt bei 37 *µs*; einzig der *Return Home*-Befehl benötigt 1,52 *ms*.

Instructions

Der HD44780 beherrscht die in Tab. 1 zusammengefassten Kommandos. Sie werden über das *Instruction Register* an den Controller übergeben (Steuerleitung RS = 0).

Wert	Befehl (*Instruction*)
0x01	*Clear Screen*
0x02	*Cursor Home*
0x04-0x07	*Entry Mode Set*
0x08-0x0F	*Display Control*
0x10-0x1C	*Cursor/Display Shift*
0x20-0x3C	*Function Set*
0x40-0x7F	*Set CGRAM Address*
0x80-0xFF	*Set DDRAM Address*

Tab. 1: Instructions des HD44780

Die Befehle *Clear Screen* (löscht das Display und setzt den Adresszähler = 0)

und *Cursor Home* (setzt nur den Adresszähler = 0, also den Cursor auf die Position links oben) kommen dabei ohne Parameter aus. Bei allen anderen, im Folgenden näher erläuterten *Instructions* ändert sich der Befehlswert in Abhängigkeit von den Parametern.

Entry Mode Set

Der Befehl *Entry Mode Set* (Bit 2 = 1) wählt die Schreibrichtung auf dem *Display* und legt fest, ob das Display bei Zeicheneingabe „geschoben“ werden soll:

- *Schreibrichtung*: rechts (Hochzählen des Display-Adresszählers): Bit 1 = 1; links: Bit 1 = 0.
- *Shift*: Verschieben des Displays (Bit 0 = 1).

Um beispielsweise die Schreibrichtung auf „rechts“ bei gleichzeitigem Verschieben des Displays zu stellen, muss der Befehlswert 0x07 an den Controller übergeben werden.

Display Control

Der Befehl *Display Control* (Bit 3 = 1) schaltet das Display an bzw. aus und legt fest, ob der Cursor sichtbar sein und blinken soll:

- *Display*: An: Bit 2 = 1; Aus: Bit 2 = 0
- *Cursor*: An: Bit 1 = 1; Aus: Bit 1 = 0.
- *Blinken*: An: Bit 0 = 1; Aus: Bit 0 = 0.

Einen sichtbar blinkenden Cursor stellt man also mit dem Befehlswert 0x0F ein.

Cursor/Display Shift

Der Befehl *Cursor/Display Shift* (Bit 4 = 1) verschiebt den Cursor oder den Displayinhalt um eine Position nach rechts oder links:

- *Shift*: Display: Bit 3 = 1, Cursor: Bit 3 = 0.
- *Richtung*: rechts: Bit 2 = 1, links: Bit 2 = 0.

So lässt sich beispielsweise mit dem Befehl 0x18 und einer Schleife eine Ausgabe in eine Laufschrift verwandeln; der Befehl 0x10 bewegt den Cursor nach links, 0x14 nach rechts. Dabei ist zu beachten, dass aufgrund der Adressierung des HD44780 (siehe unten) die Zeilen eins und drei sowie zwei und vier jeweils wie eine Zeile behandelt werden: Zeichen, die aus Zeile eins nach links „herausfallen“, werden in Zeile drei von rechts „eingeschoben“.

Function Set

Mit dem Befehl *Function Set* (Bit 5 = 1) erfolgen mehrere, für das Funktionieren der Schnittstelle zwischen PCF8574T und HD44780 wesentliche Festlegungen: die Zahl der Datenbits (4/8), die Zahl der Display-Zeilen (ein-/mehrzeilig) und der verwendete *Character Font* (5 x 8 bzw. 5 x 10 Punkte):

- *Data Length* (DL): acht Bits: Bit 4 = 1, vier Bits: Bit 4 = 0.
- *Number of Lines* (N): mehrzeilig: Bit 3 = 1, einzeilig: Bit 3 = 0.
- *Character Font* (F): 5 x 10: Bit 2 = 1, 5 x 8: Bit 2 = 0.

Die Bits 1 und 0 können beliebige Werte annehmen. Bei mehrzeiligen Displays wird nur der Font 5 x 8 unterstützt.

Das Display-Board LCD2004 arbeitet mit vier Datenbits, einem mehrzeiligen Display und dem einfachen *Character Font* (5 x 8) – einzustellen mit dem Befehlswert 0x28 (siehe Abschnitt Initialisierung).

Set DDRAM Address

Die für die Ausgabe auf dem LC-Display vorgesehenen Zeichen werden in einem 80 Byte großen RAM-Bereich (*Display Data* RAM, DDRAM) abgelegt. Der ‚Zeiger’ auf die aktuelle Schreibposition wird mit

dem Befehl *Set DDRAM Address* gesteuert. Dafür wird im *Instruction*-Byte Bit 7 gesetzt; Bit 0 bis 6 enthalten die gewählte DDRAM-Adresse.

Will man die Schreibposition auf die erste Position der ersten Zeile setzen, muss der Befehlswert 0x80 an den Controller übergeben werden.

Zeichensatz

Der HD44780 unterstützt den Zeichensatz des *American Standard Code for Information Interchange* (ASCII) [6]. Darin ist die Bedeutung der Zeichenkodes 0x20 bis 0x7E international vereinheitlicht. Die Zeichenkodes 0x00 bis 0x1F sowie 0x7F sind darin mit Steuerzeichen belegt; die 128 Zeichen von 0x80 bis 0xFF können unterschiedliche Zeichen repräsentieren.

Der HD44780 wird mit zwei verschiedenen *Character ROMs* angeboten: einer japanischen (ROM-Code A00) und einer europäischen Variante (ROM-Code A02). Die beiden Zeichensätze unterscheiden sich wie folgt (siehe Tabelle 4 der Spezifikation [1], S. 17-18):

- Der japanische Zeichensatz enthält statt des *Backslash* (‚\') in 0x5C das Währungszeichen für Yen (‚¥'), und statt der *Tilde* (‚~') steht der Zeichencode 0x7E für einen Pfeil (‚→').
- Der europäische Zeichensatz entspricht in den oberen 128 Zeichen dem Standard ISO/IEC 8859-1 [7], der japanische dem Shift-JIS-Zeichensatz von 1982 [8].
- Alle Steuerzeichen (0x00 bis 0x1F) werden im japanischen Zeichensatz als Leerzeichen angezeigt; im europäischen Zeichensatz sind sie mit Sonderzeichen und grafischen Symbolen belegt.

Das LCD2004 wird mit dem japanischen *Character ROM* geliefert. Deutsche Umlaute und Sonderzeichen finden sich daher im Zeichensatz nicht an den gewohnten Stellen, sind jedoch enthalten.

Mit dem Befehl *Set CGRAM Address* können einige wenige eigene Zeichen spezifiziert werden – entweder acht im 5 x 8-*Font* oder vier im 5 x 10-*Font*. Dazu werden die Zeichen in das 64 Byte große CGRAM geschrieben (je eine 5 bit breite Bitmuster-Zeile je CGRAM-Byte). Auf eine nähere Erläuterung dieser für die meisten Anwendungen nicht relevanten Funktion verzichte ich an dieser Stelle; Details finden sich in der Hitachi-Spezifikation [1].

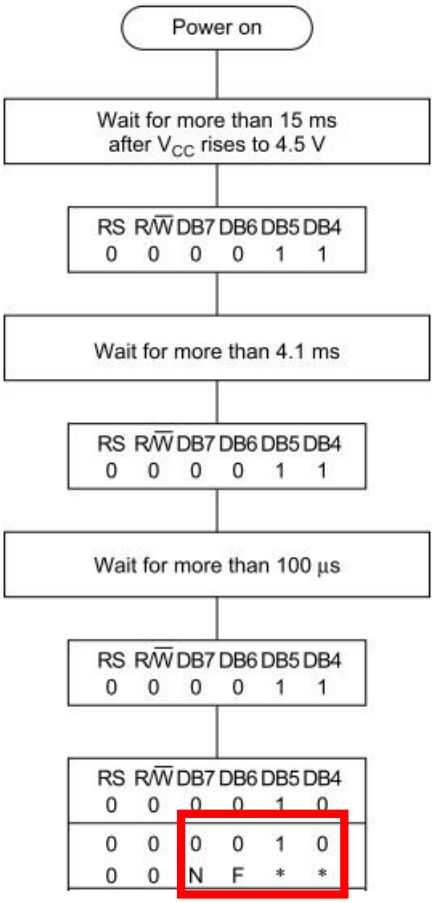


Abb. 5: Initialisierung (aus [1], Abb. 24, S. 46)

Initialisierung

Nach dem Einschalten der Stromzufuhr ist der LCD-Controller im 8-bit-Modus und erwartet ein einzeiliges Display. Display und Cursor sind ausgeschaltet, der Eingabe-Modus auf ‚*No Shift*' voreinge-

stellt, d. h. kein Scrollen des Displays, und ‚Inkrement', d. h. Schreiben von links nach rechts (S. 23 [1]).

Diese Default-Einstellung muss für unser LCD2004 zu Programmbeginn auf den 4-bit-Modus und ein mehrzeiliges Display umgestellt werden (*Function Set*). Diese Initialisierung ist trickreich – geht man nicht exakt nach der Spezifikation vor, dann sind die Resultate nicht vorhersagbar.

Abb. 5 gibt den Ablauf der Initialisierung sehr gut wieder: In dem roten Kästchen erkennt man den *Function Set*-Befehl. Daran sollte sich die Einstellung des *Entry Mode* (siehe oben) anschließen. Wer den Ablauf lieber als Programmcode möchte, dem sei [9] empfohlen.

Zeichenausgabe

Die auf dem LC-Display auszugebenden Zeichen werden vom Controller in einen 80 Byte großen Zeichenpuffer (DDRAM) geschrieben. Das erfolgt – bei entsprechender Initialisierung – mit einem Auto-Inkrement, d. h. der Adresszähler des DDRAM wird nach jedem Zeichen-Ausgabebefehl automatisch um eins erhöht, die Zeichen werden von rechts nach links geschrieben.

Bei der Initialisierung (*Clear Display*) wird der Adresszähler des DDRAMs auf Null gesetzt. Die erste Zeichenausgabe beginnt also in der linken oberen Ecke des Displays, sofern man den Adresswert nicht mit der *Set DDRAM Address Instruction* verändert hat (siehe oben).

Die Ausgabe eines einzelnen ASCII-Zeichens erfolgt durch einen Daten-Schreibbefehl (Abb. 6).

Da der HD44780 zweizeilige Displays erwartet, bilden die ersten 40 Byte (0x00-0x27) die erste Ausgabezeile, die Adressen 0x40-0x67 die zweite. Bei unserem vierzeiligen Display muss die Cursorposition nach der Zeichenausgabe korrigiert werden, wenn ein Zeilenende erreicht ist, denn der Adressraum verteilt sich wie folgt auf die Zeilen:

- Erste Zeile: Adressen 0x00-0x13
- Zweite Zeile: Adressen 0x40-0x53
- Dritte Zeile: Adressen 0x14-0x27
- Vierte Zeile: Adressen 0x54-0x67

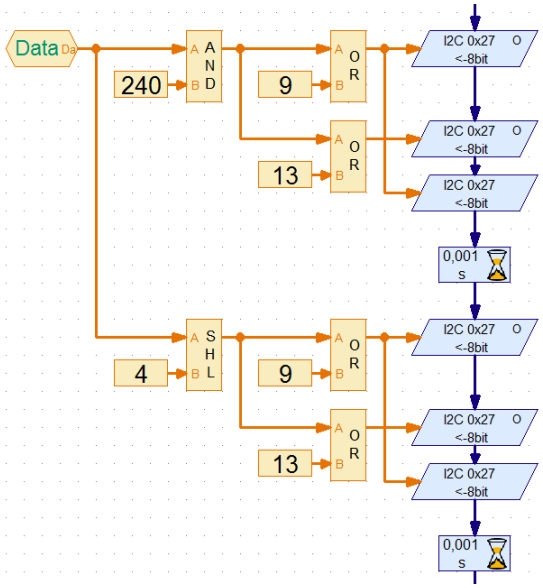


Abb. 6: Schreiben eines ASCII-Zeichens (Data) an die aktuelle Position im DDRAM

RoboPro-Treiber

Der oben erwähnte, mit Robo Pro ausgelieferte I²C-Treiber für das Conrad-Display (LCD-PCF8574-HD44780) funktioniert nicht mit dem LCD2004 – nicht nur, weil jener die I²C-Adresse 0x20 verwendet, sondern auch, weil die Anschlüsse zwischen PCF8574 und HD44780 beim LCD2004 anders belegt sind (Tab. 2).

7	6	5	4	3	2	1	0
D_3	D_2	D_1	D_0	BL	EN	RW	RS

Tab. 2: I²C-Datenbyte an den PCF8574: vier Datenbits, vier Steuerbits

Im Downloadbereich der ft:c habe ich einen Treiber für das LCD2004 hochgeladen, der – bis auf die Erzeugung eigener Zeichen – alle Kommandos des LCD2004 unterstützt. Bei Bedarf lässt sich der Trei-

ber mit wenigen Änderungen an das zweizeilige LCD1602 anpassen.

