PHYSICAL COMPUTING

BRINGEN SIE DEN ARDUINO™ AN SEINE GRENZEN

FRUIT UP YOUR FANTASY

PHILIP CAROLI
CHRISTIAN CAROLI

ARDUINO™

HANDBUCH

Platinen, Shields, Elektronik und Programmieren: Roboter, SMS-Alarmanlage, Wetterstation, automatische Gießanlage und mehr als Treibstoff für eigene Projekte

FRANZIS

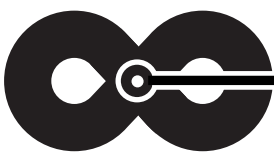
Philip Caroli studiert Informatik am KIT in Karlsruhe und beschäftigt sich schon seit vielen Jahren mit Elektronik und 3-D-Druck. Angefangen bei kleinen elektronischen Schaltungen über Reparaturen von Haushaltsgeräten bis hin zur Entwicklung von kleinen Robotern, hat Philip Caroli alles repariert, erweitert und programmiert, was ihm in die Hände fiel. Als Autodidakt hat er einen sehr praxisbezogenen Zugang zur Materie und kann seine Freude am Thema gut vermitteln. In seiner Freizeit ist er im FabLab Karlsruhe e.V. engagiert und hat dort unter anderem einen Lasercutter mit aufgebaut.

Christian Caroli ist bereits seit seiner Jugend begeisterter Elektronik-Bastler. Was zunächst mit dem Auseinanderschrauben und Wiederzusammenbauen seines Computers begann, setzte er in späteren Jahren mit Zusatzmodulen für seinen PC, diversen Mikrocontroller-Schaltungen und Kleingeräten bis zum heutigen Bau von 3-D-Druckern fort.
Vor zwei Jahren hat der Bastler aus Leidenschaft den Verein FabLab Karlsruhe e.V. mitgegründet und mitaufgebaut, dessen Ziel es ist, den Umgang mit modernen Produktionsmitteln wie 3-D-Druckern und Lasercuttern zu lehren und zu ermöglichen. Hauptberuflich entwickelt Christian Caroli Soft- und Hardware in seiner eigenen Firma.

FRUIT UP
YOUR
FANTASY

PHILIP CAROLI
CHRISTIAN CAROLI

ARDUINO™



HANDBUCH

Platinen, Shields, Elektronik und Programmieren: Roboter, SMS-Alarmanlage, Wetterstation, automatische Gießanlage und mehr als Treibstoff für eigene Projekte

FRANZIS

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte Daten sind im Internet über http://dnb.ddb.de abrufbar.

Programmleitung: Dr. Markus Stäuble
Lektorat: Ulrich Dorn
art & design: www.ideehoch2.de
Satz: DTP-Satz A. Kugge, München
Druck: C.H. Beck, Nördlingen
Printed in Germany

ISBN 978-3-645-60316-4

Vorwort

Sie haben sich also entschieden, in die wunderbare Welt der Arduino-Entwicklungsplattform einzutauchen – dazu kann man Ihnen nur gratulieren! Wir sind der festen Überzeugung, dass Sie es nicht bereuen werden, denn Arduino bietet mittlerweile eine fast unüberschaubare Menge an Möglichkeiten, die Sie ganz individuell und kreativ einsetzen können, egal ob Sie einfach nur den Umgang mit einem Mikrocontroller erlernen möchten, Elektronik mit Digitaltechnik verbinden wollen oder ein ganz konkretes Projekt im Kopf haben, für das Sie einen Arduino nur als Mittel zum Zweck verwenden – mit der Arduino-Entwicklungsplattform können Sie fast alles machen.

Mehr als nur eine Erfolgsgeschichte

Im Jahr 2002 hatte der außerordentliche Professor Massimo Banzi ein Problem. Er war als gelernter Softwarearchitekt eingestellt worden, den Studenten neue Wege des interaktiven Designs beizubringen – manchmal auch Physical Computing genannt. Seine Studenten sollten eigentlich anhand kleiner Einplatinenrechner ihre Ideen verwirklichen, aber die damals verfügbaren Geräte waren zu klein dimensioniert, veraltet und durch die Bank weg zu teuer. Die von ihm zunächst verwendete und damals verbreitete BASIC Stamp kam aus dem Silicon Valley, hatte aber zu wenig Speicherplatz und kostete stolze 100 Dollar. Außerdem arbeitete sie nicht mit den Macintoshs zusammen, die in seinem Institut verwendet wurden. So kam die Idee auf, eine eigene Lösung zu bauen.

Von einem Kollegen am MIT hatte er von der Programmiersprache Processing gehört, die es auch unerfahrenen Personen erlaubte, komplexere Programme zu schreiben. Die Programmiersprache war recht erfolgreich, was Banzis Meinung nach auch an der gelungenen und sehr einfach zu bedienenden Benutzeroberfläche lag – etwas Ähnliches wollte er für sein Projekt ebenfalls einsetzen.

Einer seiner Studenten, Hernando Barragán, entwickelte dann 2005 für ihn den ersten Prototyp, der eine einfache Platine und eine benutzerfreundliche Oberfläche beinhaltete. Ein schönes Projekt, doch Banzi dachte schon weiter, wollte eine Plattform, die noch günstiger und noch einfacher zu bedienen war.

Zu diesem Zeitpunkt war bereits abzusehen, dass die finanziellen Mittel des Projekts, an dem Banzi und seine Kollegen arbeiteten, ausliefen. Sie befürchteten, dass ihre Arbeit untergehen oder missbraucht würde. Da sie allesamt Anhänger der Open-Source-Bewegung waren, lag der Gedanke natürlich nahe, auch ihre Arbeit unter eine derartige Lizenz zu stellen. Allerdings war damals die Open-Source-Bewegung haupt-

sächlich auf Software beschränkt, kaum hingegen wurde Hardware auf dieser Basis entwickelt.

Damit das Ganze funktionieren konnte, benötigte man eine geeignete Open-Source-Lizenz, diese waren jedoch alle entweder auf Software oder aber auf Kulturgüter wie Musik und Prosatext ausgelegt. Nach einigen Recherchen stellten sie aber fest, dass sie nur ihren Blickwinkel, den sie bislang auf ihr Projekt hatten, abändern mussten, um die Creative Commons, eine weitverbreitete, offene Lizenz, verwenden zu können. Fortan wurde also die Hardware als ein Stück Kultur gesehen, die man mit anderen Menschen teilen möchte.

Natürlich war für die Entwicklungsgruppe auch wichtig, dass ihr Produkt für ihre studentische Zielgruppe attraktiv war. Sie peilten einen Preis unter 30 Dollar an – ein Äquivalent zu einem Pizzeria-Abendessen. Außerdem wollten sie die Hardwareplatine schrullig machen, wollten sie cool und herausstechend gestalten. Während andere Platinen meistens grün waren, wurde ihre blau, während andere Hersteller an Ein- und Ausgängen sparten, brachten sie so viele wie möglich unter. Als letzten Schliff platzierten sie noch die Karte von Italien auf die Rückseite der Platine.

Da die meisten Mitglieder des Teams keine Ingenieure waren, sind einige Designentscheidungen, die bei den verschiedenen Arduinos getroffen wurden, eher merkwürdig und von echten Ingenieuren nicht immer leicht zu verstehen, da sie aus einem Do-it-yourself-Ansatz heraus entstanden sind. Vielleicht sollte man daher die Arduino-Plattform auch immer mit einem leichten Augenzwinkern betrachten und nicht alles so ernst nehmen. Wenn es funktioniert, ist es gut (genug).

Wichtig war dem Team ebenso, dass das endgültige Produkt ganz einfach zu bedienen ist: Die Platine wird aus der Kiste genommen, an den Computer angeschlossen, und es kann losgehen – ganz ohne zusätzliche Hardware, Werkzeuge, Stromversorgung oder Ähnliches. Man sollte in der Lage sein, Elektronik zu lernen, ohne sich vorher mit Algebra beschäftigen zu müssen.

Nachdem das Projekt fertiggestellt war, wurden 300 unbestückte Leiterplatten hergestellt und an die Studenten verteilt. Sie sollten die Aufbauanleitung im Internet lesen, die Platine bestücken und dann für ihre Projekte einsetzen. So testete das Team seine eigene Philosophie und konnte Fehler in Dokumentation, Software und Hardware korrigieren.

Ein erstes Projekt war ein Wecker, der an der Decke an einem Kabel aufgehängt war. Drückte man die Snooze-Taste, zog er sich spöttisch ein Stück höher, bis nichts anderes mehr übrig bliebt, als sich komplett zu erheben, um ihn auszuschalten.

Nachdem die 300 Platinen verteilt waren, wurde das Projekt langsam bekannter. Immer mehr Leute wollten auch so ein Gerät haben. Aber es fehlte noch immer ein Name für das Kind. Mit geneigten Köpfen über ihren Getränken, das Problem überdenkend, kam die Lösung zu ihnen: Sie saßen in ihrer Lieblingsbar, die nach einem König benannt war: Arduino von Ivrea.

