5 Ausgänge: Licht und Lärm

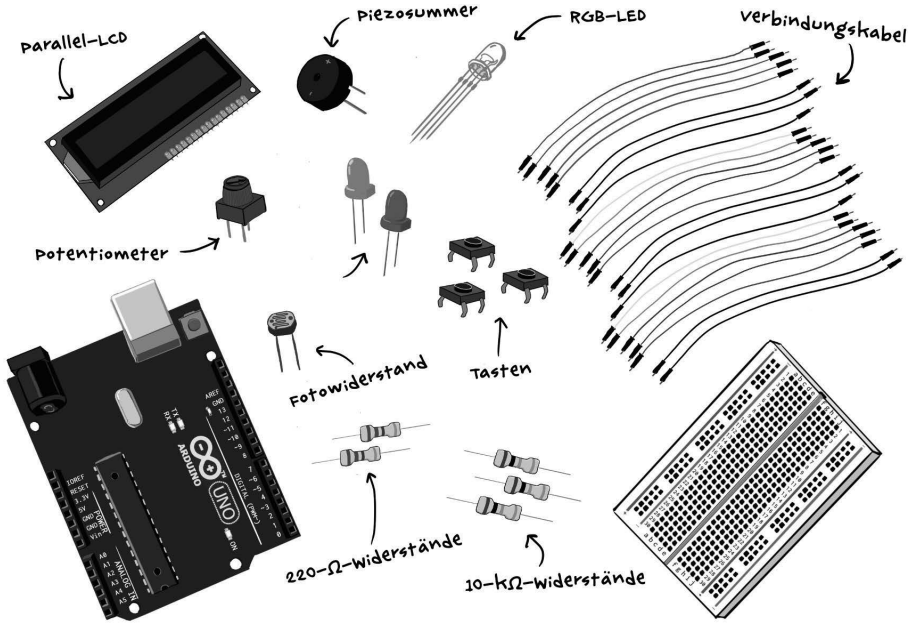
Inhalt dieses Kapitels

- Anspruchsvollere Techniken zur Steuerung von LEDs – Animation und RGB-LEDs
- Digitale Ausgangssignale mithilfe der Pulsweitenmodulation an das Verhalten von analogen Ausgangssignalen anpassen
- Grundlagen von Bitverschiebungs- und Binäroperationen in JavaScript
- Eine mehrfarbige LED-Wetterstation mithilfe einer Wetter-API von einem Drittanbieter konstruieren
- Ein LCD-Parallelmodul an den Uno anschließen und mit Johnny-Five steuern
- Einen anspruchsvollen Zeitgeber aus mehreren Eingangs- und Ausgangsbauteilen konstruieren
- Mit Piezokomponenten und Johnny-Five Geräusche machen und Melodien spielen

Wir wollen jetzt etwas Lärm machen. Oder Licht. Oder beides. Sie haben schon einige einfache Tricks mit LEDs ausprobiert, aber jetzt wollen wir uns etwas ausführlicher ansehen, wie Sie *Ausgänge* in Ihren Projekten verwenden können.

Für dieses Kapitel benötigen Sie:

- 1 Arduino Uno mit USB-Kabel
- 2 Standard-LEDs, beliebige Farbe
- 1 Fotowiderstand
- 1 RGB-LED mit gemeinsamer Kathode
- 3 Tasten
- 1 LCD-Parallelmodul 16 x 2
- 1 Drehpotentiometer
- 1 Piezosummer
- 3 10-kΩ-Widerstände
- 2 220-Ω-Widerstände
- 23 Verbindungskabel in verschiedenen Farben
- 1 Steckbrett mit ca. 400 Kontakten



5.1 Es werde Licht

LEDs scheinen noch mehr Dinge tun zu können als einfach nur an- und auszugehen. Wenn Sie sich Ihre elektronischen Geräte ansehen, können Sie beobachten, wie die darin eingebauten LEDs pulsieren und abgedunkelt werden.

