Arduino-Workshops

Eine praktische Einführung mit 65 Projekten

von
John Boxall

1. Auflage

Arduino-Workshops – Boxall

schnell und portofrei erhältlich bei beck-shop.de DIE FACHBUCHHANDLUNG

dpunkt.verlag 2013

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 86490 106 5

Inhaltsverzeichnis: Arduino-Workshops – Boxall

3

Erste Schritte

In diesem Kapitel lernen Sie Folgendes:

- Grundlagen einer sauberen Projektgestaltung
- Grundlagen der Elektrizitätslehre
- Funktion von Widerständen, Leuchtdioden, Transistoren, Gleichrichterdioden und Relais
- Verwendung von Steckplatinen für die Konstruktion von Schaltungen
- Verwendung von Integervariablen, `for`-Schleifen und digitalen Ausgängen für verschiedene Arten von LED-Effekten

Als Nächstes werden Sie Ihren Arduino zum Leben erwecken. Dabei geht es um mehr als nur um die Platine als solche. Sie werden lernen, wie Sie Ihre Projekte planen, um Ihre Vorstellungen in die Realität umzusetzen. Anschließend erhalten Sie eine Einführung in die Elektrizitätslehre. Elektrischer Strom ist die Grundlage für alles, was wir in diesem Buch tun, weshalb es wichtig ist, für Ihre eigenen Projekte ein solides Verständnis der Grundlagen mitzubringen. Außerdem sehen wir uns die verschiedenen Bauteile an, die dabei helfen, Projekte in der Praxis zu gestalten. Am Ende werfen wir einen Blick auf einige neue Funktionen, die Bausteine für Arduino-Sketche bilden.

Projekte planen

Wenn Sie damit beginnen, eigene Projekte zu entwickeln, sind Sie vielleicht versucht, den Sketch sofort zu schreiben, gleich nachdem Ihnen die Idee dafür gekommen ist. Es ist jedoch sinnvoll, zunächst einige Vorbereitungen zu treffen. Schließlich kann der Arduino keine Gedanken lesen, sondern benötigt genaue Anweisungen. Wenn Sie winzige Details übersehen, kann es sein, dass der Arduino zwar trotzdem in der Lage ist, die Anweisungen auszuführen, allerdings werden dann die Ergebnisse nicht Ihren Erwartungen entsprechen.

Unabhängig davon, ob Sie ein einfaches Projekt erstellen, das nur ein Lämpchen blinken lässt, oder ob Sie ein automatisches Signal für eine Modelleisenbahn konstruieren, ist ein ausführlicher Plan für den Erfolg unverzichtbar. Gehen Sie bei der Umsetzung von Arduino-Projekten stets nach folgenden grundlegenden Schritten vor:

1. **Legen Sie das Ziel fest.** Bestimmen Sie, was Sie erreichen wollen.
2. **Schreiben Sie den Algorithmus.** Ein *Algorithmus* ist eine Reihe von Anweisungen, die beschreiben, wie das Projekt ausgeführt werden soll. Führen Sie in dem Algorithmus die Schritte auf, die notwendig sind, um das Ziel des Projekts zu erreichen.
3. **Wählen Sie die Hardware aus.** Ermitteln Sie, wie Sie diese Hardware an den Arduino anschließen.
4. **Schreiben Sie den Sketch.** Erstellen Sie eine erste Version des Programms, das dem Arduino sagt, was er tun soll.
5. **Stellen Sie alle Verbindungen her.** Schließen Sie die Hardware, die Schaltungen und alle weiteren erforderlichen Elemente an die Arduino-Platine an.
6. **Testen Sie das Projekt und beheben Sie Fehler.** Funktioniert es? Suchen Sie in dieser Phase nach Fehlern und ihren Ursachen, ob im Sketch, in der Hardware oder im Algorithmus.

HINWEIS

Selbst bei gut geplanten Projekten tritt manchmal eine Krankheit auf, die als »Feature Creep« bezeichnet wird, also als »schleichender Funktionszuwachs«. Das passiert, wenn jemand eine neue Funktion einfällt, die er einem Projekt noch hinzufügen möchte, und deshalb versucht, neue Elemente in eine bereits vorhandene Konstruktion hineinzuzwängen. Wenn Sie ein Design abwandeln müssen, dann versuchen Sie nicht, ihm irgendwelche zusätzlichen Elemente unterzuschieben oder Änderungen in letzter Minute vorzunehmen, sondern beginnen Sie neu, indem Sie das Ziel neu definieren.

Je mehr Zeit Sie auf die Planung des Projekts aufwenden, umso leichter fallen Test und Fehlerbehebung.

Elektrizität

Da Sie in Ihren Arduino-Projekten elektronische Schaltungen bauen werden, wollen wir uns zunächst mit der Elektrizität beschäftigen. Einfach ausgedrückt, ist *Elektrizität* oder *elektrischer Strom* eine Form von Energie, die wir nutzen und in Wärme, Licht, Bewegung und Kraft umsetzen können. Elektrischer Strom hat drei Haupteigenschaften, die bei unseren Projekten zum Tragen kommen: Stromstärke, Spannung und Leistung.