Der Name schoss schnell um die Welt. Marketing und Werbung waren nicht notwendig, schon früh zog er die Aufmerksamkeit von interessierten Professoren auf sich. Vor allem der Preis von 30 Dollar war eine Sensation, aber auch die einfach zu bedienende Oberfläche und die leichte Handhabung trugen ein Übriges dazu bei. Schon nach kurzer Zeit wurde das Projekt nicht nur an Universitäten verwendet, sondern fand auch Einzug in Fach- und allgemeine Schulen, Künstler beschäftigten sich damit, und auch mancher Hobbykeller wurde bald von ihnen bevölkert.

Das Team um Massimo Banzi stand nun vor der Herausforderung, zum einen die Arduinos in ausreichender Stückzahl zu produzieren und zum anderen den wachsenden Anforderungen der Benutzer gerecht zu werden. Da sie nicht in der Lage waren, Tausende von Platinen zu lagern, ließen sie sie bedarfsgerecht in einer Firma in der Nähe von Ivrea produzieren – pro Tag zwischen 100 und 3.000 Stück.

Zudem entwickelten sie die Arduino-Reihe immer weiter. Der Ur-Arduino wurde nach unserer Zählung 13-mal überarbeitet. Die derzeitige Version ist der Arduino Uno, der selbst schon wieder drei Revisionen erfahren hat – Tendenz steigend.

Eben dieser Uno ist es auch, der als Referenzdesign für die anderen Arduinos der Familie dient. Neben dem Uno gibt es den Leonardo, den Due, den Yún, den Tre, den Micro sowie Mega, Robot, Esplora, Mini, Nano, Pro, Fio, LilyPad und bis zum Erscheinen dieses Buchs sicherlich einige weitere mehr.

Jeder Arduino dieser Familie lässt sich mit der gleichen Entwicklungsumgebung auf PC oder Mac bedienen, arbeitet nach ähnlichen Prinzipien, hat aber jeweils die eine oder andere Fähigkeit, die andere Arduinos nicht haben. Leider ist es nicht möglich, alle Arduinos in diesem Buch zu besprechen, aber die wichtigsten davon werden wir in den nächsten Kapiteln behandeln.

Doch nicht nur die Arduinos wurden von Banzis Team entwickelt, es gibt auch Erweiterungen für Arduinos. Diese sogenannten Shields lassen sich ganz einfach auf ein passendes Arduino-Board stecken und bieten dann neue Möglichkeiten. So gibt es beispielsweise Shields, die einen Arduino in ein bestehendes Computernetzwerk einbinden (Ethernet-Shield, Wi-Fi-Shield), andere bauen ein eigenes Funknetzwerk auf (Wireless-SD-Shield), steuern Motoren an (Motor-Shield) oder kommunizieren sogar mit dem Handynetzwerk (GSM-Shield). Auch diese werden wir im übernächsten Kapitel ansprechen.

Doch nicht nur das Team von Banzi ist in Sachen Arduino tätig, auch zahllose andere Hobbybastler, Universitäten und Firmen bieten mittlerweile Arduino-Platinen und Shields an. Gerade bei eBay, Alibaba und anderen Onlinehandelsplattformen kann man sich unzählig viele Arduino-Nachbauten und Shields besorgen, die preislich meist unter den Originalen liegen, aber längst nicht so cool aussehen. Auch Eigenentwicklungen, die es nicht beim Original-Arduino-Anbieter gibt, kann man hier kaufen, beispielsweise Relais-Shields, mit denen man hohe Ströme schalten kann, Display-Shields, die kleine Monitore beinhalten, oder Sensoren-Shields, die die

Schwerkraft, das Magnetfeld der Erde, Luftdruck, Temperatur, Lichteinfall oder andere Dinge messen können.

Dieses große Angebot an Shields ist auch großen Firmen nicht verborgen geblieben. So bietet beispielsweise Intel mit der Galileo-Plattform ein Arduino-kompatibles Entwicklungsboard an, das Arduino-Shields aufnehmen und ansprechen kann, dabei aber auf einem wesentlich leistungsfähigeren Prozessor (natürlich von Intel) aufsetzt, der dann auch Linux oder Windows RT als Betriebssystem benutzen kann.

Sie sehen also: Das Spielfeld der Arduino-Welt ist unüberschaubar groß, und man kann viele Hundert Stunden des Bastelns darin verbringen, ohne auch nur ansatzweise an Grenzen zu stoßen. Aus dem relativ kleinen Projekt des außerordentlichen Professors Massimo Banzi ist mittlerweile ein millionenschwerer Markt geworden, der aber immer noch auf Open Source basiert und viele Leute glücklich macht – hoffentlich auch bald Sie. (Quelle: *http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino/0*)

Für den unbeschwerten Arduino™-Genuss

Um Ihnen einen unbeschwerten Arduino-Genuss zu ermöglichen, versuchen wir in diesem Buch, Ihnen die Welt des Arduino so schmackhaft zu machen, dass Sie sich voll und ganz darauf einlassen und Spaß daran haben, die Möglichkeiten der Arduinos auszunutzen und selbst Ideen für eigene Projekte zu entwickeln. Dazu stellen wir Ihnen einige Projekte aus ganz unterschiedlichen Bereichen vor, die zeigen, was überhaupt alles möglich ist. Die meisten davon können Sie selbst auf einfache Art und Weise nachbauen – Bauplan und Source Code sind Teil des Buchs. Keines der Projekte ist bis zum Endstadium ausgebaut, alle lassen sich problemlos nach Ihren Vorstellungen erweitern, nur bei den ersten und oftmals schwierigen Schritten nehmen wir Sie ein bisschen an die Hand und erklären, warum was wie aufgebaut wurde, ohne dass wir Sie mit allzu viel Theorie überfrachten.

Dabei ist uns wichtig, dass wir Ihnen den Einstieg in die Welt der Arduinos und der Elektronik so einfach wie nur irgend möglich gestalten. Vermutlich sind Sie wie wir auch kein Fan von endlosen Theoriekapiteln, die mit wissenschaftlicher Genauigkeit jeden noch so unwichtigen Aspekt des jeweiligen Themas auseinandernehmen und mit möglichst langen und komplizierten Sätzen zu beschreiben versuchen. Wir haben daher versucht, uns bei der – leider eben doch an der einen oder anderen Stelle notwendigen – Theorie nach Möglichkeit kurzzufassen und ansonsten möglichst viel durch praktische Übungen zu erklären. Gerade am Anfang ist es wichtig, dass man das große Ganze versteht – Details lassen sich dann immer noch nachschlagen, sei es im Internet oder im Elektronikfachbuch.

Ressourcen zum Buch als Download

In diesem Buch finden Sie etliche Programme, die Bestandteil der beschriebenen Projekte sind. Natürlich sollen Sie in Zeiten des Internets nicht gezwungen werden, die Programme umständlich abzutippen, wir stellen sie Ihnen daher unter der Internetadresse

www.buch.cd

und

arduino-handbuch.visual-design.com

vollständig zur Verfügung. Leider fördert das bequeme Herunterladen nicht allzu sehr das Verständnis des jeweiligen Source Code, daher möchten wir Sie ermutigen, die Programme eigenständig zu erweitern und zu verbessern – Möglichkeiten dazu gibt es reichlich.

Bezugsquellen für passende Bauteile

Das Besorgen passender Bauteile für die Projekte kann umständlich und verwirrend sein, denn oft gibt es unzählige Varianten eines Bauteils, und die Wahl des richtigen ist manchmal schwierig.

Wenn Sie in der glücklichen Lage sind, ein Elektronikfachgeschäft in Ihrer Nähe zu haben, können Ihnen die geschulten Angestellten sicherlich durch den Dschungel der Abkürzungen und Bezeichnungen helfen. Auch nicht vorrätige Bauteile lassen sich dort bestellen. Wenn Sie das Buch mitnehmen, lassen sich auch leicht alternative Bauteile finden, denn nicht immer müssen die Schaltungen eins zu eins nachgebaut werden.

Leider aber werden diese Geschäfte immer seltener, und so ist man häufig gezwungen, sich die Bauteile im Internet zu bestellen. Hier ist man leider auf sich selbst gestellt und muss sich durch endlose Bauteilelisten klicken.

Um Ihnen den Aufwand etwas zu erleichtern, haben wir ebenso auf der Webseite *arduino-handbuch.visual-design.com* Bezugsquellen für die Bauteile zusammengetragen.

Gestatten, Philip und Christian Caroli

Bevor es losgeht, noch ein kurzes Wort zu den Autoren. Wir sind Philip und Christian Caroli, Brüder und leidenschaftliche Bastler, die viel in der Maker-Szene aktiv sind. Als Mitglieder des FabLab Karlsruhe e. V. – einer offenen Werkstatt, in der Interessierte vor allem mit computergestützten Fertigungsmethoden wie 3-D-Druckern oder CNC-Fräsen experimentieren, aber auch mit Elektronik, Textilien oder Software – sind wir sehr daran interessiert, unsere Begeisterung für alles, was man selbst machen kann, zu teilen und jeden, der sich dafür interessiert, mit diesem Virus anzustecken.

