4 Schnelleinstieg

Es ist immer besser, praktische Erfahrungen zu sammeln, als nur über etwas zu lesen – und vor allem macht es mehr Spaß. Dieses Kapitel soll Sie mit der Verwendung eines Steckbretts und einiger elektronischer Bauteile vertraut machen. Sie lernen hier, wie Sie mit Ihrem Arduino oder Ihrem Raspberry Pi LEDs und Motoren steuern.

Steckbretter

Wenn Sie externe elektronische Geräte (z. B. einen Motor) mit einem Raspberry Pi oder Arduino verbinden wollen, ist es oft nicht möglich, sie direkt an die Platine anzuschließen, denn damit alles richtig funktioniert, brauchen Sie auch noch einige weitere elektronische Bauteile. Und selbst wenn Sie nur eine einzige LED aufleuchten lassen wollen, benötigen Sie irgendeine Möglichkeit, diese LED mit dem Raspberry Pi oder dem Arduino zu verbinden.

Genau das Richtige, um solche Verbindungen ohne Löten herzustellen, sind sogenannte *Steckbretter*. Entwickelt wurden sie als Werkzeuge für Elektronikingenieure, die darauf Prototypen ihrer Neuentwicklungen bauen konnten, bevor sie sie in eine dauerhaftere Form überführten. Steckbretter machen es möglich, mit elektronischen Bauteilen zu experimentieren und eigene Projekte zu konstruieren, ohne etwas löten zu müssen.

Abbildung 4–1 zeigt ein Steckbrett mit den Bauteilen für das Experiment zur Motorsteuerung weiter hinten in diesem Kapitel. Wie Sie sehen, hält das Steckbrett die Bauteile und Kabel fest und stellt Verbindungen her.

Das in diesem Buch verwendete Steckbrett verfügt über 400 Kontakte. Es gibt auch größere und kleinere Modelle, aber diese Größe ist für die Projekte und Experimente in diesem Buch genau richtig.

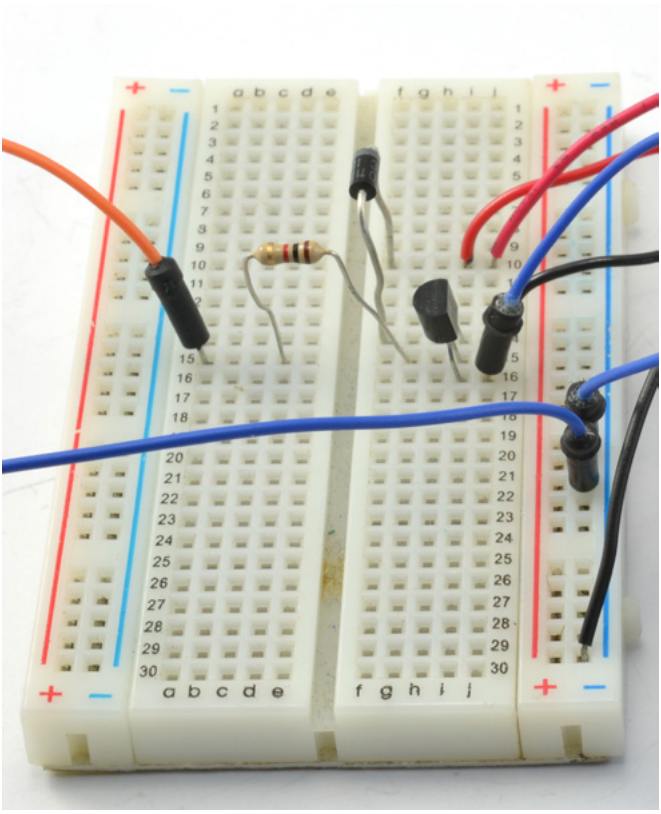

Abb. 4–1 *Verwendung eines Steckbretts*

Diese Art von Steckbrett weist an den langen Kanten jeweils zwei Reihen von Kontakten auf, die gewöhnlich mit roten und blauen Linien gekennzeichnet sind. Die Löcher in einer Reihe sind unter der Kunststoffoberfläche leitend miteinander verbunden. Sie können diese Reihen zwar auch für andere Zwecke verwenden, aber gewöhnlich werden sie für den positiven und negativen Anschluss der Stromversorgung benutzt.

