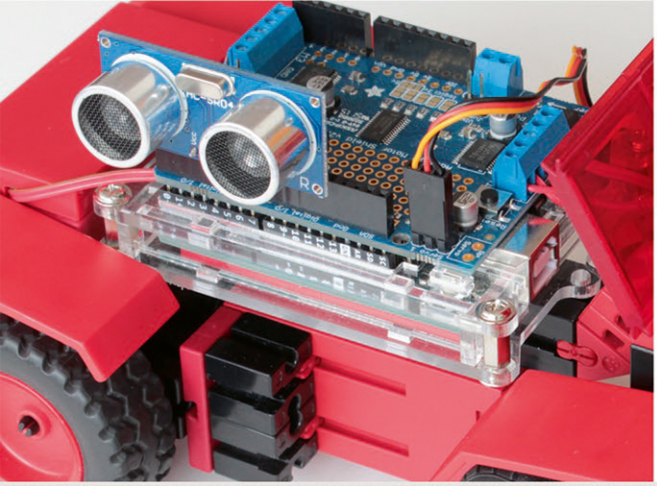
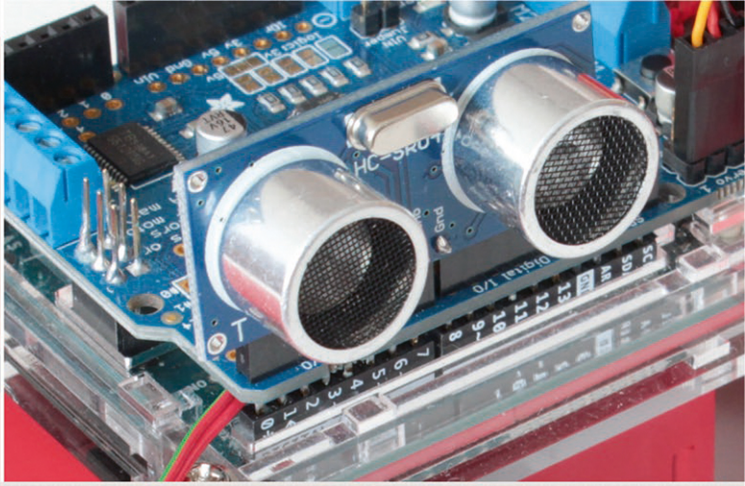




Dirk Fox
Thomas Püttmann

Bauen, erleben, begreifen: fischertechnik®-Roboter mit Arduino

Modelle, Steuerung, Programmierung

edition Make:

dpunkt.verlag

Dirk Fox ist Informatiker, Gründer und Geschäftsführer eines Beratungsunternehmens für IT-Sicherheit, Herausgeber einer Fachzeitschrift für Datenschutz und Datensicherheit, Vorstand eines großen IT-Netzwerks – und begeisterter »fischertechniker«. Er gibt die fischertechnik-Zeitschrift »ft:pedia« heraus und setzt sich für den Ausbau des Technikunterrichts an deutschen Schulen ein – mit fischertechnik.

Thomas Püttmann ist außerplanmäßiger Professor für Mathematik an der Ruhr-Universität Bochum. Zur Vermittlung von Themen aus den Bereichen Mathematik, Technik und Naturwissenschaften entwickelt er gezielt lehrreiche Modelle, wenn möglich aus fischertechnik. Als echter Mathematiker optimiert er seine Konstruktionen so lange, bis man keinen Stein mehr weglassen oder verschieben kann. Regelmäßig schreibt er Beiträge für die »ft:pedia«.

Dirk Fox und Thomas Püttmann sind auch die Autoren des erfolgreichen ersten fischertechnik-Buchs »Technikgeschichte mit fischertechnik«, das Ende 2015 im dpunkt.verlag erschienen ist.

Dirk Fox
Thomas Püttmann

Bauen, erleben, begreifen: fischertechnik®-Roboter mit Arduino

Modelle, Steuerung, Programmierung

dpunkt.verlag

Dirk Fox **dirk.fox@secorvo.de**
Thomas Püttmann **thomas.puettmann@rub.de**

Lektorat: Dr. Michael Barabas
Projektkoordinierung/Lektoratsassistenz: Anja Weimer
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Ulrich Borstelmann, *www.borstelmann.de*
Herstellung: Stefanie Weidner, Frank Heidt
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

ISBN:
Print 978-3-86490-426-4
PDF 978-3-96088-963-2
ePub 978-3-96088-964-9
mobi 978-3-96088-965-6

1. Auflage 2020

Wieblinger Weg 17
69123 Heidelberg

Hinweis:
Der Umwelt zuliebe verzichten wir auf die Einschweißfolie.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: hallo@dpunkt.de.

5 4 3 2 1 0

Zur Einführung xi

Eine kurze Geschichte der Roboter xi
fischertechnik-Roboter xvii
Arduino-Steuerung xix
Warum dieses Buch? xxi
Literatur xxii

1 Der Arduino 1

1.1 Geschichte 2
1.2 Was ist der Arduino? 4
1.3 Der Arduino im Vergleich 8
1.4 Der ftDuino 10
1.5 Anschlüsse 12
1.6 Anbau 15
1.7 Motor Shield 17
1.8 Literatur und Links 20

2 Die Programmierung des Arduino 21

2.1 Grundsätzliches 22
2.2 Bibliotheken 24
2.3 Serieller Monitor 25
Ausgabe 27
Eingabe 27
Debugging 28
2.4 I/O-Ports 29
Digitale Ports 29
Analoger Output 31
Analoge Ports 32
2.5 Interrupts 34
2.6 Timer 36
2.7 Serielle Protokolle 37
I²C-Protokoll 38
SPI-Protokoll 41
2.8 Mathematische Operationen 43

2.9 Adafruit Motor Shield 44
Gleichstrommotoren 45
Schrittmotoren 45
Servomotoren 46
2.10 Literatur und Links 48

3 Der Buggy 49

3.1 Die Geschichte des Buggys 50
3.2 Das Buggy-Basismodell 54
Mechanischer Aufbau 54
Akku, Controller und Motorsteuerung 59
3.3 Buggy-Steuerung 64
Tanzender Buggy 64
IR-Fernsteuerung 66
Bewegungssteuerung 74
Hinderniserkennung mit Bumpern 84
Hinderniserkennung mit Ultraschall 87
Spurfolger mit IR-Sensor 93
Objektfolger mit Kamera 100
Spurfolger mit Kamera 112
Tacho- und Hodometer 118
3.4 Literatur und Links 126

4 Der Flitzer 127

4.1 Einführung 128
4.2 Konstruktion des Flitzers 132
Vorderradlenkung 132
Karosserie und Beleuchtung – vorne 133
Hinterradantrieb 135
Karosserie und Beleuchtung – hinten 136
Erläuterung der Konstruktion 138
Controller, Akku und Verkabelung 142
Konstruktionsvarianten 146

4.3 Die Steuerung 147
Lenkung 147
Beleuchtung und Hupe 150
Tachometer 157
Funkfernsteuerung mit Joystick 165
Funkfernsteuerung mit Nunchuk 179
Geschwindigkeitsbegrenzer und Abstandsregeltempomat 189
Spurverlasswarner und Spurhalteassistent 201
Parkmanöverassistent 209
4.4 Literatur 217

5 Der Plotter 219

5.1 Hintergrund 220
5.2 Mechanische Konstruktion 224
Grundplatte 224
Schlitten 225
Schreibkopf 229
Konstruktionshinweise 234
5.3 Antrieb 236
5.4 Anschluss an den Arduino 240
5.5 Ansteuerung des Plotters 243
5.6 Grafik-Beschreibungssprache 245
HP-GL 246
Befehlsumfang der Implementierung 248
HP-GL-Basisbefehle 249
Bresenham-Algorithmus 252
HP-GL-Vektorgrafik- und Vieleck-Befehle 255
HP-GL-Parser 266
Plotter-Steuerungsprogramm 270
5.7 Der Plotter im Betrieb 273
Papier 273
Einige einfache HP-GL-Dateien 274
Komplexere HP-GL-Vektorgrafiken 278
5.8 Literatur und Links 280