Diese Verhaltensweisen sind aber nur Illusionen (siehe Abb. 5–1). Eine LED kann nur Licht einer einzigen Wellenlänge ausgeben, also nur Licht einer Farbe. Trotzdem sind die LEDs um uns herum scheinbar in der Lage, ihre Helligkeit oder ihren Farbton zu ändern.

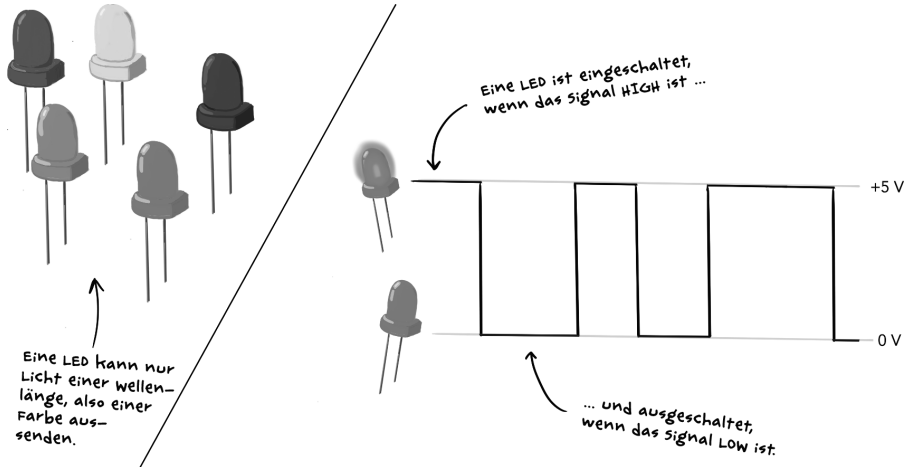


Abb. 5–1 *LEDs geben nur eine Farbe ab und sind entweder ein- oder ausgeschaltet.*

Zeit für ein wenig LED-Zauberei! Es gibt eine elektronische Technik, die wir nutzen können, um LEDs ein anspruchsvolleres Verhalten zu entlocken.

LEDs mit Pulsweitenmodulation (PWM) ausblenden

Eine LED kann nur ein- oder ausgeschaltet sein. Sie können sie zum Blinken bringen, indem Sie sie in regelmäßigen, kurzen Abständen ein- und ausschalten. Mit diesem Blinken können wir aber auch das menschliche Auge täuschen.

Erforderliches Material

- 1 Arduino Uno mit USB-Kabel
- 1 Steckbrett
- 2 Standard-LEDs beliebiger Farbe
- 2 220-Ω-Widerstände
- Gelbe (2) und schwarze (1) Verbindungskabel