Da wir beide aus der Informatik kommen, sind wir selbst Laien auf dem Gebiet der Elektronik und keine hundertprozentigen Elektroingenieure. Wir erheben ebenso wenig Anspruch darauf, perfekte Schaltungen oder Programme zu schreiben, wie es auch das Arduino-Team in seiner Anfangszeit praktiziert hat. Statt bleischwerer Überkorrektheit vertreten wir eher einen pragmatischen Ansatz: Wenn es funktioniert und dabei nicht brennt, ist es schon mal ganz gut.

Dass wir nicht alles so bierernst nehmen, werden Sie vielleicht im einen oder anderen Absatz des Buchs feststellen. Wir hoffen, dass wir damit auch Ihren Geschmack treffen, und wünschen Ihnen viel Spaß beim Lesen und Ausprobieren mit der großartigen Arduino-Plattform!

Philip und Christian Caroli, Karlsruhe im Mai 2015

Inhaltsverzeichnis

I Hardware 17

1 Ausgesuchte Arduino™-Platinen 19
1.1 Arduino™ Uno 19
1.2 Arduino™ Leonardo 22
1.3 Arduino™ Mega 2560 23
1.4 Arduino™ Esplora 25
1.5 Arduino™ Due 27
1.6 Arduino™ Robot 29
1.7 Arduino™ LilyPad 31
1.8 Intel Galileo 33
1.9 Arduino™ Yún 35
1.10 Arduino™ Micro 37
1.11 Arduino™ Fio 38
1.12 Arduino™ Zero 39
1.13 Weitere Platinen 41
1.13.1 Arduino™ Mega ADK 41
1.13.2 Arduino™ Ethernet 41
1.13.3 Arduino™ Mini 41
1.13.4 Arduino™ Nano 41
1.13.5 LilyPad Arduino™ Simple 42
1.13.6 LilyPad Arduino™ SimpleSnap 42
1.13.7 LilyPad Arduino™ USB 42
1.13.8 Arduino™ Pro 42
1.13.9 Arduino™ Pro Mini 43
1.14 Arduino™-Platinen auf einen Blick 44

2 Mit Arduino™-Shields erweitern 45
2.1 Proto-Shield 45
2.2 Ethernet-Shield 47
2.3 Motor-Shield 48
2.4 Wi-Fi-Shield 50
2.5 USB-Serial-Light-Adapter 52
2.6 Arduino™ ISP 53

II Programmierung 55

3 Arduino™-Software entwickeln 57
3.1 Installation der Entwicklungsumgebung 59
3.1.1 Installation auf einem Windows-PC 59
3.1.2 Installation auf einem Apple Mac 63
3.2 Einrichten der Entwicklungsumgebung 64
3.3 Erste Schritte in der C-Programmierung 67
3.3.1 Befehle und Kommentare einsetzen 68

3.3.2 In den Initialisierungsprozess einklinken ... 70
3.3.3 Ein Programm schreiben und installieren ... 71
3.3.4 Variablen als Platzhalter für Befehle nutzen ... 74
3.3.5 Schleifen als Kontrollinstrument einsetzen ... 78
3.3.6 Mit der if-Abfrage Bedingungen festlegen ... 81
3.3.7 Mit Funktionsaufruf Redundanzen vermeiden ... 85
3.3.8 String-Variablen für die Textausgabe nutzen ... 89
3.3.9 Felder definieren die Länge des Textfelds ... 91
3.3.10 Fehlerteufel mit serieller Ausgabe aufspüren ... 92
3.3.11 Präprozessoreinsatz vor dem Kompilieren ... 96
3.3.12 Weiterführende Hilfen für Entwickler ... 98

III Elektronik ... 99

4 Volt, Watt, Ampere und Ohm ... 101
4.1 Elektrischer Strom und Spannung ... 101
4.1.1 Gefährliche Potenzialunterschiede ... 103
4.1.2 Stromstärke elektrischer Leiter ... 104
4.2 Widerstand im Stromkreis ... 105
4.2.1 Farbcodes für Widerstände ... 106
4.2.2 Rechnen mit Volt, Watt, Ampere, Ohm ... 108
4.3 Dioden geben die Richtung an ... 110
4.4 Kondensatoren speichern Energie ... 111
4.4.1 Experiment mit Elektrolytkondensatoren ... 112
4.4.2 Ausführungen und Bauformen ... 113
4.5 Transistoren verstärken und schalten ... 114
4.6 Integrierte Schaltkreise ändern alles ... 115
4.7 Revolution im Kleinen ... 118
4.8 Reihen- und Parallelschaltungen ... 120
4.8.1 Reihenschaltung von Widerständen ... 120
4.8.2 Reihenschaltung von Kondensatoren ... 120
4.8.3 Parallelschaltung von Kondensatoren ... 121
4.8.4 Parallelschaltung von Widerständen ... 122
4.9 Spannung gezielt reduzieren ... 123
4.10 Breadboard-Schaltungen ohne Lötarbeit ... 124
4.10.1 Breadboard – Tipps und Tricks ... 125
4.11 Löten wie die Profis ... 125
4.11.1 Werkzeug zum Löten ... 126
4.11.2 Vorsichtsmaßnahmen ... 133
4.11.3 Erste Schritte: Verbinden zweier Kabel ... 134
4.11.4 Zweite Schritte: Lochrasterplatinen ... 136
4.11.5 Entlöten von Bauteilen ... 143
4.11.6 Tipps und Tricks ... 145
4.12 3-D-Drucker ... 146
4.13 Gebrauch eines Multimeters ... 148
4.13.1 Durchgangsmessung ... 148
4.13.2 Widerstandsmessung ... 149
4.13.3 Spannungsmessung ... 150
4.13.4 Strommessung ... 151

4.13.5 Tipps und Tricks ... 152
4.14 FabLabs und Hackerspaces ... 154
4.15 Schaltpläne lesen und begreifen ... 156
4.16 Datenblätter richtig lesen ... 157

IV 22 Projekte ... 159

5 Arduino™ im Praxiseinsatz ... 161
5.1 Leonardo, der Kollegenschreck ... 161
5.1.1 Motivation ... 162
5.1.2 Aufgabenstellung ... 162
5.1.3 Hintergrundwissen ... 163
5.1.4 Schaltplan ... 166
5.1.5 Source Code ... 166
5.1.6 Tipps und Tricks ... 168
5.2 Analoger Temperatursensor ... 169
5.2.1 Perfektionismus ... 170
5.2.2 Aufgabenstellung ... 170
5.2.3 Hintergrundwissen ... 171
5.2.4 Schaltplan ... 174
5.2.5 Source Code ... 175
5.2.6 Tipps und Tricks ... 177
5.3 Infrarotfernbedienung ... 178
5.3.1 TVZapPro™ ... 179
5.3.2 Aufgabenstellung ... 180
5.3.3 Hintergrundwissen ... 180
5.3.4 Schaltplan ... 182
5.3.5 Source Code ... 184
5.3.6 Tipps und Tricks ... 188
5.4 Lichtschranke ... 189
5.4.1 Gruben graben ... 190
5.4.2 Aufgabenstellung ... 191
5.4.3 Hintergrundwissen ... 191
5.4.4 Schaltplan ... 192
5.4.5 Source Code ... 194
5.4.6 Tipps und Tricks ... 195
5.5 SMS-Alarmanlage ... 196
5.5.1 Handys im Wandel der Zeit ... 197
5.5.2 Aufgabenstellung ... 197
5.5.3 Hintergrundwissen ... 197
5.5.4 Schaltplan ... 200
5.5.5 Source Code ... 202
5.5.6 Tipps und Tricks ... 205
5.6 Wedelstab ... 205
5.6.1 WedelText Maxx ... 206
5.6.2 Aufgabenstellung ... 206
5.6.3 Hintergrundwissen ... 207
5.6.4 Schaltplan ... 210

5.6.5 Source Code ... 213
5.6.6 Tipps und Tricks ... 218
5.7 Kameraauslöser ... 219
5.7.1 Die Wurzel des Übels ... 219
5.7.2 Aufgabenstellung ... 220
5.7.3 Hintergrundwissen ... 220
5.7.4 Schaltplan ... 222
5.7.5 Source Code ... 225
5.7.6 Tipps und Tricks ... 226
5.8 LED-Lichterkette ... 227
5.8.1 Variable Wandfarbe ... 228
5.8.2 Aufgabenstellung ... 229
5.8.3 Hintergrundwissen ... 229
5.8.4 Schaltplan ... 232
5.8.5 Source Code ... 234
5.8.6 Tipps und Tricks ... 235
5.9 Stoppuhr mit Sieben-Segment-Anzeige ... 236
5.9.1 Fehlende Bedarfsanalyse ... 236
5.9.2 Aufgabenstellung ... 237
5.9.3 Hintergrundwissen ... 237
5.9.4 Schaltplan ... 239
5.9.5 Source Code ... 241
5.9.6 Tipps und Tricks ... 244
5.10 Serielle LED-Lichterkette ... 246
5.10.1 Kaufen Sie die neue RitterReiter™ ... 247
5.10.2 Aufgabenstellung ... 247
5.10.3 Hintergrundwissen ... 248
5.10.4 Schaltplan ... 250
5.10.5 Source Code ... 251
5.11 Rotationsmonitor ... 253
5.11.1 Dinge, die die Welt nicht braucht ... 253
5.11.2 Aufgabenstellung ... 254
5.11.3 Hintergrundwissen ... 254
5.11.4 Schaltplan ... 257
5.11.5 Source Code ... 260
5.11.6 Tipps und Tricks ... 263
5.12 LCD-Textdisplay ... 264
5.12.1 Das Henne-Ei-Problem ... 264
5.12.2 Aufgabenstellung ... 265
5.12.3 Hintergrundwissen ... 265
5.12.4 Schaltplan ... 268
5.12.5 Source Code ... 270
5.12.6 Tipps und Tricks ... 273
5.13 Breakout auf TFT-Display ... 273
5.13.1 Notfallspiel aus dem Nichts ... 273
5.13.2 Aufgabenstellung ... 274
5.13.3 Hintergrundwissen ... 274
5.13.4 Schaltplan ... 276

