



Leseprobe

Robert Jänisch, Jörn Donges

Mach was mit Arduino!

Einsteigen und durchstarten mit Drum Machine, Roboterauto & Co.

ISBN (Buch): 978-3-446-45128-5

ISBN (E-Book): 978-3-446-45258-9

Weitere Informationen oder Bestellungen unter

<http://www.hanser-fachbuch.de/978-3-446-45128-5>

sowie im Buchhandel.

Inhaltsverzeichnis

1	Einführung	1
1.1	Maker – die Erfinder von morgen	1
1.2	Was dich in diesem Buch erwartet	3
1.3	Wie dieses Buch aufgebaut ist	4
2	Willkommen in der Arduino-Welt!	7
2.1	Was ist überhaupt der Arduino?	7
2.2	Los geht's! Installation der Arduino-Software	8
2.2.1	Die Arduino-Software downloaden	9
2.2.2	Den USB-Treiber unter Windows installieren	9
2.3	Die Entwicklungsumgebung starten und den ersten Sketch übertragen	10
3	Dein erster Schaltkreis	13
3.1	Schaltungsaufbau mit dem Breadboard (Steckbrett)	15
3.1.1	Aufbau	16
3.1.2	Versorgungsspannung	17
3.1.3	Einschränkungen	17
3.2	Vom Programm zur Schaltung: Hardwareentwicklung mit dem Arduino	17
3.3	Dein erster Stromkreis	18
3.4	„Es werde Licht!“ Eine LED zum Leuchten bringen	19
3.5	Jetzt nehmen wir Kontakt auf: Ein- und Ausgänge am Arduino	20
3.6	Wir tasten uns heran: Eine LED per Taster steuern (Teil 1)	21
3.7	Morsen mit dem Arduino: Eine LED per Taster steuern (Teil 2)	22
3.8	If-Abfragen erstellen: Eine LED per Taster steuern (Teil 3)	23
3.9	Ein Taster, zwei Wirkungen: Eine LED ein- und ausschalten	24

4	Der Schlüssel zum Verstehen aller Schaltungen	27
4.1	Der Schaltplan – die abstrakte Essenz der Schaltung	28
4.2	Spannung, Strom, Widerstand – das Dreigespann der Elektrotechnik	29
4.3	Bitte parallel in Reihen aufstellen! Das Gesetz der Reihenschaltung	31
4.4	Das Multimeter – ein Multitalent für Strom, Spannung und Widerstand	33
4.5	Schaltplanentwicklung und -zeichnung mit Fritzing	34
4.5.1	Die Steckplatinen-Ansicht	35
4.5.2	Die Schaltplan-Ansicht	36
4.5.3	Die Platinen-Ansicht	37
4.5.4	Die Code-Ansicht	38
5	Eine Ampel mit Tag- und Nachtschaltung	39
5.1	Arduino-Ampel vs. reale Ampelsteuerung	40
5.2	Die Ampel zeigt grün für den Arduino: eine Tagschaltung programmieren	41
5.3	Nachts sind alle Ampeln gelb: eine Nachtschaltung programmieren	44
5.4	Ein Spannungsteiler in Aktion: Messdaten mit einem Analog-Digital-Wandler auslesen	45
6	Eine Weltzeituhr mit Alarmfunktion	51
6.1	Das LCD-Display HD44780 anschließen	51
6.2	Text auf dem Display darstellen	56
6.3	Strings oder Rechnen mit Wörtern: Der Arduino als Digitaluhr (Teil 1)	58
6.4	Was schlägt die Stunde? Der Arduino als Digitaluhr (Teil 2)	59
6.5	Bits und Bytes bis zum Überlaufen: Der Arduino als Digitaluhr (Teil 3)	61
6.6	Wie Funktionen funktionieren: Der Arduino als Digitaluhr (Teil 4)	62
6.7	Ein wenig Zeitrechnung muss sein: Der Arduino als Digitaluhr (Teil 5)	62
6.8	Der Arduino als Weltzeituhr (Teil 1)	64
6.9	Arrays – die virtuellen Sortimentskästen: Der Arduino als Weltzeituhr (Teil 2)	65
6.10	New York, Rio, Tokyo: Der Arduino als Weltzeituhr (Teil 3)	65
6.11	Jetzt wird der Arduino laut: Weltzeituhr mit Alarmfunktion	67
7	Eine Mini-Wetterstation mit Analoganzeige	73
7.1	Der Temperatur- und Luftfeuchtigkeitssensor DHT11	74
7.2	Serielle Info für die Fehlersuche: Einsatz des seriellen Monitors	75
7.3	Jetzt kann das Wetter kommen! Aufbau der Wetterstation	78
7.4	Statusmeldungen des DHT11-Sensors ausgeben	80

7.5	Die case-Abfragetechnik: DHT11-Fehlercodes, Temperatur und Luftfeuchtigkeit ausgeben	81
7.6	Ein Statustaster gratis! Statusabfrage mit dem Reset-Taster des Arduino ...	81
7.7	Messuhr mit Stil: Analoge Temperaturanzeige	82
7.8	Die Bauteile für die Temperaturanzeige: Servo und Potentiometer	83
7.9	Das Potentiometer: Dateneingabe auf analoge Weise	84
7.10	Der Servo-Motor: Arme und Beine für den Arduino	85
7.11	Wir bauen ein Thermometer: Skala und Zeiger für die Temperaturanzeige	89
7.12	Das Thermometer ist fertig: Berechnung der Temperaturskala	90
7.13	Der Variablentyp float: Umrechnung der Temperatur in Winkelwerte	92
8	Eine temperaturgeregelte Lüftersteuerung	95
8.1	Jetzt kommt Bewegung ins Spiel: ein Gleichstrommotor als Lüfter	96
8.2	Ein Ventil für elektrischen Strom: der Transistor	97
8.3	Spannungsspitzen vermeiden: eine Diode zum Schutz des Arduino	100
8.4	Temperaturmessung mit dem TMP36-Sensor	102
8.5	Alles geregelt dank Arduino: Temperaturregelung und Lüfterschaltung verbinden	103
9	Exkurs: Internet der Dinge (IoT) mit dem Particle Photon	107
9.1	Particle Photon & Co.: Mikrocontroller von Particle.io	108
9.2	Déjà-vu für Arduino-Kenner: Die Parallelen zwischen Arduino und Particle Photon	109
9.3	Den Particle Photon einrichten und einen ersten Sketch übertragen	109
9.3.1	Ohne Particle-Account und Strom geht nichts	110
9.3.2	Ein Multifarbentalent: die Bedeutung der LED-Farben beim Particle Photon	110
9.3.3	Anmeldung per Smartphone: die Particle-App	111
9.3.4	Den ersten Sketch übertragen	112
10	Eine Pflanzenbewässerungsanlage: Kombination von Arduino und Particle Photon	115
10.1	Benötigte Bauteile	116
10.2	Aufbau und Programmierung der Pflanzenbewässerungsanlage	116
10.3	Steuerung der Pflanzenbewässerung übers Internet	118

