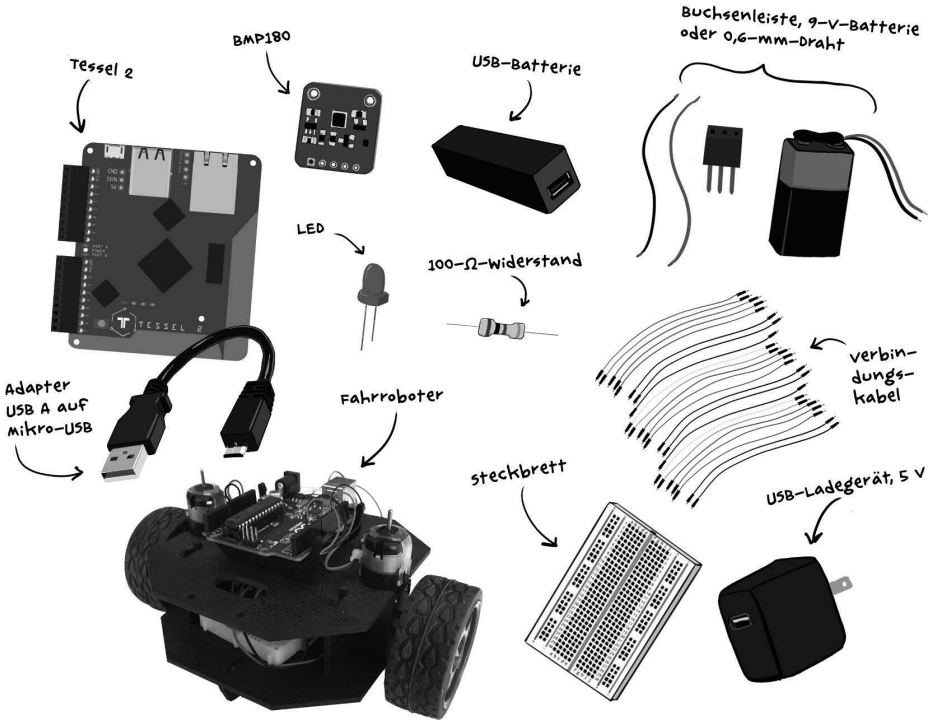
8 Projekte ohne Kabel

Inhalt dieses Kapitels

- In Hobbyelektronikprojekten auf die Kabelverbindung verzichten
- E/A-Plug-ins für verschiedene Plattformen
- Die Entwicklungsplattform Tessel 2
- Anpassung an einen Logikpegel von 3,3 V und unterschiedliche Pinkonfigurationen
- Code drahtlos auf dem Tessel 2 bereitstellen
- Mithilfe von Node.js und npm komplexere Software für den Tessel 2 schreiben
- Batterien nutzen, um ganz auf Kabel zu verzichten

Für dieses Kapitel benötigen Sie:

- 1 Tessel 2
- 1 Kabel USB A auf USB-Mikro
- 1 Standard-LED beliebiger Farbe
- 1 100-Ω-Widerstand
- 1 Multisensor-Erweiterungsplatine BMP180 von Adafruit
- Eine der folgenden Gruppen von Teilen:
- 3 Buchsenleisten
- 2 Längen eindrahtiges Kabel 0,6 mm
- 9-V-Batterie mit Clip
- Lötkolben und Zubehör
- 1 5-V-USB-Ladegerät o.Ä. für den Tessel 2
- 1 USB-Batterie (oft unter der Bezeichnung *Powerbank* vertrieben)
- Verbindungskabel
- 1 Steckbrett mit ca. 400 Kontakten
- 1 Fahrroboter (Motortreiberschaltung und Chassis) aus Kapitel 6



Unsere Projekte sind immer aufwändiger und leistungsfähiger geworden, unterliegen aber alle einer großen Einschränkung: Sie sind physisch an einen Computer gebunden. Unsere Johnny-Five-gesteuerten Arduino-Uno-Projekte waren nicht physisch unabhängig, sondern brauchen einen Hostcomputer, von dem sie ihre Anweisungen und den Strom beziehen. Bei manchen Arten von Problemen ist diese Host-Client-Bindung kein Problem, aber um autonome, eigenständige, mit JavaScript gesteuerte Projekte zu erstellen – also die Kabel loszuwerden –, müssen wir uns nach neuer Hardware umsehen.