6 Der Greifer 283

6.1 Industrieroboter in Kinderzimmern 284
6.2 Steuerung mit Potenziometern 290
Aufbau und Motorisierung 291
Anschluss der Elektronik 293
Das Auslesen des Potis 295
Die Motorsteuerung 296
6.3 Der Aufbau des Greifers 300
Der Körper 300
Der Oberarm 303
Die Greifhand 304
Der Unterarm 305
Das Schultergelenk 308
Die Kette und die Parallelführung des Greifers 309
Der Anbau der Motoren und der Potenziometer 310
Die Verkabelung 313
6.4 Die Ansteuerung der Motoren 315
Wie sich die Potenziometerwerte ändern 316
Die Anlaufwerte 317
Die Motorsteuerung 320
Die Steuerung der Greifhand 323
6.5 Die Türme von Hanoi 325
Die Lösung des Problems durch Rekursion . . . 326
Das Programmieren der Rekursion 327
Teach-in: das Anlernen von Positionen 328
Die Funktion zum Bewegen einer Scheibe 331
Der Sketch im Überblick 332
6.6 Literatur und Links 334

7 Der Delta 335

7.1 Parallel ist klasse! 336
7.2 Aufbau und Verkabelung 339
Schultergelenke 339
Oberarme 341

Schultern 342
Motorisierung 343
Der Rahmen 345
Unterarme 346
Hand 347
Verkabelung 348
7.3 Positionserkennung und Motorsteuerung 351
Auslesen der Potenziometer 351
Motorsteuerung 354
Fehlerbehebung, Genauigkeit, Geschwindigkeit 359
7.4 Elektromagnet und Nunchuk 360
Elektromagnet und Kugeln 360
Nunchuk 362
7.5 Solitaire 364
Das Spiel 364
Teach-in 366
Die Programmierung 368
7.6 Tic-Tac-Toe 370
Das Spiel 371
Die Daten 372
Die Logik 374
Die Steuerung mit dem Nunchuk 376
Das Holen und Ablegen der Kugeln 379
Der globale Programmablauf 380
Bedienung per Tastsinn 381
7.7 Kinematik: Wo ist die Hand? 385
Winkel der Oberarme 385
Geometrie 390
Direkte Kinematik 393
Der Roboter als Maus 400
7.8 Die Hand zu vorgegebenen Koordinaten bewegen 405
Inverse Kinematik 405
Koordinaten anfahren 408
Positionen geradlinig ansteuern 410

7.9 Manipulator mit Vakuumsauger 414
Die Unterdruckeinheit. 414
Anbau an den Delta-Roboter 417
Der Manipulator mit Nunchuk 419
7.10 Werkstücke sortieren mit der Pixy-Kamera 421
Die Pixy-Kamera am Arduino 421
Anbau am Delta-Roboter 422
Anlernen von Werkstücken. 424
Die Arduino-Library für die Pixy-Kamera 425
Werkstücke stapeln oder sortieren 426
7.11 Literatur und Links 430

Glossar: Akronyme **431**

Bildnachweise **433**

Zur Einführung

Eine kurze Geschichte der Roboter

Spätestens seit der Antike träumt der Mensch davon, etwas Künstliches zu schaffen, das Tätigkeiten automatisch ausführt und so den Eindruck erweckt, dass es lebt oder mittels »magischer Kräfte« funktioniert. So beschrieb schon *Heron von Alexandria* (1. Jhd. n. Chr.) in seinem Werk »Pneumatika« neben anderen einen Mechanismus, der automatisch die Türen eines Tempels öffnete, sobald ein Opferfeuer auf einem Altar angefacht wurde (Abb. E–1) [1].

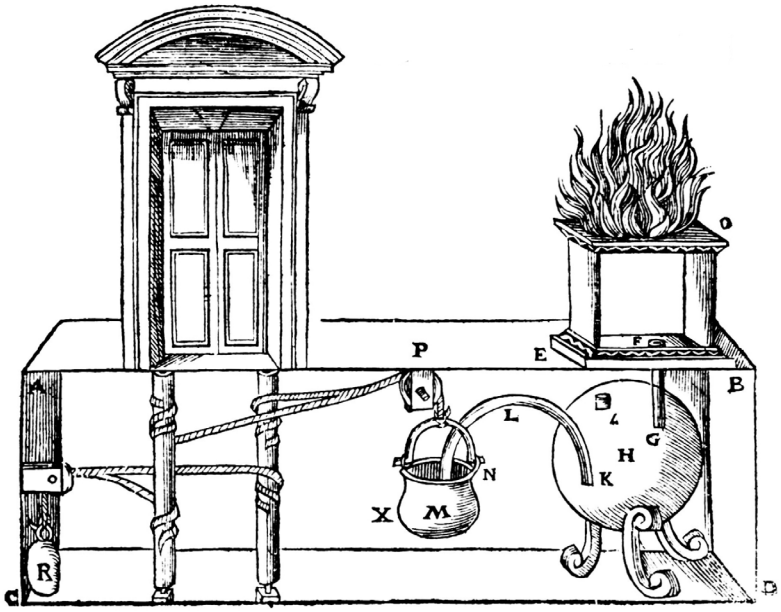


Abb. E–1 Die »automatischen« Tempeltüren nach Heron von Alexandria

Viele der Schau- und astronomischen Uhren an Kirchen und Rathäusern erhielten ab dem 14. Jahrhundert Schlag-, Spiel- und Automatenwerke, die zur vollen Stunde ausgelöst wurden und eine große Wirkung auf die Zuschauer ausübten. So kräht seit 1530 an der Berner Zytglogge ein Hahn vier Minuten vor dem Stundenschlag und schlägt mit den Flügeln; in der Folge dreht ein Bärenzug seine Runde und ein Narr schellt an zwei über ihm hängenden Glocken (Abb. E–2). Ein ähnlicher Hahn fand sich schon an der ersten Uhr des Straßburger Münsters aus dem Jahr 1353.

Abb. E–2 Figurenspiel an der Berner Zytglogge aus dem Jahr 1530

Leonardo da Vinci (1452-1519) konstruierte im 15. Jahrhundert zahlreiche Maschinen, die in Theatern und Fürstenhöfen Volk und Adel beeindruckten, darunter bewegliche Ritterrüstungen und auch ein Fahrzeug, das sich, von zwei Federn angetrieben, selbstständig fortbewegte (Codex Atlanticus, ca. 1478-1480, Abb. E–3). Für den französischen König Franz I. entwickelte er im Jahr 1516 einen mechanischen Löwen, der auf den König zulief, sich auf die Hinterläufe setzte und aus seinem Maul oder Brustkorb Lilien vor dessen Füße fallen ließ.

Die Leistungsfähigkeit solcher Konstruktionen war allerdings begrenzt, denn es existierten nur simple Energiespeicher (wie z.B. Federn, Gewichte oder Wasserdruck). Zudem erlaubte die aus Holz realisierte grobe Mechanik nur sehr einfache Steuerungen. Mit der Entwicklung von verschleißarmen Metallgetrieben für Turmuhren und den Fortschritten in der Feinmechanik wurden komplexere Getriebe in deutlich geringerer Größe und mit weit weniger Reibungsverlusten möglich. Eine Blütezeit erreichten solche mechanischen Automaten im 18. Jahrhundert. So konstruierte z.B. *Jacques de Vaucanson* (1709-1782) im Jahr 1738 eine schnatternde, fressende, trinkende, verdauende und schwimmende mechanische Ente (Abb. E–4).

Weltberühmt wurde der vorgeblich humanoide Schachautomat von *Wolfgang von Kempelen* (1734-1804) aus dem Jahr 1769 (Abb. E–5). Über mehrere Jahrzehnte blieb seine Funktionsweise ein Geheimnis, was die öffentliche Diskussion

und die Fantasie der Menschen stark anfachte. Auch wenn der Automat tatsächlich durch einen kleinwüchsigen Menschen gesteuert wurde, war dieser Apparat ein mechanisches Bravourstück, der das Gebiet stark beflügelte. Eine weit größere, aber viel weniger bekannte Leistung von Kempelens war die Konstruktion einer Sprechmaschine im Jahr 1791 – der erste funktionsfähige Sprachsyntheseapparat überhaupt.