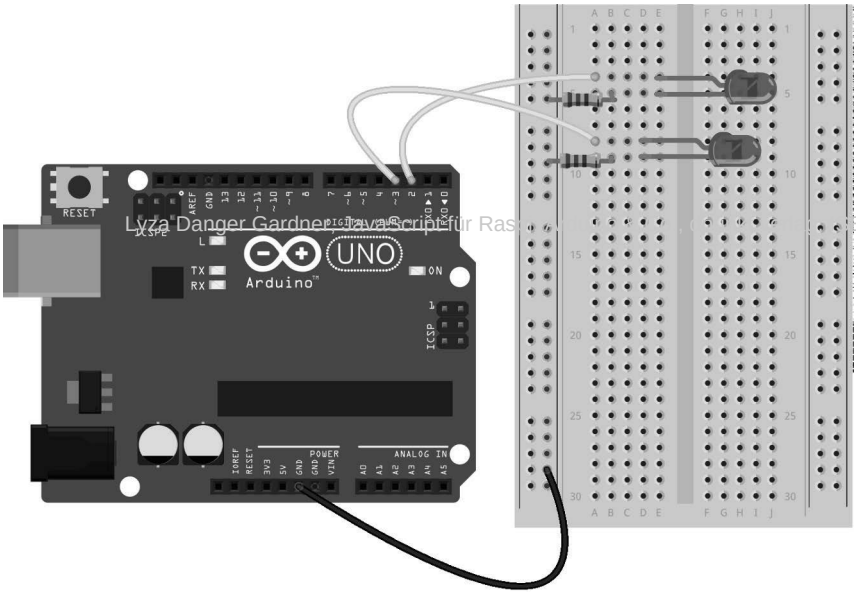


Abb. 5–2 *Verkabelungsdiagramm für das LED-Experiment*

Bauen Sie die Schaltung aus Abb. 5–2 auf. Erstellen Sie dann eine neue Datei und geben Sie den Code aus dem folgenden Listing darin ein.

Listing 5–1: ***experiment-led.js***

```
const five = require('johnny-five');
const board = new five.Board();
```

```
board.on('ready', () => {
  const led1 = new five.Led(2);
  const led2 = new five.Led(3);

  board.repl.inject({
    led1: led1,
    led2: led2
  });
});
```

Instanziiert ein Led-Objekt an Pin 2

Instanziiert ein Led-Objekt an Pin 3

Injiziert die Led-Objekt-instanzen in die REPL

Dieser Code tut nicht viel. Er instanziiert lediglich zwei `Led`-Objekte und macht sie in der Johnny-Five-REPL als `led1` und `led2` verfügbar. Führen Sie das Skript aus:

```
$ node experiment-LED.js
```

Jetzt können Sie an der REPL-Eingabeaufforderung Befehle eingeben. Auch wenn Sie mit diesem Vorgang bereits vertraut sind, wollen wir als Erstes eine der LEDs zum Blinken bringen. Geben Sie Folgendes ein und drücken Sie die Eingabetaste:

```
>> led1.blink()
```

Die erste LED sollte jetzt im 100-ms-Takt blinken (100 ms an, 100 ms aus). Das ist die Standardphasenlänge (die Länge der einzelnen Blinkperioden) für die Methode `blink`.

Heben Sie jetzt den Arduino und das Steckbrett vorsichtig mit einer Hand an und wedeln Sie sie vor Ihrem Gesicht hin und her. Wenn Sie das schnell genug machen, scheint die LED nicht mehr zu blinken, sondern sie sieht aus wie eine verschmierte Linie.

Das ist eine optische Täuschung. Wenn sich Dinge zu schnell bewegen, kommen Ihr Gehirn und Ihre Augen nicht mehr mit. Das Gehirn verbindet die einzelnen Punkte und macht daraus eine durchgezogene Linie. In der Fotografie kennen Sie vielleicht einen ähnlichen Effekt: Wenn die Belichtungszeit nicht kurz genug ist, tritt in den Bildern Bewegungsunschärfe auf. Dieses Phänomen sorgt auch dafür, dass Filme mit einer Abfolge von 24 Bildern pro Sekunde oder mehr wie eine fließende Bewegung erscheinen.

Das Gleiche gilt auch für das Erkennen einzelner Blinkvorgänge. Wenn eine Lampe schnell genug blinkt, können wir nicht mehr wahrnehmen, dass sie überhaupt an- und ausgeht (siehe Abb. 5–3). Dabei haben wir Menschen jeweils individuelle Wahrnehmungsschwellen, sodass ich fast wahnsinnig werde, wenn ich mit alten Fluoreszenz- oder Röhrenbildschirmen mit niedriger Bildwiederholrate arbeiten muss, während sich meine Kollegen nicht im Geringsten daran stören. Aber bei ca. 100 Hz (100 Zyklen pro Sekunde) verliert jeder die Fähigkeit, einzelne Blinkvorgänge zu unterscheiden.