Der Mikrocontroller ATmega328P des Arduino Uno weist zu starke Einschränkungen auf, um ein komplettes Betriebssystem zu beherbergen oder JavaScript nativ auszuführen. Um den Uno mit JavaScript zu steuern, brauchen Sie einen leistungsfähigeren externen *Host*, der den JavaScript-Code für die Platine ausführt.

Auf dem Host muss die Logik des JavaScript-Programms in Anweisungen übersetzt werden, die der Mikrocontroller verstehen kann. Diese Anweisungen müssen dann an den Uno übertragen werden, der als *Client* dient, während die Daten, die vom Uno kommen – z.B. Sensormesswerte – zur Verarbeitung an den Host geschickt werden (siehe Abb. 8–1).

Bei Johnny-Five sind die Formatierung und Übertragung von Anweisungen und Daten zwischen Host und Client die Aufgabe der *E/A-Schicht*.

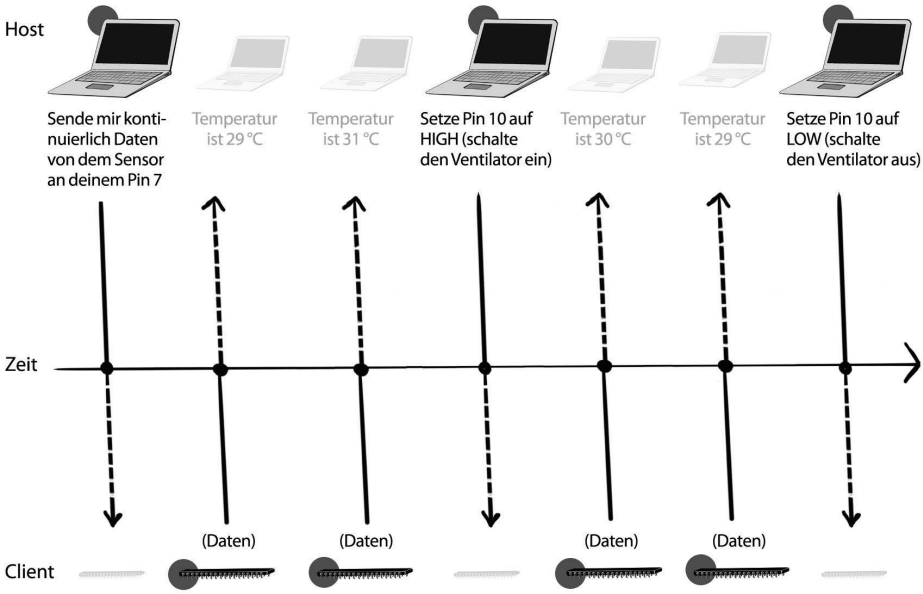


Abb. 8–1 *Diese Skizze aus Kapitel 1 zeigt die theoretische Host-Client-Kommunikation für einen automatischen Ventilator. In unseren bisherigen Experimenten hat der Arduino als Thin Client fungiert und Ihr Computer als Host.*

Arduino-Platinen einschließlich des Uno stellen für Johnny-Five einen Sonderfall dar: Firmata stellt die E/A-Schicht bereit und ist in Johnny-Five standardmäßig verfügbar. Das heißt, dass Firmata (über USB) für die Ein- und Ausgabe genutzt wird, sofern Sie bei der Instanziierung des `Board`-Objekts keine anderen Anweisungen gegeben haben (was über eine Option möglich ist). In den bisherigen Kapiteln haben Sie immer die Standard-E/A genutzt.

8.1 Gründe für die bisherige Kabelabhängigkeit

Bisher haben wir in unseren Projekten eine USB-Kabelverbindung verwendet, weil sie sowohl als Datenübertragung als auch als Stromversorgung fungieren kann. Um Ihre Projekte kabellos zu machen, müssen Sie diese beiden Anforderungen – E/A und Stromversorgung – auf andere Weisen erfüllen. Sehen wir uns also an, was dabei hinter den Kulissen geschieht.

Datenaustausch, die E/A-Schicht und E/A-Plug-ins

Johnny-Five legt die APIs und das logische Verhalten der Komponenten fest, doch wie die Daten und Anweisungen konkret zwischen der Johnny-Five-Anwendung und der Hardware ausgetauscht werden, ist Sache der *E/A-Schicht*. Da Johnny-Five die Einzelheiten kompatiblen *E/A-Plug-ins* überlässt, bleibt es selbst plattformunabhängig.