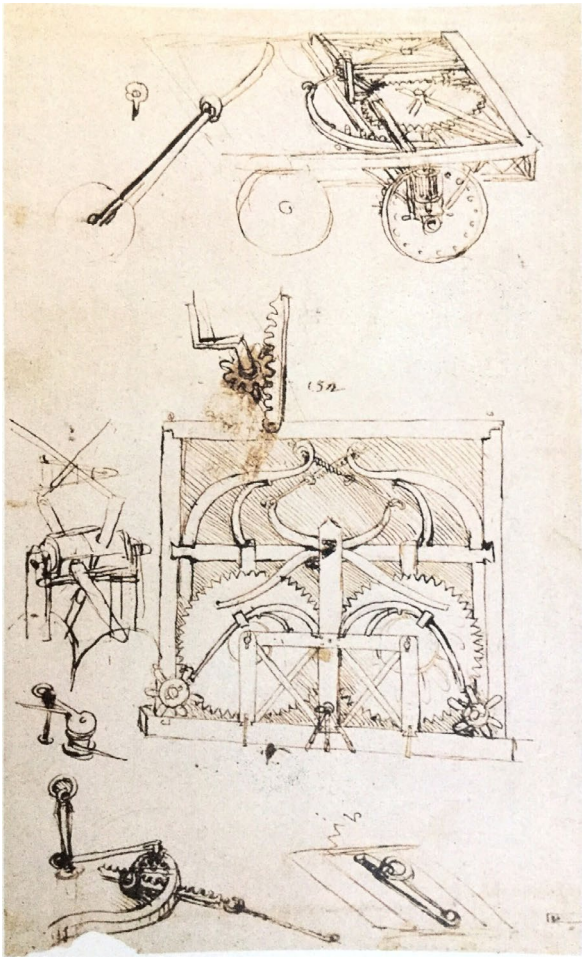


Abb. E–3 Leonardo da Vincis Konstruktion eines federgetriebenen Fahrzeugs (Codex Atlanticus) [2]

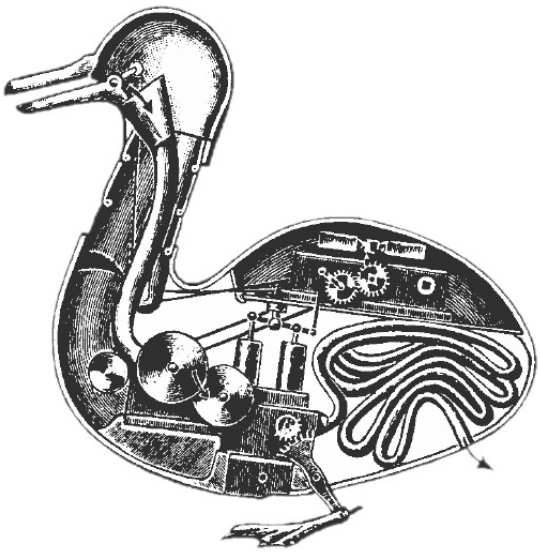


Abb. E–4 Mechanische Ente mit Verdauungsapparat von Jacques de Vaucanson

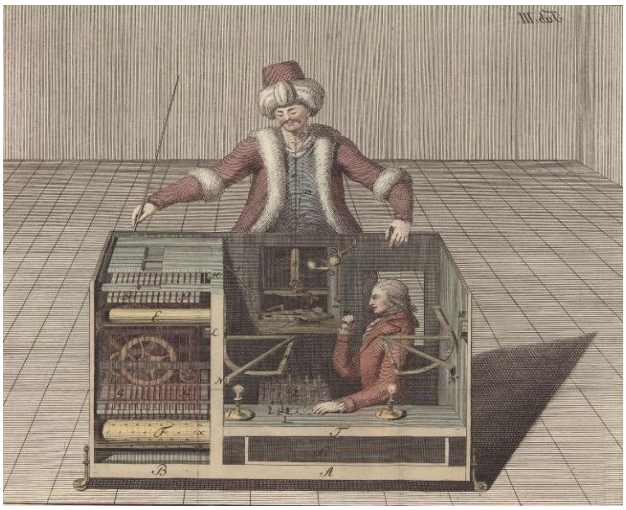


Abb. E–5 Versuch einer Nachbildung des Schachtürken von Wolfgang von Kempelen durch Joseph Friedrich von Racknitz (1789)

Mit der Dampfmaschine von *James Watt* (1736-1819) aus dem Jahr 1784 mit Planetengetriebe erhielt die Menschheit im ausgehenden 18. Jahrhundert die erste – und lange ersehnte – von Ort und Zeit unabhängige Kraftmaschine, den ersten Motor [3].

Wenig später, im Jahr 1805, stattete *Joseph-Marie Jacquard* (1752–1834) den 20 Jahre zuvor von *Edmund Cartwright* (1743-1823) erfundenen mechanischen Webstuhl mit einer Lochkartensteuerung aus – und baute damit den ersten »programmierbaren Automaten« (Abb. E–6). Mit Jacquards Webstuhl erreichte die Automation eine neue Stufe: Zum ersten Mal hatte ein Automat einen echten Zweck und Nutzen jenseits davon, Menschen zu unterhalten oder zu beeindrucken.

Abb. E–6 Modell des Jacquard-Webstuhls im Musée des Arts et Métiers (Paris)

Die Vorstellung, dass technische Automaten dem Menschen unangenehme, monotone, harte oder gefährliche Arbeiten abnehmen können, war entscheidend für den modernen Begriff »Roboter«. Er wurde 1920 vom tschechischen Künstler *Josef Čapek* (1887-1945) für die künstlichen Menschen im Theaterstück *R.U.R.* seines Bruders *Karel Čapek* (1890-1938) geprägt: »Robota« ist das tschechische Wort für »Zwangsarbeit« oder »Frondienst«.

Der wirkliche Durchbruch für die Entwicklung nützlicher und flexibel einsetzbarer Roboter gelang jedoch erst mit der Entwicklung des Computers und dessen anschließender kontinuierlicher Miniaturisierung. Durch die Erfindung des integrierten Schaltkreises (IC) im Jahr 1958/1959 durch *Jack Kilby* (1923-2005) und *Robert Noyce* (1927-1990) und die Entwicklung des Mikroprozessors im Jahr 1970 wurden sehr flexible, leichte, schnelle und verschleißfreie, vor allem aber *programmierbare* Steuerungen möglich.

In den vergangenen 50 Jahren wurden Energiespeicher, Steuerungen und Antriebe so leistungsfähig, klein und bezahlbar, dass Roboter den Menschen inzwischen zahlreiche komplexe Tätigkeiten abnehmen können. Heute durchdringen Roboter immer mehr Lebensbereiche. Moderne Produktionsanlagen sind ohne Roboter schlichtweg undenkbar geworden. Sie verrichten Tätigkeiten exakter, ohne Pausen und meist wesentlich schneller als Menschen – und sind dabei auch noch günstiger. Durch Roboter bleibt die Produktion auch in einem Hochlohnland wie Deutschland wirtschaftlich und ist die Herstellung vieler Präzisionsprodukte überhaupt erst möglich. Roboter sind daher eine moderne Ausprägung des zentralen Ziels aller menschlichen Erfindungsgabe: der Verbesserung unserer Lebensumstände.

Zunehmend werden Roboter auch für Arbeiten in für Menschen tödlichen Umgebungen eingesetzt. Denn Roboter brauchen keinen Sauerstoff und können, bei geeigneter Konstruktion, weit höhere Temperaturen, Strahlungen und Drücke als Menschen ertragen. Sie arbeiten auch im Wasser, im All oder in verseuchtem Gelände zuverlässig. Daher werden sie auch als Lebensretter und Mechaniker eingesetzt – sie können Verschüttete bergen und Bomben entschärfen. Und schließlich können Roboter einige Dinge sogar deutlich besser als Menschen: Sie erkunden als fliegende Drohnen Gegenden aus jeder Höhe und Perspektive oder tauchen in sehr große Tiefen. Auch Rechenleistung und Reaktionsfähigkeit sind größer und schneller als die eines Menschen: Inzwischen spielen sie besser Schach, gewinnen im Tischtennis und sind gerade dabei zu beweisen, dass sie die besseren Autofahrer sind. Auch als Assistenten in der Medizin bewähren sie sich: Sie sind feiner steuerbar als Menschenhände, arbeiten immer konzentriert und ermüden nicht.

Neben der Verfeinerung der mechanischen Konstruktion und immer leistungsfähigeren, schnelleren und kleineren Mikroprozessoren für die Steuerung sind es vor allem hochauflösende Sensoren und ausgefeilte Algorithmen, z.B. zur Bildauswertung, die diese Leistungen möglich machen. In die Entwicklung eines Roboters fließen daher zahlreiche Spezialkenntnisse ein: Nur ein abgestimmtes Zusammenspiel von Mechanik, Programmierung, Sensorik und Elektronik führt zum gewünschten Ergebnis. Dabei helfen auch Erkenntnisse aus der Biologie

(Stichwort »Bionik«): Viele Lebewesen haben optimierte Bewegungsapparate ausgebildet, die heute als Vorbild für die Konstruktion von Robotern dienen – z. B. für deren Aerodynamik oder die Bewegungssteuerung.

fischertechnik-Roboter

Roboter sind – das belegen zahlreiche MINT-Initiativen und Wettbewerbe – für viele Jugendliche und an Technik interessierte Junggebliebene ein faszinierender Einstieg in MINT-Themen. Sie eignen sich auch aus einem weiteren Grund als Thema von MINT-Projekten: Um einen Roboter »zum Leben« zu erwecken, braucht man Kenntnisse aus allen vier MINT-Bereichen: Mathematik, Informatik, Naturwissenschaften und Technik. An einem Roboter lässt sich daher auf hervorragende Weise und ganz konkret der praktische Nutzen dieser Kenntnisse vermitteln.