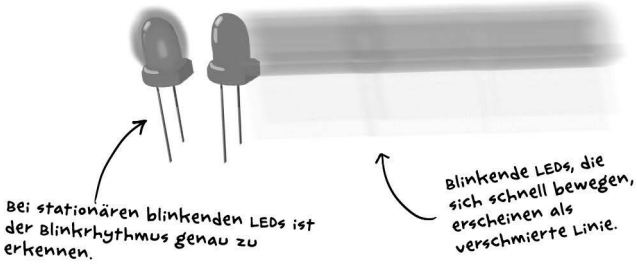


Abb. 5–3 *Wenn sich eine LED schnell bewegt, können wir die einzelnen Blinkvorgänge nicht mehr unterscheiden.*

Aufhören!

Wenn die blinkende LED Sie nervt, können Sie sie mit der Instanzmethode `Led.stop` anhalten:

```
> led1.stop()
```

Je nachdem, wann Sie den Befehl geben, kann es sein, dass die LED anschließend aus ist oder Dauerlicht gibt. Ist sie an und wollen Sie sie ausschalten, geben Sie Folgendes ein:

```
> led1.off()
```

Sie müssen beide Methoden verwenden, denn es ist nicht möglich, einfach sofort `off()` zu verwenden. Durch `stop` wird das Intervall gelöscht, das den Blinkvorgang verursacht, durch `off` nicht.

So weit, so gut. Aber welchen Sinn hat es, eine LED so schnell blinken zu lassen, dass wir die Blinkvorgänge nicht mehr unterscheiden können und es aussieht, als gäbe sie Dauerlicht? Schließlich könnten wir die LED doch genauso gut einfach einschalten, um dieselbe Wirkung zu erzielen, oder?

Es geschieht jedoch etwas Bemerkenswertes, wenn Sie ein bisschen mit dem Verhältnis der Zeiträume herumspielen, in denen die LED ein- bzw. ausgeschaltet ist. Wenn die LED sehr schnell regelmäßig blinkt, dann sind wir nicht mehr in der Lage, den Blinkvorgang zu erkennen – aber wenn sie nur ein Viertel der Zeit eingeschaltet ist (und drei Viertel der Zeit ausgeschaltet), dann erscheint sie auch erheblich dunkler.

Probieren Sie das in der REPL aus:

```
>> led1.on()
>> led2.brightness(64)
```

Bei `brightness` (»Helligkeit«) handelt es sich um eine Instanzmethode von `Led`-Objekten in Johnny-Five. Sie nimmt einen 8-Bit-Wert (0 bis 255) entgegen, sodass ein

Wert von 64 für ein Viertel der höchstmöglichen Helligkeit steht. Infolge des Aufrufs von `led2.brightness(64)` ist die LED in 75 % der Zeit aus und nur in 25 % der Zeit ein, wobei die Ein- und Ausschaltvorgänge zu schnell erfolgen, als dass wir sie erkennen könnten. Stattdessen erscheint die zweite LED jetzt deutlich weniger hell als die erste.

Der Mikrocontroller des Uno macht dies möglich, indem er die Hardwarevoraussetzungen für die sogenannte *Pulsweitenmodulation* (*PWM*) bietet und die Ein- und Ausschaltvorgänge schneller steuert, als die Software es tun könnte. Der prozentuale Anteil der Zeit, in dem ein PWM-Signal `HIGH` (also eingeschaltet) ist, wird als *Tastverhältnis* bezeichnet. Ein Ausgang, der ein Viertel der Zeit lang `HIGH` ist, hat ein Tastverhältnis von 25% (siehe Abb. 5–4).