Der Hauptteil des Steckbretts ist in zwei »Bänke« mit Reihen von je fünf Löchern aufgeteilt. Alle fünf Löcher einer Reihe sind jeweils durch eine Klammer unterhalb der Plastikoberfläche leitend miteinander verbunden. Um den Kontakt eines Bauteils mit dem Kontakt eines anderen Bauteils zu verbinden, müssen Sie die beiden Kontakte lediglich in Löcher derselben Reihe einstecken.

Wie funktioniert ein Steckbrett?

Unter den Löchern in dem Kunststoffgehäuse befinden sich Metallklammern, die Drähte und die Kontakte von Bauteilen festhalten sollen.

In Abbildung 4–2 sehen Sie ein auseinandergebautes Steckbrett, bei dem eine dieser Klammern entfernt wurde, sodass Sie sich eine Vorstellung von dem Aufbau unter dem Plastik machen können. Allerdings sollten Sie darauf verzichten, ein Steckbrett auf diese Weise auseinanderzubauen, da es nach dem Zusammenbau erfahrungsgemäß nie wieder ganz zu seiner alten Form zurückfindet.

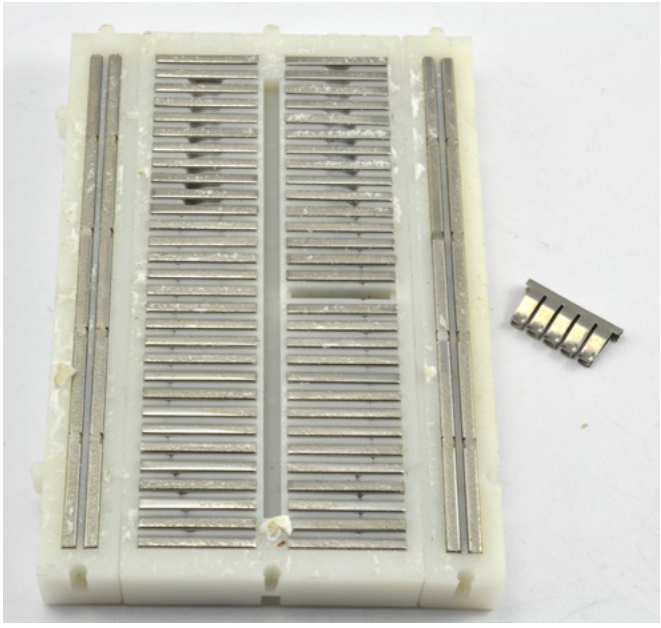

Abb. 4–2 *Der innere Aufbau eines Steckbretts mit einer ausgebauten Klammer*

Ein Steckbrett an den Arduino anschließen

Die GPIO-Anschlüsse des Arduino werden verwirrenderweise oft als »Pins«, also »Stifte«, bezeichnet, obwohl es sich in Wirklichkeit um Buchsen handelt. Um sie mit einer Reihe auf dem Steckbrett zu verbinden, verwenden Sie Schaltdraht mit Steckern, wie in Abbildung 4–3 gezeigt.