Dank der rasanten Entwicklung in der Informationstechnik ist die Konstruktion eines Roboters heute nicht mehr allein gut finanzierten Forschungsinstituten vorbehalten. Leistungsfähige Mikroprozessoren und hochwertige Sensoren gibt es inzwischen für wenige Euro – und hilfreiches Lernmaterial findet man zuhauf im Internet.

Für dieses Buch haben wir fünf Robotermodelle ausgewählt, an denen wir die praktische Anwendung von Grundlagen der Mechanik, der Regelungstechnik, der Mathematik und der Informatik vorstellen. Sie eignen sich nicht nur zum Selbststudium, sondern auch für den Einsatz im Unterricht – als Einführung in die Konstruktion und Steuerung komplexer technischer Systeme.

Inzwischen gibt es zahlreiche Bausätze für mobile Roboter. Sie ersparen dem Käufer den früher unvermeidlichen Gang in den Hobbykeller und erfreuen sich großer Beliebtheit – vor allem, weil gut ausgestattete Hobbykeller mit Werkbank, Bohrer, Säge und Feile inzwischen wahrscheinlich Seltenheitswert besitzen. Allerdings haben Bausätze einen großen Nachteil: Das Zusammenstecken eines Bausatzes spart nämlich nicht nur Zeit, sondern reduziert auch Einsichten. Die Auseinandersetzung mit Schwerpunkt, Reibung, Gewicht und Drehmoment, einwirkenden Kräften und statischer Stabilität hat der Hersteller vorweggenommen; sie ist daher am Bausatz nicht mehr nachvollziehbar.

Und ein zweiter wichtiger Nachteil beschränkt die Einsatzmöglichkeiten eines Bausatzes erheblich: Variationen und Ergänzungen des Robotermodells sind meist nur in geringem Umfang möglich, es gibt (wie – leider – üblicherweise in der Schule) nur genau »die eine« richtige Lösung – ganz anders übrigens als im wirklichen Leben. Da nachhaltige Lernprozesse aber nachweislich erst bei einer vertieften Auseinandersetzung mit einer nicht alternativlosen Lösung einer

Problemstellung einsetzen, nämlich durch Experimentieren, Ausprobieren und das Analysieren von Fehlversuchen, sind Bausätze didaktische Sackgassen: Der Lernerfolg ist auf die vom Hersteller vorgesehenen Elemente beschränkt, wirkt nicht darüber hinaus und vermeidet Lernen durch Fehler.

Daher schlossen wir die Verwendung von Bausätzen für dieses Buch aus. Eine komplette Roboter-Eigenkreation stellt jedoch schon in Ermangelung gut ausgestatteter Hobbykeller keine geeignete Alternative dar. Deshalb haben wir uns für die Verwendung eines Konstruktionsbaukastens mit sehr geringen Herstellungstoleranzen und einem professionellen Verbindungssystem (Nuten, Zapfen, Riegel) entschieden, das die Entwicklung vielfältiger mechanischer Modelle und insbesondere Roboter-Prototypen erlaubt sowie deren spätere Modifikation, Ergänzung und Erweiterung ermöglicht. Wollen wir die Gestaltungsmöglichkeiten so offen wie möglich halten, also eine stabile Mechanik mit Statik, Getrieben und Antrieben, mit Pneumatik, Sensoren und Elektronik koppeln, ist das Konstruktionssystem fischertechnik dafür eine ideale Wahl.

Zumal mit fischertechnik bereits in den frühen 80er-Jahren des vergangenen Jahrhunderts erste Robotermodelle für Schulen entwickelt und eingesetzt wurden. 1987 widmete sich ein ganzes Buch der Programmierung von fischertechnik-Robotern [4], und im selben Jahr ließ ein CHIP-Sonderheft einen fischertechnik-Portalroboter Mühle spielen (Abb. E–7, [5]).

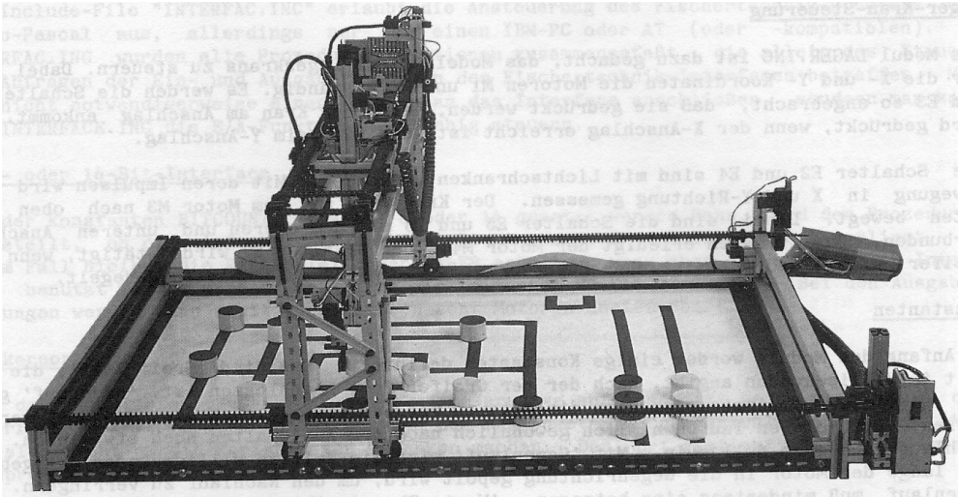


Abb. E–7 Portalroboter spielt Mühle (CHIP Special, 1987) [5]

Die in diesem Buch vorgestellten fünf fischertechnik-Roboter sind Neuentwicklungen und eine Hommage an diese ersten, vor über 30 Jahren konstruierten fischertechnik-Roboter zugleich. Sie nutzen einerseits die statische, mechani-

sche und konstruktive Leistungsfähigkeit des fischertechnik-Systems und zeigen andererseits, welche Möglichkeiten in der Verwendung heutiger Mikrocontroller und Sensoren stecken.

Die fünf Roboter lassen sich allesamt mit aktuellen fischertechnik-Bauteilen nachbauen und wurden von uns so gestaltet, dass sie mit relativ wenigen Bauteilen auskommen. Damit können sie auch mit kleineren fischertechnik-Bauteilsammlungen nachgebaut werden. Einzelne fehlende Teile lassen sich entweder durch kreative Konstruktionsvarianten ersetzen oder auch im Einzelteilvertrieb nachbestellen. Zu allen Modellen haben wir dafür Bauteillisten zusammengestellt, die auf der Webseite zum Buch zum Download bereitstehen.

Grundsätzlich sind auch zahlreiche Varianten unserer Konstruktionsvorschläge möglich – genau dazu wollen unsere Modellvorschläge anregen. Der Lerneffekt wird darunter nicht leiden, ganz im Gegenteil.

Arduino-Steuerung

Die Firma fischertechnik bietet mehrere Controller zur Steuerung von fischertechnik-Modellen an: für junge Einsteiger den Smart BT Controller (als Nachfolger des ROBO LT Controllers) und für »Profis« den (sehr leistungsfähigen) ROBOTICS TXT Controller (als Nachfolger des ROBO TX Controllers). Beide können mit der beeindruckend flexiblen und leistungsfähigen grafischen Programmiersprache ROBO Pro programmiert werden, die wir vor allem Programmiereinsteigern auch sehr empfehlen.

Zur Steuerung der in diesem Buch vorgestellten fischertechnik-Roboter verwenden wir dennoch keinen der fischertechnik-Controller, sondern einen Arduino Uno (Abb. E–8). Zwar nimmt sich ein Arduino Uno mit seinem 16-MHz-8-Bit-Mikroprozessor und mickrigen 2 KB RAM geradezu wie ein Kinderspielzeug im Vergleich mit dem »Herz« des TXT Controllers aus – einem mit 600 MHz getakteten ARM Cortex A8, ausgestattet mit 256 MB RAM und 128 MB Flash-Speicher.