Stromstärke

Elektrischer Strom fließt (definitionsgemäß) vom positiven Pol einer Stromquelle, etwa einer Batterie, durch den Stromkreis zum negativen Pol. Hierbei handelt es sich um *Gleichstrom*. Daneben gibt es noch *Wechselstrom*, den wir in diesem Buch aber nicht behandeln. In manchen Stromkreisen wird der negative Pol als *Masse* oder *Erdung* bezeichnet (*GND* für *ground*). Die *Stromstärke* wird in *Ampere* (A) gemessen, kleinere Stromstärken in Milliampere (mA), wobei 1000 mA einer Stromstärke von 1 A entsprechen.

Spannung

Die *Spannung* ist die Differenz zwischen der potenziellen Energie am positiven und negativen Ende des Stromkreises. Gemessen wird sie in *Volt*. Je höher die Spannung, desto schneller bewegt sich der Strom durch den Stromkreis.

Leistung

Die *Leistung* ist ein Maß für die Rate, mit der ein elektrisches Gerät Energie einer Form in die andere umwandelt. Gemessen wird sie in *Watt* (W). So leuchtet beispielsweise eine 100-Watt-Birne heller als eine 60-Watt-Birne, da sie mehr elektrische Energie in Licht umwandelt.

Zwischen Spannung, Stromstärke und Leistung gibt es eine einfache mathematische Beziehung:

```
Leistung (P) = Spannung (U) x Stromstärke (I)
```

Elektronische Bauteile

Nachdem Sie jetzt die Grundlagen der Elektrizitätslehre kennen, wollen wir uns ansehen, wie sich dies auf die verschiedenen elektronischen Bauteile und Geräte auswirkt. Elektronische *Bauteile* sind Einzelteile, die den elektrischen Strom in einem Stromkreis so steuern, wie wir es für unsere Projekte brauchen. So wie die verschiedenen Bauteile in einem Auto zusammenwirken, um Treibstoff, Kraft und Bewegung bereitzustellen, sodass wir fahren können, steuern die elektrischen Bauteile zusammen den Strom und machen ihn für unsere Geräte nutzbar.

Besondere Elemente werde ich in diesem Buch immer dort beschreiben, wo sie zum Einsatz kommen. In den folgenden Abschnitten dagegen lernen Sie die grundlegenden Bauteile kennen.

Widerstände

Mehrere Bauteile, etwa die LEDs des Arduino, benötigen nur eine geringe Stromstärke, gewöhnlich um 10 mA. Wird die Platine mit einer zu hohen Stromstärke beschickt, wandelt sie den Überschuss in Wärme um, was sie letztendlich zerstören kann. Um die Stromstärke zu verringern, die an solchen Bauteilen anliegt, können wir zwischen der Spannungsquelle und der Komponente einen *Widerstand* platzieren. Im normalen Kupferdraht kann der Strom frei fließen, doch in einem Widerstand wird die Bewegung verlangsamt. Ein Teil des Stroms wird in Wärme umgewandelt. Wie groß dieser Anteil ist, hängt vom Widerstandswert ab. Abbildung 3–1 zeigt typische Widerstände.

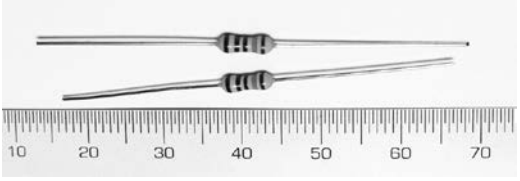


Abb. 3–1 *Typische Widerstände*

Elektrischer Widerstand

Der Widerstandswert eines solchen Bauteils kann fest oder variabel sein. Gemessen wird der elektrische Widerstand in Ohm (Ω), wobei höhere Werte in Kiloohm (kΩ = 1000 Ω) oder Megaohm (MΩ = 1.000.000 Ω) angegeben werden.

Den Widerstandswert ablesen

Widerstände sind sehr klein, weshalb auf diesen der Widerstandswert gewöhnlich nicht in Form von Zahlen abgedruckt ist. Sie können den Widerstandswert mithilfe eines Multimeters messen, aber auch direkt von dem Bauteil ablesen. Oft wird der Wert in Form einer Folge von farbigen Ringen angegeben, die von links nach rechts die folgende Bedeutung haben:

- **Erster Ring:** gibt die erste Stelle des Widerstandswerts an.
- **Zweiter Ring:** gibt die zweite Stelle des Widerstandswerts an.
- **Dritter Ring:** gibt den Multiplikator in Vielfachen von 10 (bei Vier-Ring-Widerständen) oder die dritte Stelle des Widerstandswerts an.
- **Vierter Ring:** gibt den Multiplikator in Vielfachen von 10 für Fünf-Ring-Widerstände an.
- **Fünfter Ring:** gibt die Toleranz (Genauigkeit) an.

Tabelle 3–1 führt die auf Widerständen verwendeten Farben und ihre Bedeutungen auf.