11	Der Piezoeffekt: Wie du mit dem Sound von 1880 deinen Arduino rockst	127
11.1	Ohne Sound geht nichts: Tonerzeugung mit Piezo-Summer oder Lautsprecher	127
11.2	Do Re Mi Fa So La Ti: eine Tonleiter spielen	129
11.3	Auf das Timing kommt es an: die Tondauer festlegen	130
11.4	Melodiegenerator und Lautstärkeregler	131
11.5	Rechnen mit Tönen: ein Pitch-Regler kontrolliert die Tonhöhe	133
11.6	Auf und ab: eine Melodie in allen Tonarten erklingen lassen	134
12	Echt stark! Eine Verstärkerschaltung mit Transistor für den Arduino	137
12.1	Wir bauen einen Mini-Audioverstärker	137
12.2	Das Grundprinzip der Verstärkung: So funktioniert der Transistor	139
12.3	Stromspeicher und Wechselstrom-Ventil: So funktioniert der Kondensator	140
12.4	Der Kondensator schützt die Ein- und Ausgänge	142
13	Ein Synthesizer aus Arduino und Digital-Analog-Wandler	145
13.1	Von Zahlen zu Spannungen: der Digital-Analog-Wandler	145
13.2	Sag es mit 1 und 0: die Binärdarstellung mit dem Arduino	146
13.3	Mit PORTD die digitalen Pins kontrollieren	147
13.4	Auf die Klangfarbe kommt es an: Sinus-, Rechteck- und Dreieckschwingung	150
13.5	Der Arduino gibt den Takt vor: Änderung der Wellenform	152
13.6	Jetzt wird Sound draus: Änderung der Tonhöhe	153
14	Eine Arduino-Drum Machine	155
14.1	So wird der Arduino zur Drum Machine	155
14.2	Retro-Drum Sound mit 8 bit: Samples für die Drum Machine	158
14.3	Mehrdimensionale Arrays für die Programmierung der Drum Machine	159
15	Ein autonom fahrendes Roboterauto	165
15.1	Empfohlene Starthilfe: ein Roboter-Bausatz	165
15.2	Zusammenbau des Roboter-Bausatzes	166
15.2.1	Schritt 1: Einbau von Chassis und Motoren	167
15.2.2	Schritt 2: Installation des Motortreibers	168
15.2.3	Schritt 3: Einbau des Arduino und des Batteriegehäuses	169

15.2.4	Schritt 4: Vorbereitung des Ultraschall-Sensors	170
15.2.5	Schritt 5: Einbau des Arduino Uno und des Sensor Shields	172
15.2.6	Schritt 6: Einbau des Ultraschall-Sensors	174
15.3	Programmierung des Roboterautos	176
15.3.1	Hinderniserkennung	176
15.3.2	Entfernungsmessung	177
16	Bob, der humanoide Roboter	185
16.1	Humanoide Roboter für alle: das InMoov-Projekt	186
16.2	Wir bauen einen humanoiden Roboter (Teil 1): Organisation ist alles	188
16.3	Wir bauen einen humanoiden Roboter (Teil 2): 3D-Druck und Zusammenbau der Einzelteile	189
16.4	Wir bauen einen humanoiden Roboter (Teil 3): Typische Fehler und wie man sie am besten vermeidet	191
17	Alles, was du für deine Arduino-Projekte über Programmierung wissen musst	195
17.1	Grundstruktur von Arduino-Sketches	195
17.2	Einbinden von Libraries	196
17.3	Schreiben und Auslesen von Daten	196
17.4	Variablen	197
17.4.1	Int/Long-Variablen	197
17.4.2	String-Variablen	198
17.4.3	Float/Double-Variablen	198
17.4.4	Boolean-Variablen	198
17.4.5	Arrays	199
17.5	Serieller Monitor	199
17.6	Abfragen	200
17.6.1	If-Abfragen	200
17.6.2	Case-Abfragen	201
17.7	Schleifen	202
17.7.1	For-Schleifen	202
17.7.2	Do while-Schleifen	202
17.8	Definition eigener Funktionen und Prozeduren	203
17.9	Systemvariablen und Funktionen	204
17.9.1	millis()	204
17.9.2	PORTD	204
17.9.3	tone(Frequenz)	204

17.9.4	sizeof(Variable)	204
17.9.5	delay(T) und delayMicroseconds(t)	204
18	Alles, was du für deine Arduino-Projekte über Hardware wissen musst	205
18.1	Schaltplan	205
18.2	Ohmsches Gesetz	207
18.3	Widerstand	207
18.4	Leuchtdiode (LED)	210
18.5	Potentiometer	212
18.6	Schalter und Taster	213
18.7	Fotowiderstand (LDR)	214
18.8	LCD-Display	214
18.9	Piezo-Töner	215
18.10	Temperatur- und Luftfeuchtigkeitssensor DHT11	216
18.11	Servo-Motor	217
18.12	DC-Motor	218
18.13	Diode	219
18.14	Transistor	219
18.15	Kondensator	220
19	Ausblick: Noch mehr Mikrocontroller und Projektideen	223
19.1	Weitere Arduino-Boards	223
19.2	Weitere Mikrocontroller-Plattformen	224
19.3	Löten und Platinenbau	225
19.4	Ausblick: Musik-Projekte (Audio und Midi)	226
19.5	Ausblick: Messen und Steuern im Haus (Smart Home)	226
19.6	Ausblick: Roboterbau und -programmierung	227
19.7	Vernetze dich! Projektideen teilen	228
	Stichwortverzeichnis	229

6

Eine Weltzeituhr mit Alarmfunktion

In diesem Kapitel werden wir ein Projekt verwirklichen, das einen ganz typischen Einsatz von Mikrocontrollern im Alltag zeigt. Wir bauen eine Digitaluhr mit dem Arduino und erweitern sie im nächsten Schritt um eine Weltzeituhr sowie eine Alarmfunktion. Dabei lernst du ein neues, sehr nützliches Bauteil kennen, das du auch in vielen anderen Projekten gut gebrauchen kannst: das LCD- Display HD44780. Softwareseitig wirst du ein paar neue Programmierkonzepte kennen lernen – allen voran den String-Datentyp und die Verarbeitung von Zeichenketten.



Bild 6.1 Wir bauen eine Weltzeituhr mit Alarmfunktion.