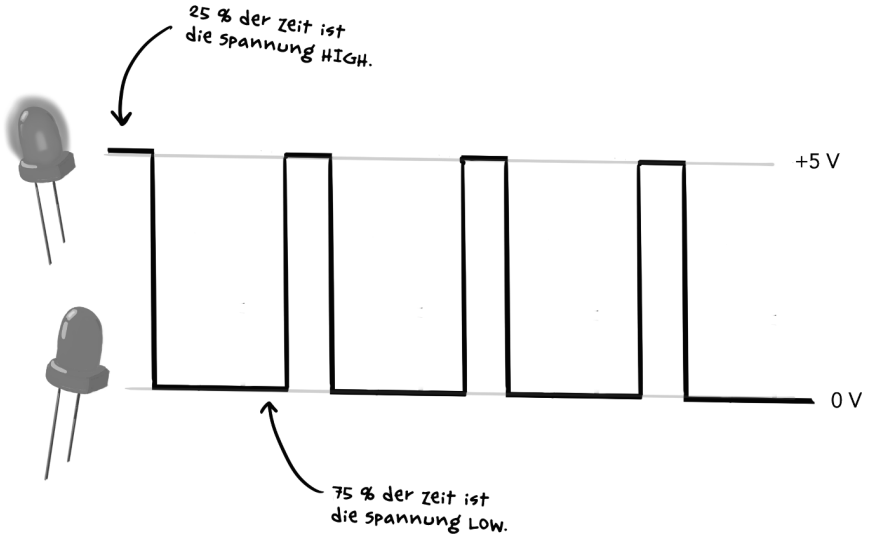


Abb. 5–4 *Ein Signal mit einem Tastverhältnis von 25%*

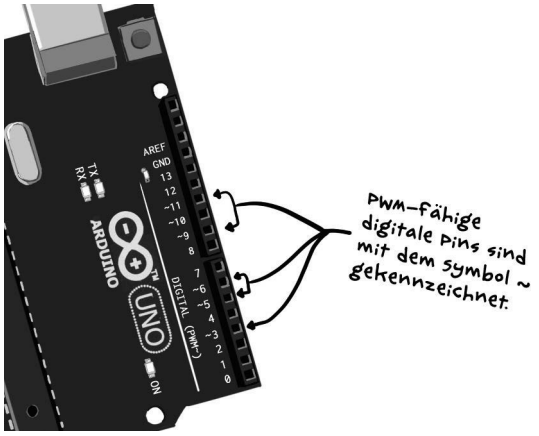


Abb. 5–5 *Nur einige der Digitalpins auf dem Arduino Uno unterstützen PWM. Sie sind mit einer Tilde gekennzeichnet.*

Hardwareunterstützung für PWM ist weit verbreitet, da es sich um eine sehr nützliche Technik handelt, allerdings unterscheidet sie sich von einer Platine zur anderen und steht gewöhnlich nur an bestimmten Pins zur Verfügung. Das ist auch beim Arduino Uno der Fall, wo nur die Pins 3, 5, 6, 9, 10 und 11 für PWM geeignet sind.

Aber keine Angst, Sie müssen sich diese Zahlen nicht merken, denn auf der Platine sind die Nummern der entsprechenden Pins mit einer vorangestellten Tilde (~) gekennzeichnet (siehe Abb. 5–5).

Probieren Sie nun in der REPL Folgendes aus:

```
>> led1.brightness(64)
```

Das funktioniert nicht. Stattdessen erhalten Sie eine Fehlermeldung, die wie folgt beginnt:

```
Error: Pin Error: 2 is not a valid PWM pin (Led)
```

Diese Meldung kommt von Johnny-Five und Firmata. Die Software weiß, welche Pins auf welchen Platinen was tun können.

Wir haben jetzt schon einige bemerkenswerte Dinge festgestellt: Menschen können zu schnelle Blinkvorgänge nicht mehr unterscheiden. Wenn wir das Tastverhältnis des Signals für eine LED ändern, scheint sie heller oder dunkler zu werden. Wir werden jetzt noch einen weiteren Punkt besprechen und dann dazu übergehen, diese Kenntnisse praktisch umzusetzen.

Anschluss von Bauteilen mit PWM-Steuerung

Methoden wie `brightness` sowie verschiedene andere Methoden von `Led`-Objekten erfordern, dass das entsprechende Bauteil an einen Pin mit PWM-Unterstützung angeschlossen ist. Johnny-Five löst eine Ausnahme aus, wenn Sie versuchen, eine Methode, die PWM benötigt, für ein Bauteil anzuwenden, das an einen Pin ohne PWM-Unterstützung angeschlossen ist. Achten Sie beim Schaltungsaufbau immer darauf, welche Pins PWM unterstützen, um Zeit und Ärger zu sparen.