Farbe	Wert
Schwarz	0
Braun	1
Rot	2
Orange	3
Gelb	4
Grün	5
Blau	6
Violett	7
Grau	8
Weiß	9

Tab. 3–1 *Werte der farbigen Ringe auf einem Widerstand*

Der fünfte Ring gibt die *Toleranz* des Widerstands an, also das Maß für die Genauigkeit. Da es sehr schwer ist, Widerstände mit exakten Werten herzustellen, können Sie beim Kauf zwischen Varianten mit einem unterschiedlichen prozentualen Fehler wählen. Ein brauner Ring steht für eine Toleranz von 1 %, ein goldenes Band für 5 % und ein silbernes für 10 %.

In Abbildung 3–2 sehen Sie die schematische Darstellung eines Widerstands. Der gelbe, der violette und der orangefarbene Ring werden, wie in Tabelle 3–1 angegeben, nacheinander als die Zahlen 4, 7 und 3 gelesen. Das steht für 47.000 Ω oder kurz 47 kΩ.

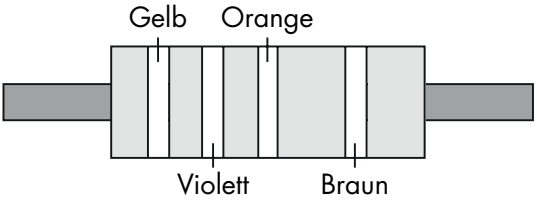


Abb. 3–2 *Beispiel für die Farbkennzeichnung auf einem Widerstand*

Chipwiderstände

Auf Widerständen für die Montage auf Oberflächen ist anstelle der Farbringe ein Code aus Zahlen und Buchstaben aufgedruckt, wie Sie in Abbildung 3–3 sehen. Die ersten beiden Zahlen bilden dabei eine Zahl, während die dritte angibt, wie viele Nullen darauf folgen. Der Widerstand in Abbildung 3–3 hat also einen Wert von 10.000 Ω oder 10 kΩ.

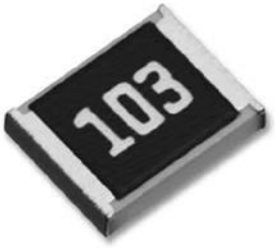


Abb. 3–3 *Beispiel für einen Widerstand zur Oberflächenmontage*

HINWEIS

Wenn auf einem kleinen Chipwiderstand ein Code aus Zahlen und Buchstaben aufgedruckt ist (beispielsweise O1C), dann suchen Sie in Google nach ***EIA-96 code calculator****, um Nachschlagetabellen für diesen Code zu finden.*

Multimeter

Ein *Multimeter* ist ein enorm praktisches und ziemlich preisgünstiges Messgerät, mit dem Sie Spannungen, Widerstände, Stromstärken und mehr bestimmen können. In Abbildung 3–4 sehen Sie beispielsweise ein Multimeter beim Messen eines Widerstands.

Abb. 3–4 *Multimeter beim Messen eines 560-Ω-Widerstands mit 1 % Toleranz*

Für Farbenblinde ist ein Multimeter unverzichtbar. Wie andere Werkzeuge sollten Sie auch dieses Messgerät von einem renommierten Händler beziehen, anstatt im Internet nach dem billigsten Modell zu suchen.

Leistungsangabe

Die Leistungsangabe eines Widerstands nennt die Leistung (in Watt), die das Bauteil verträgt, ohne sich zu überhitzen und auszufallen. Bei den Modellen in Abbildung 3–1 handelt es sich um 1/4-Watt-Widerstände, die im Arduino-System am häufigsten verwendete Art.

Bei der Auswahl von Widerständen müssen Sie das Verhältnis von Leistung, Stromstärke und Spannung berücksichtigen. Je größer Stromstärke oder Spannung, umso größer ist die Leistung des Widerstands.

Gewöhnlich wird bei wachsender Leistung auch die Bauform von Widerständen immer größer. Bei dem Modell in Abbildung 3–5 handelt es sich um einen 5-Watt-Widerstand, der 26 mm lang und 7,5 mm breit ist.

Abb. 3–5 *Ein 5-Watt-Widerstand*

Leuchtdioden (LEDs)

Eine LED ist ein weit verbreitetes, ungemein nützliches Bauteil, das elektrischen Strom in Licht umwandelt. Es gibt Leuchtdioden in unterschiedlichen Formen, Größen und Farben. In Abbildung 3–6 sehen Sie eine typische LED.

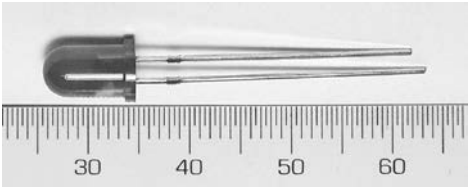


Abb. 3–6 *Rote LED mit einem Durchmesser von 5 mm*

Beim Einbau von Leuchtdioden in einen Stromkreis müssen Sie Obacht geben, da diese Bauteile *gepolt* sind. Das heißt, dass der Strom nur in einer Richtung durch die LED fließen kann. Er tritt an der positiven Seite (*Anode*) ein und an der negativen (*Kathode*) wieder aus, wie Abbildung 3–7 zeigt. Wenn Sie zu viel Strom in falscher Richtung durch eine Diode schicken, wird sie dadurch zerstört.