3 Das LCD05

Eine etwas teurere Alternative ist das LCD05 von Devantech Ltd. [10], das ebenfalls mit einer Betriebsspannung von 5 *V* arbeitet. Es ist in vier Varianten erhältlich: mit zwei Zeilen à 16 Zeichen (wie die Conrad-LCDs und das LCD1602) und vierzeilig mit je 20 Zeichen (wie das LCD2004), entweder mit weißer Schrift auf blauem oder mit schwarzer Schrift vor grünem Hintergrund (Abb. 7).

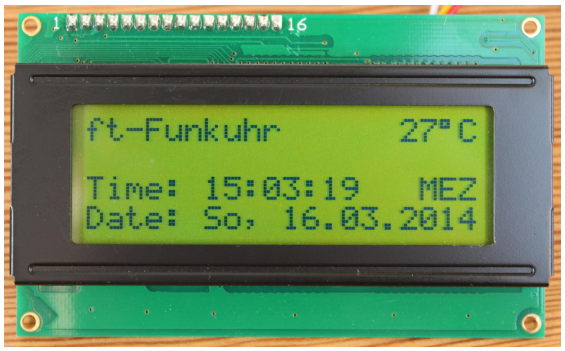


Abb. 7: LCD05 (Devantech, 4 x 20 Zeichen)

Das LCD05 wird unter anderem von nodna, manu systems und lipoly.de angeboten und kostet in der 4 x 20-Zeichen-Variante knapp 30 €.

Anschluss an den TX

Die Ansteuerung des LCD erfolgt über einen PIC18F23K22 von Microchip Technology Inc., der eine Betriebsspannung von 2,3 bis 5,5 *V* erwartet und daher direkt am I²C-Bus des TX betrieben werden kann [11]. Dazu muss der auf dem Board mitgelieferte Mode-Jumper (siehe Abb. 8) entfernt werden. Die Belegungsreihenfolge der I²C-Anschlüsse entspricht der anderer Devantech-Sensoren (siehe z. B. [12]):

- Rot: VCC (5 *V*)
- Schwarz: SDA (Datenleitung)
- Gelb: SCL (Clock)
- Weiß: GND

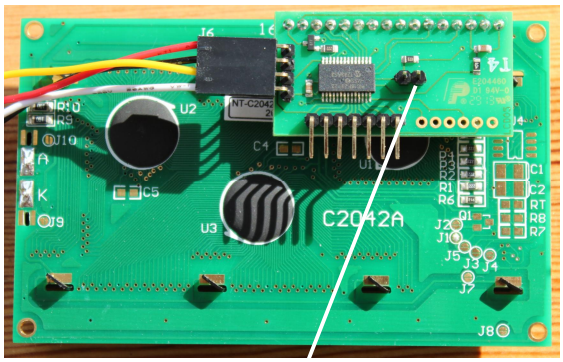


Abb. 8: Anschlüsse des LCD05 und Mode-Jumper

Die Verwendung des PIC18F23K22 wirkt ein wenig, als wolle man mit Kanonen auf Spatzen schießen, denn der IC ist ein sehr leistungsfähiger, mit 64 MHz getakteter RISC-Prozessor mit 512 Byte SRAM, 256 Byte EEPROM, 8 kB Flash Memory Programmspeicher, 25 I/O-Ports, 19 A/D-Wandlern (10 bit) sowie drei 8-bit- und vier 16-bit-Timern. Allerdings unterstützt der Controller nach Datenblatt nicht die volle Spezifikation des *I²C Fast Mode*. Der Betrieb mit 400 kHz funktionierte bei meinen Tests jedoch problemlos.

Ohne Hintergrundbeleuchtung liegt der Stromverbrauch aller vier Varianten des Displays bei lediglich 5 *mA*; mit Hintergrundbeleuchtung benötigen die grünen 80 bzw. 135 *mA*, die blauen lediglich 40 bzw. 55 *mA*. Text auf dem blauen Display ist allerdings nur mit Hintergrundbeleuchtung lesbar.

Die I²C-Adresse des Displays ist auf 0x63 voreingestellt. Sie kann via I²C-Befehl umgestellt werden auf 0x64, 0x65, 0x66 oder 0x67. Damit lassen sich bis zu fünf LCD05-Displays auf demselben Bus separat adressieren. Beim Einschalten des Displays wird die eingestellte I²C-Adresse angezeigt, sofern die Anzeige nicht unterdrückt oder geändert wurde (siehe unten).

Datenregister

Das LCD05 verfügt über insgesamt vier Datenregister. Register 0x00 enthält die Zahl der noch freien Bytes des 100 Byte

großen FIFO-Anzeigepuffer (*First In – First Out*). Dieser Wert ist relevant, um die Geschwindigkeit des Programms mit der des Displays zu synchronisieren, wenn man z. B. schnell wechselnde Anzeige- oder Lauftexte ausgeben möchte. In der Regel ist der Puffer jedoch groß genug, um auszugebende Zeichen zwischenzuspeichern, bis sie auf dem Display angezeigt werden.

Reg.	Inhalt
0x00	*Command Register* (write) Freie Bytes im FIFO-Puffer (read)
0x01	*Keypad Low Byte* (read only)
0x02	*Keypad High Byte* (read only)
0x03	Version (read only)

Tab. 3: Datenregister des LCD05

Register 0x01 und 0x02 sind nur bei gleichzeitiger Verwendung einer direkt anschließbaren Zifferntastatur relevant und werden im Folgenden nicht weiter betrachtet.[2] Register 0x03 enthält die Versionsnummer der Firmware; mein Exemplar des LCD05 liefert die Versionsnummer 1 (Tab. 3).

Befehlsregister

Wie bei anderen Devantech-Sensoren [12] ist auch beim LCD05 das Register 0x00 das *Command Register*. Auszuführende Befehle werden mit einem I²C-Schreib-Kommando in dieses Register geschrieben.

Der Controller des Displays unterstützt insgesamt 26 Kommandos (Tab. 4). Sie wurden im Wertebereich der nicht druckbaren Zeichen des ASCII-Codes (1 bis 31) untergebracht. Die sechs Steuerkommandos *Backspace* (0x08), *Horizontal Tab* (0x09), *Line Feed* (0x0A), *Vertical Tab* (0x0B), *Clear Screen* (0x0C) und *Carriage Return* (0x0D) setzt der Controller geeignet für das verwendete Display um.

Wert	Kommando
0x01	*Cursor Home*
0x02	*Set Cursor* (1-80/32)
0x03	*Set Cursor* (Zeile, Spalte)
0x04	*Hide Cursor*
0x05	*Show Underline Cursor*
0x06	*Show Blinking Cursor*
0x08	*Backspace*
0x09	*Horizonzal Tab* (*Default*: 4)
0x0A	*Smart Line Feed*
0x0B	*Vertical Tab*
0x0C	*Clear Screen*
0x0D	*Carriage Return*
0x11	*Clear Column*
0x12	*Tab Set*
0x13	*Backlight On*
0x14	*Backlight Off* (*Default*)
0x15	*Disable Startup Screen*
0x16	*Enable Startup Screen*
0x17	*Save as Startup Screen*
0x18	*Set Display Type*
0x19	*Change Address*
0x1B	*Custom Char Generation*
0x1E	*Contrast Set*
0x1F	*Brightness Set*

Tab. 4: Kommandos des LCD05 (Devantech)

Fast alle Kommandos – bis auf die im Folgenden näher erläuterten Befehle – kommen ohne Parameter aus.

[2] Mit diesen beiden Registern – respektive der anschließbaren Zifferntastatur – wird sich ein eigener I²C-Beitrag beschäftigen.

Set Cursor

Mit den beiden *Set Cursor*-Befehlen 0x02 und 0x03 wird der Cursor auf ein bestimmtes Ausgabefeld gesetzt. An dieser Stelle erfolgt die Ausgabe des nächsten Zeichens.

Als Parameter erwartet das Kommando 0x02 die absolute Zeichenposition, auf der der Cursor platziert werden soll (je nach Display-Typ also 1-80 resp. 1-32). Das Kommando 0x03 erwartet die Koordinaten: Zeile (1-4/2) und Spalte (1-20/16) in jeweils einem Byte-Parameter.

Anders als beim LCD2004 übernimmt hier der PIC18F23K22 die Umrechnung der Positionsdaten in die Display-RAM-Adresse des HD44780. Das macht die Ansteuerung wesentlich einfacher und intuitiver.

Set Display Type

Damit die Umrechnung der Cusor-Position PIC18F23K22 in Zeile und Spalte korrekt erfolgt, muss – am besten bei der Initialisierung, siehe unten – das richtige Display eingestellt sein. Vier verschiedene Display-Typen lassen sich per *Set Display Type*-Befehl (0x18) einstellen:

Kennung	LCD-Typ
0x03	20 x 4 (grün)
0x04	20 x 4 (blau)
0x05	16 x 2 (grün)
0x06	16 x 2 (blau)

Tab. 5: LC-Display-Typen

Der Wert war bei meinem Display bereits korrekt voreingestellt – daher muss man diese Setzung bei der Initialisierung eigentlich nur vornehmen, wenn man nicht sicher sein kann, dass kein anderes Programm die Einstellung verändert hat.

Tab Set

Der Befehl *Tab Set* (0x12) ändert den Tabulatorwert (default: 4) auf einen Wert zwischen 0 und 10. Er wird als Byte-Parameter übergeben. Damit ändert sich die Schrittweite des *Horizontal Tab*-Kommandos.

Contrast und Brightness Set

Mit den beiden Kommandos *Contrast Set* (0x1E) und *Brightness Set* (0x1F) werden Kontrast und Helligkeit der Zeichen auf dem Display zwischen 0 und 255 eingestellt. Als Parameter wird jeweils ein Byte übergeben. *Brightness Set* stellt die Helligkeit der Hintergrundbeleuchtung ein. Ist sie ausgeschaltet, wird sie durch dieses Kommando automatisch wieder eingeschaltet.

Zeichenausgabe und Zeichensatz

Wird ein Wert größer oder gleich 32 (0x20) in das Befehlsregister geschrieben, wird das entsprechende ASCII-Zeichen direkt auf dem Display ausgegeben.

Der Zeichensatz ist wie beim LCD2004 die japanische Variante des *Character ROMs* (siehe Abschitt 2).

Mit dem Befehlscode 0x1B können acht eigene Zeichen (Format 5 x 8) erzeugt und im Zeichensatz des Controllers an den Adressen 0x80 bis 0x87 abgelegt werden. Dazu wird der Befehl *Custom Char Generation* in das *Command Register* geschrieben, gefolgt von der Adresse des Zeichensatzes (ein Byte) und acht Bytes, die die 5 x 8-bit-Matrix des Zeichens zeilenweise (in den niederstwertigen fünf Bit) beschreiben. In diesen acht Byte muss jeweils das höchstwertige Bit gesetzt sein.

Anschließend können die neu generierten Zeichen einfach durch Übermittlung der Speicheradresse auf dem Display ausgegeben werden.

Initialisierung

Vor der Nutzung des Displays sollte – z. B. in einer Initialisierungsfunktion – das korrekte Display eingestellt werden (*Set Display Type*), damit bei der Ausgabe die

Modell	Conrad-LCD	LCD2004	LCD05
I²C-Adressen	8	1	5
Geschwindigkeit	100 kHz	100 kHz	400 kHz
Anschluss	TX-Ext.-Kabel	Wannenstecker	Wannenstecker
Zeichen	2 x 16	4 x 20	4 x 20
Helligkeit	Poti	Poti	Software
Kommandos	6	10	20
Besonderheiten	–	Laufschrift	Start-Screen, Anschluss Keypad
Preis	25-37 €	20 €	30 €

Tab. 6: Eigenschaften der drei vorgestellten LC-Displays

Displaypositionen der Zeichen richtig berechnet werden. Außerdem empfiehlt es sich, die Helligkeit und den Kontrast des Bildschirms einzustellen.

Möchte man mehrere LCD05-Displays an einem I²C-Bus betreiben oder kollidiert die Adresse 0x63 mit einem anderen I²C-Device, muss zuvor noch eine I²C-Adressänderung bei den Displays vorgenommen werden, damit sie einzeln adressiert werden können. Devantech-Sensoren verwenden dafür eine längere Befehlsfolge, die mit dem *Change Address*-Kommando 0x19 beginnt:

0x19 0xA0 0xAA 0xA5,

gefolgt von der gewünschten (8-bit-) I²C-Adresse (0xC6, 0xC8, 0xCA, 0xCC oder 0xCE). Sofern das Start-Display nicht per Kommando deaktiviert wurde, wird beim nächsten Einschalten des Displays die I²C-Adresse des LCD angezeigt.

RoboPro-Treiber

Für das Vorgängermodell von Devantech, das LCD03, gibt es in der Robo Pro-Bibliothek der aktuellen Programmversion – Rubrik I²C – einen Treiber von Rei Vilo, der auch mit dem LCD05 genutzt werden kann, da die Befehle des LCD03 auch vom LCD05 unterstützt werden. Allerdings beschränkt sich der Treiber auf die Befehle *Set Cursor*, *Clear Screen* und *Backlight On/Off*.

Im Download-Bereich der ft:c findet sich ein umfangreicher LCD05-Treiber, der (bis auf den Zeichengenerator) alle 26 Kommandos des LCD05 über Unterprogramme zugänglich macht. Einige zusammengehörige Kommandos wurden darin kombiniert – so z. B. die Darstellung des Cursors (0x04-0x06) oder die Aktivierung bzw. Deaktivierung der Hintergrundbeleuchtung (0x13, 0x14).