5.13.5 Source Code ... 278
5.13.6 Tipps und Tricks ... 286
5.14 Wetterstation ... 287
5.14.1 Augen auf! ... 287
5.14.2 Aufgabenstellung ... 288
5.14.3 Hintergrundwissen ... 288
5.14.4 Schaltplan ... 290
5.14.5 Source Code ... 292
5.14.6 Tipps und Tricks ... 296
5.15 Automatische Gießanlage ... 296
5.15.1 Karlsruher Student konserviert Zimmerpflanzen! ... 297
5.15.2 Aufgabenstellung ... 298
5.15.3 Hintergrundwissen ... 299
5.15.4 Schaltplan ... 301
5.15.5 Source Code ... 303
5.15.6 Tipps und Tricks ... 304
5.16 Der Arduino™ Robot ... 305
5.16.1 Kaufen Sie die Virtual Robo-Leash™ ... 305
5.16.2 Aufgabenstellung ... 306
5.16.3 Hintergrundwissen ... 306
5.16.4 Source Code ... 309
5.16.5 Tipps und Tricks ... 311
5.17 Analoge Uhr ... 312
5.17.1 Steampunk ... 313
5.17.2 Aufgabenstellung ... 313
5.17.3 Hintergrundwissen ... 314
5.17.4 Schaltplan ... 320
5.17.5 Source Code ... 324
5.17.6 Tipps und Tricks ... 329
5.18 Der Arduino™ Yún ... 330
5.18.1 Der Kollege im Nachbarabteil ... 330
5.18.2 Aufgabenstellung ... 331
5.18.3 Hintergrundwissen ... 331
5.18.4 Inbetriebnahme des Arduino™ Yún ... 331
5.18.5 Source Code ... 342
5.18.6 Tipps und Tricks ... 344
5.19 Blauer Herzschlag ... 345
5.19.1 Schöne neue Welt ... 345
5.19.2 Aufgabenstellung ... 345
5.19.3 Hintergrundwissen ... 346
5.19.4 Schaltplan ... 347
5.19.5 Source Code ... 348
5.19.6 Tipps und Tricks ... 353
5.20 Mobiler Temperaturlogger ... 355
5.20.1 Klobige Allzweckwaffe ... 355
5.20.2 Aufgabenstellung ... 355

5.20.3 Hintergrundwissen ... 356
5.20.4 Schaltplan ... 365
5.20.5 Source Code ... 366
5.21 Breadboard-Arduino™ ... 370
5.21.1 Meister Suns weise Worte ... 370
5.21.2 Aufgabenstellung ... 371
5.21.3 Hintergrundwissen ... 371
5.21.4 Schaltplan ... 375
5.21.5 Tipps und Tricks ... 378
5.22 Arduino™ und Windows ... 378
5.22.1 Schwarz-Weiß ... 379
5.22.2 Aufgabenstellung ... 379
5.22.3 Installation des Windows-PCs ... 379

Stichwortverzeichnis ... 397

Teil I
Hardware

1.14 Arduino™-Platinen auf einen Blick

Arduino	Prozessor	Tolerante Eingangs-spannung	Betriebs-spannung	CPU-Geschwin-digkeit	Digitale Ausgänge	Davon analoge Ausgänge	Davon PWM-Ausgänge	Analoge Ausgänge	EEPROM (KByte)	SRAM (KByte)	Flash (KByte)	USB	UART
Uno	ATmega328	7–12 V	5 V	16 MHz	20	6	6	0	1	2	32	Typ B	1
Due	AT91SAM3X8E	7–12 V	3,3 V	84 MHz	66	12	12	2	–	96	512	2 Micro B	4
Leonardo	ATmega32U4	7–12 V	5 V	16 MHz	32	12	7	0	1	2,5	32	Micro B	1
Mega 2560	ATmega2560	7–12 V	5 V	16 MHz	70	16	15	0	4	8	256	Typ B	4
Mega ADK	ATmega2560	7–12 V	5 V	16 MHz	70	16	15	0	4	8	256	Typ B	4
Micro	ATmega32U4	7–12 V	5 V	16 MHz	32	12	7	0	1	2,5	32	Micro B	1
Mini	ATmega328	7–9 V	5 V	16 MHz	22	8	6	0	1	2	32	–	–
Nano	ATmega168 ATmega328	7–9 V	5 V	16 MHz 16 MHz	22	8	6	0	0,512 1	1 2	16 32	Mini B	1
Ethernet	ATmega328	7–12 V	5 V	16 MHz	20	6	4	0	1	2	32	Typ B	–
Esplora	ATmega32U4	7–12 V	5 V	16 MHz		–		0	1	2,5	32	Micro B	–
ArduinoBT	ATmega328	2,5–12 V	5 V	16 MHz	20	6	6	0	1	2	32	–	1
Fio	ATmega328P	3,7–7 V	3,3 V	8 MHz	14	8	6	0	1	2	32	Mini B	1
Pro (168)	ATmega168	3,35–12 V	3,3 V	8 MHz	20	6	6	0	0,512	1	16	–	1
Pro (328)	ATmega328	5–12 V	5 V	16 MHz	20	6	6	0	1	2	16	–	1
Pro Mini	ATmega168	3,35–12 V 5–12 V	3,3 V 5 V	8 MHz 16 MHz	20	6	6	0	0,512	1	16	–	1
LilyPad	ATmega168V ATmega328V	2,7–5,5 V	2,7–5,5 V	8 MHz	20	6	6	0	0,512	1	16	–	–
LilyPad USB	ATmega32U4	3,8–5 V	3,3 V	8 MHz	9	4	4	0	1	2,5	32	Micro B	–
LilyPad Simple	ATmega328	2,7–5,5 V	2,7–5,5 V	8 MHz	9	4	5	0	1	2	32	–	–
LilyPad SimpleSnap	ATmega328	2,7–5,5 V	2,7–5,5 V	8 MHz	9	4	5	0	1	2	32	–	–
Yún	ATmega32U4	5 V	5 V	16 MHz	32	12	7	0	1	2,5	32	Micro B	1
Robot (Control, oben)	ATmega32U4	5 V	5 V	16 MHz	*	*	*	0	1	2,5	32	Typ B	
Robot (Motor, unten)	ATmega32U4	9 V	5 V	16 MHz	4	1	1	0	1	2,5	32	Typ B	
Intel Galileo V1	Intel Quark SoC X1000	5 V (Gen 1) 7–15 V (Gen 2)	5 V 5 V	400 MHz	20	6	6	0	8	512	8192	Micro B Micro A	1
Zero	ATSAMD21G18		3,3 V	48 MHz	20	6	12	0	16	32	256	2 x Micro B	1

Teil IV
22 Projekte

5

Arduino™ im Praxiseinsatz

Gut gerüstet geht es jetzt an die praktische Umsetzung von 22 kleinen und großen Arduino-Projekten, die einen Querschnitt der vielen Einsatzmöglichkeiten des Arduino zeigen und dazu animieren, mit neuen eigenen Ideen daran anzuschließen.

5.1 Leonardo, der Kollegenschreck

Auf den ersten Blick kann man den Arduino Leonardo leicht mit einem Uno verwechseln, denn von den äußeren Ausmaßen und der Position der Steckleisten sind die beiden sehr gleich. Der große Vorteil des Leonardo gegenüber dem Arduino Uno ist jedoch, dass er auf sehr einfache Art und Weise mit dem PC oder einem Macintosh kommunizieren kann, denn durch eine besondere Programmierung simuliert der Mikrocontroller des Leonardo eine Tastatur oder eine Maus, die über USB an den entsprechenden Computer angeschlossen ist.

Damit ist der Leonardo problemlos in der Lage, den Mauszeiger des Computers zu verschieben oder Texte an den Computer zu senden, die dort genau so erscheinen, als ob sie über die Tastatur eingetragen worden wären. Der Leonardo dreht also als erster Arduino seiner Familie den Spieß um: Er bekommt nicht nur Befehle vom Computer, sondern kann genau solche auch an den Computer zurücksenden.

5.1.1 Motivation

Das Telefon klingelt. Eine von Fett und Chipskrümeln übersäte Hand greift zum Hörer. »Ja? … Ihr Mauszeiger dreht Kreise? Jetzt verarschen Sie mich nicht! … Oh Mann, muss man denn alles alleine machen? Ja, ich komme.« Mit einem abgrundtiefen Stöhnen landet der Hörer auf der Gabel. Ein massiger Körper wuchtet sich aus dem Drehstuhl, und das spärliche T-Shirt mit der Aufschrift »Ich Admin, du nix« wird in gekonnter Star-Trek-TNG-Manier glatt gezogen, was die bislang blank liegenden Speckröllchen gnädigerweise versteckt.