Zum Glück sind LEDs so aufgebaut, dass Sie die beiden Enden unterscheiden können. Das »Bein« der Anode ist länger und der Rand an der Unterseite ist an der Kathodenseite flach abgeschnitten (siehe Abb. 3–8).

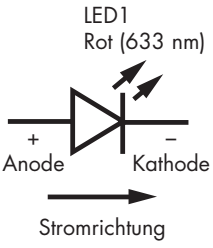


Abb. 3–7 *Stromfluss durch eine LED*

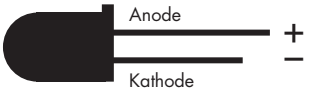


Abb. 3–8 *Durch die äußere Gestaltung der LED lassen sich Anode (langes Bein) und Kathode (abgeflachter Rand) unterscheiden.*

Wenn Sie für ein Projekt LEDs benötigen, müssen Sie die Betriebsspannung und die Stromstärke berücksichtigen. Beispielsweise sind für übliche rote LEDs etwa 1,7 V und 5 bis 20 mA erforderlich. Das stellt uns vor ein kleines Problem, da der Arduino eine feste Ausgabespannung von 5 V mit einer viel höheren Stromstärke liefert. Glücklicherweise können wir die Stärke des Stroms, der die LED erreicht, mit einem *Begrenzungswiderstand* verringern.

Welchen Widerstandswert brauchen wir dazu aber? Um das zu berechnen, wenden wir das ohmsche Gesetz an.

Um den erforderlichen Begrenzungswiderstand für eine LED zu bestimmen, verwenden wir die folgende Formel:

$$R = (U_V - U_D) / I$$

Hierbei ist U_V die Versorgungsspannung (der Arduino gibt 5 V ab), U_D die Durchlassspannung (hier 1,7 V) und I die erforderliche Stromstärke für die LED (10 mA). Der Wert von I muss aber in Ampere angegeben werden, weshalb wir 10 mA zunächst in 0,01 A umrechnen.

Wenn wir die angegebenen Werte für unsere LED in die Formel einsetzen, erhalten wir den Widerstand $R = 330\ \Omega$. Die LED leuchtet jedoch auch fröhlich vor sich hin, wenn wir sie mit weniger als 10 mA beschicken. Es ist immer eine gute Idee, die Stromstärke zu senken, um empfindliche elektronische Bauteile zu schonen, weshalb wir einen Widerstand von 560 Ω und 1/4 W verwenden. Dadurch gelangt ein Strom von 6 mA zu unseren LEDs.

HINWEIS

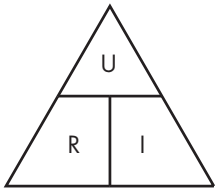
Wählen Sie im Zweifelsfall immer einen Widerstand mit einem leicht höheren Wert – besser eine trübe LED als eine kaputte!

DIE DREIECKSDARSTELLUNG DES OHMSCHEN GESETZES

Das ohmsche Gesetz gibt die Beziehung zwischen Stromstärke, Widerstand und Spannung an:

```
Spannung (U) = Widerstand (R) x Stromstärke (I)
```

Wenn Sie zwei dieser Größen kennen, können Sie daraus die dritte berechnen. Eine weit verbreitete Eselsbrücke für das ohmsche Gesetz ist das Dreieck in Abbildung 3–9.



U = R × I
I = U ÷ R
R = U ÷ I

Abb. 3–9 *Die Dreiecksdarstellung des ohmschen Gesetzes*

Dieses Dreieck ist ein bequemes Hilfsmittel für die Berechnung von Spannung, Stromstärke oder Widerstand, wenn zwei der drei Werte bekannt sind. Wollen Sie beispielsweise den Widerstand berechnen, decken Sie das ***R*** *mit dem Finger ab. Was übrig bleibt, ist Spannung dividiert durch Stromstärke. Um die Spannung zu berechnen, decken Sie* ***U*** *ab, sodass noch Stromstärke mal Widerstand zu sehen ist.*

Steckplatinen

Für unsere wechselnden Schaltungen brauchen wir eine Plattform, die sie zusammenhält und auf der wir sie aufbauen können. Hervorragend für diesen Zweck geeignet sind *Steckplatinen*. Dabei handelt es sich um ein Kunststoffbrett mit Reihen von Einsteckbuchsen, die über elektrische Leiter miteinander verbunden sind. Wie Sie in Abbildung 3–10 sehen, gibt es sie in unterschiedlichen Größen, Formen und Farben.