■ 6.1 Das LCD-Display HD44780 anschließen

Das LCD-Display HD44780 kann zwei Zeilen mit jeweils 16 Zeichen darstellen. Es ermöglicht dir, verschiedenste Daten und Werte anzuzeigen, und ist quasi so etwas wie ein kleiner Bildschirm für den Arduino. HD44780 ist eigentlich die Bezeichnung für den Control-

lerchip, der die Daten vom Arduino annimmt und damit die Anzeige ansteuert. Das Display hat 16 Eingangspins, von denen acht für die Übertragung der Anzeigedaten vorgesehen sind. Die restlichen acht Pins dienen der Steuerung, Spannungsversorgung und Beleuchtung des Displays.

Die Pinbelegung dieses Bauteils folgt einem weit verbreiteten Standard und wird von der Arduino-Programmiersprache bereits mit vielen Funktionen unterstützt, die das Darstellen von Daten auf dem Display zum Kinderspiel machen. Die meisten Displays, die du im Elektronikhandel für wenige Euro kaufen kannst, sind schon mit Bausteinen bestückt und können direkt am Arduino betrieben werden. Am besten besorgst du dir ein Modul mit angelöteter Stiftleiste, die du direkt ins Breadboard stecken kannst.

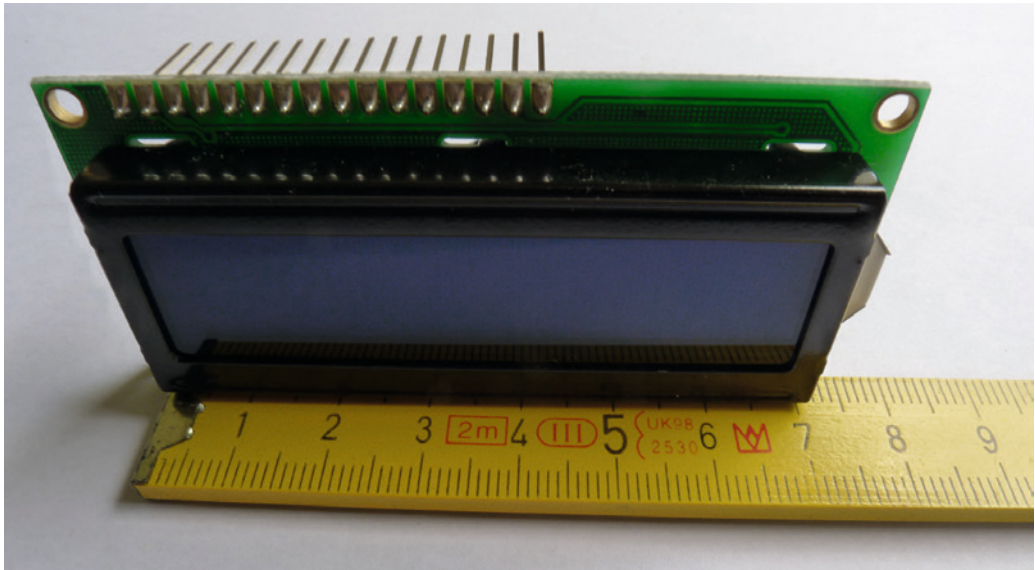


Bild 6.2 LCD-Display HD44780

Um das LCD-Display HD44780 zu verdrahten und an den Arduino anzuschließen, brauchst du eine ganze Menge von Drahtstückchen. Bei voller Beschaltung belegt das Display neun Pins am Arduino. Außerdem musst du auf dem Breadboard noch einige Pins miteinander verbinden. Nun ist der Zeitpunkt gekommen, zu dem du aus einer Rolle Klingeldraht kurze Verbindungsstücke schneidest, so wie es in Kapitel 3 erklärt wurde. Es gibt im Elektronikhandel auch fertige Sets sogenannter Jumperkabel, die du verwenden kannst.

Die Pins am Display werden von links nach rechts gezählt. An die ersten beiden Pins wird die 5 V-Versorgungsspannung angelegt. Pin 3 ist für die Einstellung des Kontrasts verantwortlich. Manche Schaltungen, die du im Internet findest, sehen hier einen veränderbaren Widerstand vor, um den Kontrast der Anzeige einstellen zu können. Normalerweise reicht es aber aus, den Pin einfach auf GND zu legen. Damit erzielt man meist schon eine gute Darstellung. Solltest du dennoch Probleme mit einer zu blassen Anzeige haben, kannst du

Die folgende Tabelle zeigt die Verdrahtung noch einmal in der Übersicht.

LCD-Pin	LCD-Bezeichnung	Arduino-Anschluss	Funktion
1	VSS	GND	Spannung –
2	VDD	+ 5 V	Spannung +
3	V0	GND	Kontrast (0 – 5V)
4	RS	DIGITAL 12	Register Select
5	R/W	GND	Read/Write
6	E	DIGITAL 11	Enable
7–10	DB0–DB3	nicht verwendet	Datenpins
11	DB4	DIGITAL 5	Datenpins
12	DB5	DIGITAL 4	Datenpins
13	DB6	DIGITAL 3	Datenpins
14	DB7	DIGITAL 2	Datenpins
15	A	+ 3.3 V	LED Backlight +
16	K	GND	LED Backlight –

Bild 6.4 zeigt den Schaltplanaufbau für das LCD-Display.

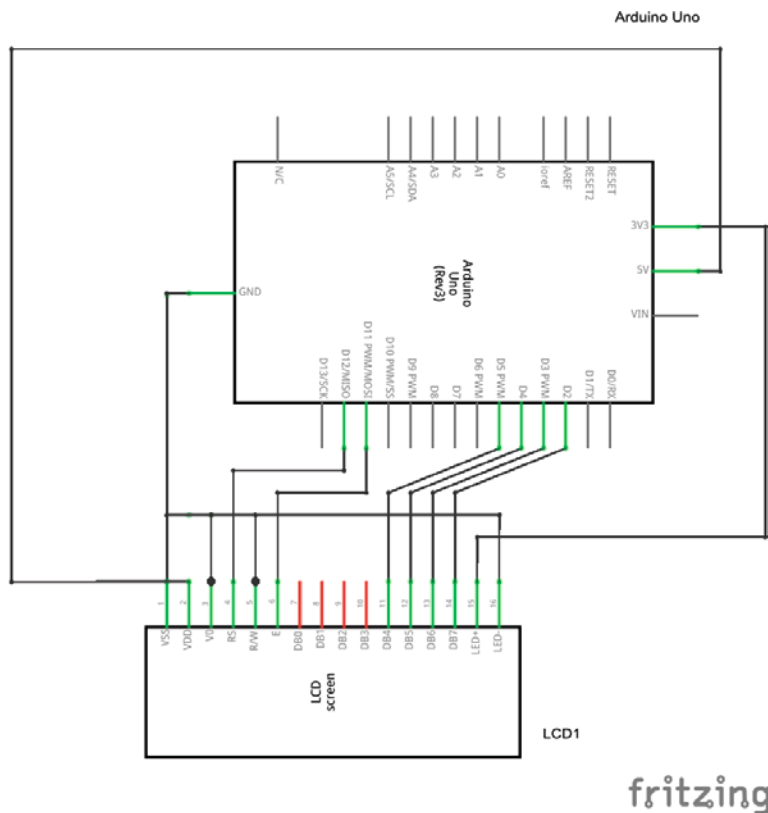


Bild 6.4 Schaltplan für den LCD-Display-Aufbau

Bild 6.5 bis Bild 6.7 zeigen, wie du den Arduino an das LCD-Display anschließt.