Bedeutungsschwere Schritte nähern sich einem klein gewachsenen Angestellten, der zwar nicht zitternd, so doch voller Sorge in seine Richtung schaut: »Ehrlich, ich habe nichts gemacht!« Der Administrator würdigt ihn keines Blickes, greift sich die Tastatur, plotzt auf den Sitz. Tatsächlich, der Mauszeiger beschreibt einen exakten Kreis und verhält sich dann wieder normal. Das ist doch bestimmt wieder ein Virus, den sich dieser Idiot beim Surfen auf illegalen Seiten eingefangen hat!

Der Task-Manager wird geöffnet, der eine oder andere verdächtige Prozess wird beendet. Der kleine Kollege zuckt nur einmal kurz auf, als die Arbeit des Nachmittags im digitalen Nirwana verschwindet – brav. Da! Schon wieder, ein runder Kreis! Aber wo ist denn jetzt der blöde Virus, er hat doch schon jedes in Frage kommende Programm gekillt! Und schon wieder! Erste Schweißperlen bilden sich – das gibt's doch nicht, bis jetzt hat er noch jeden Virus plattgemacht. Ah, ein Erfolg, keine Kreisbewegung mehr! Aber dafür läuft der Computer insgesamt nicht mehr. Also ein Neustart …

Zwei Stunden später entschwindet ein völlig entnervter Admin vorzeitig in den Feierabend, nachdem er dem Angestellten noch blumenreich mitgeteilt hat, dass er auf ein schwächeres Gerät wechseln muss und der Rechner auseinandergenommen wird – morgen.

Kurz vor Feierabend zieht ein dritter Kollege ein kleines USB-Kabel aus dem Rechner, das – unerreichbar für bückscheue Zeitgenossen – hinten am Rechner angebracht war. Sicherlich wird der kleine Kollege morgen melden, dass der Rechner jetzt wieder problemlos arbeitet. Und falls der Admin diesen Erfolg für sich verbuchen will, wird bestimmt die unschuldige Frage auftauchen, wie genau er das denn nur wieder geschafft hat und wie man das in Zukunft vermeiden kann.

5.1.2 Aufgabenstellung

Wir wollen ein kleines Programm schreiben, das einen Arduino Leonardo dazu veranlasst, den Mauszeiger einen Kreis ziehen zu lassen, nachdem eine Taste gedrückt wurde. Nach Beendigung der Kreisbewegung soll die Maus wieder freigegeben werden.

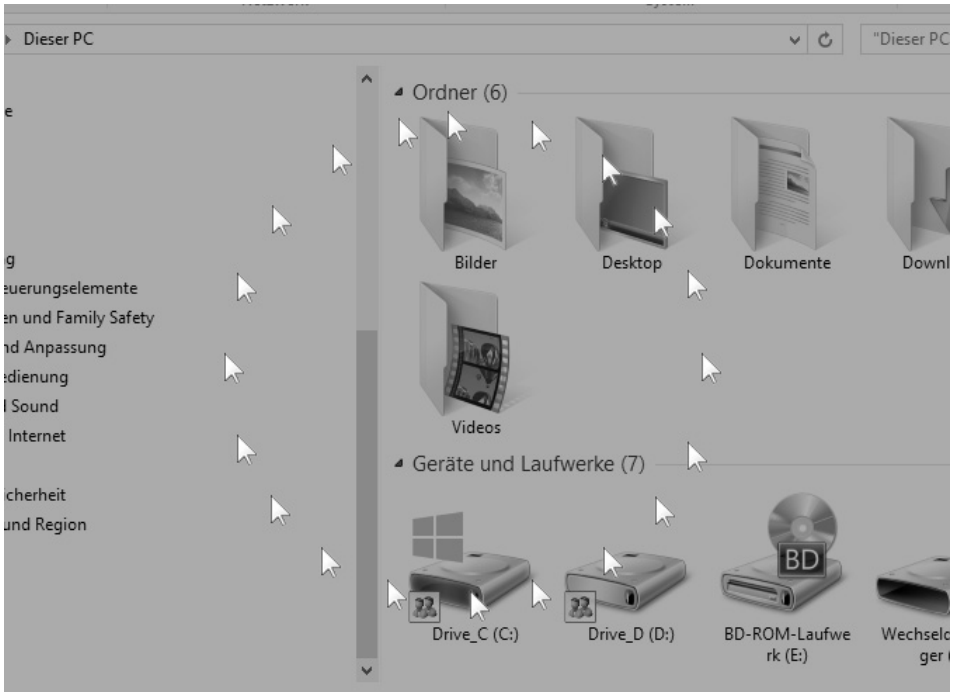


Bild 5.1: Der Arduino Leonardo bewegt den Mauszeiger im Kreis.

5.1.3 Hintergrundwissen

Es ist ja normal, dass ein Arduino über USB mit dem Computer kommunizieren kann. Wann immer ein Programm auf den Arduino geladen werden muss, läuft dies über USB, und auch der serielle Monitor, über den man Informationen vom Arduino erhalten kann, muss diesen Weg gehen.

Kommunikation eines normalen Arduino mit dem Computer

Das geschieht unter Zuhilfenahme der RS232-Schnittstelle, die für einen Arduino deutlich leichter zu handhaben ist als die USB-Schnittstelle. Der Grund hierfür liegt hauptsächlich darin, dass das USB-Datenaustauschprotokoll zwischen Gerät und Computer um ein Vielfaches komplexer ist als das der RS232-Schnittstelle und man gerade in der Anfangszeit des Arduino diese komplexe Thematik einem eigens dafür konstruierten Chip überlassen hat. Dieser Chip ist es, der das komplizierte USB-Protokoll ganz für sich allein abhandelt und dem Arduino das deutlich vereinfachte RS232-Protokoll zur Verfügung stellt.

In den früheren Arduinos wurde hierfür ein spezieller Chip verwendet, der aber keine hohen Geschwindigkeiten zuließ und noch ein paar andere Nachteile hatte. Als dann der Arduino immer älter und fortschrittlicher wurde, ist man umgestiegen auf einen

eigenen Mikrocontroller, der das USB-Protokoll in höherer Geschwindigkeit und mit mehr Kompatibilität abhandelte. Dennoch ist es heute immer noch so, dass der eigentliche Mikroprozessor des Arduino über die RS232-Schnittstelle mit dem Computer kommuniziert.

Kommunikation eines Leonardo mit dem Computer

Natürlich soll ein Arduino Leonardo genauso programmiert werden können wie ein Uno, auch er muss also über die RS232-Schnittstelle mit dem Computer kommunizieren können. Gleichzeitig aber soll er noch auf einem anderen Weg mit dem Computer in Verbindung treten, nämlich als Tastatur und Maus.

Mäuse und Tastaturen für Computer gibt es mittlerweile am Markt unübersehbar viele. Da alle die gleiche Funktion innehaben, nämlich Mausbewegungen bzw. Tastendrücke zu übermitteln, ist es nicht sinnvoll, für jede einzelne Tastatur oder Maus einen eigenen Treiber für Macintosh, Linux und Windows zu schreiben. Daher hat man sich schon vor geraumer Zeit auf einen Standard geeinigt, mit dem Mäuse und Tastaturen mit den Computern kommunizieren müssen. Als »Human Interface Devices« oder auch HID-Geräte werden sie heute von allen Computern mit modernen Betriebssystemen automatisch erkannt.

Auch der Leonardo, der Micro und der Esplora aus der Arduino-Familie gehen diesen Weg und geben sich beim Computer als HID-Gerät aus, das sowohl Maus- als auch Tastatursignale abgeben kann.

Die Kommunikation mit dem Computer übernimmt in diesem Fall kein externer Chip, sondern ein spezieller Mikrocontroller (ATmega32U4), der neben den normalen Aufgaben eines Arduino auch noch in der Lage ist, die USB-Kommunikation zu übernehmen.

Doch jetzt gibt es ein Problem: Der Arduino selbst mit seiner Möglichkeit, Programme zu empfangen und den seriellen Monitor anzusprechen, ist aus Sicht des Computers ein Gerät, die Simulation einer Tastatur oder Maus ist aber ein anderes, zusätzliches USB-Gerät. Es ist nicht möglich, beide Funktionen in einem USB-Anschluss zu vereinen.

Natürlich gäbe es die Möglichkeit, den Arduino mit zwei USB-Kabeln an den Computer anzuschließen. Das wäre aber umständlich und natürlich auch nicht besonders billig, weshalb die Erschaffer des Arduino Leonardo einen anderen Weg gewählt haben: Ist der Arduino an den Computer angeschlossen, um ein Programm zu empfangen oder die serielle Schnittstelle zu bedienen, befindet sich der Arduino in dem einen USB-Modus, soll er hingegen Tastatur- oder Maussignale an den Computer senden, befindet er sich in einem anderen.