Bei den Tests des Treibers zeigte es sich, dass es im Download-Mode zu Fehlern (oder Verzögerungen) bei den I²C-Kommandos kommen kann. Mit der Option „Wiederholen bis erfolgreich“ arbeiten die Befehle jedoch zuverlässig.

Beispielanwendungen

Auf dem LC-Display lassen sich deutlich mehr Informationen unterbringen als auf dem TX oder der LED-Anzeige aus ft:pedia 4/2012 [13]. Eine nahe liegende Anwendung ist daher ein Display für die *Real Time Clock* (RTC) oder die ft-Funkuhr (ft:pedia 4/2013 [3] bzw. ft:pedia 3/2012 [14]). Im Downloadbereich der ft:c findet ihr einige Robo Pro-Beispielprogramme.

Auch der GPS-Empfänger aus ft:pedia 3/2013 [15] wäre für ein größeres Display

dankbar. Und für Abfahrts-Anzeigen an S-Bahn-Modell-Haltestellen oder –Bahnhöfen eignen sich die LC-Displays ebenfalls hervorragend – mit Laufschriften für Fahrplanabweichungen oder Meldungen. Besonders interessant ist in diesem Zusammenhang, dass sich an das LCD05 ein Zifferntastenblock anschließen und über die Datenregister auslesen lässt. Damit kann man die ft-Funkuhr mit RTC DS3231 zu einem Alarm-Wecker erweitern… – mehr dazu in der nächsten Ausgabe der ft:pedia.

Empfehlung

Die vorgestellten Displays haben unterschiedliche Vor- und Nachteile. Um eure Kaufentscheidung zu erleichtern, habe ich die meiner Ansicht nach wesentlichen Kriterien in einer Tabelle zusammengefasst (siehe Tab. 6).

Quellen

[1] Hitachi: *HD44780U (LCD-II): Dot Matrix Liquid Crystal Display Controller/Driver*, Datasheet Rev. 0.0, September 1999.

[2] Philips: *PCF8574: Remote 8-bit I/O expander for I2C-bus*, Datasheet, September 1994.

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

[4] NXP: *PCF8574; PCF8574A: Remote 8-bit I/O expander for I2C-bus with interrupt*, Product data sheet v.5, 27.05.2013.

[5] Wikipedia: *HD44780*.

[6] Wikipedia: *American Standard Code for Information Interchange*.

[7] Wikipedia: *ISO 8859-1*.

[8] Microsoft: *Windows Codepage 932 – Japanese Shift-JIS*.

[9] Tim Starling (et.al.): *LiquidCrystal_I2C*, Arduino-Code (C++), dfrobot.com, 21.12.2011.

[10] Devantech: *LCD05 – I2C/Serial LCD*, Technical Documentation.

[11] Microchip: *PIC18F23K22*, Data Sheet, Revision F, 05.06.2012.

[12] Dirk Fox: *I²C mit dem TX – Teil 8: Ultraschall-Sensor*. ft:pedia 4/2013, S. 35-40.

[13] Dirk Fox: *I²C mit dem TX – Teil 2: LED-Display*. ft:pedia 4/2012, S. 32-37.

[14] Dirk Fox, Dirk Ottersmeyer: *Bau einer ft-Funkuhr*. ft:pedia 3/2012, S. 4-10.

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

Pneumatik

Druckluftsteuerungen (Teil 1)

Stefan Falk

Wie in ft:pedia 4/2013 versprochen, beginnen wir mit diesem Artikel eine kleine Reihe zu pneumatischen Steuerungen. Im Gegensatz zur einfachen manuellen Ansteuerung eines Zylinders mit dem aktuellen Handventil werden wir Logikschaltungen und Steuerungen in reiner Pneumatik realisieren.

Einführung

Fangen wir mit einem einfachen Beispiel an: Unter Verwendung eines Pneumatik-Handventils können wir einen Zylinder nach Belieben ein- oder ausfahren lassen:

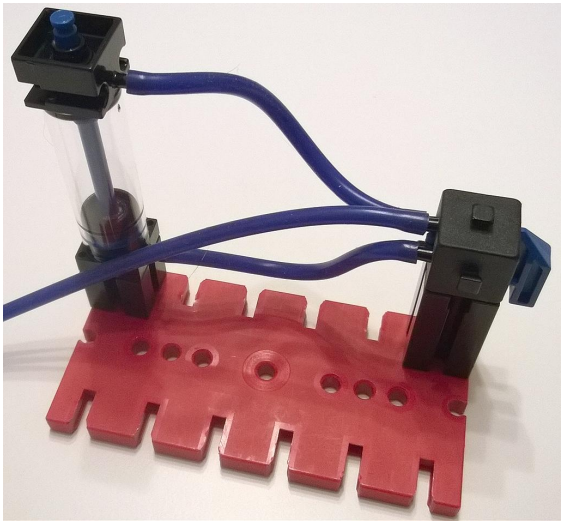


Abb. 1: Handventil steuert Zylinder

Was genau macht das Handventil hier? Zunächst mal besitzt es drei deutlich sichtbare Anschlüsse für Schläuche: Am oberen wird die Druckluft zugeführt. Diese wird je nach Stellung des Drehknopfes auf eine der beiden anderen verteilt.

In der mittleren Stellung wird die Druckluft an keinen der beiden Ausgänge weitergeleitet. In den beiden gedrehten Stellungen hingegen bekommt jeweils einer der Ausgänge die Druckluft zugeführt:

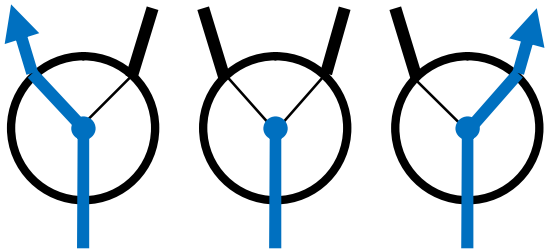


Abb. 2: Scheinbare Funktionsweise des Handventils

Wenn das ein Schalter für elektrischen Strom wäre, endete die Beschreibung der Funktionsweise hier schon. Bei Druckluft müssen wir aber an eine wesentliche Ergänzung denken: Was würde passieren, wenn wir eine Seite des Zylinders unter Druckluft setzen und diesen Anschluss beim Umschalten des Ventils einfach nur zusperren? Die Druckluft wäre ja immer noch im Zylinder, und er würde sich nicht so bewegen, wie wir das wollten, weil nun einfach nur auf beiden Seiten Druckluft anliegt. Die Druckluft muss also unbedingt auch wieder aus dem Zylinder heraus!

Abb. 2 zeigt uns also nur die halbe Wahrheit. Sie stellt nur dar, wie die Druckluft in die Zylinderhälften hinein, aber nicht, wie sie wieder heraus kommt. Tatsächlich hat das Handventil nämlich einen vierten Anschluss: Eine kleine Öffnung auf der Seite der Schlauchanschlüsse. In der mittleren Stellung ist diese Luftauslass-Öffnung mit keinem der drei anderen

Anschlüsse verbunden. In den gedrehten Stellungen aber besteht eine Verbindung zwischen dem gerade nicht mit Druckluft verbundenen Ausgang und der Entlüftung. Auf diese Weise kann die vorher hineingepumpte Luft aus derjenigen Zylinderöffnung ins Freie entweichen, die gerade nicht mit Druckluft beaufschlagt wird.

Damit haben wir mit dem Handventil tatsächlich ein Ventil mit vier Anschlüssen („Wegen“) und drei Stellungen, im Pneumatik-Jargon also ein „4/3-Wege-Ventil“ vorliegen. Das entsprechende Schaltbild macht die Sache deutlicher, als wir es in einer Skizze wie Abb. 2 darstellen können:

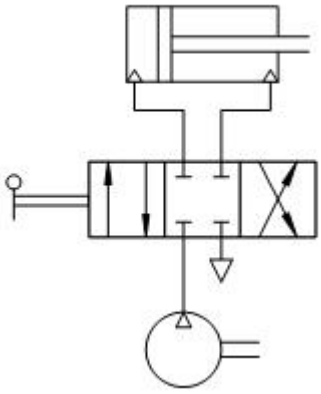


Abb. 3: Schaltbild der Handventil-Nutzung

Das Ventil ist in der mittleren Stellung gezeigt: Alle vier Anschlüsse (Druckluftanschluss, die beiden Ausgänge in Richtung Zylinder sowie die Abluft ins Freie) sind verschlossen, also mit nichts anderem verbunden.

Der Betätigungshebel links vom Ventil im Schaltbild – sinnbildlich für den Drehknopf an unserem Handventil – kann das Ventil in eine von drei Stellungen schalten. Jedes der drei Quadrate im Ventil-Schaltzeichen steht für eine Schaltstellung.

Abb. 4 und 5 zeigen den Druckluftverlauf in den beiden gedrehten Knopfstellungen. Die Pfeile im Schaltbild stellen also jeweils dar, von und nach wo Druckluft strömen kann:

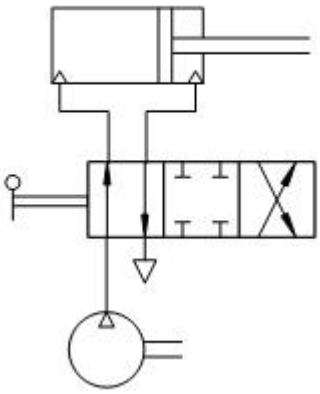


Abb. 4: Zylinder wird ausgefahren

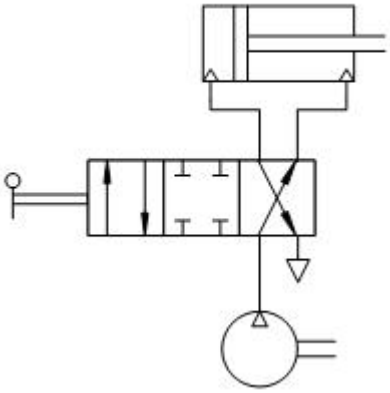


Abb. 5: Zylinder wird eingefahren

Solange wir den Zylinder lediglich ein- und ausfahren wollen, benötigen wir die mittlere Schaltstellung nicht. Für diesen Zweck reicht ein 4/2-Wege-Ventil, also eines mit immer noch vier Wegen/Anschlüssen, aber nur zwei Schaltstellungen, völlig aus:

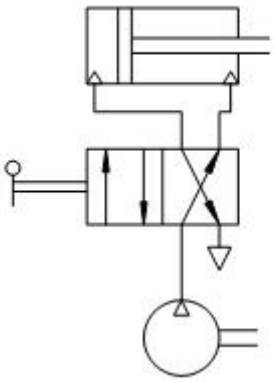


Abb. 6: Zylinderansteuerung mit 4/2-Wege-Ventil

Beim fischertechnik-Handventil entspricht das der ausschließlichen Nutzung der beiden gedrehten Schalterstellungen; die mittlere Stellung wird nicht verwendet.

Automatische Rückstellung

Im Sinne der Elektromechanik-Sprechweise ist unser Handventil ein *Schalter*. Jede gerade eingenommene Stellung bleibt erhalten, bis aktiv eine andere gewählt wird. Eine automatische Rückstellung findet nicht statt. Die pneumatische Entsprechung eines fischertechnik-*Tasters*, der ja nach dem Loslassen automatisch in seine Ruhestellung zurückkehrt, ist ein Ventil mit automatischer Rückstellung. Die in einem Taster verwendete Feder wird im entsprechenden Pneumatik-Schaltzeichen explizit dargestellt:

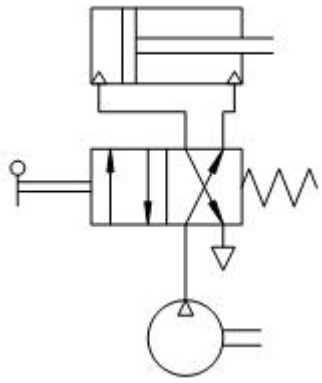


Abb. 7: 4/2-Wege-Ventil mit Rückstellfeder

Zudem muss die Betätigung durchaus nicht manuell geschehen, sondern kann ebenso gut per Druckluft erfolgen – auch wenn diese enorm nützliche Möglichkeit in den aktuellen Pneumatik-Kästen bedauerlicherweise völlig unterschlagen wird.

Ein Beispiel zeigt Abb. 8. Wir haben hier einen „pneumatischen Taster“, also ein 3/2-Wege-Ventil mit Rückstellung, dessen Ausgang per Druckluftleitung das 4/2-Wege-Ventil (ebenfalls mit Rückstellfeder) umschaltet.