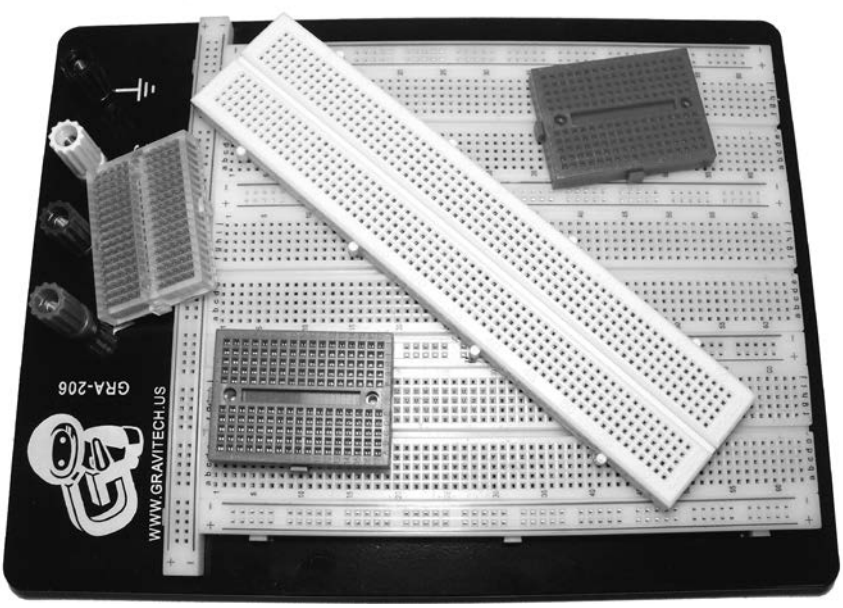


Abb. 3–10 *Steckplatinen in verschiedenen Formen und Größen*

Um eine Steckplatine verwenden zu können, müssen Sie wissen, wie die Buchsen verbunden sind – entlang der kurzen oder der langen Reihen in den einzelnen Abschnitten am Rand und in der Mitte. Das ist je nach Platine unterschiedlich. Beispielsweise sind bei der Platine oben in Abbildung 3–11 jeweils fünf Löcher in vertikaler Richtung verbunden, aber in horizontaler Richtung von den Nachbarn isoliert. Wenn Sie zwei Drähte in Löcher einer vertikalen Reihe stecken, sind sie elektrisch leitfähig verbunden. Die langen Reihen in der Mitte zwischen den horizontalen Abschnitten sind dagegen horizontal verbunden. Häufig werden Sie eine Schaltung an die Versorgungsspannung und an Masse anschließen müssen, und dazu sind diese horizontalen Lochreihen ideal geeignet.

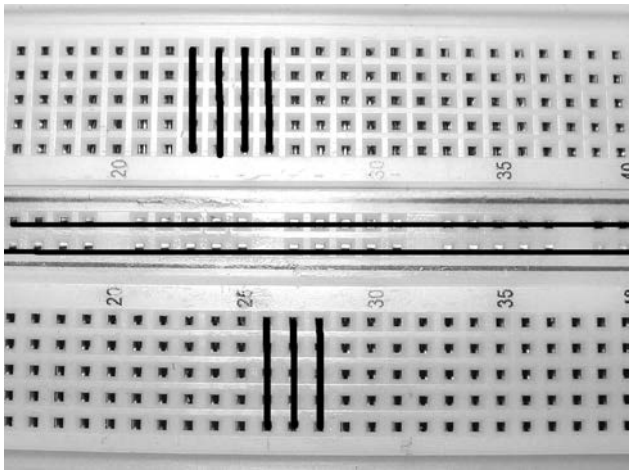


Abb. 3–11 *Interne Verbindungen einer Steckplatine*

Bei komplizierten Schaltungen kann die Steckplatine schnell überfüllt werden, sodass Sie nicht unbedingt in der Lage sind, Bauteile genau dort zu platzieren, wo Sie sie haben wollen. Dieses Problem lässt sich mit kurzen Verbindungsdrähten jedoch leicht lösen. Anbieter, die Steckplatinen verkaufen, haben gewöhnlich auch Sortimente von Drähten verschiedener Länge im Programm (siehe Abb. 3–12).

Abb. 3–12 *Sortiment von Verbindungsdrähten für Steckplatinen*

Projekt Nr. 1: LED-La-Ola

Wir wollen nun praktisch mit LEDs und Widerständen arbeiten. In diesem Projekt sollen fünf LEDs ein Lichtmuster anzeigen, das sich in La-Ola-Form weiterbewegt (ähnlich wie vorn an KITT aus der Fernsehserie *Knight Rider*).

Der Algorithmus

Der Algorithmus für dieses Projekt sieht wie folgt aus:

1. Schalte LED 1 ein.
2. Warte eine halbe Sekunde.
3. Schalte LED 1 aus.
4. Schalte LED 2 ein.
5. Warte eine halbe Sekunde.
6. Schalte LED 2 aus.
7. Fahre fort, bis LED 5 eingeschaltet wird, und kehre den Vorgang dann von LED 5 zu LED 1 um.
8. Wiederhole den Vorgang unendlich oft.