Man ist also nicht gezwungen, das USB-Kabel ständig zu wechseln oder gar mit zwei Kabeln zu hantieren, es gibt aber auch ein paar Nachteile. Aufgrund der besonderen Architektur des Leonardo bricht bei einem Reset die serielle Verbindung zum Computer zusammen. Viel nervenaufreibender ist jedoch der Umstand, dass es der

Arduino-Entwicklungssoftware nicht immer gelingt, ein selbst geschriebenes Programm auf den Arduino hochzuladen.

Wenn nämlich der Arduino mit seinem bisherigen Programm die Tastatur- oder Mausfunktion des Computers anspricht, kann das zum einen dazu führen, dass die Bedienung des Programms erschwert wird – weil zum Beispiel der Mauszeiger gerade in eine Kreisbahn gedrängt wird –, zum anderen ist aber der Arduino gar nicht bereit dafür, neue Programme vom Computer entgegenzunehmen.

Auch wenn es hierfür einen kleinen Trick gibt, ist es doch angeraten, bei jedem Programm, das die Tastatur- oder Mausfunktion des Leonardo verwendet, eine Möglichkeit vorzusehen, die HID-Funktion seitens des Arduino ordentlich zu beenden und somit den Weg freizumachen für die normale Kommunikation mit dem Arduino. Das geschieht üblicherweise über die Befehle `Mouse.end()` und `Keyboard.end()`.

Pull-up-Widerstände

Wenn ein Taster gedrückt wird, besteht eine elektrische Verbindung zwischen seinen beiden Anschlüssen, wird er losgelassen, besteht auch keine Verbindung mehr. Wenn ein Taster mit einem Anschluss an die Masse gelegt wird und mit dem anderen an einen digitalen Pin des Arduino, wird bei gedrücktem Taster mit dem Befehl `digitalRead(pin_taster)` immer ein `LOW` gemessen. Was passiert aber, wenn der Schalter nicht gedrückt wird?

Der Pin des Arduino ist über ein Kabel mit dem Taster verbunden, aber es besteht keine Verbindung zur gegenüberliegenden Seite des Tasters. In diesem Fall werden wir bei 100 Aufrufen von `digitalRead(pin_taster)` ungefähr 50-mal ein `LOW` und 50-mal ein `HIGH` messen. Das liegt daran, dass wir mit dem Arduino-Pin und dem Kabel zum Taster eine kleine Antenne aufgebaut haben, die auf viele unterschiedliche Störquellen reagiert.

Für einen Taster ist das natürlich weniger gut. Beheben kann man das, indem man den Arduino-Pin mit einer Spannung verbindet, beispielsweise der Versorgungsspannung. Wenn man dabei noch einen hochohmigen Widerstand benutzt, wird bei gedrücktem Taster trotzdem ein `LOW` gemessen.

Auf solche Widerstände trifft man in der digitalen Elektronik häufiger, man nennt sie auch Pull-up-Widerstände. Freundlicherweise sind im Mikrocontroller des Arduino solche Pull-ups bereits intern eingebaut, wir müssen sie nur anschalten. Das geschieht, indem der entsprechende Pin durch den Befehl `digitalWrite(Pinnummer, HIGH)` initialisiert wird.

Da nur Pull-up-Widerstände eingebaut sind, die den Pin mit der Versorgungsspannung verbinden, und keine Pull-down-Widerstände für eine Verbindung mit der Masse, können Taster nicht zwischen Arduino-Pin und der Versorgungsspannung geschaltet werden, da sonst sowohl bei gedrücktem als auch bei nicht gedrücktem Schalter ein `HIGH` gemessen werden würde.

Benötigte Bauteile
- 1 Arduino Leonardo
- 1 Breadboard
- 1 Mikrotaster
- 2 Drahtbrücken

5.1.4 Schaltplan

Um den Taster an den Arduino anzuschließen, müssen Sie eine Seite mit der Masse verbinden – auf dem Arduino mit GND bezeichnet – und die andere Seite mit dem digitalen Pin 2 des Arduino. Bei einem üblichen Taster mit vier Anschlusspins sind jeweils zwei Pins im Bauteil intern miteinander verbunden, weshalb er nicht beliebig angeschlossen werden kann. Sie müssen zwei auf gegenüberliegenden Seiten herausstehende Pins zum Anschluss benutzen. Sollte Ihre Schaltung nicht funktionieren, versuchen Sie einmal, den Taster um 90 Grad zu drehen.

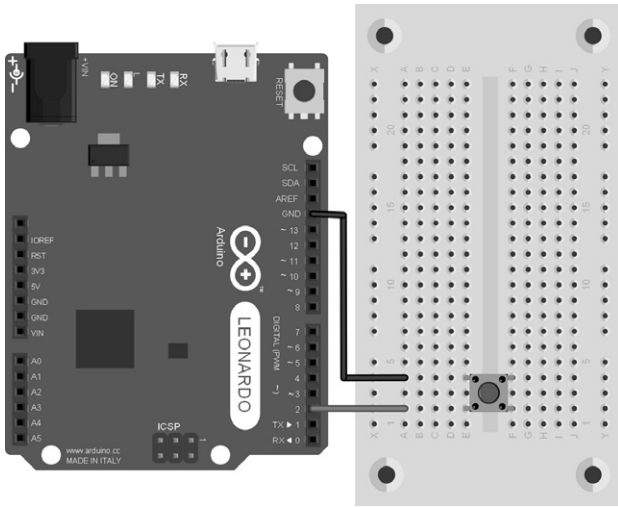


Bild 5.2: Der Breadboard-Aufbau der Schaltung.

Download Source Code
`Mauswandern.ino` und `MauswandernEsplora.ino`
- *buch.cd*
- *arduino-handbuch.visual-design.com*

5.1.5 Source Code

```
001 void setup() {
002   pinMode(2, INPUT);            //Pin 2 auf Ausgang setzen
003   digitalWrite(2, HIGH);        //Pull-up-Widerstand einschalten
```

```
004   Mouse.begin();                //Mauskontrolle übernehmen
005 }
006
007 bool StartKreis = false;
008 float GradZahl = 0;             //Aktuelle Position (Grad)
009 float SchrittWeite = 0.1;       //in Grad
010 double KreisDurchmesser = 100; //300 Pixel Durchmesser
011 double XPos = 0;                //Aktuelle absolute Position (in Pixel)
012 double YPos = 0;
013 double AlteXPos = 0;            //Vorherige absolute Position (in Pixel)
014 double AlteYPos = 0;
015
016 void loop() {
017   if (digitalRead(2) == LOW)    //Bei Knopfdruck Vorgang starten
018   {
019     StartKreis = true;
020   }
021
022   if (StartKreis) {             //Wenn Vorgang gestartet ist
023     //Berechnung der neuen Position
024     GradZahl = GradZahl + SchrittWeite;
025     XPos = round(sin(GradZahl) * KreisDurchmesser);
026     YPos = round(cos(GradZahl) * KreisDurchmesser);
027     // Maus an neue Position setzen
028     Mouse.move(XPos - AlteXPos, YPos - AlteYPos, 0);
029     AlteXPos = XPos;            //Aktuelle Position merken
030     AlteYPos = YPos;
031   }
032
033   if (GradZahl > 2 * PI) {      //Nach einer Umdrehung anhalten
034     GradZahl = 0;               //Position auf 0
035     StartKreis = false;         //Vorgang beenden
036     Mouse.end();                //Mauskontrolle wieder abgeben
037   }
038
039   delay(20);                    //Einige Millisekunden warten
040 }
```

In der `setup()`-Routine des Programms wird der Modus des zweiten Pins des Arduino auf Eingang geschaltet, und durch den `digitalWrite`-Befehl wird der im Prozessor vorhandene Pull-up-Widerstand aktiviert. Außerdem wird die Leonardo-spezifische Klasse `Mouse` durch den `begin()`-Befehl initialisiert, und damit wird auch die Kommunikation zum Computer als HID-Gerät aufgebaut.

Anschließend werden etliche Variablen definiert, die wir im weiteren Verlauf des Programms benutzen und über die man auch einige Einstellungen vornehmen kann. Interessant ist hierbei vor allem die Variable `KreisDurchmesser`, über die die Größe des Kreises bestimmt wird, die der Mauszeiger abfahren soll.

In der `loop`-Anweisung wird dann zuerst einmal abgefragt, ob Pin 2 auf `LOW` steht, was gleichbedeutend mit einem Tastendruck ist. Wurde die Taste gedrückt, wird die Variable `StartKreis` auf den Wert `true` gesetzt.

Eben diese Variable wird anschließend auf den Wert `true` geprüft, um sicherzugehen, dass auch immer ein Kreis abgeschlossen wird, egal wann die Taste wieder losgelassen wird.

Ist dies also der Fall, beginnt die Berechnung der aktuellen Position des Mauszeigers, die vor allem von der Variablen `GradZahl` abhängt, die den derzeitigen Winkel im Kreis angibt und die bei jedem Durchlauf des `loop`-Bereichs um eine `SchrittWeite` erhöht wird.