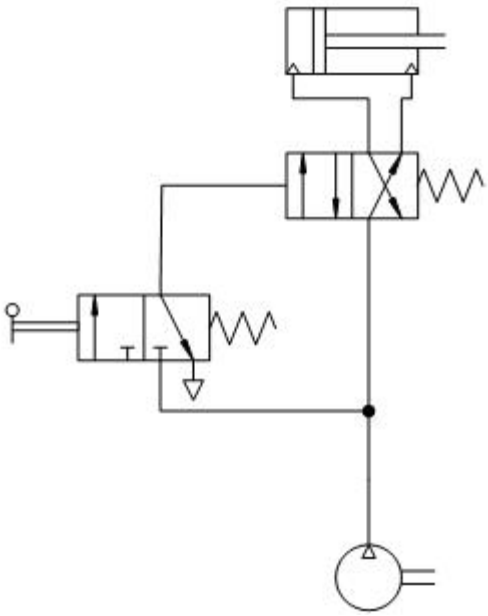


Abb. 8: Ansteuerung eines Ventils durch ein anderes

In den 1981 eingeführten Pneumatik-Kästen von fischertechnik waren tatsächlich öffnende und schließende 3/2-Wege-Ventile enthalten [1]:

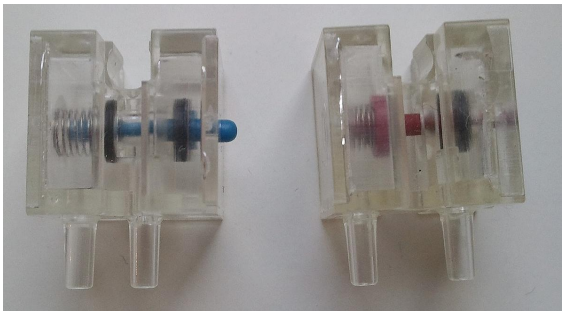


Abb. 9: Öffnende und schließende Festo-Ventile aus der fischertechnik Ur-Pneumatik

Mit den aktuellen Pneumatik-Teilen von fischertechnik ist das leider nicht ohne größeren Aufwand zu realisieren. Als schaltendes Element steht uns lediglich das Handventil zur Verfügung, und das ist einer maschinellen Betätigung nur schlecht, um nicht zu sagen miserabel, zugänglich: Wir können zwar mit etwas Ausdauer Konstruktionen finden, die das Ventil z. B. per Motorkraft umsteuern, aber die Ansteuerung per Druckluft gelingt mangels Alternativen nur über einen

Pneumatikzylinder und endet in einem klobigen, uneleganten Teilegrab. Die Abbildungen 10 bis 13 zeigen immerhin eine Möglichkeit:

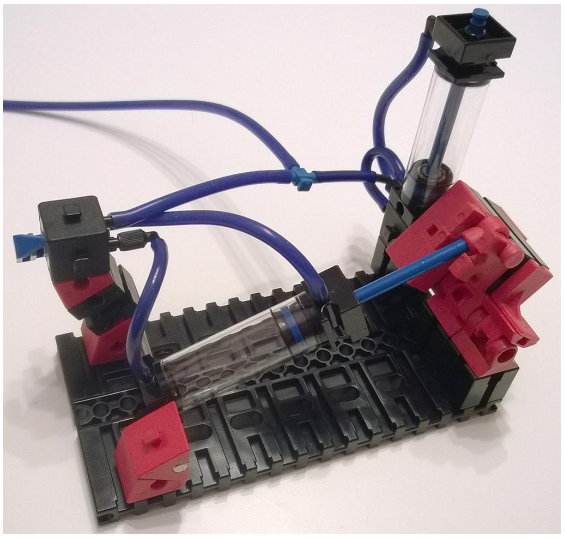


Abb. 10: Pneumatikzylinder steuert Handventil

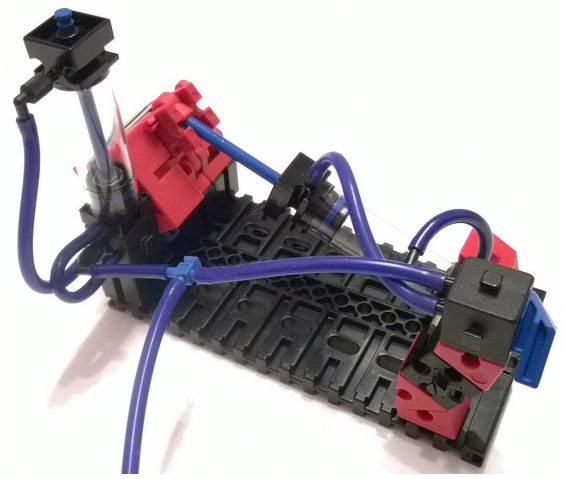


Abb. 11: Rückansicht

Die Mechanik muss für die auftretenden Kräfte robust genug ausgeführt werden, weil das Pneumatik-Ventil recht schwergängig ist. Deshalb wird der schwenkbare Teil von zwei Gelenken gehalten. Das äußere sitzt auf einem BS15, das innere auf einem BS15 mit zwei Zapfen. Der BS30 darauf trägt zwei Bausteine 15 · 30 · 5 mit drei Nuten, per Federnocken mit dem BS30 verbunden. Am unteren Ende sitzen innen zwei BS5, die den Griff des Ventils eng genug umschließen.

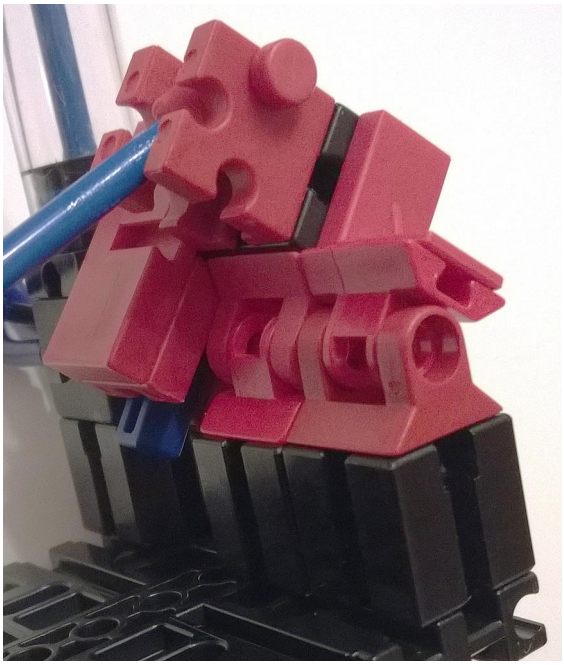


Abb. 12: Detailansicht

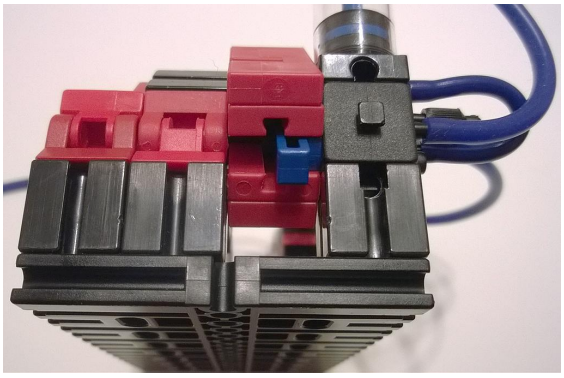


Abb. 13: Ansicht von unten

Durch den langen Weg, den der Zylinder verfahren muss, um das Ventil ganz durch zu schalten, sind die Schaltzeiten leider auch nicht berauschend (zum Glück ist das für Funktionsmodelle meist nicht so schlimm). Der größte Nachteil aber ist die Kombination aus langem Verfahrweg und großer Kraft, die notwendig ist, das recht schwergängige Handventil zu schalten. Das Ventil besteht ja nur aus zwei ineinander gesteckten Plastikteilen, und damit es luftdicht ist, müssen diese sehr stramm sitzen. Ein ft-Zylinder mit eingebauter Rückstellfeder reicht leider keinesfalls aus, um das Ventil zurück zu stellen.

Das aber bedeutet, dass wir für das Umschalten eines Handventils beide Eingänge des steuernden Zylinders benötigen. Wir kommen also genau nicht mit einer einzigen Steuerleitung aus, sondern brauchen immer Druckluft auf einem von zwei Anschlüssen anstatt nur Druck oder keinen Druck an einem einzigen Anschluss. Wir müssen uns also etwas einfallen lassen, wenn wir pneumatisch angesteuerte Pneumatik-Ventile verwenden möchten.

3/2-Wegeventile im Eigenbau

Fangen wir wieder klein an: Das Mindeste, was wir zum wahlweisen Beaufschlagen einer Leitung mit Druckluft und die Entlüftung derselben benötigen, ist ein 3/2-Wege-Ventil. In einer Stellung muss der Ausgang mit der Abluft verbunden und die Druckluftzufuhr dicht abgeschlossen sein, in der anderen Stellung muss der Ausgang mit der Druckluft und nicht mit der Abluft verbunden sein. Wir können also das 3/2-Wege-Ventil (vgl. das linke Ventil in Abb. 8) auch durch die Zerlegung in zwei getrennte 2/2-Wege-Ventile darstellen (Abb. 14).

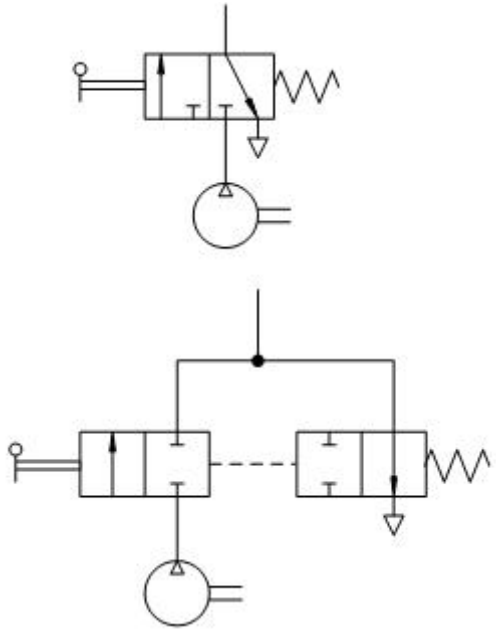


Abb. 14: Zerlegung eines 3/2-Wegeventils in zwei 2/2-Wegeventile

Vergewissert euch, dass die obere Darstellung dasselbe Resultat bringt wie die untere. Die gestrichelte Linie zwischen den beiden Ventilen steht für eine mechanische Kopplung: Wenn der Betätigungshebel das linke Ventil umschaltet, soll gleichzeitig auch das rechte umschalten, und wenn die Rückstellfeder das rechte zurückstellt, soll das ebenfalls aufs linke wirken.

Die beiden 2/2-Wegeventile sind die einfachsten möglichen Ventile überhaupt: Da besteht entweder Durchgang zwischen zwei Anschlüssen, oder der Durchgang wird verschlossen (beide Anschlüsse enden damit „dicht“).

Das Beste ist nun: Ein solches 2/2-Wegeventil existiert auch unter den aktuellen Pneumatikteilen! Bevor ihr nun aber vergeblich eure Kästen und die Stücklisten danach durchsucht: Wir können dafür einfach einen simplen Pneumatikschlauch verwenden, den wir nach Bedarf zu klemmen. Allerdings ist es gar nicht so einfach, den aktuellen ft-Schlauch so dicht abzuklemmen, dass auch die ca. 1 bar des ft-Kompressors nicht mehr durchkommen. Anstatt den Schlauch also – mit zwar wenig „Schaltweg“, aber viel Kraftaufwand – abzu*klemmen*, verwenden wir das Ab*knicken* des Schlauches – mit wenig Kraft auf Kosten eines etwas längeren Schaltweges. Einen Bauvorschlag zeigt Abb. 15.

Das Standard-Pneumatik-Ventil steuert den schräg stehenden Pneumatikzylinder nur „einfach“ an, also über nur einen einzigen Schlauch. Von den drei Ventilanschlüssen sind nur die Druckluftzuleitung und ein Ausgang verwendet. Der zweite Ausgang ist mit einem P-Stopfen verschlossen. Wir haben es also nur mit einer einzigen Steuerleitung zu tun.

Der Zylinder wiederum ist ein „einfach wirkender“, also einer mit eingebauter Rückstellfeder. Der dreht das Gebilde aus drei mittels Verbinder 45 und (auf der

unteren Seite) zwei Federnocken stabilisierten BS7,5. Dieses wiederum ist drehbar auf einer *Rastaufnahmeachse 22,5* (130593) in einem BS30 mit Bohrung gelagert. Auf die Rückseite der Achse kommt ein Rastkegelzahnrad oder eine Rastkupplung zur Fixierung.

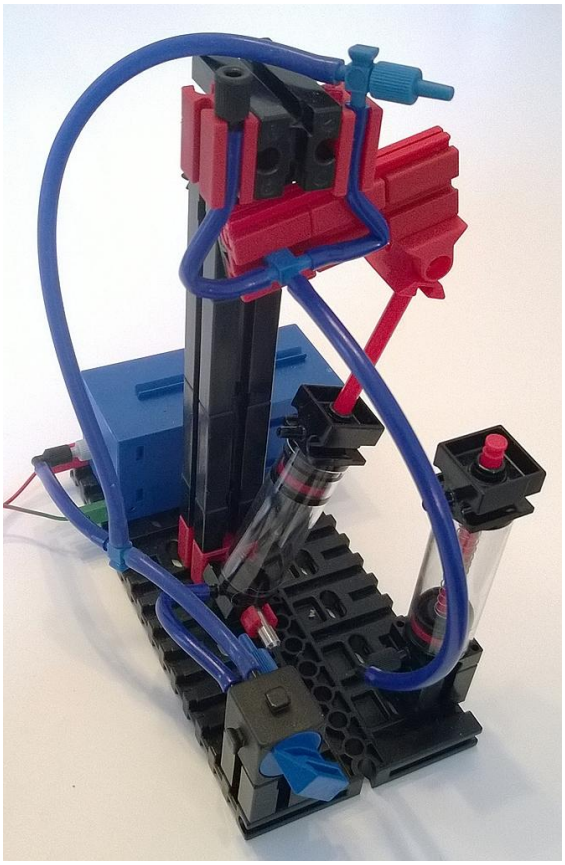


Abb. 15: 3/2-Wege-Selbstbauventil

Jetzt kommt der Trick: Auf der Drehachse fluchtend sitzt ein Pneumatik-T-Stück. Von ihm gehen seitlich zwei 5 cm lange Schlauchstückchen ab, die beide oben in je einem BS7,5 gehalten werden. Ein Schlauch endet in einem schwarzen *Pneumatik-Anschluss* (36200) und ist somit frei mit der Abluft verbunden. Der andere wird von einem weiteren Pneumatik-T-Stück im BS7,5 fixiert und bekommt Druckluft vom Kompressor. Ein Anschluss dieses T-Stücks ist wieder mit einem blauen P-Stopfen verschlossen.