Bis jetzt haben Sie noch keine E/A-Plug-ins verwendet, da die gängigen Arduino-Platinen, darunter auch der Uno, für Johnny-Five einen Sonderfall darstellen. Wenn Sie ein `Board`-Objekt instanziieren, ohne anzugeben, welches E/A-Plug-in verwendet werden soll, greift Johnny-Five automatisch auf Firmata über USB zurück. Darum funktioniert die E/A einfach, ohne dass wir etwas dazu tun müssen. Im Gegensatz zu den E/A-Plug-ins für andere Plattformen, die einzeln installiert werden müssen, ist Firmata im Lieferumfang von Johnny-Five enthalten (technisch ausgedrückt, handelt es sich um eine Abhängigkeit). Das mag zwar so aussehen, als seien Firmata und die Einzelheiten der E/A Bestandteil des Johnny-Five-Codes, doch das ist nicht der Fall. Wenn Sie Johnny-Five auf anderen Plattformen einsetzen wollen, müssen Sie die entsprechenden E/A-Plug-ins installieren. Mehr darüber erfahren Sie in Kürze.

In der bisherigen Host-Client-Anordnung mit Johnny-Five und dem Arduino Uno bildete das USB-Kabel die Nabelschnur für den seriellen Austausch der Firmata-Nachrichten. Das war einer der Gründe für die physische Anbindung.

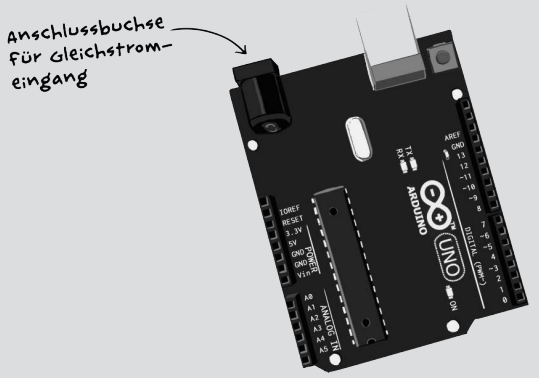
USB als Stromquelle

Die Schaltkreise in einem Projekt müssen mit Strom versorgt werden. Die bisher verwendete USB-Verbindung liefert konstante 5 V. Außer für einige induktive Elemente wie Motoren und Servomotoren, die höhere Stromstärken oder Spannung benötigen als die Entwicklungsplatine liefern kann, haben Sie für die Stromversorgung ebenfalls das USB-Kabel genutzt.

Gleichstrom für Entwicklungsplatinen

Der Arduino Uno verfügt über eine eingebaute Gleichstrombuchse, mit der Sie ihn an einen Netzadapter anschließen können. Solche Adapter sind allgegenwärtig. Sie wandeln die Wechselspannung des Stromnetzes in Gleichspannung um, welche viele elektronische Komponenten benötigen.

Die Stecker moderner Netzadapter erlauben den Anschluss an alle möglichen Geräte von Anrufbeantwortern bis zu Entwicklungsplatinen wie den Uno.

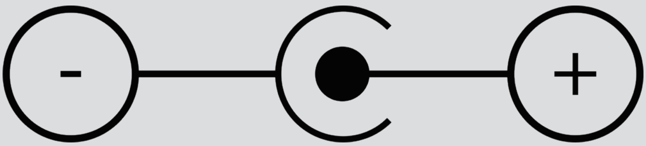


Der Uno verfügt über eine Buchse für den Gleichstromanschluss. →

Die meisten Entwicklungsplatinen (unter anderem auch der Uno) verfügen über eingebaute *Spannungsregler* für den Stromeingang. Dadurch ist es möglich, den Uno über die Gleichstrombuchse mit Spannungen zwischen 9 V und 12 V zu versorgen. Die Platine regelt diese Eingangsspannung selbst auf die erforderlichen 5 V herunter.

Spannungsregler müssen mit einer Spannung versorgt werden, die höher als die Zielspannung ist, denn wie bei einer LED fällt auch bei einem Regler etwas Spannung ab: Die Regelschaltung der Platine verbraucht einen Teil der Eingangsspannung. Im Fall des Uno sind mindestens 7 V erforderlich, um konstante 5 V zu erhalten.

Jeder 9- bis 12-V-Gleichstromadapter mit positivem Innenleiter und einem Stecker der Größe 5,5 mm x 2,5 mm (die am weitesten verbreitete Größe) eignet sich für die Versorgung des Uno.