Die Hardware

Für dieses Projekt brauchen Sie folgende Teile:

- fünf LEDs
- fünf 560-Ω-Widerstände
- eine Steckplatine
- Verbindungsdrähte
- Arduino und USB-Kabel

Die LEDs schließen wir über die 560-Ω-Begrenzungswiderstände an die Digitalpins 2 bis 6 an.

Der Sketch

Kommen wir jetzt zu unserem Sketch. Geben Sie in der IDE den folgenden Code ein:

```
// Projekt 1: LED-La-Ola
void setup() ❶
{
   pinMode(2, OUTPUT);       // der Steuerpin für LED 1 wird
                             // als Ausgang festgelegt
   pinMode(3, OUTPUT);       // Dito für LED 2 bis LED 5
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
   pinMode(6, OUTPUT);
}

void loop() ❷
{
   digitalWrite(2, HIGH);    // Schaltet LED 1 ein
   delay(500);               // Wartet eine halbe Sekunde
   digitalWrite(2, LOW);     // Schaltet LED 1 aus
   digitalWrite(3, HIGH);    // Wiederholt den Vorgang für LED 2 bis 5
   delay(500);
   digitalWrite(3, LOW);
   digitalWrite(4, HIGH);
   delay(500);
   digitalWrite(4, LOW);
   digitalWrite(5, HIGH);
   delay(500);
   digitalWrite(5, LOW);
   digitalWrite(6, HIGH);
   delay(500);
   digitalWrite(6, LOW);
   digitalWrite(5, HIGH);
   delay(500);
   digitalWrite(5, LOW);
   digitalWrite(4, HIGH);
   delay(500);
   digitalWrite(4, LOW);
   digitalWrite(3, HIGH);
   delay(500);
   digitalWrite(3, LOW);
   // Die Schleife loop() kehrt an den Anfang zurück und
   // wird wiederholt ausgeführt
}
```

In `void setup()` an der Stelle ❶ werden die Digital-E/A-Pins als Ausgänge festgelegt, da wir die LEDs bei Bedarf mit Strom versorgen wollen. Mit der Funktion `digitalWrite()` im Abschnitt `void loop()` (❷) schalten wir die einzelnen LEDs ein.

Der Schaltplan

Als Nächstes konstruieren wir die Schaltung. Der Aufbau solcher Schaltungen lässt sich auf verschiedene Weise darstellen. Für die ersten Projekte in diesem Buch verwenden wir fotografische Schaltpläne wie in Abbildung 3–13.

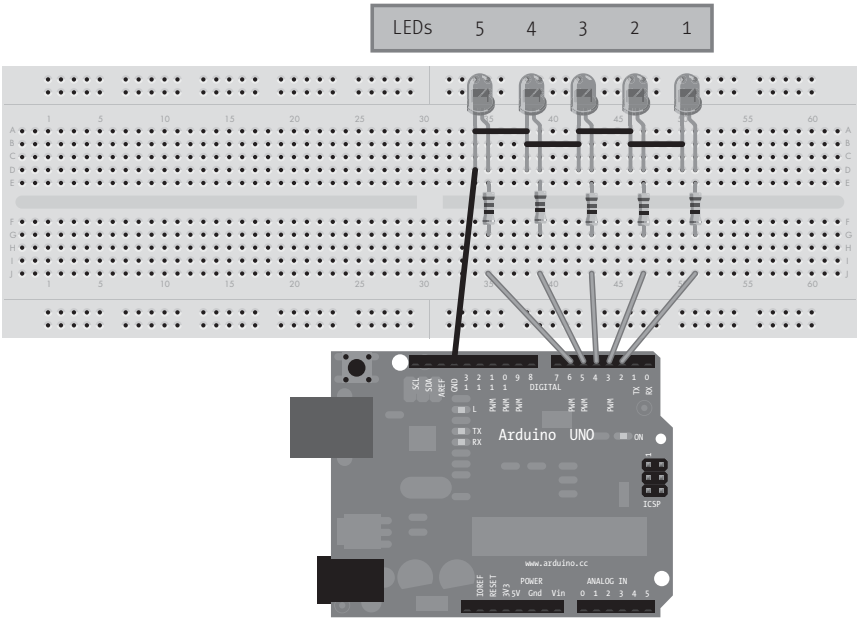


Abb. 3–13 *Schaltplan für Projekt 1*

Wenn Sie die Verdrahtungsskizze mit den Funktionen im Sketch vergleichen, können Sie erkennen, wie die Schaltung funktioniert. Beispielsweise wird bei der Verwendung von `digitalWrite(2, HIGH)` eine hohe Spannung von 5 V an den Digitalpin 2 angelegt. Der Strom verläuft durch den Begrenzungswiderstand über die Anode in die LED, über die Kathode wieder heraus und schließlich zurück zum GND-Anschluss des Arduino, um den Kreis zu schließen. Anschließend verwenden wir `digitalWrite(2, LOW)`, um den Strom und damit die LED wieder abzuschalten.

Den Sketch ausführen

Schließen Sie jetzt den Arduino an und laden Sie den Sketch hoch. Nach ein oder zwei Sekunden sollten die LEDs von links nach rechts und zurück aufleuchten. Erfolg ist doch etwas Schönes – genießen Sie ihn!