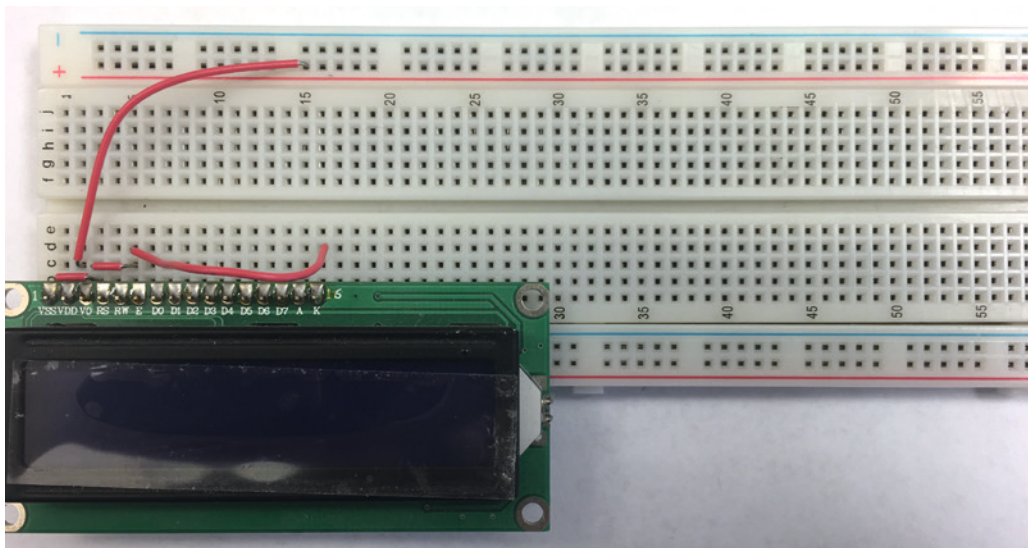


Bild 6.5 Die ersten Anschlüsse auf dem LCD-Display

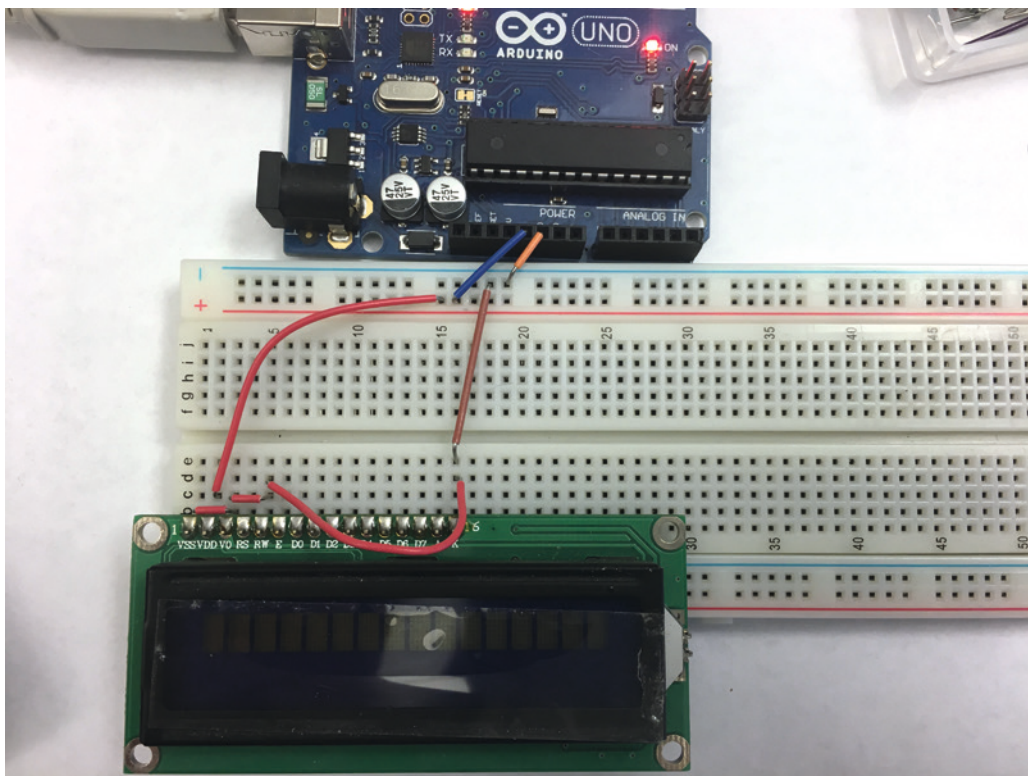


Bild 6.6 Jetzt schließt du den Arduino an.

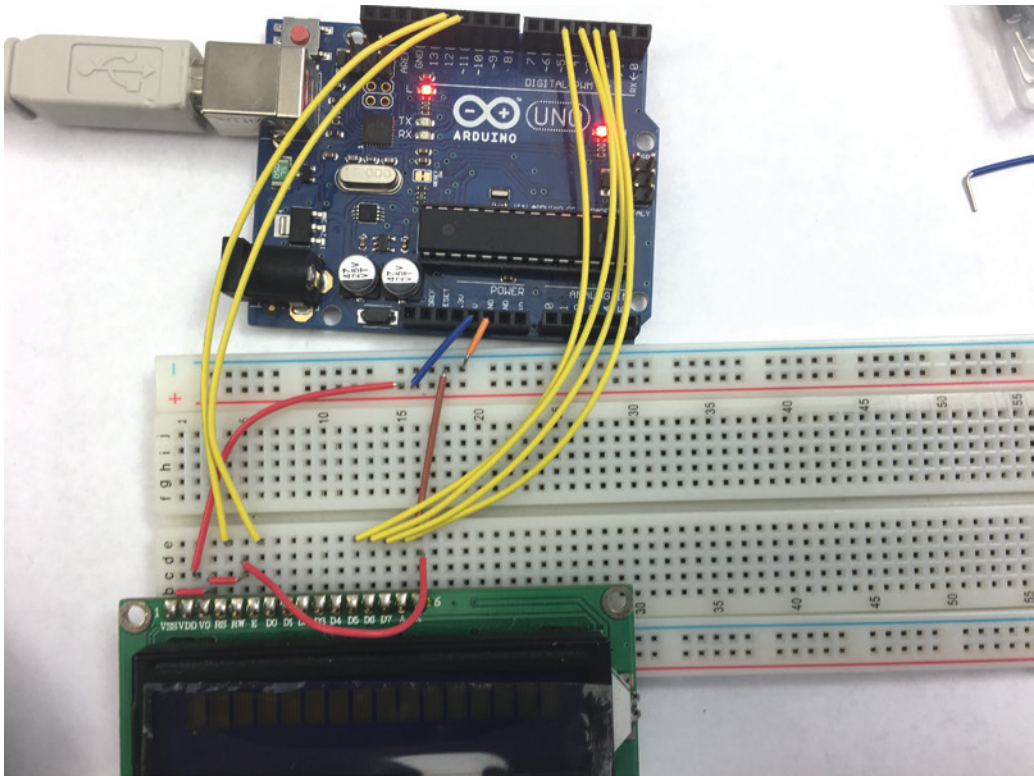


Bild 6.7 Das LCD-Display ist nun komplett angeschlossen.