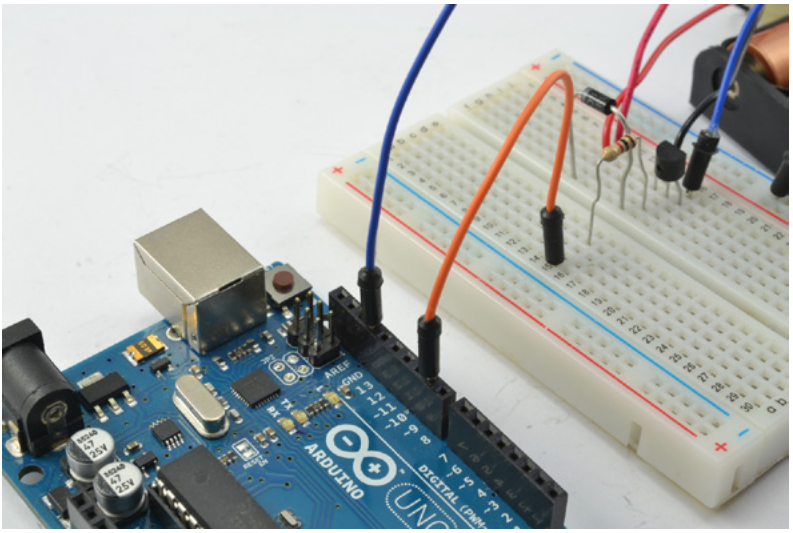

Abb. 4–3 *Verbindung zwischen Arduino und Steckbrett*

Dabei handelt es sich um biegsame Kabel mit kleinen Steckern an beiden Enden. Es ist ratsam, sich einen guten Vorrat davon in verschiedenen Längen anzulegen.

Es gibt Schaltdrahtsortimente in verschiedenen Farben und Größen zu kaufen (siehe Anhang A). Es lohnt sich, sich eine Grundausstattung mit Steckbrett und verschiedenen Schaltdrähten anzuschaffen. Verschiedene Farben machen es einfacher, zu erkennen, wo die Verbindungen verlaufen, insbesondere, wenn Sie sehr viele Verbindungen auf dem Steckbrett haben.

Ein Steckbrett an den Raspberry Pi anschließen

Die GPIO-Anschlüsse des Raspberry Pi sind tatsächlich Pins und keine Buchsen. Daher können Sie die Schaltdrähte für den Arduino auf dem Raspberry Pi nicht einsetzen. Stattdessen benötigen Sie Schaltdraht, an dessen einem Ende sich eine Buchse befindet, die auf einen Pin des GPIO-Headers auf dem Raspberry Pi passt, und am anderen Ende ein für das Steckbrett geeigneter Stecker.

Abbildung 4–4 zeigt, wie Sie mit solchem Schaltdraht Verbindungen von den GPIO-Pins des Raspberry Pi zu einer Reihe auf einem Steckbrett herstellen.

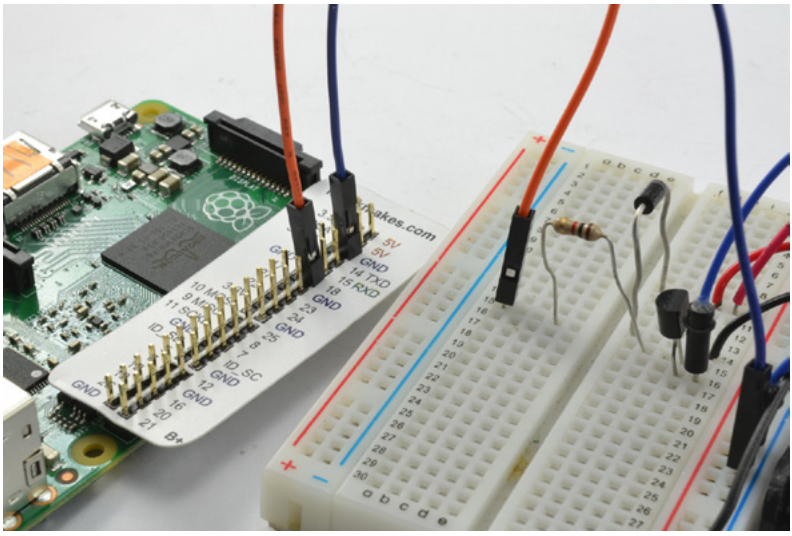

Abb. 4–4 *Verbindung zwischen Raspberry Pi und Steckbrett*

Die GPIO-Pins auf dem Raspberry Pi identifizieren

Die GPIO-Pins des Raspberry Pi sind auf der Platine nicht beschriftet. Um nicht mühselig die Pins zählen zu müssen, können Sie eine GPIO-Schablone verwenden, die über den Header gezogen wird. Die Schablone in Abbildung 4–4 ist ein Raspberry Leaf, das Sie bei Adafruit und bei MonkMakes.com bekommen. Es gibt jedoch auch noch andere Schablonen.