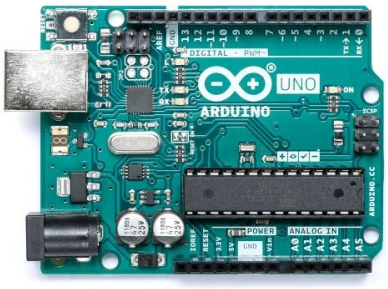
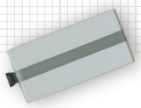


Abb. E–8 Arduino Uno (Rev. 3)

Für die Wahl des Arduino sprechen jedoch vor allem zwei gewichtige Gründe:

- Der fischertechnik-Controller ist selbst zum »Straßenpreis« von etwa 200 € kein Schnäppchen: Für einen Einsteiger, der die Leistungsfähigkeit des Controllers so bald nicht ausreizen wird, ist das eine hohe Investition. Der Smart BT Controller für rund 80 € wiederum besitzt zahlreiche Beschränkungen



(wie die geringe Zahl an Ein- und Ausgängen und das Fehlen einer Möglichkeit zum Programm-Upload), die ihn für die Steuerung eines autonomen mobilen Roboters schlichtweg ungeeignet machen. Wer günstig einsteigen will, dem bleibt nur die Wahl eines anderen Mikrocontrollers.

- Für Einsteiger in die (Roboter-)Programmierung, die eine gängige Programmiersprache wie C, Java oder Python verwenden möchten, ist die Verwendung des TXT Controllers eine Herausforderung: Mit Linux und einem C-Compiler sollte man umgehen können, bevor man hier die ersten Schritte wagt, auch wenn die fischertechnik-»Community« eine eigene Firmware entwickelt hat, die z. B. die Programmierung mit Python durch eine Schnittstelle unterstützt.[1]

Der Preis eines Arduino Uno liegt mit 10-25 € bei einem Bruchteil des Preises eines TXT und er verfügt über eine Entwicklungsumgebung (IDE), in der Steuerungen in einem Derivat der Programmiersprache C/C++ entwickelt werden können. Entwicklungsumgebung, Hardware und Programmiersprache sind sehr ausgereift, schließlich gibt es den Arduino bereits seit dem Jahr 2005. Schon in den ersten drei Jahren wurden mehr als 50.000 Arduino-Boards verkauft; seitdem wird das Open-Source-Projekt kontinuierlich weiterentwickelt. Da das Board über einen 9-V-Anschluss verfügt, können wir den Arduino sogar mit einem fischertechnik-Netzteil betreiben.

Allerdings erkauft man sich den niedrigen Preis des Arduino mit verschiedenen Einschränkungen:

- Der Arduino kommt ohne ein fischertechnik-kompatibles Gehäuse daher, das einen Einbau in fischertechnik-Modelle erleichtern würde und die Platine vor Beschädigungen und Kurzschlüssen schützt.
- Er benötigt für die Ansteuerung der fischertechnik-Motoren sowie für die Kommunikation über WLAN oder Bluetooth zusätzliche aufsteckbare Boards (*Shields*), die den Preisvorteil abschmelzen.
- Die Ein- und Ausgänge des Arduino sind nicht kurzschlussfest – wer sie verwendet, sollte daher wissen, was er tut, sonst ist der Controller schnell zerstört.
- Zwar werden die Steuerungsprogramme in das RAM des Arduino geladen, aber der äußerst begrenzte Speicher erlaubt keine besonders umfangreichen Programme.

1 Die Community Edition der Firmware für den TXT Controller einschließlich einer Installationsanleitung findet sich im Github: *https://github.com/ftCommunity/ftcommunity-TXT*.

Dafür gibt es für fast jede denkbare Anwendung einen Beispiel-Sketch und für jeden verfügbaren Industriesensor eine kostenlose, in der Community getestete Treiberbibliothek, die man lediglich herunterladen und in die IDE des Arduino importieren muss. Der Arduino verfügt auch über deutlich mehr Ports (Input-Output-Anschlüsse) als die fischertechnik-Controller. Außerdem wartet er im Detail mit dem einen oder anderen Pluspunkt auf, den er selbst einem TXT Controller voraushat – mehr dazu im ersten Kapitel.

Warum dieses Buch?

Mit diesem Buch wollen wir nicht nur zeigen, welche faszinierenden Möglichkeiten die Kombination eines ausgereiften technischen Baukastensystems mit einem komfortabel programmierbaren Mikrocontroller und zahlreichen Sensoren für den Roboterbau eröffnet. Wir möchten außerdem an konkreten Beispielen belegen, dass sich die Beschäftigung mit den Grundlagen der Informatik, Mathematik und Mechanik lohnt: Wer zuerst rechnet, seine Mechanik systematisch konstruiert und der Programmierung eine geeignete Modellbildung vorausschickt, kommt zu erheblich leistungsfähigeren Lösungen.

In unseren Robotermodellen verwenden wir ausgewählte, frei verfügbare Arduino-Bibliotheken und konzentrieren uns bei der Programmierung auf den Kern der jeweiligen Anwendung. Dadurch sind die Arduino-Sketche in diesem Buch überwiegend übersichtlich und kurz. Wir ersparen Ihnen außerdem die Suche nach einer passenden Bibliothek: Auf der Webseite zum Buch finden Sie nicht nur alle im Buch abgedruckten Programme (»Sketche«), sondern auch die aktuellen Bezugsquellen der von uns ausgewählten Arduino-Bibliotheken als anklickbare Links.

Außerdem haben wir dort 3D-Bauanleitungen der im Buch vorgestellten fischertechnik-Modelle zum Download bereitgestellt. Sie wurden mit dem Programm »fischertechnik Designer« entwickelt und lassen sich mit einem kostenlosen Viewer (für OS/X) bzw. einer Demoversion des Programms (für Windows) nutzen. Links auf Viewer und Demoversion finden sich ebenfalls auf der Webseite zum Buch. Zu den Schaltungen haben wir mit dem Open-Source-Programm »fritzing« eine grafische Darstellung der Anschlüsse von Motoren und Sensoren am Arduino bzw. an den verwendeten Shields erzeugt, die Sie ebenfalls auf der Webseite als editierbare Datei finden. Ergänzt haben wir Bezugsquellen für die Sensoren, Breadboards und Shields, die wir in unseren Modellen verwenden.

Die Programmierung des Arduino erfolgt in der Programmiersprache C (bzw. in C++). Das Buch enthält keine Einführung in die C-Programmierung, denn es gibt zahlreiche hervorragende Bücher und Online-Tutorials, die genau das

bieten.[2] Speziell für die Programmierung des Arduino existiert auch ein hervorragendes und umfangreiches Online-Referenz-Manual (in deutscher und englischer Sprache).[3] Für den Nachbau der Modelle sind tiefere Kenntnisse der Programmiersprache C auch nicht erforderlich; die abgedruckten Sketche können direkt verwendet werden und vermitteln auch einem Programmiereinsteiger ein erstes Verständnis von der Programmiersprache. Wer zur Weiterentwicklung der vorgestellten Modelle und Programme tiefer in den Arduino und seine Möglichkeiten eintauchen möchte, dem empfehlen wir die Lektüre eines Grundlagenbuchs über den Arduino wie z. B. [6] und [7].

Für erfahrene C/C++-Programmierer hält der Arduino ein paar Besonderheiten bereit, denn wir haben es hier mit einem sogenannten *Embedded System* zu tun – einem Mikroprozessor ohne Bildschirm und Tastatur; die Ein-/Ausgabe erfolgt allein über I/O-Pins. Auf diese Besonderheiten gehen wir im zweiten Kapitel ein und stellen den Befehlsumfang des Arduino und einzelner, von uns verwendeter Bibliotheken vor. Das Kapitel erleichtert das Verständnis unserer Sketche und ermöglicht es Ihnen, diese zu modifizieren und weiterzuentwickeln. Dem fortgeschrittenen C-Programmierer wollen wir damit ersparen, ein Arduino-Buch zu erwerben und es mühsam nach eben diesen Besonderheiten zu durchforsten.

Webseite zum Buch: *fischertechnik-roboter-mit-arduino.de*

Literatur

[1] Sigvard Strandh: *Die Maschine. Geschichte, Elemente, Funktion*. Weltbild Verlag, Augsburg 1992.

[2] Mark Rosheim: *Leonardo's Lost Robots*. Springer-Verlag Berlin Heidelberg, Florenz 2006.

[3] Dirk Fox, Thomas Püttmann: *Technikgeschichte mit fischertechnik*. dpunkt.verlag, Heidelberg 2015.

[4] J. P. M. Steemann: *Robotik mit dem Heimcomputer*. Elektor Verlag, Aachen 1987.

[5] Gerhard Bader: *fischertechnik und Computer*. CHIP Special, Vogel Verlag, Würzburg 1987.

[6] Benjamin Kappel: Ard*uino. Elektronik, Programmierung, Basteln*. Rheinwerk Verlag, Bonn 2016.