■ 6.2 Text auf dem Display darstellen

Hast du das LCD-Display erfolgreich angeschlossen? Prima, dann können wir loslegen. Dein Aufbau wird wahrscheinlich so ähnlich aussehen wie in Bild 6.8. Mit Folgenden wollen wir nun das Display per Software ansteuern.

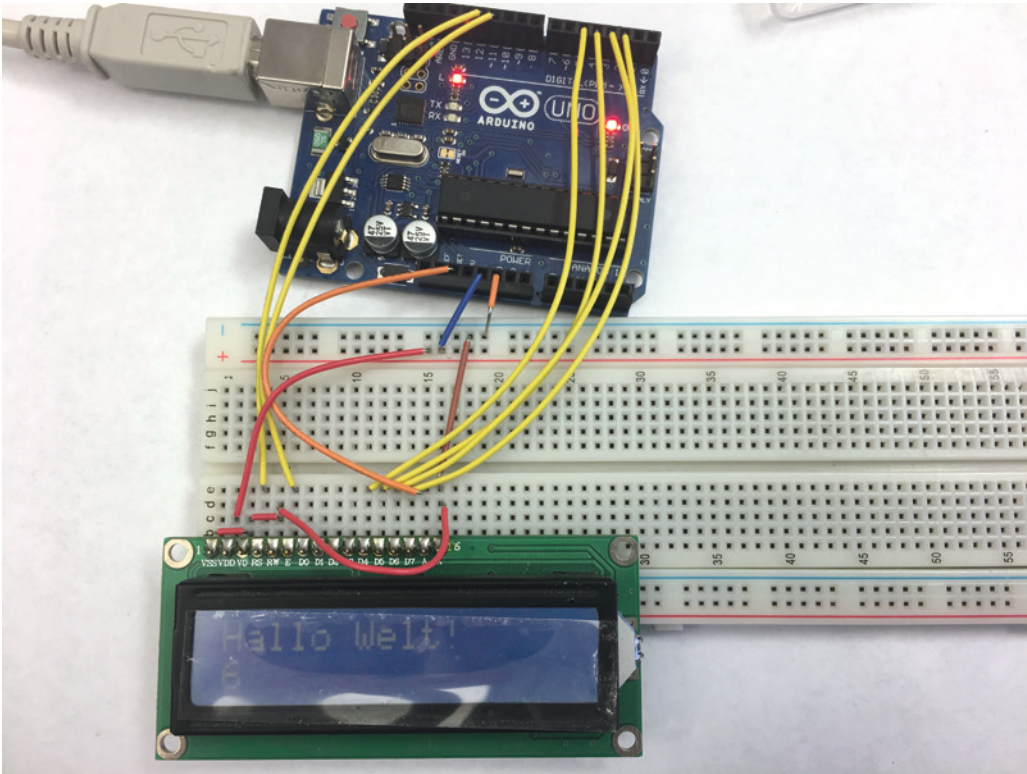


Bild 6.8 „Hallo Welt!“

Bevor wir uns um die Funktionen der Uhr kümmern, testen wir das Display erst einmal mit einem Sketch, der eine Meldung auf dem Display ausgibt. Wir entscheiden uns für das klassische „Hallo Welt“, aber du kannst natürlich einen beliebigen Text wählen.

```

1 // Testsketch für LCD-Display
2 #include <LiquidCrystal.h>
3 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
4 void setup() {
5   // LCD mit 16 Spalten und zwei Zeilen initialisieren
6   lcd.begin(16, 2);
7   // Text ausgeben
8   lcd.print("Hallo Welt!");
9 }
10 void loop() {
11   // Cursor in der zweiten Zeile an erster Stelle
12   positionieren
13   lcd.setCursor(0, 1);
14   // Gib die Anzahl der Sekunden seit dem letzten Reset
15   aus:
16   lcd.print(millis()/1000);
17 }

```

In diesem Code sind einige neue Elemente enthalten, die wir im Folgenden kurz durchgehen werden.

- **Zeile 2:** Die Anweisung `#include <LiquidCrystal.h>` dient dazu, den Inhalt einer anderen Datei in deinen Code einzubinden. Der Compiler behandelt das so, als ob der Text aus der Datei direkt an dieser Stelle stehen würde. Meist verwendet man diese Anweisung, um externe Programmbibliotheken einzubinden. In diesem Fall binden wir die Funktionen zur Ansteuerung des LCD-Displays ein, die in der Datei `LiquidCrystal.h` stehen.
- **Zeile 3:** Hier erzeugen wir ein Objekt vom Typ `LiquidCrystal` mit Namen `lcd`. Wir übergeben der Software mit `LiquidCrystal lcd(12, 11, 5, 4, 3, 2);` gleich das Pin-Layout unserer Schaltung. Immer wenn wir in Zukunft auf das Display schreiben, passiert das über dieses Objekt. Eine ausführliche Dokumentation findest du unter dem Stichwort „LiquidCrystal Library“ auf der Arduino-Website: <https://www.arduino.cc/en/pmwiki.php?n=Reference/LiquidCrystal>
- **Zeile 6:** Hier initialisieren wir das Display, d. h., wir legen die Zeilen- und Spaltenzahl fest und löschen die Anzeige.
- **Zeile 8:** Mit `lcd.print()` schicken wir eine Zeichenkette auf das Display.
- **Zeile 13:** Die Funktion `lcd.setCursor()` erhält Zeile und Spalte als Parameter und legt fest, an welcher Stelle auf dem Display der nächste `print`-Befehl ausgegeben wird.

Im `loop`-Block des Testsketches nutzen wir den Arduino schon als Timer: Die Funktion `millis()` liefert zurück, wie viele Millisekunden seit dem letzten Reset oder Einschalten des Arduino vergangen sind. Wir schicken diesen Wert aufs Display und teilen ihn durch 1000, das heißt, wir haben einen Sekundenzähler. Diesen Wert werden wir als Basis für das Programmieren der Uhr-Funktionen verwenden.