Bei der Berechnung selbst wird eine virtuelle Position des Mauszeigers berechnet. Virtuell deshalb, weil wir die tatsächliche Position des Mauszeigers nicht wissen, denn der Computer nimmt zwar Mauspositionierungsdaten vom Leonardo an, gibt aber keinerlei Informationen über die aktuelle Position zurück. Aus der virtuellen Position berechnen wir eine relative, denn nur das akzeptiert der Computer als Mauseingabe: Es ist also nicht möglich, dem Computer zu befehlen, an die Position x = 100 und y = 100 zu gehen, vielmehr akzeptiert er lediglich das Kommando, die Maus vom aktuellen Standpunkt aus um beispielsweise fünf Pixel nach oben und sieben nach rechts zu bewegen.

Die relative Position errechnen wir aus der virtuellen ganz einfach, indem wir die im vorherigen `loop`-Durchgang (`AlteXPos` bzw. `AlteYPos`) gespeicherte Position abziehen.

Nun benötigen wir noch eine Möglichkeit, die Kontrolle der Maus wieder abzugeben und damit die Kommunikation mit dem Arduino zum Upload eines neuen Programms wieder freizugeben. In unserem Fall wird einfach geprüft, ob die Variable `GradZahl` den Wert `2 * PI` angenommen hat, also die Kreisbewegung vollendet ist.

Im letzten Befehl des Programms lassen wir den Leonardo noch 20 ms warten, sodass der Mauszeiger insgesamt ca. 50 Mal in der Sekunde neu gesetzt wird.

5.1.6 Tipps und Tricks

Man kann dieses Projekt übrigens auch auf einem Arduino Esplora zum Laufen bringen. Zwar hat dieser keine Möglichkeit, einen Schalter anzubringen – dafür bringt er aber selbst mehr als genug davon mit.

Verwendung eines Arduino Esplora

Wenn Sie über einen Esplora verfügen, brauchen Sie also keinen externen Schalter anzuschließen und müssen lediglich den Source Code etwas abwandeln. Vor den gesamten Code schreiben Sie dazu die Zeile `#include <Esplora.h>`, die die Entwicklungsumgebung dazu veranlasst, die Esplora-Bibliothek zu verwenden. Anschließend müssen Sie nur noch die erste Zeile des `loop`-Bereichs

```
017 if (digitalRead(2) == LOW)    //Bei Knopfdruck Vorgang starten
```

durch diese hier ersetzen:

```
017 if (Esplora.readButton(SWITCH_DOWN) == LOW)
```

Sie bewirkt, dass statt des extern angeschlossenen Schalters der *Unten*-Schalter des rechten Tastenfelds verwendet wird.

Das war's dann auch schon, das Programm läuft jetzt auch auf Ihrem Esplora.

Uploadprobleme beheben

Wie Sie gesehen haben, können Sie mit dem Leonardo sehr einfach Tastatur- und Mauskommandos an den Computer senden. Wenn Sie aber vergessen haben, eine Möglichkeit einzubauen, diese auch abzuschalten, kann es schwierig werden, ein neues Programm auf den Leonardo zu laden, weil dann die Tastaturcodes und Mausbewegungen die Bedienung der Entwicklungsumgebung sehr erschweren. Glauben Sie uns, wir haben da einschlägige Erfahrungen.

Zum Glück gibt es einen Trick: Wenn Sie den Leonardo an den PC anschließen und die Reset-Taste drücken, wird der Bootloader des Arduino acht Sekunden lang aktiviert. In dieser Zeit ist der Leonardo unter einem anderen COM-Port erreichbar als üblich und sendet noch keine Tastatur- oder Maussignale. Wenn Sie es schaffen, den COM-Port der Entwicklungsumgebung auf den neuen Port umzustellen und den Uploadvorgang während dieser acht Sekunden zu starten, klappt der Upload.

Als Vorgehensweise empfiehlt sich, zuerst den Reset-Schalter zu drücken und dann in der Entwicklungsumgebung unter *Werkzeuge/Port* den neuen Port einzustellen. Welcher das ist, können Sie übrigens ganz gut im Geräte-Manager von Windows in der Abteilung *Anschlüsse* herausfinden.

Wenn Sie das geschafft haben, merkt sich die Entwicklungsumgebung die Portnummer, und Sie können den Uploadvorgang starten. Da zuvor das Programm kompiliert wird und das Kompilieren leider oft länger als acht Sekunden dauert, sollten Sie währenddessen alle fünf Sekunden den Reset-Knopf drücken, bis der eigentliche Uploadvorgang startet. Vielleicht brauchen Sie ein paar Versuche, bis es klappt, aber wenn Sie einmal den Bogen raus haben, ist es eigentlich recht leicht.

5.2 Analoger Temperatursensor

Im Prinzip ist ein Computer ein recht dummes Gerät – während jede Stubenfliege aus einem brennenden Haus flüchten würde, ist ein Arduino noch nicht einmal in der Lage, zu merken, dass er gerade brennt. Sensoren können das ändern. Sie messen die Umwelteinflüsse und wandeln sie in Signale um, die ein Mikrocontroller auswerten und anschließend entsprechend reagieren kann. So können Rollladen bei beginnender Dunkelheit heruntergefahren, Scheibenwischer bei einsetzendem Regen angeschaltet und Airbags bei einem Unfall ausgelöst werden.

Stichwortverzeichnis

Symbole

#define 96
3-D-Drucker 146, 154

A

ADC-Rauschunterdrückung 358
Addition 88
ADK-Version 24
Akku 254
Alkaline-Batterie 256
Ampere 104
Analoge Uhr 312
Anode 111
Arduino
  Elektronik 101
  Entwicklungsumgebung 57
  Geschichte 5
  Hardware 19, 45
  Programmierung 57
  Projekte 161
Arduino Due 27
Arduino Esplora 25, 168
Arduino Ethernet 41
Arduino Fio 38
Arduino ISP 53
Arduino Leonardo 22
Arduino LilyPad 31, 42
Arduino Mega 2560 23
Arduino Mega ADK 41
Arduino Micro 37
Arduino Mini 41
Arduino Nano 41
Arduino Pro 42
Arduino Pro Mini 43
Arduino Robot 29, 305
Arduino Uno 7, 19
Arduino Yún 35
Arduino Zero 39
Arduino-Befehle 98
Arduino-Entwicklungsumgebung 59
Arduino-Software
  Apple Mac-Installation 63
  Download 59
  Windows-Installation 59
Arduino-Supportseite 67
Arduino-Websites 98
Array 91
ASCII 89
Atheros AR9331 35
AT-Kommandos 197
ATmega 53
ATmega16U2 21
ATmega328 20
ATmega32U4 22, 35
Ausgang 71
Automatische Gießanlage 296

B

Banzi, Massimo 5
Barragán, Hernando 5
BASIC Stamp 5
Basis 114
Batterie 254
Bauteile besorgen 9
Befehle 68
Beschleunigungssensor 209
Betriebssystem 59
Bewegungssensor 200, 206
Bibliothek 96
Blei 127
Bleifreies Löten 145
bool 81
Booleans 81
Breadboard 124
Breakout auf TFT-Display 273
Bridge-Bibliothek 35
Brown-out-Detektor 359
Bug 96
byte 77

C

C 57, 93
C# 57
C++ 93
Chip 115
Compiler 58, 66
C-Programmierung 67
Creative Commons 6

D

Datenblätter lesen 158
Debugger 92
Debugging 96, 394
delay 73
delayMicroseconds() 77
digitalWrite 72
DIL 116
Diode 110
double 77
Download
  Arduino-Software 59
  Source Code 8
Dünnschichttransistor 274
Durchgangsmessung 148

E
EEPROM 361
Elektrischer Leiter 104
Elektrolytkondensatoren 112
Elektromotor 48, 299
Elektroniklötzinn 127
Emitter 114
Energiesparen 378
Entlötlitze 143
Entlötsaugpumpe 143
Entwicklungsumgebung 57, 59
  einrichten 64
Erweiterungen 7
Erweiterungsplatinen 45
Ethernet-Shield 47

F
FabLabs 154
FALSCH 81
false 81
Farad 113
Farbcode, Widerstand 106
Farbraum 229, 248
Feld 91
Feldeffekttransistor 231
Fernbedienung 178
Feuchtigkeitssensor 302
Firmware 288
Fliegende Schaltung 353
float 77
Flüssigkristallanzeige 265
Flussmittel 145
for 78, 79
Formatieren 382
for-Schleife 79
Freilaufdiode 300
Funktion 71, 85
Funktionsaufrufe 85
Fuse-Bits 377

G
Gleichstrommotor 49
Gleitkommazahl 76
GSM 197

H
Hackerspaces 154
Halbschritt 316
HD G 104-Prozessor 51
Heißleiter 171
Helligkeitssensor 302
HIGH 72
Hohe Stromstärken 230
HSV 248

I
I^2C 290
IC 115
ICSP-Anschluss 20
Idle 358
if 81
if-Abfrage 81
Infrarot 180, 181
Infrarotleuchtdiode 183
Infrarotsensor 308
Initialisierung 79
Initialisierungsprozess 70
Inkrementierung 79
int Siehe Integer
Integer 76, 79
Integrierter Schaltkreis 115, 118
Intel Galileo 33, 378
int-Zahl 77
IRremote-Bibliothek 184
Isolierung 131
ISP 371

K
Kabel 131
Kalte Lötstelle 145
Kaltleiter 171
Kameraauslöser 219
Kathode 111
Kleidung 354
Kollektor 114
Kommandos 68
Kommandozeile 336
Kommentare 68
Kompilierung 58
Kondensator 111
  Parallelschaltung 121
  Reihenschaltung 120
Kupferlitze 131, 143

L
Lasercutter 154
Lautsprecher 172
LCD-Textdisplay 264
LED 19
Leonardo 161
Leuchtdioden 220
Lichtdetektor 193
Lichterkette 227
Lichtschranke 189, 191
LiIon-Akkus 256
LilyPad 345
LilyPad Arduino Simple 42
LilyPad Arduino SimpleSnap 42
LilyPad Arduino USB 42
Linux 59
Liquid Crystal Display 265
Lithium-Polymer-Akku 347
Litze 131
Lochrasterplatine 125, 136, 211
long 77
loop 71
loop-Bereich 86
Lötbrücke SJVCC 54
Löten 125
  Vorsichtsmaßnahmen 133
Lötfett 145
Löthilfe, helfende Hand 130
Lötkolben 126
Lötschwamm 129

Lötstation 126
Lötwasser 145
Lötzange 144
Lötzinn 127
LOW 72
Luftdruck 101, 290
Luftfeuchtigkeit 289
Lufttemperatur 289

M
Master 276
Memory-Effekt 256
Messwandler 171
Milliampere 104
MISO 276
Mobiler Temperaturlogger 355
MOSI 276
Motor-Shield 48, 319
Multimeter 148
Multiplexing 237

N
New Line 94
NiMH-Akkus 256
NPN-Transistor 115
NTC-Widerstand 171

O
Objektorientiertes Programmieren 93
ODER 81
Ohm 106
Ohm'sches Gesetz 109
Omega 106
Open-Source-Bewegung 5
Open-Source-Lizenz 6
OpenWrt-Yun 331
OS X 59
OUTPUT 72

P
Parallelschaltung 121
Physical Computing 5
Pin 13 19, 71
pinMode 72
PNP-Transistor 115
Potenzialausgleich 103
Potenziale 102
Potenziometer 124, 266
Power-Down 358
Power-Save 358
Präprozessor 96
Processing 5
Programm
  hochladen 66
  installieren 73
  kompilieren 66
Programmiersprache C 57, 68, 93
Programmierung 67
  Befehle 68
  Funktion 71
  Initialisierungsprozess 70
  Kommandos 68
  Kommentare 68
Projekte
  Analoge Uhr 312
  Analoger Temperatursensor 169
  Arduino Robot 305
  Arduino und Windows 378
  Automatische Gießanlage 296
  Blauer Herzschlag 345
  Breakout auf TFT-Display 273
  Infrarotfernbedienung 178
  Kameraauslöser 219
  LCD-Textdisplay 264
  LED-Lichterkette 227
  Leonardo 161
  Lichtschranke 189
  Mobiler Temperaturlogger 355
  Rotationsmonitor 253
  Serielle LED-Lichterkette 246
  SMS-Alarmanlage 196
  Stoppuhr 236
  Wedelstab 205
  Wetterstation 287
Proto-Shield 45
Prozessor 67
PTC-Widerstand 171
Pull-down-Widerstand 165
Pull-up-Widerstand 165
Pulsweitenmodulation 75
PVC 309
PWM 362
PWM (Software) 352

Q
Quelltext 58

R
Rechnen
  Ohm'sches Gesetz 109
  Spannung 108
  Stromstärke 108
Register 356
Reihenschaltung 120
Reset-Schalter 19
RGB 229
RoHS 127, 145
Rotationsmonitor 253
RS232-Protokoll 163
RS232-Schnittstelle 92, 163
Rückgabeparameter 85
Rückschlagventil 110
Ruhezustand 357

S
Schaltplan lesen 156

Schieberegister 207, 217
Schleife 78
Schrittmotor 49, 314, 316, 319, 325
Schrittmotortreiber 317, 318
Schrumpfschlauch 132
SCLK 276
Seitenschneider 129
Sensor 289, 290, 326
Serial.print() 93
Serielle LED-Lichterkette 246
Serielle Schnittstelle 93
Serieller Monitor 92
Servomotor 48
Setup-Bereich 70
Shields 7, 45
Shutdown 396
Sieben-Segment-Anzeige 236
Silberdraht 132
SIM-Karte 199
Slave 276
SMD-Gehäuse 21
SMS-Alarmanlage 196
Source Code 58
Spannung 101
  einstellen 150
  messen 150
Spannungsteiler 123, 177
SPI-Protokoll 275
Spule 299
SSH 334
Stahlwolle 131
Stand-by 358
Stoppuhr 236
String 89
Strom 101
Stromkreis 105
Strommessung 151

T
Temperatursensor 169
Test 79
Texteditor 58
TFT 274
TFT-Display 307
TFT-Display-Shield 26
Thermistor 171, 174
Transistor 114, 221
TVZapPro 179

U
Überdruck 111
UND 81
Unterdruck 111
Ur-Arduino 7
USB-Schnittstelle 92
USB-Serial-Light-Adapter 52, 346
UVC 340

V
Variable 74
Variablentypen 76
Verstärker 115
Visual Studio 380
Vorwärtsspannung 207
Vorwiderstand 207, 220

W
WAHR 81
wartezeit 76
Wasserpumpe 302
Watchdog 360
Wearables 346
Webcam 339
Webserver 288
Wedelstab 205
WEP 51
Wetterstation 287
Widerstand 105
  Farbcode 106
  Parallelschaltung 122
  Reihenschaltung 120
Widerstandsmessung 149
Wi-Fi-Shield 50
Windows 59, 386
Wiznet-W5100-Chip 48
WLAN 50, 288
WPA2 51

X
XBee-Modul 38

Y
Yún 330

Z
Zange 129
Zeichenkette 91

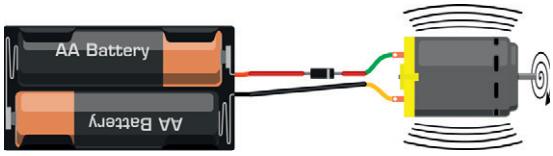
PHILIP CAROLI / CHRISTIAN CAROLI

ARDUINO™ HANDBUCH

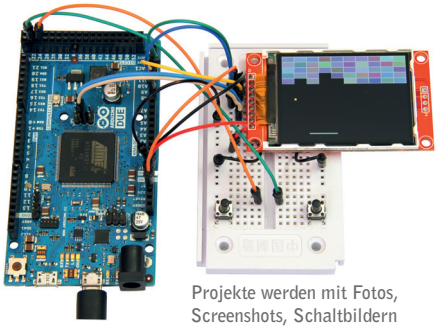
Internet der Dinge, Physical Computing – Umgebungseinflüsse sammeln, verarbeiten und darauf reagieren. Der Arduino™ ist prädestiniert dafür, egal ob digital oder analog. Erfahren Sie in diesem Buch alles, um den Arduino™ als Schaltzentrale in Ihren Projekten einzusetzen: ob Hausautomation, Roboter oder Wetterstation – Sensoren sind immer dabei.

Aus dem Inhalt:

- Arduino™-Platinen und -Shields im Überblick
- Sketches entwickeln
- Elektronikwissen für eigene Projekte
- Analoger Temperatursensor
- Lichtschranke
- SMS-Alarmanlage
- LED-Lichterkette
- Stoppuhr mit Sieben-Segment-Anzeige
- Rotationsmonitor
- Wetterstation
- Arduino Robot™
- Arduino Yún™ nutzen
- Arduino™ im Selbstbau



Elektronikwissen wird einsteigerfreundlich mit Schaubildern erklärt.



Projekte werden mit Fotos, Screenshots, Schaltbildern und Quellcode beschrieben.

Den Einstieg meistern: Hardware, Software, Programmierung und Elektronik

Für jedes Projekt den passenden Arduino™: Im Einführungskapitel lesen Sie, welches der Boards am besten zu Ihrem Projekt passt. Danach erfahren Sie, wie Sie die Arduino™-IDE installieren, und lernen die notwendigen C-Grundlagen. Da Arduino™-Projekte viel mit Elektronik zu tun haben, befasst sich ein eigenes Kapitel mit Elektronikgrundlagen – die wichtigen Themen Löten und die Nutzung eines Multimeters fehlen dabei nicht.

22 Projekte zeigen die Vielfalt von Arduino™

Anhand von 22 Praxisprojekten zeigen die Caroli-Brüder, wie Sie die verschiedenen Arduinos in eigenen Projekten nutzen. In jedem Projekt wird die notwendige Theorie, wie z. B. die Funktionsweise des GSM-Protokolls, vermittelt und das Projekt nachvollziehbar über Fotos, Screenshots, Schaltbilder und Quellcode dokumentiert. Die Quellcodes müssen Sie nicht abtippen, sie stehen kostenlos zum Download zur Verfügung.

Der komplette Quellcode aus dem Buch auf www.buch.cd

9 783645 603164

Besuchen Sie unsere Website
www.franzis.de

FRANZIS