Die beiden kurzen Schlauchstücke sind mit 5 cm Länge gerade so bemessen, dass in den beiden Endlagen des Steuerungszylinders immer einer abgeknickt und einer durchlässig ist. Je nachdem, in welche Endlagenstellung diese Mechanik gebracht wird, wird also mal der eine, mal der andere Schlauch abgeknickt. Der jeweils andere ist hinreichend offen, um Druckluft durch zu lassen.

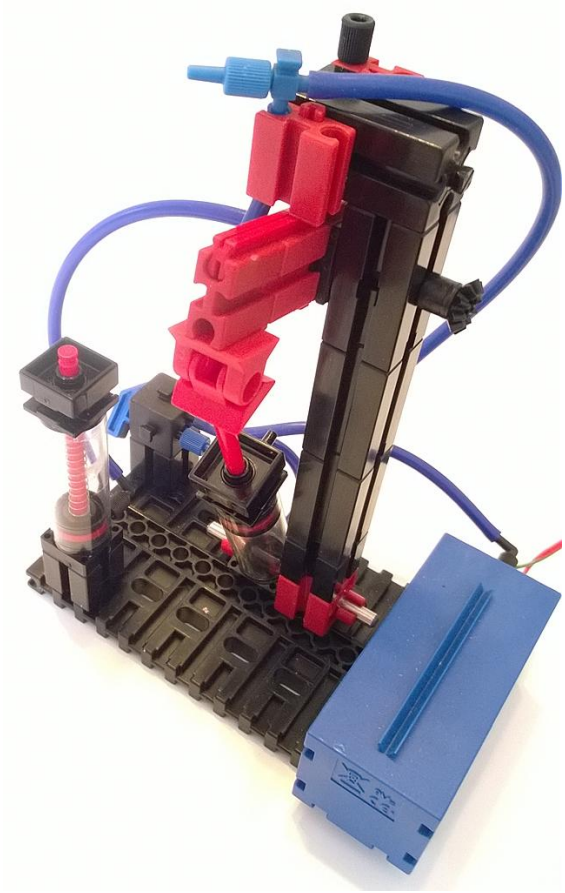


Abb. 16: 3/2-Wege-Selbstbauventil

Der zentrale Anschluss des drehbar gelagerten T-Stücks ist der Ausgang des Ventils. Dieser wird also je nach Stellung unseres Selbstbau-Ventils entweder mit Druckluft verbunden und der Abluftausgang durch seinen geknickten Schlauch verschlossen, oder er wird mit der Abluft verbunden und die Druckluftzufuhr ist verschlossen. Abb. 17 und 18 zeigen die beiden Ventilstellungen:

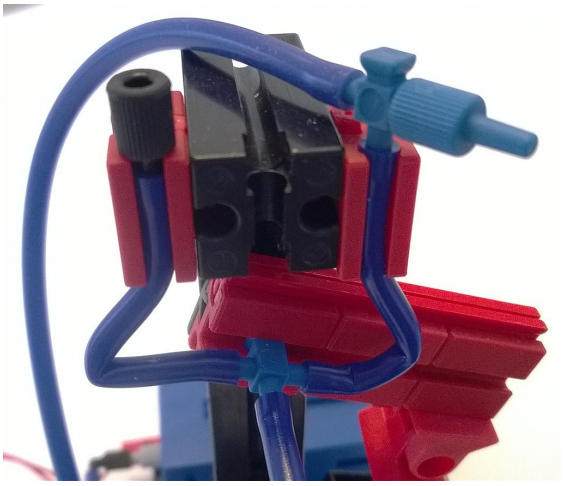


Abb. 17: Ventil in Ruhestellung

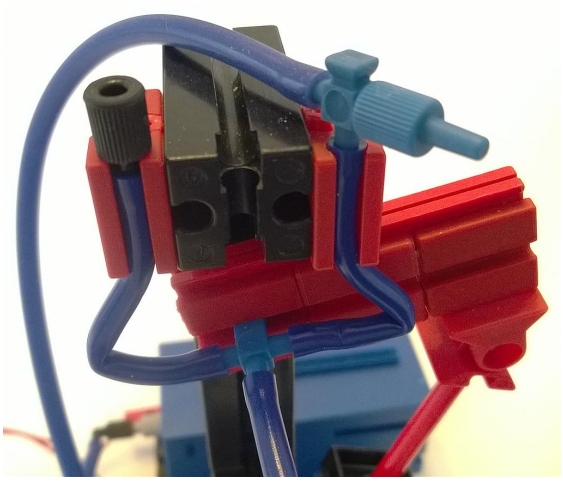


Abb. 18: Ventil in Arbeitsstellung

Und es funktioniert! Probiert den Aufbau aus und gebt dem Steuerzylinder durch das ft-Pneumatikventil abwechselnd Druckluft und Abluft. Der als Signalanzeige verbaute zweite Zylinder (wiederum ein einfach wirkender mit Rückstellfeder) fährt damit aus, wenn der Ventilausgang Druckluft führt. In der anderen Ventilstellung ist er – sonst würde der Zylinder nie mehr einfahren können – korrekt mit der Abluft verbunden; die im Zylinder enthaltene Druckluft kann abströmen und der Signalzylinder fährt wieder ein.

Identität und Negation

Damit haben wir schon unseren ersten pneumatischen Logikbaustein geschaffen. Je nachdem nämlich, wie wir die beiden oberen Anschlüsse beschalten, führt das Ventil eine bestimmte logische Operation aus:

- Beschalten wir die Anschlüsse so wie in den Abbildungen gezeigt, führt der Ventilausgang genau dann Druckluft, wenn der Eingang des Steuerzylinders *nicht* mit Druckluft beaufschlagt wurde. Ihr seht das deutlich daran, dass Steuer- und Signalzylinder immer andersherum stehen: Ist der Steuerzylinder eingefahren, das Ventil also in Ruhestellung, bekommt der Signalzylinder Druckluft und fährt aus, und umgekehrt. Wir haben es hier also mit einer logischen „Negation“, einem „Nicht-Glied“ zu tun.
- Vertauschen wir die beiden oberen Anschlüsse des Selbstbau-Ventils, setzen also den schwarzen P-Anschluss und das blaue T-Stück auf das jeweils andere Schlauchstückchen, drehen sich die Verhältnisse um: In der Ruhestellung des Ventils, wenn also am Steuerzylinder keine Druckluft anliegt, wird auch der Signalzylinder keine bekommen. Beide Zylinder fahren damit also gleichzeitig aus bzw. ein. Diese logische Operation nennt man „Identität“, weil das Eingangssignal unverändert auch am Ausgang herauskommt. Das ist durchaus nicht unnütz, denn wenn der Druck zum Ansteuern des Ventil-Zylinders zwar für dessen ordnungsgemäße Funktion ausreicht, aber kleiner ist als der Druck, der am Eingang des damit gesteuerten 3/2-Wege-Ventils anliegt, haben wir damit eine Möglichkeit, ein Signal – bei unserer Pneumatik also einen Druckpegel – zu *verstärken.*

Tipps zum Aufbau

Bei korrekter Justierung funktioniert unser Selbstbau-Ventil tadellos. Allerdings kann es nach einigen Stunden oder Tagen Stillstand schon vorkommen, dass die abge-

knickten Schlauchstücke sich an ihren Zustand „gewöhnt“ haben und nicht mehr sauber schalten. Immerhin ist das ja nur ein aus der Not geborener Behelf, weil die originalen mechanisch gut ansteuerbaren Ventile von fischertechnik – sehr zum Bedauern aller, die sie kennen – nicht mehr hergestellt werden und nie ein adäquater Ersatz kam. Aber: Auch nach längeren Stillstandszeiten kann man die Ventile doch immer wieder mit etwas „Einspielen“ zur korrekten Funktion bewegen. Vor allem aber ist das selbstgebaute 3/2-Wegeventil ja erst der Anfang!

Das pneumatische Relais

Unser nächstes Zwischenziel lautet: Wir wollen einen Pneumatik-Zylinder dadurch ein- und ausfahren lassen können, dass wir einen einzigen (!) Anschluss mit Druckluft beaufschlagen (der Zylinder soll aktiv ausfahren) oder mit der Abluft verbinden (der Zylinder soll aktiv einfahren).

Dazu verwenden wir *zwei* der selbstgemachten 3/2-Wegeventile und steuern diese durch *einen einzigen* Pneumatikzylinder mit Rückholfeder an. Unser Aufbauvorschlag zeigt eine weitere Möglichkeit, das Abknicken von Schläuchen geschickt zum Schalten zu verwenden:

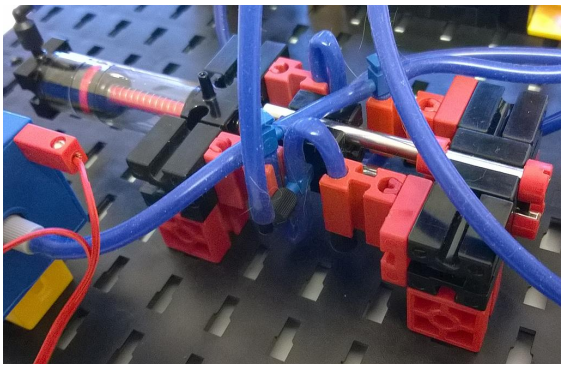


Abb. 19: Eigenbau eines pneumatisch betätigten 4/2-Wegeventils

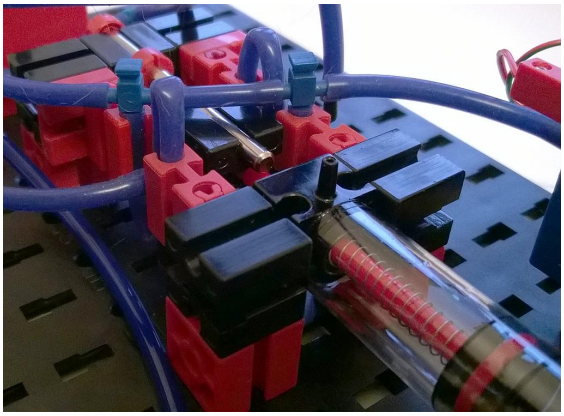


Abb. 20: Eigenbau eines pneumatisch betätigten 4/2-Wegeventils

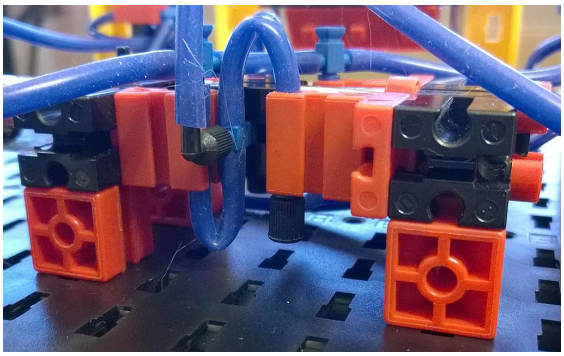


Abb. 21: Seitenansicht

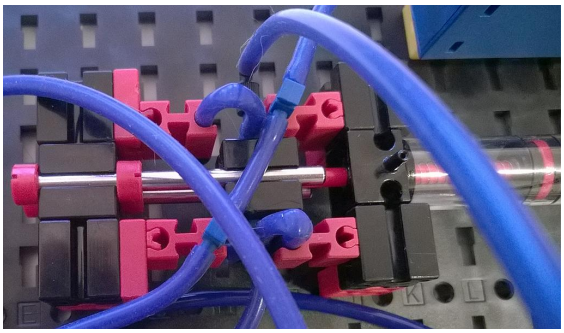


Abb. 22: Draufsicht

Ein Pneumatik-Zylinder (hier wieder ein einfach wirkender) verschiebt einen BS15, der längs von zwei Achsen geführt wird. Am Anfang und Ende des Verfahrweges werden wieder BS7,5 in geeignetem Abstand angebracht, in denen die (jetzt vier) 5 cm langen Schlauchstücke gehalten werden. Auf zwei der BS7,5 stecken links und rechts je ein Pneumatik-T-Stück.