■ 6.3 Strings oder Rechnen mit Wörtern: Der Arduino als Digitaluhr (Teil 1)

Nachdem das Display nun einsatzbereit ist, kommen wir zu unserem eigentlichen Vorhaben. Wir möchten zunächst einmal die aktuelle Uhrzeit auf dem Display darstellen – und zwar wie bei einer Digitaluhr üblich im Format HH:MM:SS (je zwei Stellen für Stunde, Minute und Sekunde, getrennt durch Doppelpunkte).

Dazu bedienen wir uns eines neuen Datentyps, der Zeichenketten abspeichern und verarbeiten kann: der `String`-Typ. Eine `String`-Variable speichert eine Zeichenkette. Du hast bereits in Kapitel 5 gelernt, dass man Zahlen als Variablen unter einem bestimmten Namen abspeichern kann und dann im Sketch unter diesem Namen auf sie zugreifen und

mit ihnen rechnen kann. Dazu haben wir den Datentyp `int` verwendet. Diese Abkürzung steht für Integer. Solche Variablen sind immer ganzzahlige Werte. Wir können Texte oder Zeichen aber auch als Variable abspeichern. Diese sind dann vom Typ `String`. Man kann sie – genauso wie Zahlen – in Funktionsaufrufen verwenden oder Operationen damit ausführen. Du kannst das ausprobieren, indem du im Testsketch den `lcd.print()`-Befehl ersetzt:

```
1 String Teil1="Hallo";  
2 String Teil2="Welt!";  
3 lcd.print(Teil1+" "+Teil2);
```

Im Ergebnis steht wieder „Hallo Welt!“ auf dem Display, doch wir haben die Ausgabe aus zwei `String`-Variablen und einer `String`-Konstanten zusammengesetzt. Mit dem `+`-Operator kannst du beliebige Ausgabezeilen aus konstanten Textstückchen und `String`-Variablen zusammensetzen. Das wollen wir in Abschnitt 6.4 für die Anzeige der Uhrzeit nutzen.

■ 6.4 Was schlägt die Stunde?

Der Arduino als Digitaluhr (Teil 2)

Im Folgenden wirst du den Sketch für die digitale Anzeige der Uhrzeit kennenlernen. Bevor wir die einzelnen Funktionen genauer durchgehen, kannst du den ganzen Code erst einmal per Copy & Paste in die Arduino-Entwicklungsumgebung kopieren und dort abspeichern.

Dann setzt du in Zeile 40 (siehe folgender Code) die aktuelle Uhrzeit ein und schickst den Sketch an den Arduino. Wir müssen dem Programm die Startzeit mitgeben, da der Arduino keine eingebaute Uhr hat wie zum Beispiel dein PC oder Laptop. Außerdem kannst du in Zeile 49 eintragen, was in der ersten Zeile angezeigt werden soll (zum Beispiel deine Stadt). Auf dem Display deines Arduino ist nun eine Zeitanzeige zu sehen (Bild 6.9).

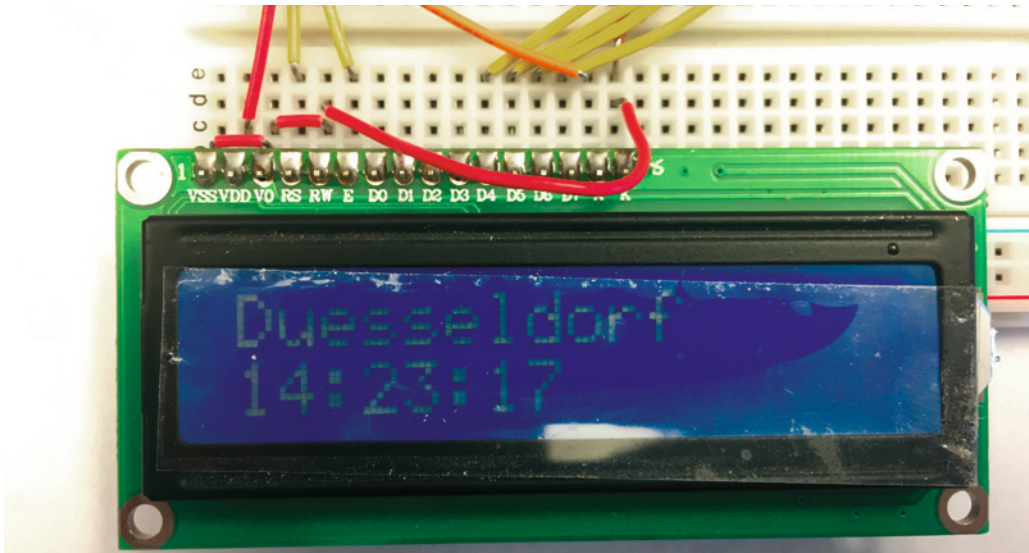


Bild 6.9 Ort und Zeit werden auf dem LCD-Display angezeigt.

Dies ist der komplette Code für die Programmierung der Zeitanzeige:

```

1  // Arduino als Digitaluhr mit dem LCD-Display
2
3  #include <LiquidCrystal.h>
4
5  String number_to_string(long secs, int DIV, int MOD){
6  // Rechnet die Sekundenzahl um und wandelt das Ergebnis
7  // in einen String
8  // DIV=3600: Rechnet Sekunden in Stunden um
9  // DIV=60: Rechnet Sekunden in Minuten um
10 // Der Wert MOD gibt an, ab welcher Zahl die Anzeige
11 // wieder auf Null springt
12 // (Normalerweise 12 oder 24 beim Stundenzähler, 60 bei // Minuten und Sekunden
13
14 String Str;
15 unsigned long number=secs/DIV%MOD;
16
17 // Bei einstelliger Zahl noch eine Null voranstellen,
18 // sonst direkt umwandeln
19 if(number<10) Str='0'+String(number);
20 else Str=String(number);
21 return Str;
22 }
23
24 String timeString(unsigned long secs){
25 // Wandelt die Sekundenzahl secs um in einen String im
26 // Uhrzeit-Format
27 return
28 number_to_string(secs,60*60,24)+':'+number_to_string(secs,60,60)+':'+
29 }
30
31 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