Dieses Symbol bedeutet, dass der Adapter einen positiven Innenleiter hat, d.h., die Spitze des Steckers ist positiv, die Hülse negativ. Die Mehrzahl der Gleichstromadapter weist Stecker mit positivem Innenleiter auf. Diesen Typ benötigen Sie für die Stromversorgung des Uno und der meisten anderen Entwicklungsplatinen.

Nicht bei allen Adaptern ist ein Polungssymbol aufgedruckt. Das ist schlecht, denn man kann nicht unbedingt davon ausgehen, dass ein Adapter einen positiven Innenleiter hat. Zwar ist die Bauform mit positivem Innenleiter weit verbreitet, aber sie ist trotzdem keine Norm.

Für meine Projekte halte ich eine Reihe von Gleichstromadaptern und 5-V-USB-Ladegeräten auf Lager. Ich verwende sie schon lange nicht mehr für die Geräte, für die sie ursprünglich gedacht waren, so erfüllen sie in ihrem zweiten Leben noch einen praktischen Zweck als Stromquelle für verschiedene Prototypen und Projekte.

Möglichkeiten zur drahtlosen Kommunikation in Projekten

Wie Sie gesehen haben, gibt es vor allem zwei Gründe für die bisherige Kabelanbindung unserer Projekte: Datenaustausch und Stromversorgung. Als Erstes werden wir uns verschiedene Möglichkeiten ansehen, um in unseren Projekten auf physische Datenverbindungen zu verzichten. Mit der Stromversorgung beschäftigen wir uns später.

Drahtlose Host-Client-Einrichtung

Manche Entwicklungsplatinen mit einfachen Mikrocontrollern bieten Möglichkeiten für einen drahtlosen Anschluss wie WLAN und Bluetooth.

Zur Ausführung der JavaScript-Logik ist immer noch ein Host erforderlich, aber die Daten und Anweisungen können jetzt drahtlos zwischen ihm und dem Client übertragen werden, sodass die physische USB-Datenverbindung nicht mehr nötig ist.

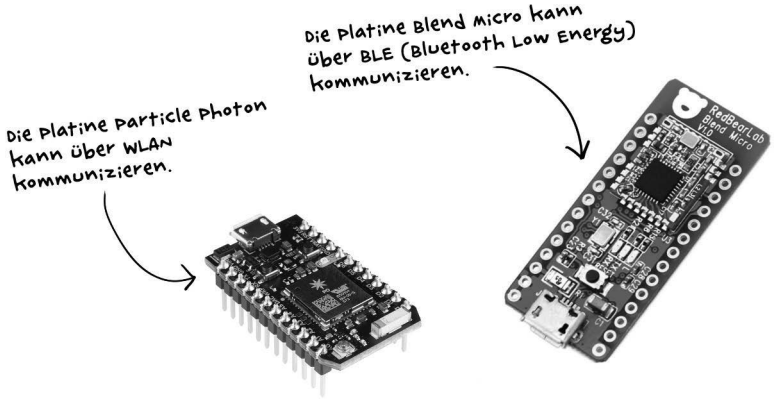


Abb. 8–2 *Diese Platinen verfügen ebenso wie der Uno nur über einfache Mikrocontroller, können aber drahtlos kommunizieren.*

Beispielsweise können Sie mit Johnny-Five drahtlos auf den Blend Micro zugreifen, indem Sie das E/A-Plug-in `blend-micro` sowie die Firmware BLEFirmata auf der Platine verwenden. Eine weitere BLE-fähige kleine Platine, den Espruino Puck.js, lernen Sie in Kapitel 12 kennen.

Eingebettetes JavaScript

Es gibt Entwicklungsplatinen mit einfachen bis mittleren Mikrocontrollern, die für die native Ausführung von JavaScript, einer Teilversion von JavaScript oder von JavaScript-artigem Code geeignet sind. Ein vollständiges Betriebssystem können die Mikrocontroller auf diesen Platinen zwar noch nicht ausführen, aber sie sind gewöhnlich sparsam im Stromverbrauch, billig und klein, was sie zu idealen Kandidaten für eingebettete Systeme macht.

Der Ablauf hängt von der jeweiligen Plattform ab (siehe Abb. 8–3). In einigen Fällen wird das von Ihnen geschriebene JavaScript-Programm in maschinennäheren und effizienteren Code kompiliert, bevor es auf das Gerät übertragen wird. Es kann aber auch sein, dass der Mikrocontroller auf der Platine in der Lage ist, zumindest eine Teilversion von JavaScript direkt auszuführen. Das bedeutet, dass Sie den Code in JavaScript schreiben, dabei aber nicht alle Merkmale der Sprache nutzen können. Plattformen mit eingeschränktem eingebetteten JavaScript sehen wir uns in Kapitel 10 genauer an.