Vergleichen wir diese Konstruktion mit dem Schaltbild eines elektromechanischen Relais [2]:

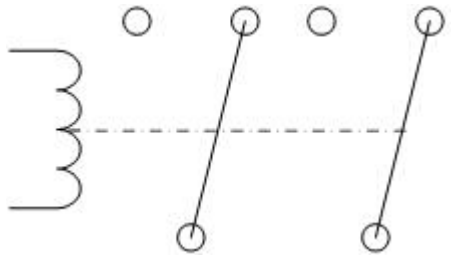


Abb. 23: Elektromechanisches Relais

Genau wie bei einem elektromechanischen Relais besitzt unser pneumatischer Aufbau:

- eine Steuerleitung: Das ist der einzige benutzte Druckluftanschluss des Zylinders. Anstelle des zweiten Anschlusses des Elektromagneten in einem Relais gibt es in der Pneumatik ja das Verbinden dieses Anschlusses mit der freien Abluft.
- zwei getrennte „Umschaltkontakte“: Die beiden Zentralkontakte sind die beiden auf dem beweglichen Schlitten befestigten T-Stücke; die Arbeits- bzw. Ruheanschlüsse sind die fest gehaltenen vier Schlauchenden.

Damit gelingt es, mit nur einem Eingangssignal am Zylinder einen weiteren Zylinder komplett vor und zurück zu steuern. Wir schließen die vier feststehenden Enden der abknickbaren kurzen Schlauchstückchen so an, dass immer genau einer der zentralen Ausgänge der beiden T-Stücke mit der Druckluft und der jeweils andere mit Abluft verbunden ist. In Abb. 22 wird die Druckluft an beide feststehenden T-Stücke durch das schräg über das Ventil verlaufende kurze Schlauchstückchen zugeführt. Die anderen, in diesem Bild unten liegenden, freien Enden werden wieder mit schwarzen Pneumatik-Anschlüssen in den BS7,5 festgeklemmt und sind Abluftausgänge. An die beiden zentralen (beweglich gelagerten) Ausgänge der T-Stücke am verschiebbaren BS15 wird schließlich der zu steuernde Zylinder angeschlossen.

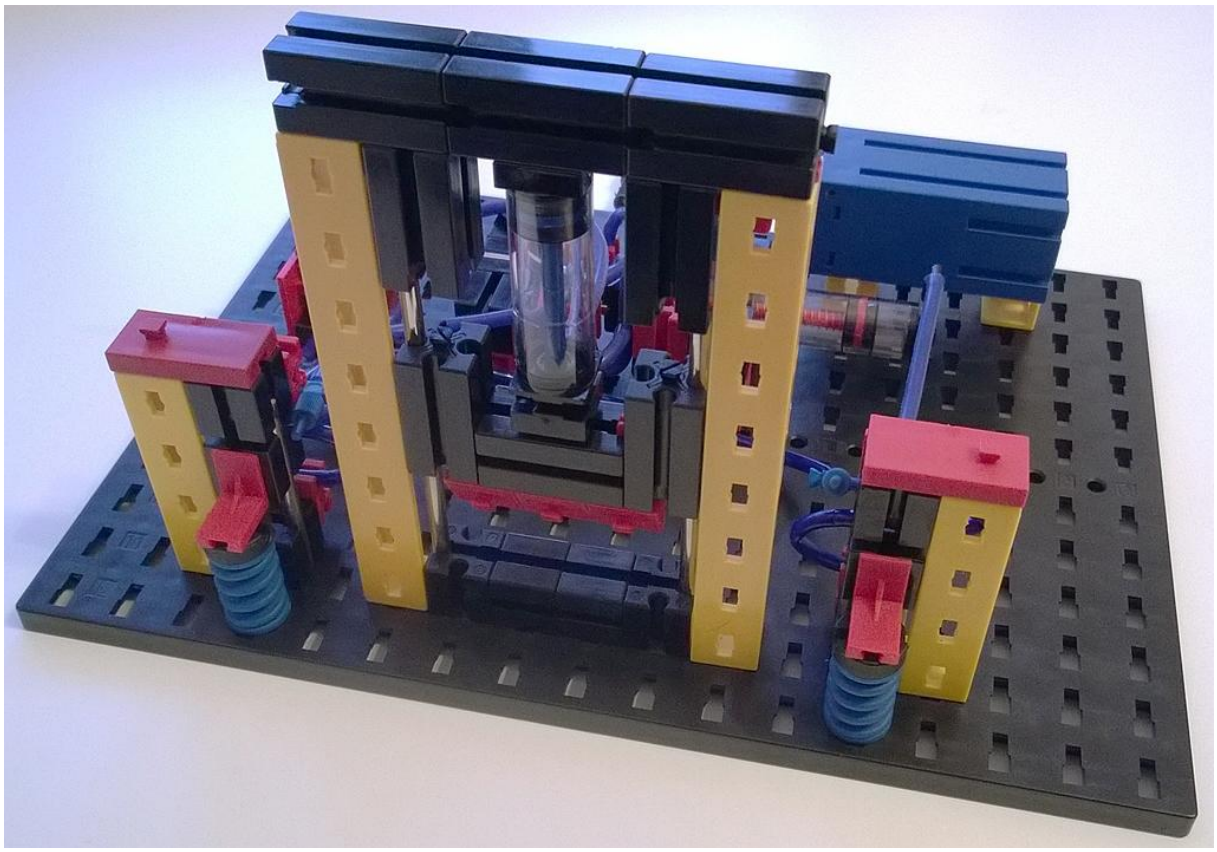


Abb. 24: Pneumatisch gesteuerte Presse

Eine pneumatisch gesteuerte Presse

Das wollen wir zu einem vollständigen Modell ausbauen und dabei gleich noch ein paar Pneumatik-Kniffe kennenlernen. Abb. 24 zeigt eine pneumatische Presse mit folgenden Eigenschaften, die wir nach und nach entwickeln werden:

1. Ein Pneumatik-Zylinder soll einen Press-Kopf zum Pressen eines Teiles herunterdrücken und danach wieder hochziehen.
2. Aus Sicherheitsgründen benötigen wir eine „Zwei-Hand-Steuerung“: Die Presse darf sich nur dann absenken, wenn zwei links und rechts neben der Presse angebrachte Druckknöpfe gleichzeitig niedergedrückt werden. Das verhindert, dass etwa eine Hand des Bedienpersonals in die Presse gelangt.
3. Der Pressvorgang soll *langsam* ablaufen. Der Press-Zylinder muss also den Presskopf nicht nach unten schlagen, sondern langsam und sanft aufsetzen. Natürlich muss er aber trotzdem mit vollem Druck pressen.

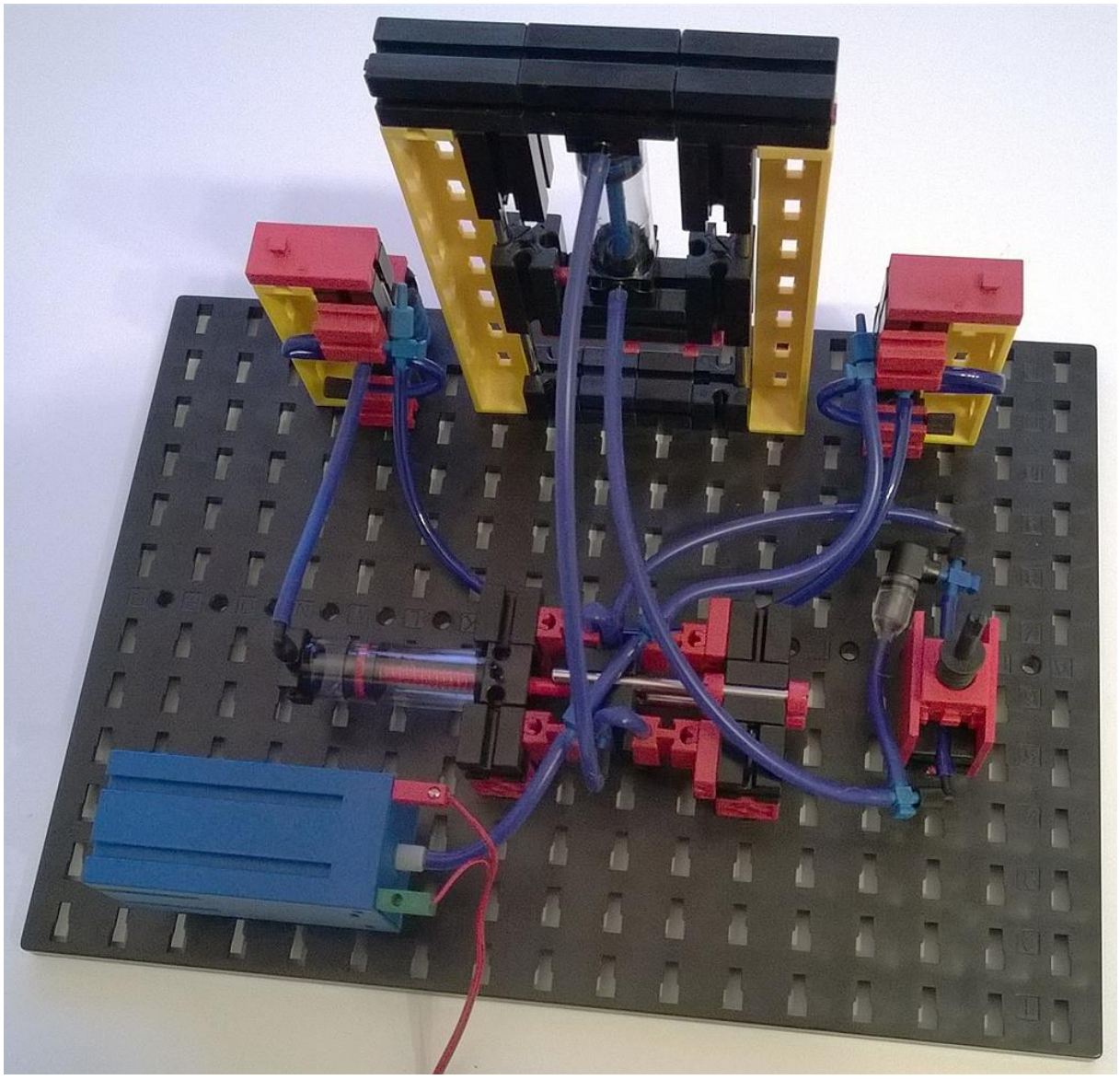


Abb. 25: Überblick über die pneumatisch gesteuerte Presse

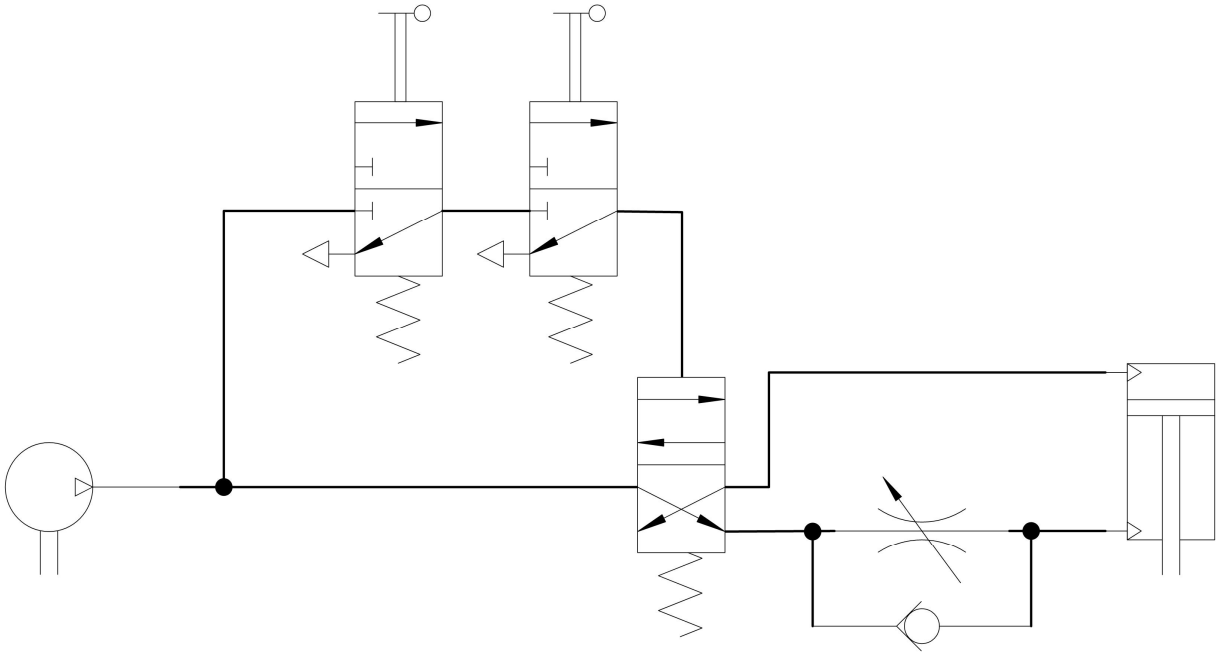


Abb. 26: Schaltbild der Presse

4. Sobald auch nur einer der beiden Druckknöpfe vom Personal losgelassen wird, soll sich die Presse *sofort und schnell* heben. Also: Langsam absenken, aber schnell anheben!

Abb. 25 zeigt das Modell von der Rückseite. Die wesentlichen Bestandteile sind:

- die eigentliche Presse,
- unser selbstgebautes 4/2-Wege-Ventil,
- zwei selbstgebaute und als Taster realisierte 3/2-Wege-Ventile,
- eine Drossel/Rückschlagventil-Kombination, auf die wir später noch ausführlich zu sprechen kommen sowie
- ein Kompressor als Druckluftquelle.