```

```

32
33 // Unter diesem Kommentar die aktuelle Uhrzeit einsetzen.
34 // Die Variable Offset ist die Anzahl der Sekunden von 0
35 // Uhr bis zum gegenwärtigen Zeitpunkt.
36 // Sie wird zu den Sekunden seit Start addiert. Dadurch
37 // zeigt der Arduino beim Start nicht 0:00:00 an, sondern // Uhrzeit, die hier definiert
   wird:
38
39 // -----UHRZEIT EINSETZEN-----
40 long init_stunde=17; long init_minute=35; long init_sekunde= 0;
41 // -----
42 unsigned long Offset=init_stunde*3600+init_minute*60+init_sekunde;
43
44
45 void setup() {
46 // LCD mit 16 Spalten und 2 Zeilen initialisieren
47 lcd.begin(16, 2);
48 // Text für erste Zeile ausgeben
49 lcd.print("Hamburg");
50 }
51
52 void loop() {
53 // Anzahl Sekunden seit dem letztem Reset berechnen
54 unsigned long secs=millis() /1000 ;
55
56 // Die mit der Funktion timeString aus den Sekunden und
57 // der Startzeit berechnete Uhrzeit ausgeben
58 lcd.setCursor(0, 1);
59 lcd.print(timeString(secs+Offset));
60 }

```

■ 6.5 Bits und Bytes bis zum Überlaufen: Der Arduino als Digitaluhr (Teil 3)

Sehen wir uns nun den Code zur Programmierung der Zeitanzeige aus Abschnitt 6.4 einmal genauer an. Als Erstes fällt auf, dass wieder ein neuer Datentyp benutzt wird – und zwar `unsigned long`. Hierbei handelt es sich nur um ganze Zahlen, genau wie beim Typ `int`. Allerdings bedeutet `long`, dass doppelt so viel Speicherplatz für die Variable reserviert wird. Statt den 16 Bit einer `int`-Variable besitzt eine `long`-Variable 32 Bit. Wir benötigen die `long`-Variable, weil wir die Sekunden seit dem Start des Programms zählen wollen. Eine `int`-Variable kann maximal Werte bis $2^{16} = 65536$ annehmen. Diese Anzahl von Sekunden ist bereits nach 18 Stunden gezählt. Dann würden die Variablen überlaufen und wieder auf null springen. Mit 32-Bit-Variablen passiert das erst nach 50 Tagen. Daher müssen wir für diese Anwendung die längeren, speicherintensiven `long`-Typen verwenden. Aus demselben Grund sagen wir dem Compiler mit dem Schlüsselwort `unsigned`, dass wir keine negativen Zahlen verwenden wollen, denn auch dadurch erweitert sich der Bereich bis zum Überlauf.



Leider können wir unsere Ausführungen zu Variablen und Datentypen an dieser Stelle nicht weiter vertiefen, da dies dem Rahmen des Buches sprengen würde. Nutze die Möglichkeit und vertiefe dein Wissen in Einstiegsbüchern oder Programmierkursen zu den Sprachen C und C++.

■ 6.6 Wie Funktionen funktionieren: Der Arduino als Digitaluhr (Teil 4)

In Zeile 5 des Codes aus Abschnitt 6.4 definieren wir übrigens eine eigene Funktion. Das ist ein wichtiges Konzept bei der Programmierung. Bislang haben wir immer bereits vordefinierte Funktionen verwendet und aufgerufen. Wir können aber auch eigene Funktionen definieren und auf diese Weise sich wiederholende Aufgaben effizient von einem Funktionsaufruf erledigen lassen.

In unserem Falle definieren wir die Funktion `number_to_string()`, die eine Zahl (eine `int`- oder `long`-Variable) in einen `String` umwandelt. Daraus werden wir die Uhrzeit auf dem LCD-Display zusammensetzen. Es wird immer zuerst der Rückgabebetyp genannt, dann der frei wählbare Funktionsname und in Klammern werden die Parameter angegeben, die an die Funktion übergeben werden sollen (immer mit Typangabe und Name). Das sieht wie folgt aus:

```
String number_to_string(long secs, int DIV, int MOD)
```

Danach folgt ein Block in geschweiften Klammern. Dort wird implementiert, was diese Funktion macht.

■ 6.7 Ein wenig Zeitrechnung muss sein: Der Arduino als Digitaluhr (Teil 5)

Wir bleiben bei der Analyse des Codes aus Abschnitt 6.4. Für die Funktion `number_to_string` schauen wir uns erst einmal Zeile 15 an:

```
unsigned long number=secs/DIV%MOD;
```

Die Zeile sieht etwas kryptisch aus, aber wir werden sie dir Schritt für Schritt erklären. Wir nehmen die Sekundenzahl in der Variablen `secs` und teilen sie erst einmal durch den Wert `DIV`, der beim Aufruf der Funktion mit übergeben wurde. Damit kann man entweder Sekunden in Minuten oder Sekunden in Stunden umrechnen, je nachdem, was man für `DIV` einsetzt.

Danach folgt ein neuer Operator: Das Zeichen `%` nennt man Modulo-Operator. Das klingt vielleicht etwas kompliziert und mathematisch, aber eigentlich ist es ganz einfach. Die Zahl, die hinter dem `%` steht, ist wie eine Grenze, die nicht überschritten wird. Genauer gesagt ergibt die Modulo-Operation den Rest, der beim Teilen entsteht. Für den Sekundenzähler benutzen wir z. B. `%60`. Dies bewirkt, dass nach 59 Sekunden nicht 60 kommt, sondern der Zähler wieder bei 0 anfängt. Denn 60 geteilt durch 60 geht glatt auf und der Divisionsrest ist 0. Bei 120 passiert das Ganze wieder usw. Anstatt die Sekunden immer weiterzuzählen, springen wir also immer bei 60 wieder auf 0 zurück – genau wie eine Uhr eben. Das erledigt der Modulo-Operator ganz automatisch für uns.

In den Zeilen 19 und 20 sorgen wir nun dafür, dass der Rückgabe-String immer genau zwei Zeichen hat. Bei Zahlen kleiner als 10 ist er jedoch einstellig. Dann setzen wir einfach eine „0“ vorne dran.

Das in der Variablen `Str` gespeicherte Ergebnis wird dann in Zeile 21 an den Funktionsaufrufer zurückgegeben. Wir können nun überall im Code auf diese Funktion zurückgreifen, indem wir einfach `number_to_string()` mit den entsprechenden Werten aufrufen.

In Zeile 24 definieren wir noch eine weitere Funktion. Diese Funktion ist die wichtigste des ganzen Programms. Sie nimmt die absolute Sekundenanzahl als Grundlage und wandelt diese in eine Uhrzeit im HH:MM:SS-Format um. Dies erfolgt, indem ein String mit dem `+`-Operator zusammengesetzt wird, und dreimal die Funktion `number_to_string()` aufgerufen wird, jeweils mit den Parametern für Stunden, Minuten und Sekunden.

In Zeile 42 definieren wir einen Wert namens `offset`, den wir noch auf die Sekundenanzählung des Arduino aufschlagen. Da der Arduino mit 0 zu zählen beginnt, wäre die angezeigte Uhrzeit beim Start immer 00:00:00, was wir nicht möchten. Stattdessen soll hier die aktuelle Zeit angezeigt werden. Dazu behelfen wir uns mit einem Trick: Wir addieren die Zahl der Sekunden, die seit der Anzeige „0:00:00“ vergangen sind, zum Timer-Wert hinzu.