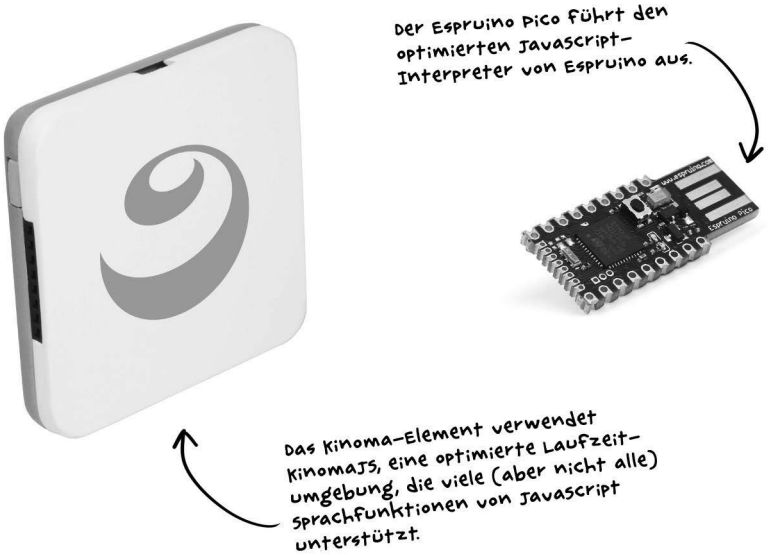


Abb. 8–3 *Geräte dieser Art haben einfache Mikrocontroller mit optimierten Varianten der JavaScript-Laufzeit bzw. eine JavaScript-ähnliche Laufzeit. Sowohl den Pico als auch das Element sehen wir uns in Kapitel 10 genauer an.*

Den Client über einen Ein-Platinen-Computer steuern

Ein-Platinen-Computer können echte Multitasking-Betriebssysteme ausführen. Das Angebot an Rechenleistung und an eingebauten Peripheriegeräten ist dabei breit gestreut. Solche Geräte können eine Menge tun, unter anderem auch Node.js ausführen, aber sie verbrauchen dementsprechend auch mehr Strom, sind teurer und größer als andere Entwicklungsplatinen für eingebettete Projekte.

Bei Ein-Platinen-Computern verwischt die Grenze zwischen Computer und Entwicklungsplatine. Einige Modelle, darunter diejenigen aus der beliebten Raspberry Pi-Serie, verbinden die Allzweckfunktion eines Computers mit den E/A-Einrichtungen von Entwicklungsplatinen. Fällt das bereits unter Ausführung eingebetteter Logik oder handelt es sich um eine Miniaturversion eines Host-Client-Aufbaus (bei der der Prozessor des Pi als Host zur E/A-Steuerung fungiert)? Außerdem ist es möglich, mit dem Raspberry Pi einen Arduino zu steuern, der mit einem USB-Anschluss des Pi verbunden ist, was nun eindeutig einen Host-Client-Aufbau im Kleinen darstellt. Mit Ein-Platinen-Computern beschäftigen wir uns eingehender in Kapitel 11.

8.2 Kabellose Projekte mit dem Tessel 2

Der Tessel 2 (*https://tessel.io/*) ist eine Open-Source-Entwicklungsplattform (sowohl die Hardware als auch die Software sind quelloffen), die auf Node.js und den Paketmanager npm ausgerichtet ist (siehe Abb. 8–4). Neben den grundlegenden Formen der E/A, mit denen wir bisher gearbeitet haben – digital, analog,

PWM, I²C usw. – bietet der Tessel auch anspruchsvollere Peripheriegeräte wie USB- und Ethernetanschlüsse und – hurra! – WLAN. (Da der Tessel 2 das einzige zurzeit erhältliche Tessel-Modell ist, werde ich ihn im Allgemeinen nur kurz als Tessel bezeichnen.)

Der Tessel ist ein interessantes und nützliches Gerät. Was das Kriterium »Kann ein vollständiges Betriebssystem ausführen« angeht, fällt er in die Kategorie der Ein-Platinen-Computer, denn er wird mit einem vorinstallierten *OpenWrt* ausgeliefert, einer Linux-Distribution, die häufig auf Routern verwendet wird.