Wenn nichts geschieht, sollten Sie sofort das USB-Kabel vom Arduino abziehen und prüfen, ob Sie den Sketch richtig eingegeben haben. Falls Sie einen Fehler finden, korrigieren Sie ihn und laden den Sketch erneut hoch. Stimmt der Sketch, blinken die LEDs aber trotzdem nicht, überprüfen Sie die Verkabelung auf der Schaltplatine.

Sie können jetzt zwar mit dem Arduino ein Lichtmuster aus LEDs erzeugen, aber der Sketch ist doch noch recht klobig. Wenn Sie das Lichtsignal beispielsweise schneller hin- und herlaufen lassen wollten, müssten Sie jedes einzelne Vorkommen von `delay(500)` ändern. Es gibt jedoch bessere Möglichkeiten.

Verwenden von Variablen

Um Daten in Computerprogrammen festzuhalten, verwenden wir *Variablen*. Im dem Sketch für Projekt 1 haben wir die Funktion `delay(500)` eingesetzt, damit die LEDs eingeschaltet bleiben. Diese Vorgehensweise ist jedoch nicht sehr flexibel. Wenn wir die Verzögerungszeit ändern wollen, müssen wir jeden Eintrag manuell umschreiben. Besser ist es, eine Variable anzulegen, die den Wert für die Funktion `delay()` festhalten soll.

Geben Sie im Sketch von Projekt 1 über der Funktion `void setup()`, unmittelbar hinter dem einleitenden Kommentar, folgende Zeile ein:

```
int d = 250;
```

Dadurch wird die Zahl 250 der Variable `d` zugewiesen.

Ersetzen Sie anschließend jedes Vorkommen von `500` in dem Sketch durch `d`. Wenn der Sketch jetzt ausgeführt wird, verwendet der Arduino den in `d` gespeicherten Wert für die `delay()`-Funktionen. Die LEDs werden jetzt viel schneller ein- und ausgeschaltet, da die Verzögerung mit dem Wert 250 nur noch halb so groß ist.

Das Schlüsselwort `int` deutet schon an, dass die Variable einen *Integer* enthält, also eine Ganzzahl zwischen –32.768 und 32.767. Einfach gesagt, ist ein Integer kein Bruch und weist auch keine Dezimalstellen auf.

Wenn Sie die Verzögerung jetzt ändern wollen, müssen Sie lediglich die Variablendeklaration am Anfang des Sketches bearbeiten. Wenn Sie beispielsweise wie folgt den Wert `100` für die Verzögerung festlegen, läuft die Sache noch schneller:

```
int d = 100;
```

Spielen Sie ein bisschen mit dem Sketch herum, indem Sie beispielsweise die Verzögerungszeiten oder die Abfolge von HIGH und LOW ändern. Bauen Sie die Schaltung aber noch nicht auseinander, da wir sie noch für weitere Projekte in diesem Kapitel gebrauchen werden.

Projekt Nr. 2: Wiederholungen mit for-Schleifen

In einem Sketch müssen Sie häufig eine Funktion wiederholt ablaufen lassen. Dazu könnten Sie die Funktion einfach kopieren und erneut einfügen, aber das wäre umständlich und außerdem eine Verschwendung des Programmspeichers auf dem Arduino. Stattdessen sollten Sie for-Schleifen verwenden. Deren Vorteil besteht darin, dass Sie bestimmen können, wie oft der Code in der Schleife wiederholt werden soll.

Um sich anzusehen, wie eine for-Schleife funktioniert, geben Sie den folgenden Code als neuen Sketch ein:

```
// Projekt 2: Wiederholungen mit for-Schleifen
int d = 100;

void setup()
{
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
   pinMode(6, OUTPUT);
}

void loop()
{
for ( int a = 2; a < 7 ; a++ )
   {
      digitalWrite(a, HIGH);
      delay(d);
      digitalWrite(a, LOW);
      delay(d);
   }
}
```

Die for-Schleife wiederholt den Code, der in den geschweiften Klammern unter dem Schlüsselwort steht, so lange, wie eine bestimmte Bedingung wahr ist. Hier verwenden wir die neue Integervariable a, die zu Anfang den Wert 2 hat. Bei jeder Ausführung des Codes wird der Wert von a mit a++

um 1 erhöht. Die Schleife wird auf diese Weise ausgeführt, solange der Wert von a kleiner als 7 ist (dies ist die Bedingung). Sobald a größer oder gleich 7 ist, fährt der Arduino mit dem Code fort, der hinter der for-Schleife steht.