Probieren Sie Folgendes in der REPL aus:

```
>> led1.on()
>> led2.brightness(128)
```

Da 128 genau in der Mitte des Helligkeitsbereichs liegt, sollte man erwarten, dass die zweite LED infolge dieser beiden JavaScript-Ausdrücke nur halb so hell scheint wie die erste. Allerdings werden Sie kaum einen Unterschied erkennen, sofern Sie die Augen nicht ziemlich weit zukneifen. Möglicherweise ist ein kleiner Unterschied zu bemerken, aber kein sehr starker.

Das liegt daran, dass Helligkeit im menschlichen Gehirn Vorrang hat. Zwar ruft `brightness(128)` ein Tastverhältnis von 50 % hervor, sodass die LED nur

während der Hälfte der Zeit eingeschaltet ist, doch verzerrt das Gehirn diese Wahrnehmung zu einer größeren Helligkeit. Anders ausgedrückt: Die 8-Bit-Helligkeitsskala von Johnny-Five, die von 0 bis 255 reicht, ist nicht linear, was aber nur an der menschlichen Wahrnehmung liegt.

LEDs mit PWM animieren

Sie wissen jetzt, wie Sie eine LED ein- und ausschalten, wie Sie sie zum Blinken bringen und sogar, wie Sie ihre Helligkeit ändern, sofern sie an einen PWM-Pin angeschlossen ist. Als weitere Ergänzung zu all diesen Tricks sehen wir uns nun an, wie Sie die Helligkeit einer LED *animieren*, sodass sie zu pulsieren, zu zittern, zu atmen oder langsam einzuschlafen scheint. Ihrer Fantasie sind keine Grenzen gesetzt!

In Johnny-Five gibt es die Klasse `Animation`, die eine detaillierte Steuerung von Animationen ermöglicht, und zwar zurzeit sowohl für die Helligkeit von LEDs als auch für die Bewegung von Servomotoren (mit denen wir uns in Kapitel 6 beschäftigen werden).

Die Arbeit mit Animationen in Johnny-Five erfolgt in mehreren Schritten, wie das folgende Listing zeigt:

Listing 5–2: Die einzelnen Schritte zur Animation in Johnny-Five

```
const pulsingLED = new five.Led(3);
const options    = { /* Einzelheiten der Animation */ };
const animation  = new five.Animation(pulsingLED);
animation.enqueue(options);
```

Erstellt ein `options`-Objekt mit den Einzelheiten der Animation; mehr darüber in Kürze.

Instanziiert ein `Animation`-Objekt und übergibt ihm die Zielkomponente, die animiert werden soll.

Beginnt die Animation mit `enqueue`.

Zur Veranschaulichung wollen wir die LED *pulsieren* lassen. In Johnny-Five müssen wir dazu zunächst angeben, wie sich die Animation verhalten soll.

Unsere pulsierende LED soll auf ansprechende Weise ein- und ausgeblendet werden. Sogenannte *Easing-Funktionen* (zum weichen Anlaufen und Auslaufen) ändern die Animationsgeschwindigkeit, während sie aktiv sind. Beispielsweise bewegt sich eine Animation mit weichem Auslaufen (»ease out«) zunächst schnell und wird dann langsamer.

Easing-Funktionen sind gewöhnlich nicht linear und nutzen Sinus-, Kubik-, Exponential- und andere Kurven. Easing-Funktionen mit Sinus etwa führen zu dem Anlauf- und Auslaufverhalten aus Abb. 5–6. Mithilfe solcher Funktionen können Animationen organischer gestaltet werden oder unterschiedliche Qualitäten annehmen. Natürlich bewegen sich die LEDs nicht, aber die Änderung ihrer Helligkeit wird animiert.