Am Schaltbild (Abb. 26) gehen wir die einzelnen Baugruppen Schritt für Schritt durch:

Zum Bau der Presse

Der grundsätzliche Aufbau der Presse ist auf Abb. 25 gut erkennbar mit folgenden Details:

- Die Statikträger sind oben per Federnocken mit den waagerecht liegenden BS30 verbunden. Sie bleiben ein kleines bisschen nach außen verschoben, damit der auf und ab gleitende Presskopf sich nicht an ihnen reibt.
- Die langen Metallachsen, an denen der Presskopf gleitet, sind gegen Herausrutschen aus ihrer unteren Führung dadurch gesichert, dass in den beiden oberen senkrechten BS30 zwischen Achse und Statikträger-Abschluss noch je ein S-Riegel steckt. Der verhindert, dass die Achsen beim Hochziehen des Presskopfes so hoch mitgezogen werden können, dass sie unten aus den führenden BS15 heraus kommen.
- Die drei Winkelsteine 60° sind mit einem Verbinder 45 mit den Grundbausteinen verbunden.

Zweihandsteuerung

Damit wie verlangt jemand mit beiden Händen je einen „Taster“ niederdrücken muss, bauen wir also zwei gleiche 3/2-Wege-Ventile und schalten sie – gerade wie elektrische Taster! – in Serie, damit Druckluft am letzten Ausgang nur dann anliegt, wenn beide Taster gleichzeitig niedergedrückt werden. Ist auch nur einer der Taster losgelassen, muss der letzte Ausgang mit Abluft verbunden sein. Dazu

verwenden wir wieder einen auf Achsen verschiebbaren BS15, auf dem ein Pneumatik-T-Stück sitzt. Am BS15 bringen wir einen roten Winkel als Betätigungsknopf an, und diesen lassen wir von einer ft-Druckfeder automatisch nach oben zurückstellen (vgl. Abb. 24).

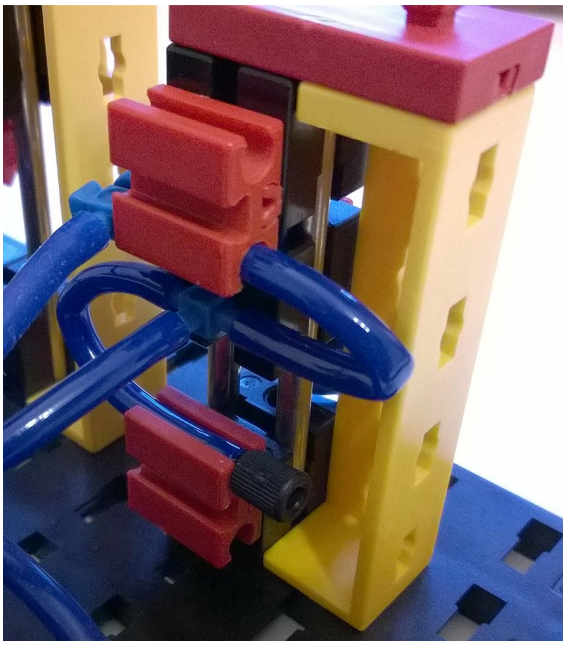


Abb. 27: Taster 1

Wieder verwenden wir 5 cm lange Schlauchstücke als abklemmbares Element, lagern sie in je einem BS7,5 und fixieren sie darin mit einem schwarzen Anschlussstück in Richtung Abluft bzw. mit einem T-Stück (mit Stopfen auf einem Anschluss) für die Druckluftzufuhr. Der mittlere Anschluss des bewegten T-Stücks ist der Ausgang des Ventils. Damit das Abklemmen richtig funktioniert, müssen die BS7,5 nahe genug am beweglichen BS15 angebracht werden.

Im Ruhezustand des Ventils, also in seiner durch die Feder erzwungenen oberen Stellung, ist er mit der Abluft verbunden. Im betätigten Zustand, also wenn der rote Winkelbaustein auf der Vorderseite niedergedrückt wird, kommt der verschiebbare BS15 mit seinem T-Stück darauf unten zu liegen, wodurch der zentrale Ausgang des T-Stücks mit Druckluft anstatt Abluft verbunden wird.

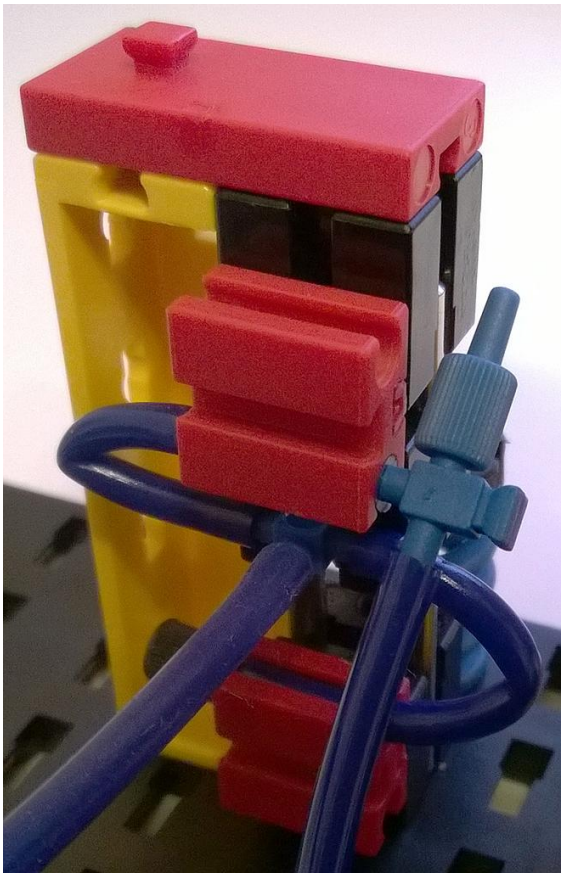


Abb. 28: Taster 2

Der zentrale Ausgang des T-Stücks des ersten Ventils (Abb. 27) führt nun zum Eingang des zweiten Taster-Ventils auf der anderen Seite der Presse (Abb. 28). Sein Druckluftzugang – in Abb. 28 das fix angebrachte T-Stück mit Stopfen rechts oben – ist also mit dem Ausgang des ersten Ventils verbunden. Erst der Ausgang dieses zweiten Tasters geht auf den Steuerungs-Zylinder unseres selbstgebauten 4/2-Wege-Ventils mit Rückstellung, das in der Mitte der Bauplatte 500 Platz findet.

Ansteuerung des Pressenzylinders

Vom auf diese Weise angesteuerten 4/2-Wege-Ventil gehen also die Ausgänge – die zentralen Anschlüsse der beiden beweglichen T-Stücke – an den Pressenzylinder. Bis hier hin könnt ihr das Modell

schon aufbauen und testen. Fast alle unsere Anforderungen haben wir damit schon erfüllt: Nur wenn beide Taster gleichzeitig betätigt werden, senkt sich die Presse ab. Lässt man einen oder beide Taster los, wird die Presse unverzüglich wieder angehoben.

Langsames Verfahren eines Zylinders

Wenn wir den Pressen-Zylinder direkt mit den Ausgängen unseres 4/2-Wege-Ventils verbinden, fährt die Presse allerdings noch sehr schnell auf und ab. Wir wollten aber, dass sie sich ganz langsam absenkt und sanft aufsetzt, ohne zu „schlagen“.

Dafür müssen wir die Stärke des Luftstroms drosseln, also weniger Luftmenge pro Zeit durch die Schläuche strömen lassen. Dafür gibt es ein Pneumatik-Bauteil unter dem treffenden Namen *Drossel*:

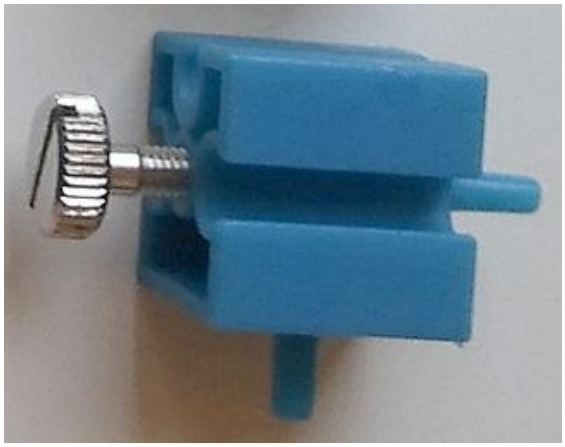


Abb. 29: Die 36077 Pneumatik-Drossel von 1981

Diese Bauteile werden allerdings leider, wie die Festo-Ventile, nicht mehr hergestellt. Allerdings ist deren Funktion wirklich leicht nachzubilden: Wir brauchen ja nur ein Stück Schlauch mehr oder weniger stark zusammen zu drücken. Das gelingt leicht mit einem Aufbau wie in Abb. 30.

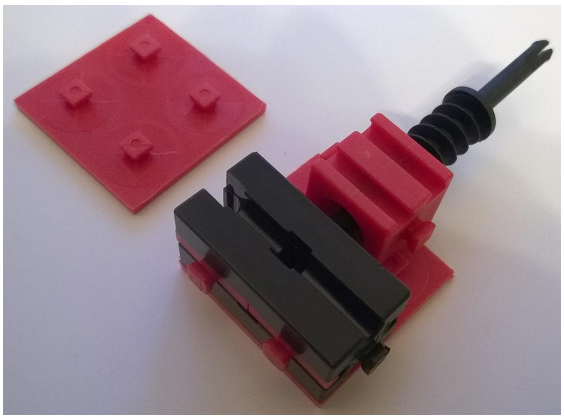


Abb. 30: Selbstgebaute Pneumatik-Drossel

Ein Baustein 30 mit Bohrung wird mittels zweier Verkleidungsplatten und einer *Schneckenmutter 15x15x15 m1* (35973) so mit einer *Rastschnecke 57,5 m1* (35977) verbunden, dass die Rastschnecke in die Bohrung des BS30 hinein geschraubt werden kann.

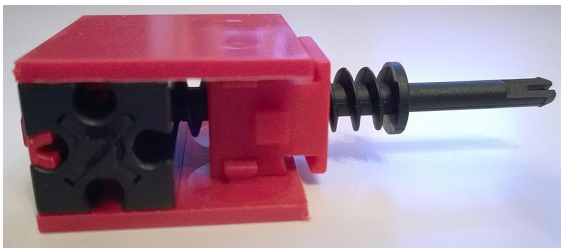


Abb. 31: Selbstgebaute Pneumatik-Drossel

Durch die Längsnut des Bausteins führen wir einen Pneumatikschlauch, der dann durch Verdrehen der Schnecke im Querschnitt verengt werden kann. Für unser Modell trägt der BS30 noch zwei Federnocken zur Befestigung auf der Bauplatte.

Zunächst könnten wir also eine Schaltung wie folgt aufbauen (zum Ausprobieren könnt ihr als 4/2-Wege-Ventil auch das aktuelle, fertige ft-Pneumatikventil mit Handsteuerung verwenden – im Moment geht es ja nur um die Drossel):

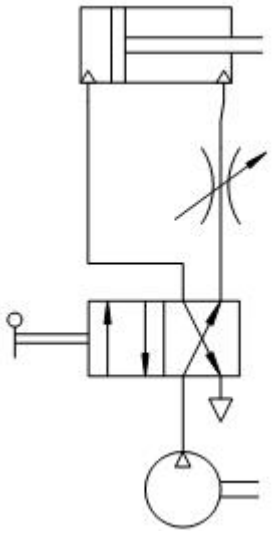


Abb. 32: Einfache Verwendung einer Drossel

Die in Abb. 31 in der Schlauchführung zu einem Zylinderanschluss eingeführte Drossel bewirkt also, dass durch diesen Schlauch die Luft nicht mehr so schnell strömen kann, wie es ohne Drossel der Fall war. Damit der Effekt tatsächlich merkbar wird, müssen wir die Drosseln übrigens fast ganz zu drehen und recht fein justieren.

Verlangsamung in nur einer Bewegungsrichtung

Auf diese Weise haben wir ein weiteres Teilziel erreicht: Die Zylinderbewegungen gehen nun langsamer. Allerdings wollten wir ja, dass nur das Absenken der Presse langsam geht, während das Anheben so schnell wie möglich geschehen soll. Wir müssen also dafür sorgen, dass die in Abb. 32 gezeigte Drossel nur dann wirkt, wenn der Zylinder gerade ausfährt. Beim Einfahren soll die Luft hingegen ungehindert strömen dürfen.

Das gelingt durch ein pneumatisches Bauteil, das in der Elektronik der Diode [3] entspricht: Ein *Rückschlagventil* lässt Luft nur in eine Richtung durch strömen, während es den Luftstrom in die andere Richtung vollständig verhindert. Ein solches Bauelement gibt es noch recht aktuell von fischertechnik. Es handelt sich um ein (manchen vielleicht seltsam vorkommendes) Bauteil aus dem Selbstbau-Kompressor mit dem S-Motor, das 32061 *Pneumatik-Rückschlagventil:*

Abb. 33: fischertechnik-Rückschlagventil

Seine Funktionsweise wird sofort klar, wenn wir uns sein Schaltbild anschauen:

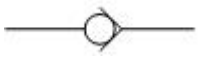


Abb. 34: Schaltzeichen des Rückschlagventils

Da sitzt tatsächlich eine Kugel in einer Aufnahme. Die wird (das zeigt das Schaltbild nicht) von einer Druckfeder so abdichtend in eine Spitze gedrückt, dass keine Luft durchströmen kann. Im fischertechnik-Rückschlagventil ist diese Feder deutlich zu sehen.