Wie oft eine for-Schleife ausgeführt werden soll, können Sie auch festlegen, indem Sie von einer höheren zu einer niedrigeren Zahl übergehen. Um das zu veranschaulichen, fügen Sie hinter die erste for-Schleife den folgenden Code in den Sketch von Projekt 2 ein:

```
for ( int a = 5 ; a > 1 ; a-- ) ❶
   {
      digitalWrite(a, HIGH);
      delay(d);
      digitalWrite(a, LOW);
      delay(d);
   }
```

Hier legt die for-Schleife bei ❶ den Wert von a auf 5 fest und zieht dann bei jeder Ausführung der Schleife mithilfe von a-- 1 davon ab. Die Schleife wird durchlaufen, solange der Wert von a größer als 1 ist (a > 1). Fällt der Wert auf oder unter 1, wird die Schleife beendet.

Damit haben wir Projekt 1 mit deutlich weniger Code umgeschrieben. Laden Sie den Sketch hoch und probieren Sie es selbst aus.

Die Helligkeit der LEDs durch Pulsbreitenmodulation ändern

Anstatt die LEDs nur mithilfe von digitalWrite() ein- und auszuschalten, können Sie deren vermeintliche Helligkeit auch ändern, indem Sie die Zeitdauer zwischen dem Ein- und Aus-Zustand mithilfe der *Pulsbreitenmodulation* (PBM) regeln. Damit können Sie die Illusion einer veränderten Helligkeit der LED hervorrufen, indem Sie die LED extrem schnell, etwa 500 Mal die Sekunde, ein- und ausschalten. Die Helligkeit, die wir wahrnehmen, wird durch das Verhältnis der Zeit bestimmt, in der der digitale Ausgangspunkt ein- bzw. ausgeschaltet ist, also die LED leuchtet bzw. nicht leuchtet. Da unser Auge ein schnelleres Flackern als 50 Mal in der Sekunde nicht mehr wahrnehmen kann, scheint es so, als hätte die LED eine konstante Helligkeit.

Je größer das *Tastverhältnis* (die Einschaltzeit des Pins im Vergleich zur Ausschaltzeit in jedem Zyklus), umso höher ist die wahrgenommene Helligkeit der an den digitalen Ausgangs-Pin angeschlossenen LED.

Abbildung 3–14 zeigt verschiedene Tastverhältnisse der Pulsbreitenmodulation. Die grau gefüllten Bereiche stehen für die Zeit, in der das Licht leuchtet. Wie Sie sehen, ist die Einschaltzeit länger, je höher das Tastverhältnis ist.

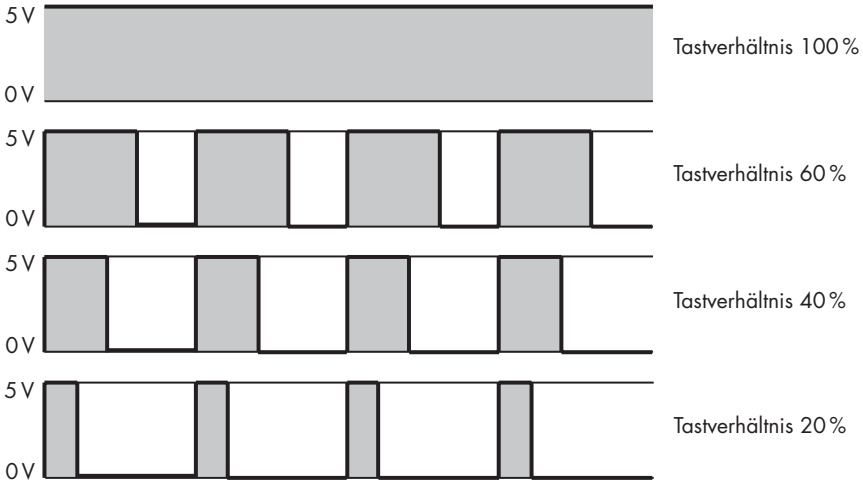


Abb. 3–14 *Unterschiedliche PBM-Tastverhältnisse*

Auf einem regulären Arduino lassen sich nur die Digitalpins 3, 5, 6, 9, 10 und 11 für die Pulsbreitenmodulation verwenden. Wie Sie in Abbildung 3–15 sehen, sind sie auf der Platine mit einer Tilde (~) markiert.

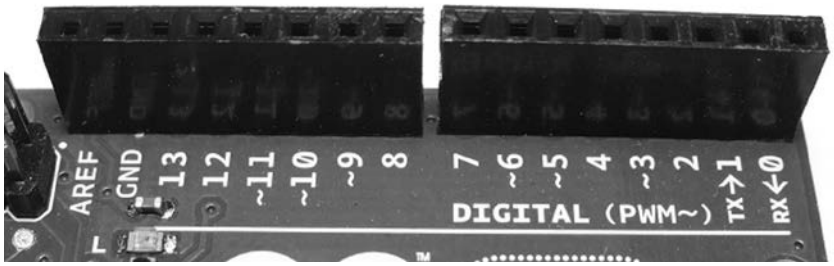


Abb. 3–15 *Die PBM-Pins sind mit einer Tilde gekennzeichnet.*

Um ein PBM-Signal zu erzeugen, verwenden wir die Funktion `analogWrite(x, y)`. Dabei ist `x` die Nummer des Digitalpins und `y` ein Wert für das Tastverhältnis zwischen 0 (0 %) und 255 (100 %).