Für unsere pulsierende LED verwenden wir die Easing-Funktion `inOutSine`. Dabei beginnt die Helligkeitsänderung langsam und nimmt dann immer mehr Fahrt auf. Ein Puls ist eine *metronomische* Animation, d.h. nachdem sie vorwärts durchgelaufen ist, läuft sie anschließend rückwärts und pendelt in einer *Schleife* ständig zwischen dem Anfangs- und dem Endzustand.

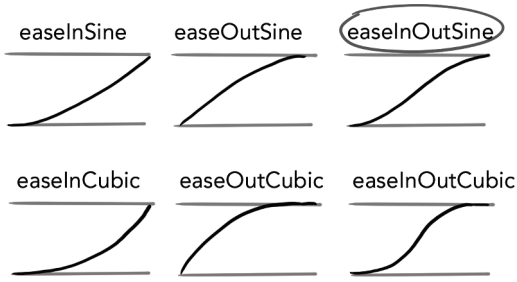


Abb. 5–6 `inOutSine` *ist eine von mehreren Dutzend Easing-Funktionen, die über das npm-Paket* `ease-component` *in Johnny-Five zur Verfügung stehen.*

Außerdem müssen wir sogenannte *Keyframes* (Schlüsselbilder) definieren (siehe Abb. 5–7). Sie geben die Zustände an (vergleichbar mit Einzelbildern in einer Filmanimation), deren Zwischenraum die Animation mithilfe der Easing-Funktion ausfüllen soll. Der Vorgang, Bilder oder Zustände zwischen den Keyframes zu erstellen, wird als *Tweening* bezeichnet. Für unseren einfachen Pulsationsvorgang brauchen wir auch nur einfache Keyframes: ganz aus- (Helligkeit 0) und ganz eingeschaltet (Helligkeit 255).

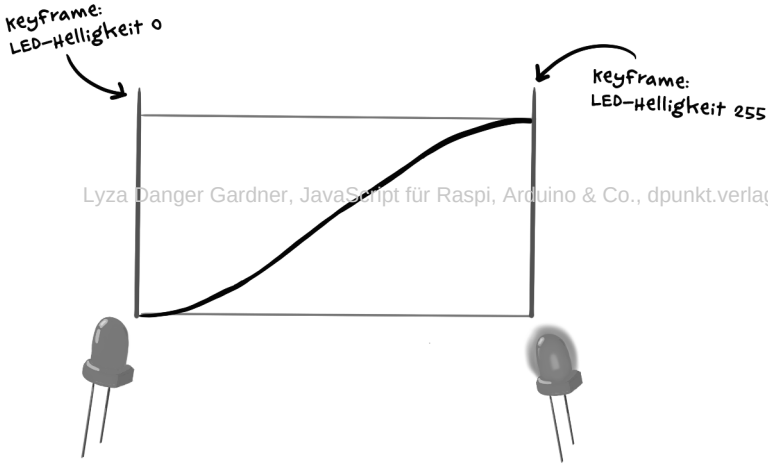


Abb. 5–7 *Die Pulsationsanimation pendelt in einer Schleife zwischen den beiden Keyframes (Helligkeit 0 und Helligkeit 255) mit der Easing-Funktion* `inOutSine` *hin und her.*

Des Weiteren müssen wir die *Dauer* der Animation in Millisekunden angeben. Jeder Abschnitt der Animation – der Wechsel von dunkel zu hell und der Wechsel von hell zu dunkel – soll jeweils eine Sekunde dauern. Insgesamt sieht unser Code für die Animationsoptionen also wie folgt aus:

Listing 5–3: Animationsoptionen für die Pulsation

```
const options = {
  easing    : 'inOutSine',
  metronomic: true,
```

```
  loop      : true,
  keyFrames : [0, 255],
  duration  : 1000
};
```