Die Software herunterladen

Die gesamte Software für dieses Buch – sowohl die Arduino-Sketche als auch die Python-Programme für den Raspberry Pi – ist auf dem GitHub-Repository zu diesem Buch erhältlich (*https://github.com/simonmonk/make_action*).

Wie Sie die Arduino-Sketche von Ihrem regulären Computer auf den Arduino hochladen, erfahren Sie im Abschnitt »Der Code zu diesem Buch« in Kapitel 2.

Um die Python-Programme auf den Raspberry Pi zu laden, folgen Sie den Anweisungen aus dem Abschnitt »Der Code zu diesem Buch« in Kapitel 3.

Experiment: Eine LED steuern

Traditionell wird als erste Übung mit dem Arduino eine LED zum Blinken gebracht. In unserem ersten Experiment werden wir genau das tun, und zwar zunächst mit dem Arduino und dann mit dem Raspberry Pi.

Dies ist ein sehr einfaches Projekt, um warm zu werden. Auf dem Steckbrett müssen nur zwei Bauteile hinzugefügt werden, nämlich die LED und ein Widerstand. Bei allen LEDs ist ein Widerstand erforderlich, um den Strom zu begrenzen, der sie durchfließt. Mehr darüber erfahren Sie in Kapitel 6.

Stückliste

Unabhängig davon, ob Sie das Experiment mit dem Raspberry Pi oder dem Arduino durchführen, benötigen Sie die folgenden Teile:

Teil	Bezugsquelle	alternative Bezugsquelle
Rote LED	Adafruit: 297 Sparkfun: COM-09590	Conrad: L 53 HD Flikto: COM-12062
Widerstand 470 Ω, 1/4 W	Mouser: 291-470-RC	Mouser: 291-470-RC
Steckbrett mit 400 Kontakten	Adafruit: 64	Conrad: EIC-801 Flikto: flik.to/189
Schaltdraht mit Steckern an beiden Enden (nur Arduino)	Adafruit: 758	Flikto: 758 electronic-shop: 758
Schaltdraht mit Buchse und Stecker (nur Pi)	Adafruit: 826	Flikto: PRT-12794 electronic-shop: 826

Diese Tabelle gibt auch Bezugsquellen und die jeweiligen Bestellnummern für die Teile an. Weitere Informationen über sämtliche in diesem Buch verwendeten Bauteile erhalten Sie in Anhang A.

Schaltungsaufbau

Den Schaltungsaufbau auf dem Steckbrett sehen Sie in Abbildung 4–5. Sowohl für den Arduino als auch den Raspberry Pi verwenden Sie dieselbe Schaltung. Der Anschluss an die Platine unterscheidet sich jedoch.

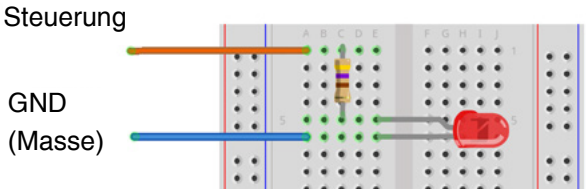

Abb. 4–5 *Schaltungsaufbau für die Steuerung einer LED*

Wie Sie in Abbildung 4–5 sehen, fließt der Strom vom Arduino bzw. Raspberry Pi erst durch den Widerstand und dann durch die LED, um sie zum Leuchten zu bringen.