Der Arbeitsablauf aber erinnert eher an Host-Client-Aufbauten und Geräte mit eingebettetem JavaScript: Sie schreiben Code auf Ihrem Computer und stellen ihn auf dem Tessel bereit, anstatt ihn unmittelbar auf dem Tessel zu schreiben. Obwohl der Tessel OpenWrt ausführt und einiges an praktischer Zusatzsoftware enthält, ist er doch stärker eingeschränkt als andere Ein-Platinen-Computer wie der Raspberry Pi. Er hat einen Arbeitsspeicher von nur 64 MB sowie 32 MB Flash-Speicher für Programme. Das geht zwar meilenweit über das hinaus, was Platinen mit einem ATmega328P wie der Uno bieten, reicht aber noch längst nicht an die Leistung gängiger Desktopcomputer heran.

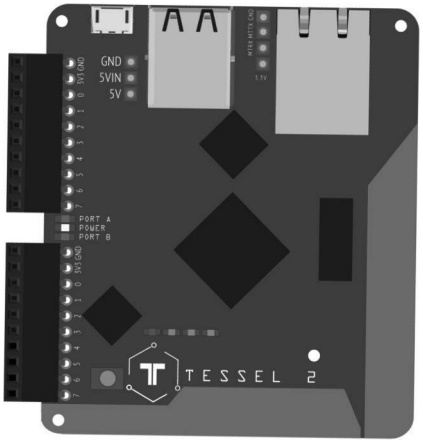


Abb. 8–4 *Die Open-Source-Entwicklungsplattform Tessel 2*

Da Sie bereits Johnny-Five-Projekte für den Uno geschrieben haben, wird Ihnen beim Umgang mit dem Tessel einiges vertraut erscheinen. Die Mechanismen sollten Ihnen bekannt vorkommen, wenn Sie schon einmal ganz allgemein etwas mit oder für Node.js entwickelt haben.

Es gibt jedoch einen entscheidenden Unterschied zwischen dem Tessel und dem Arduino Uno: Der Tessel verwendet eine Spannung von 3,3 V und nicht wie der Uno 5 V.

Tessel 2 mit 3,3 V

Der Tessel 2 arbeitet mit 3,3 V. Das müssen Sie bei der Auslegung Ihrer Schaltungen berücksichtigen und für 3,3 V geeignete Bauteile auswählen. Wenn Sie 5-V-Komponenten oder eine 5-V-Spannungsquelle an die Pins des Tessel 2 anschließen, kann die Platine dadurch beschädigt werden. Aber keine Angst, wir werden uns diese Einzelheiten im weiteren Verlauf genauer ansehen.

8.3 Den Tessel einrichten

Erforderliches Material

- 1 Tessel 2
- 1 Kabel USB A auf USB-Mikro

Um den Tessel startbereit zu machen, müssen Sie ihn wie den Uno über ein USB-Kabel an Ihren Laptop anschließen. Es gibt jedoch einige große Unterschiede. Erstens wird der von Ihnen geschriebene JavaScript-Code auf den Tessel übertragen und dort ausgeführt – der Tessel braucht keinen Host, der für ihn die Denkarbeit übernimmt. Zweitens verfügt der Tessel über eine WLAN-Einrichtung, sodass Sie ihn nach der Einrichtung nicht mehr über USB anschließen müssen. Sie können Programme auch drahtlos auf ihm bereitstellen.

Node.js LTS

Der Tessel unterstützt die LTS-Version (Long Term Support) von Node.js, die bei Version 6.11 angelangt ist, während ich diese Zeilen schreibe (und wahrscheinlich bei einer viel höheren Versionsnummer, wenn Sie sie lesen). Bei den Codebeispielen in diesem Buch wird mindestens Version 6.11 vorausgesetzt. Die Beispielskripte nutzen die in dieser Version verfügbaren Sprachmerkmale. Jüngere Node.js-Versionen als die aktuelle LTS sind möglicherweise nicht mit dem Tessel kompatibel.

Auf dem Tessel selbst läuft wahrscheinlich eine andere (ältere) Version von Node.js als auf Ihrem System. Nachdem Sie den Tessel konfiguriert haben, können Sie sich mit dem Befehl `t2 version` anzeigen lassen, welche Version von Node.js und welche Firmwareversion auf der Platine ausgeführt wird. (Mehr über den Befehl `t2` erfahren Sie in Kürze.)

Den Tessel konfigurieren

Wenn Sie die folgende Anleitung lesen, mag es so aussehen, als sei die Einrichtung ein langwieriger Vorgang, doch in Wirklichkeit nimmt er meistens nur wenige Minuten in Anspruch.