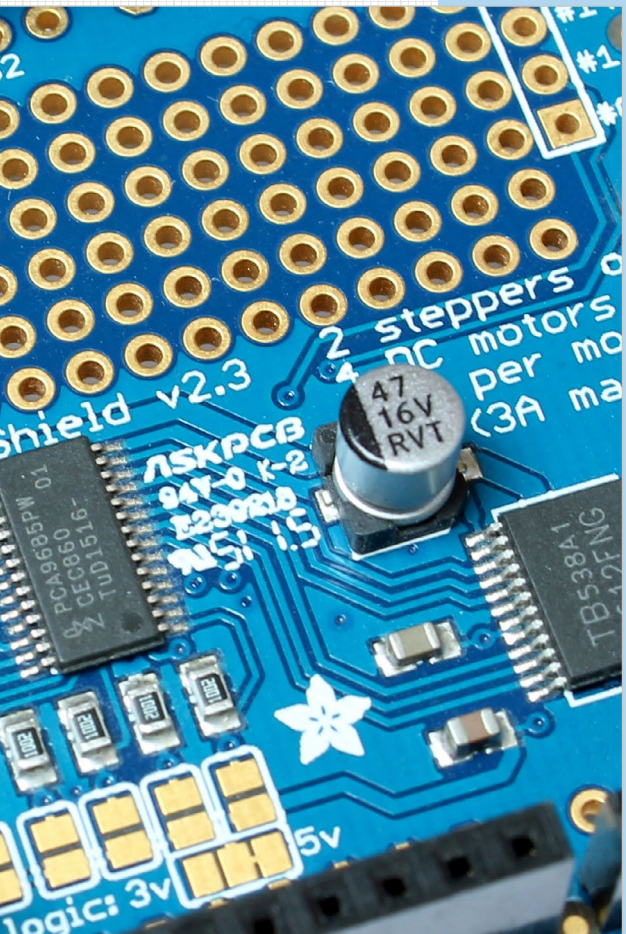
[7] Danny Schreiter: *Arduino Kompendium. Elektronik, Programmierung und Projekte*. BMU Verlag, Landshut 2019.

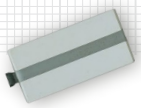
2 Siehe z. B. das C-Tutorial *http://www.c-howto.de/tutorial/*

3 Arduino-Online-Referenz: *https://www.arduino.cc/reference/de/*

1 Der Arduino

Die Steuerung der Modelle in diesem Buch erfolgt mit einem Arduino. Was genau ist der Arduino? Welche Vor- und Nachteile besitzt er im Vergleich mit den originalen fischertechnik-Controllern? Wie lässt er sich in einem fischertechnik-Modell verbauen und wie kann man fischertechnik-Sensoren daran anschließen und nutzen?





1.1 Geschichte

Der Arduino wurde von den Italienern *Massimo Banzi* und *David Cuartielles* entwickelt und im Jahr 2005 als Open-Source-Projekt veröffentlicht, bestehend aus einem Hardwareentwurf und einer Programmierumgebung. Seinen Namen verdankt der Arduino wahrscheinlich dem Markgrafen Arduin von Ivrea (ca. 955-1015), ab 1002 König von Italien – oder aber der gleichnamigen Bar in der Nähe des *Interaction Design Institute* von Ivrea, dem die beiden Entwickler angehörten.

Die Hardware besteht aus einem einfachen Mikrocontroller-Board mit analogen und digitalen Ein- und Ausgängen (Ports) auf 5-V-Basis. Für die vom Amerikaner *David Mellis* konzipierte, auf C++ basierende Arduino-Entwicklungssprache wurde eine auf *Processing* basierende und für die Betriebssysteme Windows, Linux und Mac OS X kostenlos verfügbare Entwicklungsumgebung (*Integrated Development Environment*, IDE) entworfen, in der Anwendungsprogramme, *Sketche* genannt, programmiert, getestet, übersetzt und auf den Arduino geladen werden können.

Gedacht ist der Arduino für alle Arten kleiner elektronischer Steuerungsprojekte, bei denen mit Sensoren Daten erhoben und verarbeitet oder Aktoren angesteuert werden sollen. Dank seines günstigen Preises (etwa 10-25 €), seiner vielseitigen Verwendbarkeit und der leicht zu erlernenden Programmiersprache eignet er sich besonders für Einsteiger. Er hat daher weltweit in vielen Schulen Einzug in den Unterricht gehalten.

Die von Banzi, Cuartielles und Mellis gegründete Arduino LLC pflegt die Entwicklungsumgebung, die – begleitet von zahlreichen Tutorials, einem Forum und einem Blog mit zahlreichen Projektideen – auf der Arduino-Webseite[1] zum Download angeboten wird.

Für alle Boards, die unter dem Namen »Arduino« verkauft werden und auf der Platine das Arduino-Logo tragen, erhält die Arduino LLC eine Lizenzgebühr. Wer also einen solchen originalen Arduino (und keinen der meist etwas günstigeren Klone unter anderem Namen) kauft, unterstützt damit das Open-Source-Projekt und leistet einen Beitrag zur Finanzierung der Weiterentwicklung und Pflege der IDE.

1 Webseite des Arduino-Projekts: *https://www.arduino.cc*

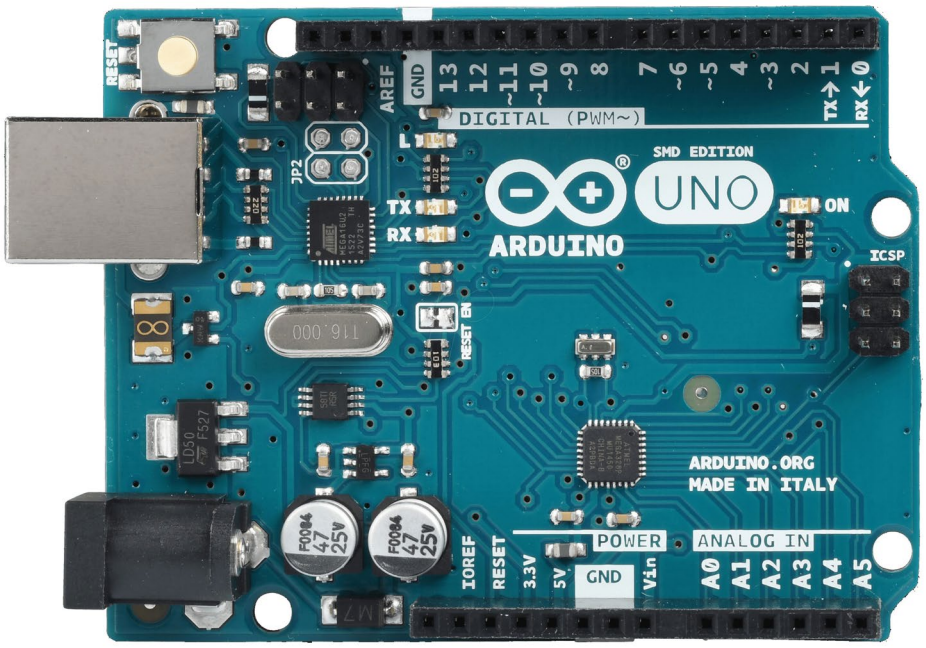


Abb. 1–1 Arduino Uno R3 (SMD-Version)

In den vergangenen 15 Jahren erschienen viele Varianten und Weiterentwicklungen des ursprünglichen Arduino-Boards, darunter die Modelle Due, Ethernet, Industrial 101, Leonardo, Lilipad, Mega, Micro, Nano, Pro, Yún und Zero. Sie sind wie der Lilipad für spezielle Anwendungen wie beispielsweise den Einsatz als *Wearables* vorgesehen, besonders klein wie der Nano, verwenden leistungsstärkere Prozessoren wie der Arduino Industrial 101 oder bieten mehr Speicher und zusätzliche I/O-Ports wie der Arduino Mega oder der Arduino Due.

In diesem Buch verwenden wir aufgrund der weiten Verbreitung und zugunsten einer möglichst großen Einsetzbarkeit unserer Steuerungen und Modelle das ursprüngliche Board, den Arduino Uno, in der aktuellen Revision 3 (R3). Die in diesem Buch vorgestellten Modelle funktionieren auch mit den meisten anderen Arduino-Varianten, allerdings sind dafür in der Regel bauliche Anpassungen am Modell erforderlich, da die Boards unterschiedliche Maße besitzen, und ggf. müssen kleine Korrekturen an der Software vorgenommen werden.

Anders als für die meisten Arduino-Varianten sind für den Arduino Uno geschlossene Gehäuse (*Cases*) erhältlich, die das Board vor Kurzschlüssen schützen und den Anbau an ein fischertechnik-Modell erleichtern – ein weiteres Argument für die Wahl des Arduino Uno.