Probieren wir nun aus, wie dies in der Praxis aussieht. Dafür können Sie den Aufbau aus den letzten Experimenten verwenden (auch wenn die erste LED dabei nicht genutzt wird). Erstellen Sie die neue Datei *animate-LED.js*. Kopieren Sie den Code aus *experiment-LED.js* in *animate-LED.js*, um einen Ausgangspunkt zu haben, und bearbeiten Sie ihn so, dass Sie ein einziges Objekt für die LED an Pin 3 erstellen. Fügen Sie nach der Instanziierung des `Led`-Objekts das `options`-Objekt hinzu. Insgesamt sieht der Code nun wie folgt aus:

Listing 5–4: *animate-led.js*

```
const five = require('johnny-five');
const board = new five.Board();

board.on('ready', () => {
  const pulsingLED = new five.Led(3);
  const options = {
    easing    : 'inOutSine',
    metronomic: true,
    loop      : true,
    keyFrames : [0, 255],
    duration  : 1000
  };
  // ...
});
```

Nun folgen Schritt 2 und 3: Sie erstellen ein `Animation`-Objekt und starten es mit `enqueue`.

Listing 5–5: Das `Animation`-Objekt instanziieren und starten

```
const five = require('johnny-five');
const board = new five.Board();

board.on('ready', () => {
  const pulsingLED = new five.Led(3);
  const options = {
    easing    : 'inOutSine',
    metronomic: true,
    loop      : true,
    keyFrames : [0, 255],
    duration  : 1000
  };
  const animation = new five.Animation(pulsingLED);
  animation.enqueue(options);
});
```

Übergibt die zu animierende Zielkomponente, in diesem Fall die LED (`pulsingLED`).

Stellt die Animation in die Warteschleife und übergibt ihr die Animationsoptionen.

Führen Sie das Skript jetzt aus:

```
$ node animate-LED.js
```

Tatsächlich erhalten wir das gewünschte Ergebnis. Allerdings habe ich Ihnen hier eine übermäßig komplizierte Vorgehensweise gezeigt, denn der Code im folgenden Listing macht genau das Gleiche:

Listing 5–6: Eine einfachere Möglichkeit, um eine LED pulsieren zu lassen

```
const five = require('johnny-five');
const board = new five.Board();

board.on('ready', () => {
  const pulsingLED = new five.Led(3);
  pulsingLED.pulse(1000);
});
```

`Led.prototype.pulse` nimmt eine Dauer in Millisekunden entgegen (Standardwert 1000).

Die zugrunde liegende Implementierung von `pulse` ähnelt unserem vorherigen Code, aber diese Animation ist so gebräuchlich, dass für Johnny-Five-Benutzer eine einfachere Methode erstellt wurde.

Pulsierende LEDs bieten eine gute Möglichkeit, um die Aufmerksamkeit zu erregen, ohne zu störend zu wirken. Unsere jetzigen Kenntnisse erlauben es uns, mit nur wenigen Codezeilen einen einfachen Timer zu erstellen, der zu pulsieren beginnt, wenn die Zeit abgelaufen ist. Den Code dafür finden Sie im folgenden Listing.

Listing 5–7: Der einfachste Timer der Welt


```
const five = require('johnny-five');
const board = new five.Board();

board.on('ready', () => {
  const pulsingLED = new five.Led(3);
  const timerLength = 10000;
  setTimeout(() => {
    pulsingLED.pulse();
  }, timerLength);
});
```

Legt die Dauer in Millisekunden fest (hier zehn Sekunden).

Richtet ein Timeout ein, sodass die LED nach zehn Sekunden zu pulsieren beginnt.

Dieser Timer ist natürlich nur von begrenztem Nutzen. So können Sie weder die Ablaufdauer noch den Startzeitpunkt ändern noch einen neuen Timer starten. Aber keine Sorge, wir werden schon bald für Verbesserungen sorgen!

Eingänge mit dem LED-Ausgang verbinden

Ausgänge sind natürlich viel interessanter, wenn sie auf Eingaben reagieren. Es sind gerade die Verbindungen zwischen Eingängen und Ausgängen, die das Internet der Dinge ausmachen.