Es spielt keine Rolle, wie herum Sie den Widerstand einbauen. Bei der LED muss jedoch der positive Kontakt zur Oberkante des Steckbretts weisen. Dieser Kontakt ist etwas länger als der negative. Außerdem ist das Gehäuse der LED an der Seite des negativen Kontakts abgeflacht.

Mehr über Widerstände erfahren Sie in Kapitel 5.

Verbindungen mit dem Arduino

Schließen Sie die Schaltung wie in Abbildung 4–6 gezeigt an die Arduino-Buchsen GND und D9 an. Ein Foto des Aufbaus sehen Sie in Abbildung 4–7.

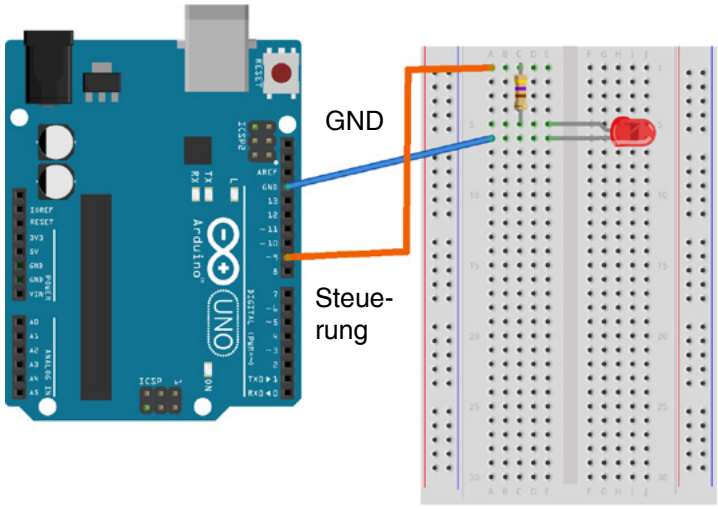

Abb. 4–6 *Schaltungsaufbau für die Steuerung einer LED mit dem Arduino*

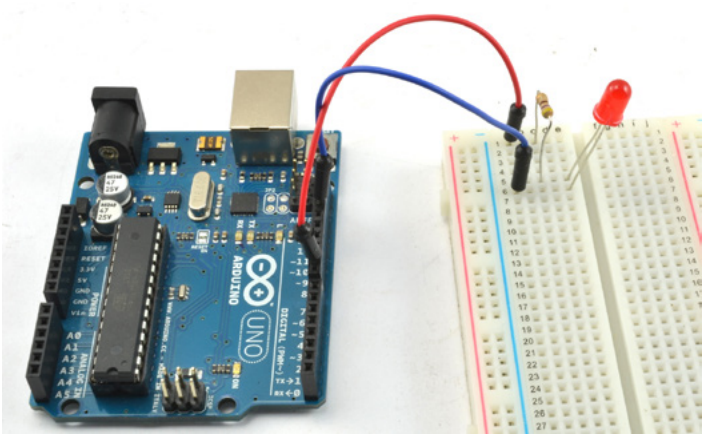

Abb. 4–7 *Anschluss des Steckbretts an den Arduino*

Die Software für den Arduino

Den Arduino-Sketch für dieses Experiment finden Sie im Verzeichnis *arduino/experiments/on_off_control* an dem Speicherort, an dem Sie den heruntergeladenen Code zu diesem Buch abgelegt haben (siehe »Die Software herunterladen« weiter vorn in diesem Kapitel).

Das Programm schaltet die LED fünf Sekunden lang ein, dann zwei Sekunden lang aus und wiederholt diesen Vorgang dann. Der komplette Code sieht wie folgt aus:

```
const int controlPin = 9; // ❶

void setup() { // ❷
   pinMode(controlPin, OUTPUT);
}

void loop() { // ❸
   digitalWrite(controlPin, HIGH);
   delay(5000);
   digitalWrite(controlPin, LOW);
   delay(2000);
}
```

❶ In der ersten Zeile wird die Konstante `controlPin` als Pin 9 definiert. Auf dem Arduino Uno gibt es zwar sowohl die Digitalpins 0 bis 13 als auch die Analogpins 0 bis 5, doch vereinbarungsgemäß wird im Code auf Digitalpins nur mit der Zahl verwiesen (hier also 9). Wollen Sie einen der sechs Analogpins ansprechen, stellen Sie der Pinnummer den Buchstaben A voran.