1.2 Was ist der Arduino?

Man kann den Arduino[2] vereinfacht beschreiben als ein Mikroprozessor-Board mit 14 digitalen Ein- und Ausgängen und sechs analogen Eingängen, über die es Werte von Sensoren einlesen und Aktoren ansteuern kann. Sechs der digitalen Ein-/Ausgänge lassen sich via Hardware als PWM-Ausgang (*Pulse Width Modulation*, Pulsweitenmodulation) nutzen.

Der Mikroprozessor des Arduino ist ein mit 16 MHz getakteter ATmega328P der Firma Microchip Technology Inc. (bis Anfang 2016 Atmel Corp.): ein 8-Bit-Prozessor mit 32 kByte Flash-Speicher für Programme, 2 kByte *Random Access Memory* (RAM) für Daten und 1 kByte *Electrically Erasable Programmable Read-Only Memory* (EEPROM), das mit einer eigenen Library beschrieben werden kann. 0,5 kByte des Flash-Speichers sind von einem Bootloader vorbelegt, der das Hochladen und Starten von selbst entwickelten Programmen im Flash-Speicher übernimmt [1].

Der Arduino benötigt eine Gleichstrom-Spannungsversorgung von 7-12 V und 1 A. Dafür können z. B. die fischertechnik-Steckernetzteile, eine 9-V-Blockbatterie oder ein fischertechnik-Akku genutzt werden. Der Stromanschluss (5,5/2,1-mm-Buchse) ist gegen falsche Verpolung geschützt.

Für den »Online«-Betrieb am PC genügt zumindest für die meisten Sensoranwendungen auch die 5-V-Stromversorgung durch den USB-Anschluss (Typ B). Aber Vorsicht: Die maximale Belastbarkeit des USB-Anschlusses am PC (je nach Typ zwischen 100 und 900 mA) darf von Board und Sensoren nicht überschritten werden, sonst nimmt der PC-Anschluss Schaden. Viele Aktoren (insbesondere Motoren) benötigen allerdings eine höhere Spannung (fischertechnik: 9 V) und haben meist auch eine deutlich höhere Stromaufnahme. Da die Belastbarkeit der Ausgänge des Arduino sehr begrenzt ist – maximal 20 mA Dauerlast, in der Spitze bis 40 mA an den digitalen Ausgängen und höchstens 50 mA am 3,3-V-Anschluss –, müssen Aktoren mit größerem Strombedarf direkt am Vin- und GND-Pin angeschlossen werden. An diese Pins wird die externe Stromversorgung durchgeschleift; sie dürfen mit bis zu 200 mA belastet werden. Die Steuerung muss jedoch vom Arduino entkoppelt werden, da deren Strombedarf die PWM-Ausgänge überlasten würde. Daher verwendet man zur Ansteuerung von Motoren üblicherweise ein *Motor Shield* (siehe Abschnitt 1.7).

Die 14 digitalen Ports D0-D13 können jeweils als Eingang oder als Ausgang genutzt werden. Als Eingang liefern sie einen binären Wert (0 oder 1); dabei kann auch die Länge des anliegenden Signals bestimmt werden (zwischen 0,01 ms und 3 min.). Auch die Ausgänge des Arduino sind digital – sie liefern eine Spannung

2 Mit »Arduino« bezeichnen wir im Folgenden das Modell Arduino Uno R3.

von entweder 0 oder 5 V. Über Pin D13 lässt sich der einzige Aktor, der auf dem Arduino-Board direkt montiert ist, ansteuern: eine LED (siehe Abb. 1–2: Pin 13 LED).

Damit darüber auch Gleichstrommotoren mit wechselnden Geschwindigkeiten oder LED mit unterschiedlicher Helligkeit angesteuert werden können, verwendet der Arduino (genau wie die fischertechnik-Controller) die Pulsweitenmodulation (*Pulse Width Modulation*, PWM): Abhängig von einem sogenannten PWM-Wert zwischen 0 und 255 erzeugt der Controller an dem gewählten Ausgang ein periodisches Signal mit einer festen, aber konfigurierbaren Frequenz/Periode. Der PWM-Wert gibt die Länge eines 5-V-Rechteckimpulses an, der in jeder Periode erzeugt wird. Beträgt der PWM-Wert 0, so gibt es keine Rechteckimpulse und am Ausgang liegt konstant 0 V an. Beträgt der PWM-Wert 255, so füllt der Rechteckimpuls die gesamte Periode aus und am Ausgang liegen konstant 5 V an.

Wie sich der am Digitalausgang angeschlossene Aktor bei den dazwischen liegenden PWM-Werten verhält, ist vom jeweiligen Aktor abhängig. Eine LED mit Vorwiderstand wird einfach ein- und ausgeschaltet. Bei PWM-Frequenzen oberhalb von ca. 20 Hz ist das Auge zu träge, um die einzelnen Blinkvorgänge wahrzunehmen und sieht nur noch ein durchgehendes Leuchten. Bei einem PWM-Wert von 128 ist der Impuls halb so lang wie die Periode. Die Leistung, die von LED und Vorwiderstand umgesetzt wird, ist daher nur halb so groß, als wenn eine konstante Spannung von 5 V anliegen würde.

Das Verhalten eines am Ausgang angeschlossenen Motors ist schwieriger zu verstehen. Im Ergebnis dreht er sich umso schneller, je höher der PWM-Wert ist. Die Induktivität des Motors glättet den Strom, der durch den Motor fließt. Dieser Glättungseffekt ist umso größer, je höher die eingestellte PWM-Frequenz ist. Bei hohen PWM-Frequenzen ist der Strom nahezu konstant. Der Motor verhält sich dann ungefähr so, als ob eine Gleichspannung angelegt wäre. Die Größe dieser Ersatzspannung lässt sich näherungsweise aus dem Verhältnis der Impulsdauer zur Periodendauer berechnen. Bei einer Ausgangsspannung von 5 V, einer Frequenz von 50 Hz (ein Impuls alle 20 ms) und einer Impulsdauer von 4 ms beträgt sie ungefähr

$$\frac{4ms}{20ms} \cdot 5V = 1V$$

Eine hohe Frequenz hat allerdings nicht nur Vorteile. Durch die häufigeren Schaltvorgänge steigt die Verlustleistung (Schaltverluste in den Transistoren, Ankerwicklungen im Motor). Einige Motoren können daher nur mit Frequenzen von 30-200 Hz betrieben werden; verbreitet sind PWM-Frequenzen von 50 Hz

(die auch für die Ansteuerung von Servos verwendet werden, s.u.). Für andere Motoren liegt die ideale Frequenz liegt deutlich höher (1-2 kHz); ab etwa 200 Hz erzeugen viele PWM-angetriebene Motoren hörbare Pfeifgeräusche. In [2] werden für die fischertechnik-Motoren Frequenzen von 70 Hz bis 1,7 kHz empfohlen; der ROBO TX Controller arbeitet mit 200 Hz [3].