In Abb. 34 kann Luft nicht von links nach rechts strömen, weil die Kugel den Ausgang verschließt. Der anliegende Luftdruck sorgt sogar dafür, dass die Kugel nur noch fester gedrückt wird. In die andere Richtung wird die Kugel aber durch die Druckluft gegen die Federkraft weg von der Spitze gedrückt, und die Luft kann strömen.

Dieses Bauelement schalten wir nun einfach parallel zur Drossel und erreichen damit den gewünschten Effekt. In eine Strömungsrichtung sperrt das Rückschlagventil, und die Luft muss durch die enge Drossel hindurch. In die andere Richtung kann die Luft praktisch frei an der Drossel vorbei durch das Rückschlagventil strömen. Das ergibt in der Gesamtschaltung des Modells den gewünschten Effekt: Beim Pressen senkt sich der Press-Kopf je nach Einstellung der Drossel langsam ab, während die Aufwärtsbewegung ungehindert schnell abläuft.

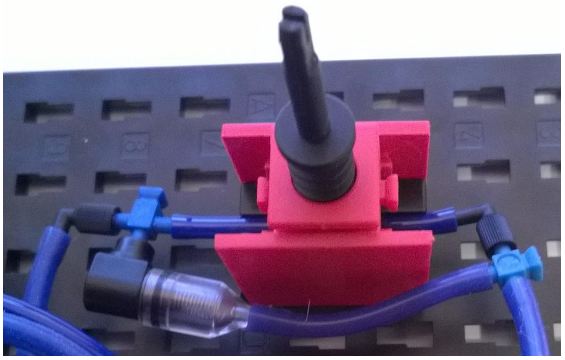


Abb. 35: Parallelschaltung von Drossel und Rückschlagventil

Richtiges Drosseln – Abluftdrosseln

Wenn man Zylinder majestätisch langsam ein- und ausfahren lassen will, ist es wichtig, auf der richtigen Seite zu drosseln: Gedrosselt wird die Abluft, nicht die Druckluftzufuhr! Dadurch steht der Pneumatikzylinder nämlich immer von beiden Seiten unter Druck. Er wird zwischen der zuströmenden Druckluft auf der einen, und der durch die Drossel nur langsam abströmenden Luft auf der anderen Seite fest eingespannt. Während der gesamten Bewegung wird er also von beiden Seiten mit hoher Kraft gehalten *und ruckelt deshalb nicht.*

Würden wir die Zuluft drosseln und die Abluft ungehindert abströmen lassen, würde sich der Zylinder (wie immer) erst dann bewegen, wenn der Luftdruck hinreichend groß ist, um die Haftreibung des Zylinders und der zu bewegenden Teile zu überwinden. Dann würde es einen kurzen, kleinen Ruck geben, bis der Zylinder so weit bewegt wurde, dass der Druck nicht mehr ausreicht, die Bewegung aufrecht zu erhalten – die Zuluft kann ja nur langsam einströmen. Das ergäbe je nach Reibungs- und Lastverhältnissen eine unschön ruckelnde Bewegung.

Die Drosselung der Abluft aber ergibt eine wunderbar gleichmäßige, ruhige und je nach Drosseleinstellung extrem langsame Bewegung. Deshalb ist es wichtig, dass

a) die Drossel in diejenige Leitung zum Zylinder eingebaut wird, durch die in der langsam gewünschten Bewegungsrichtung die Abluft strömt und

b) dass bei schneller anderer Bewegungsrichtung das Rückschlagventil auch in der richtigen Richtung parallel zur Drossel geschaltet wird, nämlich so, dass nur die Abluft von a) gedrosselt wird, die zum schnellen Anheben der Presse notwendige Zuluft aber ungehindert durchströmen kann.

Drosseln und ggf. parallel geschaltete Rückschlagventile würden auch den pneumatischen Bagger- und Traktormodellen aus den einschlägigen fischertechnik-Kästen gut zu Gesicht stehen. Deren durch Zylinder bewegte Arme agieren nämlich unnatürlich schnell und schlagen eher, als gleichmäßig zu ziehen. Ein paar Drosseln an den richtigen Stellen würde die Sache erheblich vorbildgetreuer realisieren und wäre viel eleganter anzuschauen.

Quellennachweis

[1] Falk, Stefan: *Perlentauchen (5)*, ft:pedia 4/2013, S. 6-15.

[2] Falk, Stefan: *Motorsteuerungen (4)*, ft:pedia 4/2011, S. 6-20.

[3] Falk, Stefan: *Motorsteuerungen (3)*, ft:pedia 3/2011, S. 4-13.

Modell

Detail Engineering R2D3 (1) – Gleitring-Lager

Andreas Gail

Im Rahmen des Baus des Robotermodells R2D3 wurde eine Reihe von ganz unterschiedlichen Detaillösungen erarbeitet, die durchaus Lösungsansätze bei diversen anderen Bauprojekten sein könnten. In einer kleinen Serie werden sie vorgestellt. Den Anfang macht ein Gleitring-Lager.

Aufgabenstellung

Wie in Abb. 4 (nächste Seite) dargestellt, sollte der violett abgebildete Teil des Roboters drehbar gelagert werden, gleichzeitig aber auf der darunter befindlichen Konstruktion aufliegen. Das sich dabei ergebende Problem bestand darin, dass durch minimale Kanten auf den Kreisringflächen die Drehbewegung immer wieder blockiert wurde. Es sollte ein Weg gefunden werden, um eine störungsfreie Drehbewegung reproduzierbar zu ermöglichen.

Lösung

Aus Kunststoffplatten (im Baumarkt erhältlich) wurde ein Kreisring mit 23,4 cm Außendurchmesser und 20,4 cm Innendurchmesser ausgefräst. Dieser Ring wird als *Gleitring* bezeichnet.

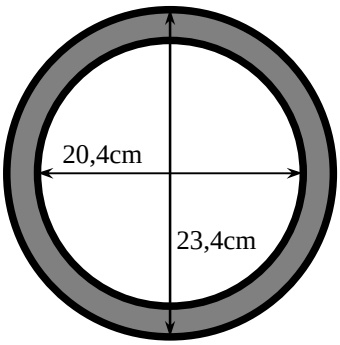


Abb. 1: Gleitring mit Bemaßung (Skizze nicht maßstäblich)

Bearbeitung

Zur Bearbeitung bietet sich z. B. eine Oberfräse mit Zirkelanschlag eines beliebigen Herstellers an (siehe Abb. 2). Bei der Einstellung des Abstands von der Zirkelnadel zum Fräser ist der Durchmesser des Fräsers selbst mit einzubeziehen. Zunächst wird der Außendurchmesser ausgefräst, dann der Innendurchmesser. So geht während der Bearbeitung der mittige Bezugspunkt nicht verloren.

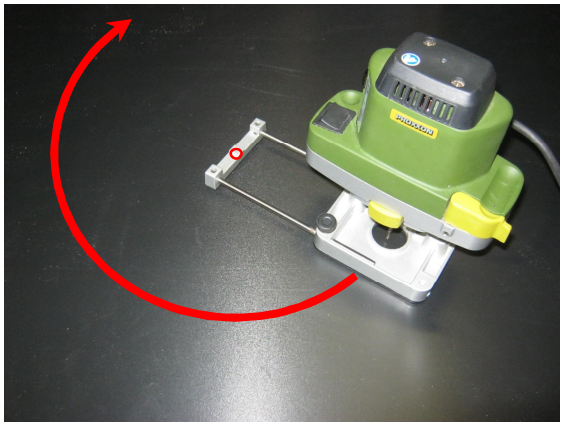


Abb. 2: Herstellung des Gleitrings per Oberfräse mit Zirkelanschlag

Auf ähnliche Weise wie zuvor der Gleitring kann auch eine runde Bodenplatte hergestellt werden. Die erforderlichen Löcher können je nach Härte des Materials ausgebohrt, per Lochzange ausgestanzt oder mithilfe eines Locheisens ausgeschlagen werden (Abb. 3).

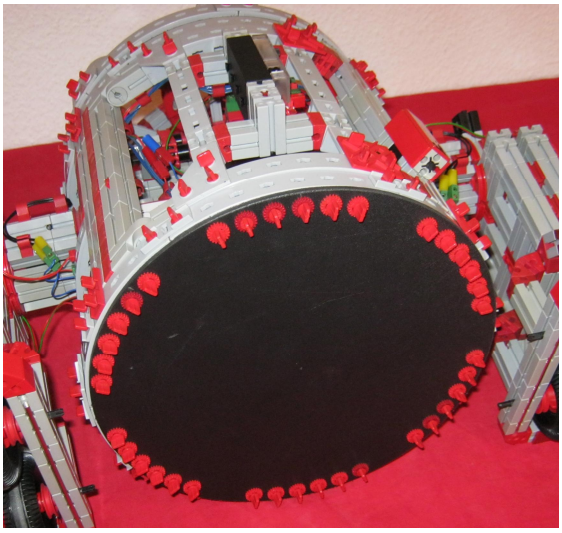


Abb. 3: kreisrunde Bodenplatte, Durchmesser 23,4 cm

Ausblick

In der nächsten Ausgabe der ft:pedia wird es darum gehen, wie mithilfe einer Ansteuerung von Leistungsmotoren über den Robo TX Controller die volle Beweglichkeit des Roboters erreicht wird.

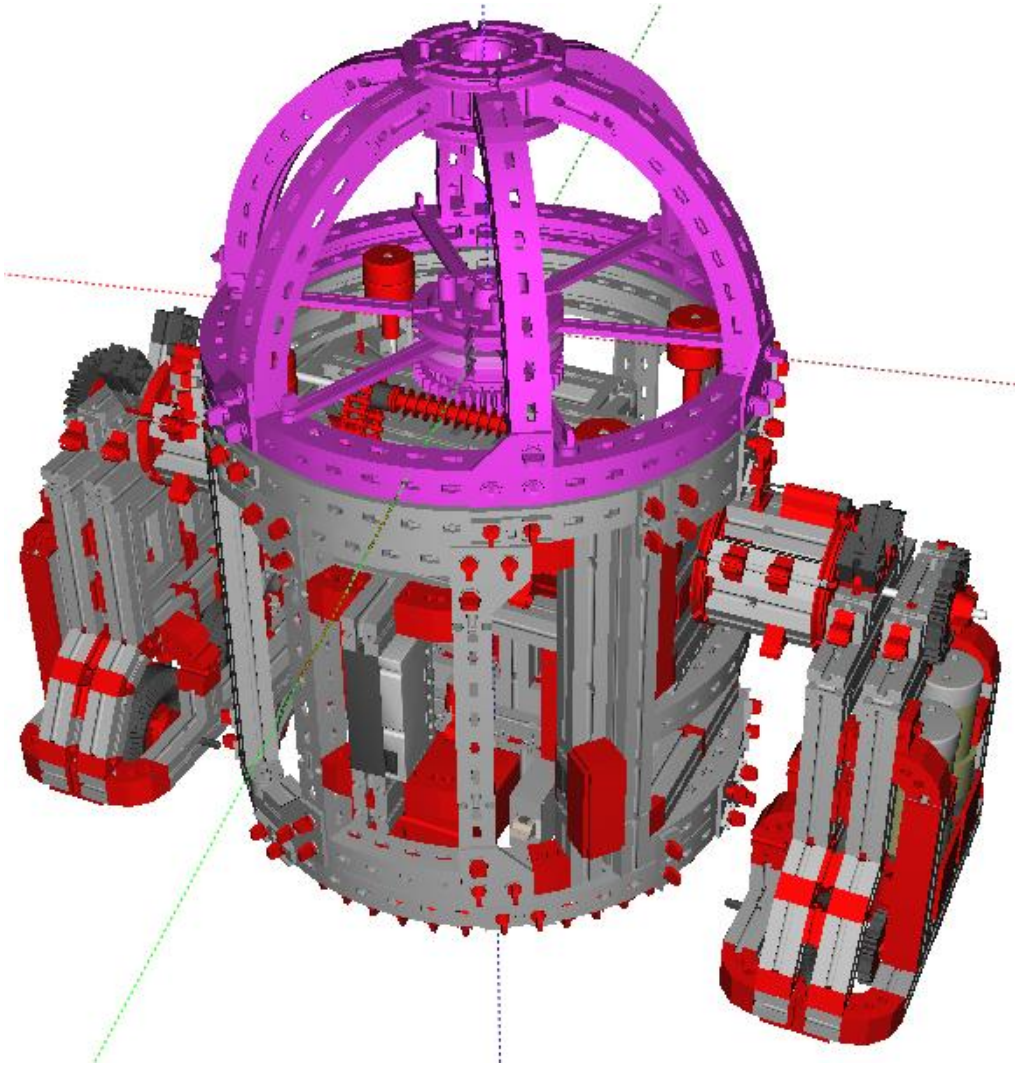


Abb. 4: Roboter mit drehbarer Halbkugel (violett dargestellt)

Kugelbahn des neuen Kastens Dynamic XL (Bild: fischertechnik)