❷ Die Funktion `setup()` richtet den Pin mit `pinMode` als digitalen Ausgang ein.

❸ Die Funktion `loop()`, die endlos wiederholt wird, setzt den Steuerpin 9 (`controlPin`) zunächst auf `HIGH` (5 V), um die LED einzuschalten. Anschließend legt sie eine Pause von 5000 ms (5 s) ein und setzt `controlPin` danach auf `LOW` (0 V), um die LED wieder abzuschalten. Die nächste Zeile sorgt für eine Verzögerung von 2 s. Daraufhin beginnt die Schleife von vorn.

Experimentieren mit dem Arduino

Laden Sie das Programm auf den Arduino hoch. Sobald der Arduino im Rahmen dieses Vorgangs neu startet, wird der Code ausgeführt und die LED beginnt zu blinken.

Sollte die LED nicht blinken, überprüfen Sie alle Verbindungen. Vergewissern Sie sich auch, dass die LED richtig herum eingesetzt ist (der längere Kontakt muss zur Oberkante des Steckbretts weisen).

Probieren Sie auch andere Zahlenwerte in den `delay()`-Funktionen aus, um zu variieren, wie lange die LED in den einzelnen Zyklen eingeschaltet bleibt. Jedes Mal, wenn Sie eine solche Änderung vorgenommen haben, müssen Sie das Programm neu hochladen.

Verbindungen mit dem Raspberry Pi

Anders als beim Arduino weisen die GPIO-Pins auf dem Raspberry Pi keine Beschriftungen auf, die Ihnen sagen, welcher Pin welcher ist. Um die einzelnen Pins zu identifizieren, haben Sie zwei Möglichkeiten: Sie können das Schema der Pinbelegung (siehe Anhang B) zurate ziehen und die Pins auf der Platine abzählen, um den richtigen zu finden, oder aber eine Schablone nutzen, die über die Stiftleiste gezogen wird, beispielsweise das Raspberry Leaf aus Abbildung 4–8. In Abbildung 4–9 sehen Sie den Schaltungsaufbau auf dem Steckbrett und die Verbindungen zum Raspberry Pi.

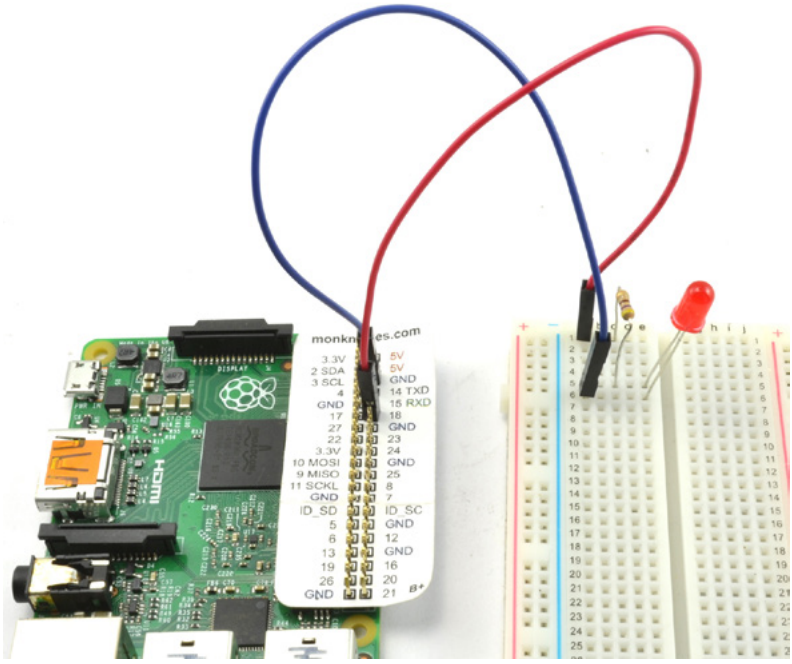

Abb. 4–8 *Anschluss des Steckbretts an den Raspberry Pi*

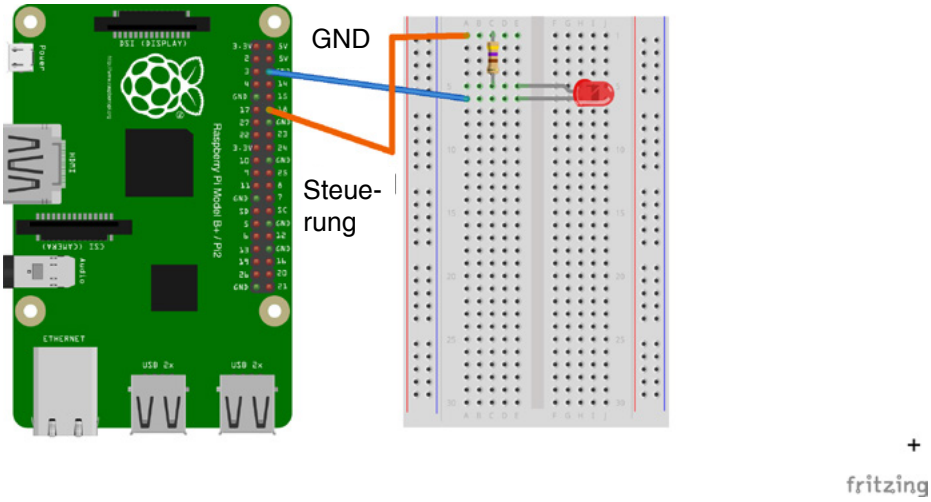

Abb. 4–9 *Schaltungsaufbau für die Steuerung einer LED mit dem Raspberry Pi*

Die Software für den Raspberry Pi

Da Sie Programme auf dem Raspberry Pi selbst schreiben können, brauchen Sie für die Programmierung keinen zweiten Computer. Verwenden Sie für dieses Experiment das folgende Programm, das Sie im Verzeichnis *python/experiments* in der Datei *on_off_control.py* finden:

```
import RPi.GPIO as GPIO # ❶
import time             # ❷

GPIO.setmode(GPIO.BCM)  # ❸

control_pin = 18        # ❹
GPIO.setup(control_pin, GPIO.OUT)

try:            # ❺
   while True: # ❻
      GPIO.output(control_pin, False) # ❼
      time.sleep(5)
      GPIO.output(control_pin, True)
      time.sleep(2)

finally:
   print("Cleaning up")
   GPIO.cleanup()
```

Dieses Programm ist seinem Gegenstück auf dem Arduino sehr ähnlich:

❶ Für den Zugriff auf die GPIO-Pins des Raspberry Pi gibt es die Python-Bibliothek `RPi.GPIO`, die der Raspberry Pi-Fan Ben Croston geschrieben hat. In der ersten Codezeile wird diese Bibliothek importiert, sodass sie in dem Programm verwendet werden kann. In allen jüngeren Versionen der Standarddistribution von Raspbian ist diese Bibliothek vorinstalliert, sodass Sie sie nur dann zu installieren brauchen, wenn Sie eine alte Raspbian-Version verwenden. In diesem Fall besteht die einfachste Möglichkeit zur Installation darin, Ihr System zu aktualisieren, was ohnehin angebracht wäre, indem Sie in einem Terminalfenster den folgenden Befehl eingeben:

```
$ sudo apt-get upgrade
```

❷ Sie müssen auch die Bibliothek `time` importieren, da sie für die Verzögerungen zwischen dem Ein- und Ausschalten der LED verwendet wird.

❸ In jedes Programm, das GPIO-Pins steuert, müssen Sie die Zeile `GPIO.setmode(GPIO.BCM)` einfügen, bevor Sie den Pinmodus festlegen oder die Pins in irgendeiner Weise verwenden. Dieser Befehl teilt der GPIO-Bibliothek mit, dass die Pins mit ihren Broadcom-Namen (BCM) bezeichnet werden und nicht anhand ihrer Positionen. Die Bibliothek `RPi.GPIO` kann zwar mit beiden Bezeichnungssystemen umgehen, aber da die Broadcom-Benennung weiter verbreitet ist, verwenden wir sie auch in diesem Buch.

Anders als bei der Programmierung des Arduino gibt es die Funktionen `setup()` und `loop()` hier nicht. Alles, was auf dem Arduino in `setup()` gehört, befindet sich hier am Anfang des Programms. Eine `while`-Endlosschleife führt die Dinge aus, die auf dem Arduino in `loop()` stehen würden.

❹ Die Variable `control_pin` legt GPIO-Pin 18 als denjenigen fest, der zur Steuerung der LED verwendet werden soll. Anschließend wird er mit `GPIO.setup()` als Ausgang definiert.

❺ Jetzt kommen wir zum Gegenstück der `loop()`-Funktion des Arduino. Der Code dafür ist in eine `try/finally`-Konstruktion eingeschlossen. Wenn in dem Programm Fehler auftreten oder wenn Sie es abbrechen, indem Sie im Terminalfenster Strg + C drücken, kann dadurch der Aufräumcode im `finally`-Block ausgeführt werden.

Sie könnten auch auf diesen Aufräumcode verzichten und einfach nur die `while`-Schleife ausführen. Da der Aufräumcode jedoch alle GPIO-Pins auf den sicheren Zustand als Eingänge zurücksetzt, senkt er die Wahrscheinlichkeit dafür, dass ein versehentlicher Kurzschluss oder Verkabelungsfehler auf dem Steckbrett den Raspberry Pi beschädigt.

❻ Die Bedingung der `while`-Schleife lautet `True`. Das mag merkwürdig erscheinen, ist aber eine übliche Maßnahme in Python, um dafür zu sorgen, dass Code unendlich oft ausgeführt wird. Das Programm führt die Befehle in der `while`-Schleife immer wieder aus, bis Sie das Programm mit Strg + C abbrechen oder den Raspberry Pi von der Stromversorgung trennen.

❼ Der Code innerhalb der Schleife ähnelt sehr stark seinem Gegenstück auf dem Arduino. Der GPIO-Pin wird auf `True` (also `high`) gesetzt und nach einer Verzögerung von fünf Sekunden auf `False` (`low`). Nach einer weiteren Verzögerung von zwei Sekunden beginnt der Zyklus von vorn.

Experimentieren mit dem Raspberry Pi

Für den Zugriff auf die GPIO-Pins sind Superuser-Rechte in Linux erforderlich. Um das Programm auszuführen, wechseln Sie in das Verzeichnis, in dem sich die Datei *on_off_control.py* befindet, und geben dann folgenden Befehl ein:

```
$ sudo python on_off_control.py
```

Durch den Befehl `sudo` am Anfang wird das Programm mit Superuser-Rechten ausgeführt. Wenn Sie der LED lange genug beim Blinken zugesehen haben, drücken Sie Strg + C, um das Programm zu beenden.

Der Code im Vergleich

Die grobe Struktur der beiden Programme ist zwar ähnlich, doch unterscheiden sich natürlich der in Arduino C und der in Python geschriebene Code. Auch die Benennung von Variablen und Funktionen wird unterschiedlich gehandhabt: In C werden die einzelnen Wortbestandteile zusammengeschrieben, wobei alle Wörter außer dem ersten mit einem großen Anfangsbuchstaben beginnen, während die Bestandteile in Python durch Unterstriche getrennt werden (was auch als »snake_case« bezeichnet wird).

Tabelle 4–1 führt die wichtigsten Unterschiede zwischen den beiden Programmen auf.