Den Rest wirst du jetzt sicherlich schnell verstehen. Im `loop`-Block nehmen wir die Laufzeit des Arduino aus der Funktion `millis()`, berechnen daraus die absolute Sekundenzahl, packen unseren `offset` drauf und schicken das Ganze an die Funktion `timeString()`, die einen String im Uhrzeitformat daraus macht. Dieser String wird mittels `lcd.print()` ans LCD-Display geschickt.

Stichwortverzeichnis

Symbole

3D-Drucker 186

A

Alarmfunktion 51, 67, 203

Ampel 39f., 210, 214

Ampelschaltung 40

Ampere 30

Analoganzeige 73, 94

Analog-Pin 45

Arduino 7

Arduino Mega 186

Arduino-Plattform 7

Arduino Uno 8

Arrays 65, 199

Atmel 7

Ausgang 20

B

Belichtungsmesser 49

Bibliotheken 196

Bits 61

Bob (humanoider Roboter) 185

Breadboard 15

Bytes 61

C

case-Abfragen 201

Chip 7, 13, 224

Cura 190

D

Datentypen 198

DC-Motor 96, 218

Debugging 76

delay 25, 43, 159

delay() 12, 204

delayMicroseconds() 204

DHT11 73f., 81, 216

DHT22 104

Digital-Analog-Wandler 5, 145

Digital-Pin 45

digitalRead 23

digitalRead() 196

Digitaluhr 51, 58

digitalWrite 23

digitalWrite() 196

Diode 219

Display 56

Double 198

do while-Schleife 202

Dreieckswelle 156

Drohnen 2

Drum Machine 155

E

Eingang 20

Electron 108

Elektrotechnik 17, 29

else-Schleife 23

Entprellen 25

Entwicklungsumgebung 10, 113

F

FabLab 2

Fehlercodes 81

float 92, 198

Fotowiderstand 40, 214

Frequenz 129

Fritzing 34

– Code-Ansicht 38

– Platinen-Ansicht 37

– Schaltplan-Ansicht 36

Funktionen 203

G

Gary Fisher 1

Gleichstrommotor 96

Gordon Moore 14

H

Hallo Welt! 57

Hardware 1

Hardwareentwicklung 17

HD44780 51

HIGH 23

humanoider Roboter 185

I

if-Abfrage 23, 200

if-Schleife 23

include 58

– LiquidCrystal.h 58

InMoov 186, 190
 int 59, 61 f.
 Integer 59
 integrated circuits 13
 Intel 14
 Internet der Dinge 107
 Internet of Things 107
 Int/Long-Variablen 197
 IoT (Internet of Things) 107
 I (Strom) 30

K

Kinect 186
 Klingeldraht 14
 Kondensator 220

L

Lautsprecher 127
 Lautstärkeregler 131, 143
 LCD-Display 51, 214
 lcd.print() 58
 lcd.setCursor() 58
 LDR (Fotowiderstand) 40, 214
 LED 19, 40, 210
 Leiterplatte 37
 Leuchtdiode 29
 Libraries 196
 Linux 9
 long 61 f.
 loop 12
 LOW 23
 Lüfter 96
 Lüftersteuerung 95
 Luftfeuchtigkeitssensor 73 f., 216

M

Maker 1
 Maker-Community 3
 Melodie 134
 Melodiegenerator 131
 Messdaten 40
 Messdaten auslesen 76
 Microchip 7

Mikrocontroller 7
 – Arduino 7
 – Electron 108
 – Photon 107
 millis() 204
 Minuspol 29
 Mooresches Gesetz 14
 Morsen 22
 Multimeter 33
 myServo.write() 88

N

Nachtschaltung 44
 noTone() 67
 number_to_string() 62

O

Ohm 30
 Ohmsches Gesetz 30, 207
 Operator 59
 Ottomar von Mayenburg 1

P

Parallelschaltung 32
 Particle 107, 115
 Particle-App 111
 Particle-Cloud 120
 Particle.function() 120
 Particle.io 108
 Particle.variable() 120
 Pflanzenbewässerungsanlage 115
 Photon 107
 Piezo 215
 Piezoeffekt 127
 Piezo-Lautsprecher 68
 Piezo-Summer 127
 Pin 45
 Pitch-Regler 133
 Platine 34
 Platinenlayout 34
 Pluspol 29
 PN2222-Transistor 98 f.
 PORTD 204

Potentiometer 83, 133, 157,
 212
 Prozeduren 203

R

Ranga Yogeshwar 1
 Rapid Prototyping 16
 Rechteckwelle 67, 156
 Regelkreise 95
 Reihenschaltung 31 f.
 Roboter 165, 185
 Roboterauto 165
 Roboter-Bausatz 165
 R (Widerstand) 30
 RX 119

S

Schalter 213
 Schaltkreis 13
 Schaltlitze 14
 Schaltplan 28, 205
 Schleife 202
 – do while 202
 – else 23
 – for 202
 – if 23
 SD-Karte 190
 Serial.begin() 76
 serialEvent() 119
 Serial-Funktion 76
 Serial.print() 76
 serieller Monitor 199
 serielle Schnittstelle 119
 Servo 83, 171
 Servo-Bibliothek 88
 Servo.h 87
 Servo-Motor 85, 217
 setup 12
 Signal 68, 71, 137
 Sinuswelle 156
 sizeof() 204
 Sketch 11, 109, 112, 195
 Smart Home 226
 Smartphone 111

Software 1
Sound 155, 158
Spannungsquelle 29
Spannungsteiler 31, 45
Spannung (U) 30
Steckbrett 15
String 58, 62
String-Variablen 198
Strom (I) 29f.
Stromkreis 13, 18
Stromversorgung 17
Sythesizer 145

T

Tagschaltung 41
Taster 14, 213
Temperaturanzeige 83
Temperaturmessung 102
Temperaturregelung 95
Temperatursensor 73f., 216
Threshold 48

Tinker-App 107
TMP36-Sensor 102
TO-92-Gehäuse 98
Toggeln 24
Tonarten 134
tone() 67
tone(Frequenz) 204
Tonerzeugung 127
Transistor 97, 219
TX 119

U

Übertragungsrate 76
Ultimaker 2 186
Umrechnung 92
unsigned long 61
URI 30
USB-Schnittstelle 76
USB-Treiber 9
U (Spannung) 30

V

Variable 58, 61
Versorgungsspannung 17
void 12

W

Weltzeituhr 51
Wetterstation 73
Wettervorhersage 73
Widerstand (R) 29f., 207
Windows 9
WLAN 110, 118

Z

Zeitrechnung 62