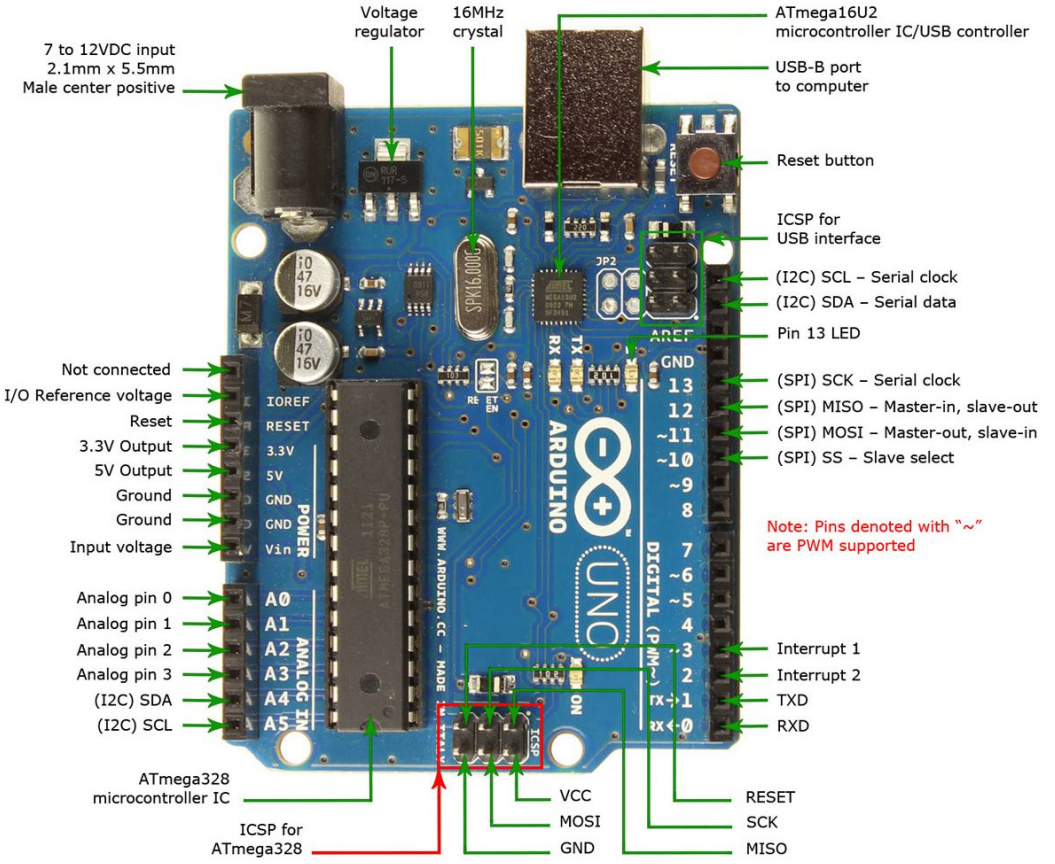


Abb. 1–2 Die Anschlüsse des Arduino Uno

Über die digitalen Ausgänge D3, D5, D6, D9, D10 und D11 des Arduino (auf dem Board mit einer »Tilde« ~ gekennzeichnet, Abb. 1–2) kann ein Analogwert von 0 bis 255 als 5-V-PWM-Signal ausgegeben werden. Die Ausgänge D5 und D6 verwenden dabei eine voreingestellte Frequenz von knapp 980 Hz, die Ausgänge D3, D9, D10 und D11 arbeiten mit einer PWM-Frequenz von 490 Hz. Diese Frequenzen können durch Änderung der Frequenzteiler auf einen Wert zwischen 31 Hz und 62 kHz geändert werden. Per Software können auch alle anderen digitalen Ausgänge mit einem PWM-Signal belegt werden.

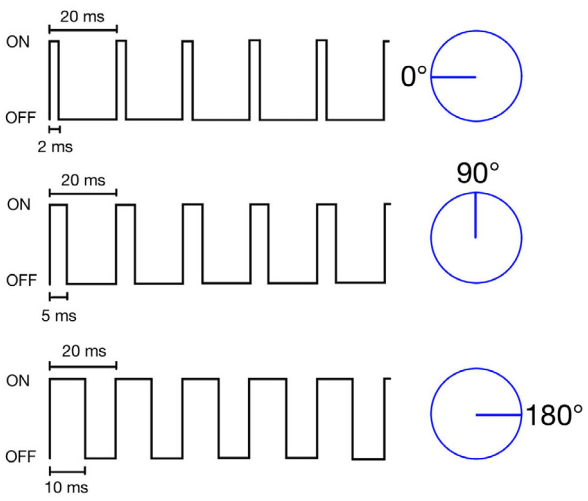


Abb. 1–3: PWM-Signal zur Ansteuerung von Servomotoren

Mit PWM-Signalen kann der Arduino auch auf direktem Wege bis zu zwei Servomotoren aus dem Modellbau ansteuern, die mit den fischertechnik-Controllern nur mit einem separaten I²C-Servo-Adapter genutzt werden können. Servos sind spezielle 5-V-Gleichstrommotoren mit einer Steuer- und Regelungselektronik, die es erlaubt, die Winkelposition der Motorwelle meist zwischen 0° und 180° über ein PWM-Signal einzustellen. Üblich ist ein 50-Hz-Signal, das zwischen 0,5 ms (linker Anschlag des Servos, 0°) und 2,5 ms (rechter Anschlag, ca. 180°) einen High-Pegel und den Rest der Periodenlänge von 20 ms einen Low-Pegel einnimmt (Abb. 1–3).

Für die Servo-PWM-Signale werden beim Arduino die Pins D9 und D10 ge-nutzt.

Die analogen Eingänge A0 bis A5 bilden eine 5-V-Eingangsspannung auf einen 10-Bit-Wert (0-1023) ab. Ist der Maximalwert kleiner, kann man den Pin AREF mit dieser Spannung belegen; der A/D-Wandler wird dann bezüglich dieser Referenzspannung auflösen. Die A/D-Wandlung des Analogwertes benötigt etwa 0,1 ms; die maximale Abfragerate liegt damit bei rund 10.000 pro Sekunde.

Der Arduino unterstützt das Protokoll *Inter-Integrated Circuit* (I²C, beim Arduino auch als *Two Wire Interface*, TWI, bezeichnet) im *Standard Mode* (100 kbit/s) und *Fast Mode* (400 kbit/s). Der Anschluss von I²C-Sensoren (oder auch Aktoren) erfolgt am Arduino Uno R3 über die Pins SDA (Daten) und SCL (Taktsignal), die direkt oberhalb des Pins AREF liegen. Bei älteren Arduino-Uno-Boards sind beide Anschlüsse nur über die Analogeingänge A4 (SDA) und A5 (SCL) zugänglich. Wie der Arduino müssen die verwendeten I²C-Sensoren für 5-V-Logik ausgelegt sein – anderenfalls muss ein *Level Shifter* dazwischengeschaltet werden. GND und Vcc (5 V) können an den entsprechenden Anschlüssen des Arduino-Boards abgegriffen werden.

Die schnellste Datenverbindung zu einem Sensor oder einem anderen Mikrocontroller kann mit dem *Serial Peripheral Interface* Protocol (SPI) erreicht werden: Damit lässt sich eine Übertragungsbandbreite von 4 Mbit/s realisieren – die zehnfache Geschwindigkeit des I²C-Busses. Die SPI-Anschlüsse (SS, MOSI, MISO, SCK) stehen an den Pins D10-D13 bereit. Über den sechspoligen, separat ausgeführten ICSP-Anschluss (*In-Circuit Serial Programming*; in Abb. 1–2 unten

in der Mitte des Arduino-Boards) kann die ICSP-Variante des SPI-Protokolls mit einem sechspoligen Pfostenstecker genutzt werden.

1.3 Der Arduino im Vergleich

Stellt man die Leistungsdaten des Arduino und die der fischertechnik-Controller ROBO TX und ROBOTICS TXT gegenüber, ist auf den ersten Blick klar, dass der Arduino weit hinter der Rechenleistung der fischertechnik-Controller zurückbleibt (Tab. 1–1). Das darf auch nicht verwundern, denn einen Arduino erhält man für einen Bruchteil des Preises eines ROBOTICS TXT Controllers. Vor allem der Prozessortakt und die Speicherkapazität des Arduino liegen im Vergleich deutlich niedriger.

	Arduino (Uno)	ROBO TX	ROBOTICS TXT
Prozessor	ATmega328P (8 bit)	ARM 9 (32 bit)	ARM Cortex A8 (32 bit)
Takt	16 MHz	200 MHz	600 MHz
RAM	2 kB	8 MB	256 MB
Flash	32 kB	2 MB	128 MB
Ports	6 analog (5 V) 14 digital (5 V, 10 kHz), davon 6 PWM-Ausgänge (5 V)	8 Universaleingänge (9 V/5 kOhm) 4 PWM-Ausgänge (9 V) 4 Zählereingänge (1 kHz)	8 Universaleingänge (9 V/5 kOhm) 4 PWM-Ausgänge (9 V) 4 Zählereingänge (1 kHz)
Last	20 mA je Port 200 mA gesamt	250 mA je Port 1 A gesamt	250 mA je Port 1 A gesamt
Schnittstellen	I^2C Fast Mode (5 V, 400 kHz), SPI (4 MHz)	I^2C Fast Mode (5 V, 400 kHz), Bluetooth	I^2C Fast Mode (3,3 V, 400 kHz), Bluetooth 2.1, WLAN
Programmier-sprache	C/C++, Scratch	ROBO Pro (C, Java)	ROBO Pro, Scratch (Python, C, Java, …)

Tab. 1–1 Leistungsdaten des Arduino im Vergleich mit ROBO TX und ROBOTICS TXT

Dennoch lassen sich mit dem Arduino zahlreiche Anwendungen mindestens so gut wie mit den fischertechnik-Controllern realisieren. Das liegt zum einen daran, dass die Prozessorgeschwindigkeit und die Speicherkapazität des TX/TXT von den meisten Anwendungen bei Weitem nicht ausgereizt werden. Zum anderen hat der Arduino – neben dem günstigeren Preis – ein paar Vorteile gegenüber den originalen fischertechnik-Controllern: