11 Schall

Schallwellen sind Verdichtungen und Verdünnungen im Medium. Ihre Bewegung kann gemessen und analysiert werden.

Der Raspberry Pi kann Schall aufzeichnen und analysieren. In einem Projekt in diesem Kapitel setzen Sie eine schnelle Fourier-Transformation ein, um Schall in Frequenzen aufzuteilen. Anhand dieser Berechnungen können Sie die einzelnen Frequenzen aus einer Schwingung entnehmen. Außerdem messen wir mithilfe eines an den Arduino angeschlossenen Mikrofons die Lautstärke.

11.1 Experiment: Stimmen hören/Lautstärke

Mit einem Mikrofon lässt sich der Schallpegel bestimmen. Im ersten Experiment lesen wir einfach Werte von einem Mikrofon und geben sie aus.

In vielen Projekten müssen Sie darüber hinaus die von einem Mikrofon abgerufenen Werte bearbeiten, indem Sie beispielsweise einen Schwellenwert festlegen (wie in einem nachfolgenden Beispiel in diesem Kapitel), einen gleitenden Mittelwert anwenden oder die Mindest- und Höchstwerte bestimmen. Das Bauteil, das wir verwenden, ist die Breakout-Platine für das Electret-Mikrofon von SparkFun (BOB-09964), die Sie in Abbildung 11–1 sehen.

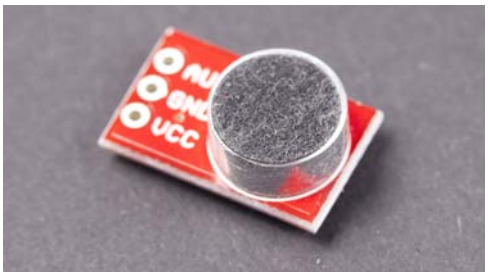

Abb. 11–1 *Ein Mikrofon auf einer Breakout-Platine*

11.1.1 Code und Schaltung für das Mikrofon am Arduino

Abbildung 11–2 zeigt die Arduino-Schaltung für das Mikrofon. Stellen Sie die Verbindungen her und führen Sie den Code aus Listing 11–1 aus.

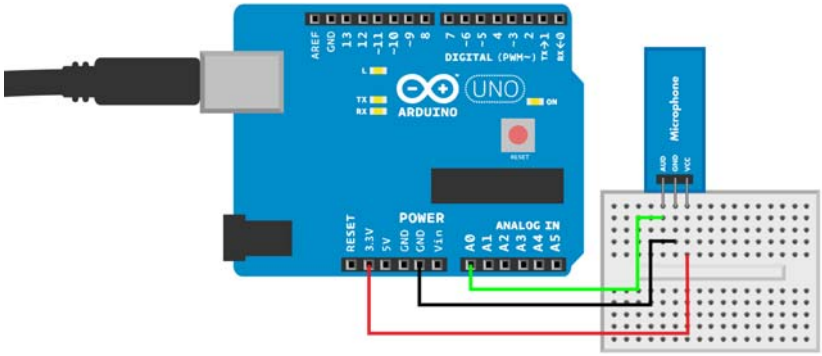

Abb. 11–2 *Die Arduino-Schaltung für das Mikrofon*

```
// microphone.ino - Gibt am seriellen Monitor den Lautstärkepegel und bei
// hoher Lautstärke zusätzlich die Meldung "Sound!" aus.
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int audioPin = A0;
const int sensitivity = 850;

void setup() {
   Serial.begin(115200);
}

void loop() {
   int soundWave = analogRead(audioPin);          1
   Serial.println(soundWave);
   if (soundWave>sensitivity) {                   2
      Serial.println("Sound!");
      delay(500);
   }
   delay(10);
}
```

Listing 11–1 *microphone.ino*

1 Die Breakout-Platine für das Mikrofon können Sie wie einen analogen Widerstandssensor ablesen. `analogRead()` gibt Rohwerte zwischen 0 und 1023 aus, wobei höhere Zahlen eine höhere Lautstärke bedeuten. Der Aufbau und das Programm ähneln einer Potenziometerschaltung.

2 Führt eine Aktion aus, wenn eine bestimmte Lautstärke erreicht ist.

11.1.2 Code und Schaltung für das Mikrofon am Raspberry Pi

Abbildung 11–3 zeigt den Schaltplan für den Raspberry Pi. Stellen Sie alle Anschlüsse her und führen Sie den Code aus Listing 11–2 aus.

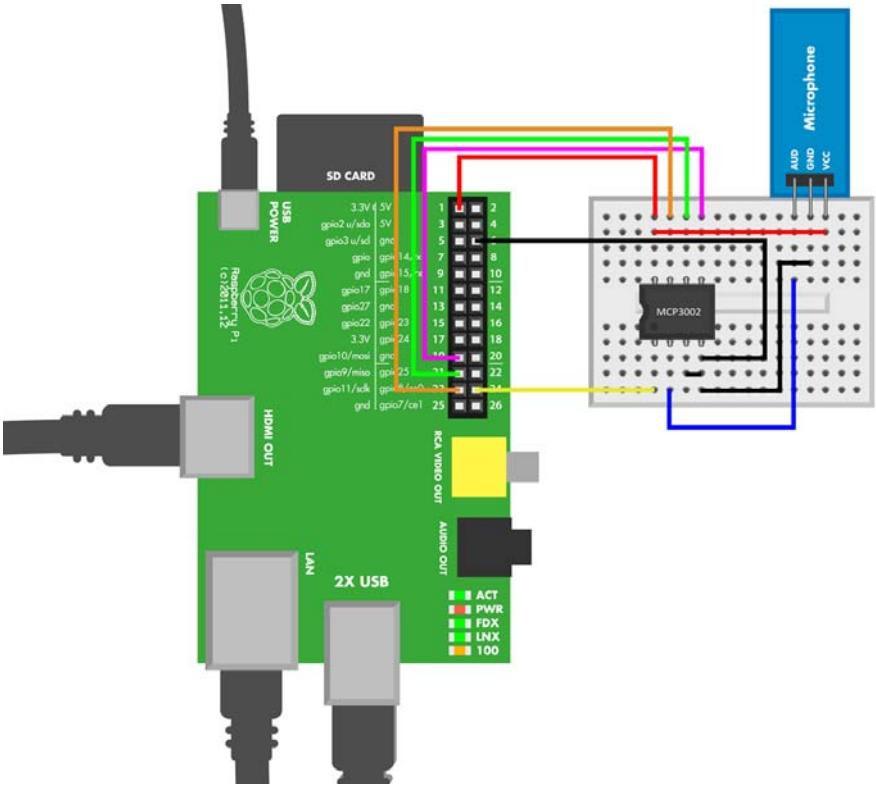

Abb. 11–3 *Die Raspberry-Pi-Schaltung für das Mikrofon*

```
# microphone.py - Liest die Lautstärke am analogen Eingang ab und
# gibt sie aus
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp                    1

def readSound(samples):
   buff = []                                     2
   for i in range(samples):                      3
      buff.append(mcp.readAnalog())              4
   return buff

def main():
   while True:
      sound = readSound(1024)                    5
```

```
        print(sound)
        time.sleep(1) # s

if __name__ == "__main__":
    main()
```

Listing 11–2 *microphone.py*

1. Die Bibliothek *botbook_mcp3002.py* muss sich im selben Verzeichnis befinden wie dieses Programm. Außerdem müssen Sie die Bibliothek `spidev` installieren, die von `botbook_mcp3002` importiert wird. Mehr darüber erfahren Sie in den Kommentaren zu Beginn von *botbook_mcp3002/botbook_mcp3002.py* und in Abschnitt 3.5.3.
2. Deklariert eine neue leere Liste.
3. Wiederholt den nachfolgenden Block, sodass `i` alle Werte zugewiesen werden. In diesem Programm wird dies zu `for i in [0, 1, 2 ... 1023]`.
4. Liest einen Wert vom Mikrofon und hängt einen neuen Eintrag an die Liste (`buff`) an.
5. Liest 1024 Messwerte und ruft eine Liste der zurückgegebenen Werte ab.

11.2 Praxisexperiment: Eine Stecknadel fallen hören

Können Sie eine Stecknadel fallen hören? Mit einem Schallsensor werden wir diese Frage nun ein für allemal klären. Schließen Sie den Sensor wie in Abschnitt 11.1.1 an und laden Sie den Code hoch. Stellen Sie den Sensor auf eine feste Oberfläche, z. B. auf einen Tisch oder den Fußboden, sodass das Mikrofon zu der Stelle weist, an der Sie die Stecknadel fallen lassen wollen. Dämpfen Sie so weit wie möglich alle Hintergrundgeräusche. Ändern Sie den Empfindlichkeitswert im Code so, dass die Meldung »Sound!« nicht ausgelöst wird, wenn gar nichts zu hören ist. Wählen Sie sorgfältig einen Wert aus, der auf leiseste Töne anspricht, aber nicht auf Stille.

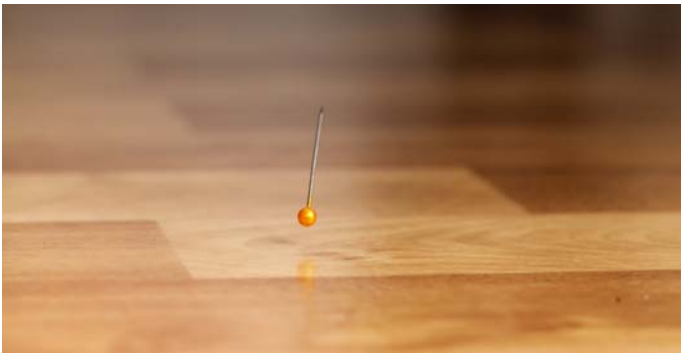

Abb. 11–4 *Eine Stecknadel fällt auf den Fußboden.*

Öffnen Sie den seriellen Monitor in der Arduino-IDE und lassen Sie eine Stecknadel auf die Oberfläche fallen. Wird auf dem Monitor die Meldung »Sound!« angezeigt? Wenn ja, dann können Sie tatsächlich eine Stecknadel fallen hören. Um Abhilfe zu schaffen, falls der Sensor nicht reagiert, können Sie für noch mehr Stille sorgen, die Nadel näher am Sensor fallen lassen und die Empfindlichkeit besser einstellen.

Die Empfindlichkeit gegenüber leisen Tönen hängt vom Pegel der Hintergrundgeräusche ab. Versuchen Sie, laute Musik aufzulegen und die Empfindlichkeit nachzujustieren, bevor Sie die Stecknadel fallen lassen. Nimmt der Sensor sie immer noch wahr?

11.3 Testprojekt: Töne über HDMI sichtbar machen

Haben Sie sich schon immer einen grafischen Equalizer mit einem 50-Zoll-Bildschirm gewünscht? In diesem Projekt analysieren Sie Schall mit dem Raspberry Pi und zeigen das Ergebnis auf dem Fernseher an. Dabei werden die Schallfrequenzen als farbige, animierte Graphen angezeigt (siehe Abb. 11–5).

Abb. 11–5 *Ein animierter Graph auf einem großen Bildschirm*

11.3.1 Lernziele

In diesem Projekt zur Visualisierung von Schall lernen Sie Folgendes:

- Schall numerisch analysieren
- Mithilfe von `SciPy` sehr schnelle Berechnungen in Python durchführen
- Frequenzen durch schnelle Fourier-Transformationen herausziehen

Außerdem frischen Sie Ihre Kenntnisse sowohl in `pyGame` auf als auch darin, HD-Grafiken über HDMI auszugeben, also beispielsweise auf Ihren Fernseher oder über einen Beamer (siehe Abschnitt 6.7).

11.3.2 Den seriellen Port des Raspberry Pi aktivieren

Da der serielle Port auf dem Raspberry Pi normalerweise von der Anmeldeshell genutzt wird, die Sie über ein serielles Kabel aufrufen können, müssen Sie ihn erst freistellen, bevor Sie ihn selbst verwenden können:

```
$ sudoedit /etc/inittab
```

Kommentieren Sie in der Datei die letzte Zeile aus, die den seriellen Port mit Beschlag belegt. Um eine Zeile auszukommentieren, stellen Sie ihr ein Nummernzeichen voran, damit sie ignoriert wird:

```
# T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Starten Sie den Raspberry Pi neu.

11.3.3 Code und Schaltung für die Visualisierung am Raspberry Pi

Installieren Sie alle erforderlichen Komponenten. Öffnen Sie dazu auf dem Raspberry Pi das Terminal und führen Sie die folgenden Befehle aus:

```
$ sudo apt-get update
$ sudo apt-get -y install python-pygame python-numpy
```

Abbildung 11–6 zeigt den Schaltplan für den Raspberry Pi. Stellen Sie die Verbindungen her und führen Sie den Code aus Listing 11–3 aus.

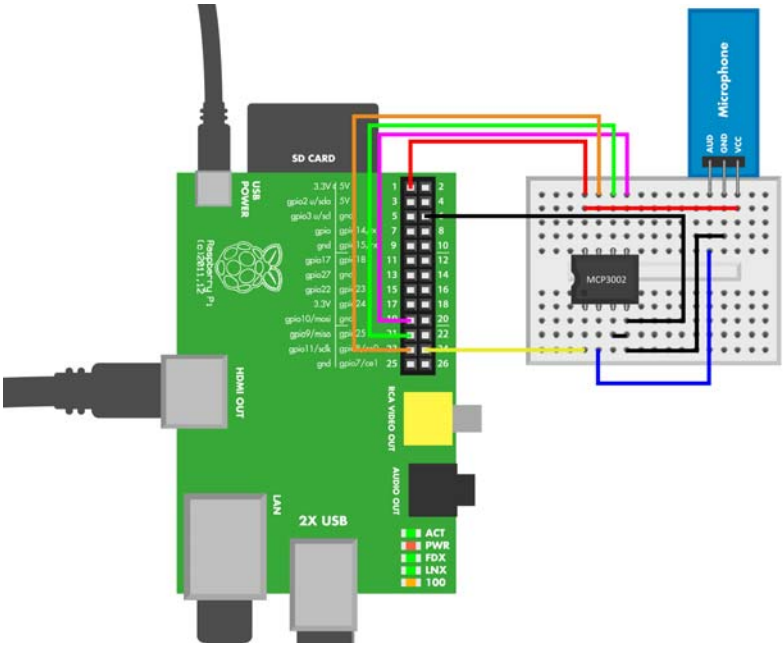

Abb. 11–6 *Die Mikrofonschaltung am Raspberry Pi für den Equalizer*

```
# equalizer.py - Zeigt über den Mikrofoneingang einen
# Equalizer an
# (c) BotBook.com - Karvinen, Karvinen, Valtokari


import pygame # sudo apt-get -y install python-pygame
import math
import numpy # sudo apt-get -y install python-numpy

import microphone            1
from pygame.locals import *

pygame.init()

width = 800
height = 640

size = width, height
background = 0, 0, 0

screen = pygame.display.set_mode(size, pygame.FULLSCREEN)
fullBar = pygame.image.load("equalizer-full-small4.jpg")          2
emptyBar = pygame.image.load("equalizer-empty-small4.jpg")
clock = pygame.time.Clock()
pygame.mouse.set_visible(False)
mainloop = True

barHeight = 36
barWidth = 80
barGraphHeight = 327
barPos = [55, 130]

sampleLength = 16

def fftCalculations(data):              3
   data2 = numpy.array(data) / 4        4
   fourier = numpy.fft.rfft(data2)      5
   ffty = numpy.abs(fourier)            6
   ffty = ffty / 256.0              7
   return ffty


while mainloop:
   buff = microphone.readSound(sampleLength)         8

   barData = fftCalculations(buff)         9

   for event in pygame.event.get():
      if event.type == pygame.QUIT:
         mainloop = False
      if (event.type == KEYUP) or (event.type == KEYDOWN):
         if event.key == K_ESCAPE:
            mainloop = False
```

```
screen.fill(background)
# Gibt Daten als Säulendiagramm aus
for i in range(8):                                    10
   bars = barData[i+1]                                11
   bars = int(math.floor(bars*10))                    12
   if bars > 10:
          bars = 10
   bars -= 1
   screen.blit(emptyBar, (barPos[0]+i*(barWidth+10), barPos[1]),
                         (0, 0, barWidth, barHeight*(10-bars)))
      if bars >= 0:
         barStartPos = (barPos[0] + i * (barWidth + 10),
                        barPos[1] + barGraphHeight - barHeight * bars + 6)
         barSourceBlit = (0, barGraphHeight - barHeight * bars+6,
                          barWidth, barHeight*bars)
         screen.blit(fullBar, barStartPos, barSourceBlit)      13
   pygame.display.update()
```

Listing 11–3 *equalizer.py*

1 Das Programm *microphone.py* aus Abschnitt 11.1.2 muss sich im selben Verzeichnis befinden wie dieses Programm (*equalizer.py*).

2 `PyGame` kann normale JPG-Bilder laden, die Sie in beliebigen Programmen erstellen können, z. B. in Inkscape, GIMP, Photoshop oder Illustrator.

3 Wenn diese Funktion vom Hauptprogramm aus aufgerufen wird, enthält `data` 16 Messungen, von denen jede einen Wertebereich von 0 bis 1024 aufweist.

4 Dividiert die Messwerte, um sie auf den Bereich von 0 bis 256 abzubilden.

5 Unterzieht die aufgezeichneten Schallmesswerte einer schnellen Fourier-Transformation (Fast Fourier Transformation, FFT). Das ergibt die Frequenzbereiche der Messwerte. Die Funktion `rfft()` gibt ein Array mit 16/2+1 (= 9) Zellen zurück, die jeweils für eine Frequenz stehen.

6 Verwirft den Imaginärteil der Zahlen, damit sie später grafisch dargestellt werden können. Beispielsweise ist `abs(1+1j)` ungefähr 1,4.

7 Konvertiert die fouriertransformierten Werte in anteilige Werte (im Bereich 0,0 bis 1,0).

8 Liest unter Verwendung der Mikrofon-Bibliothek von *botbook.com* 16 Messwerte vom Mikrofon ab.

9 Speichert die schnelle Fourier-Transformation der Schallmesswerte in `barData`.

10 Ruft für jede der acht Säulen (0 bis 7) ...

11 ... den Anteil der Frequenz ab. Wegen `i+1` wird die erste Zellennummer (0) ignoriert. Diese Zelle enthält die Gleichstromkomponente, also den Mittelwert der positiven und negativen Ausschläge der Wechselstromschwingung.

12 Zählt die Anzahl der erforderlichen Balken (für `1.0` (100 %) ergibt sich das Maximum von neun Balken).

13 Zeichnet die Säulen (mit den Balken) auf den Bildschirm.

11.3.4 Schnelle Fourier-Transformation

Mit einer schnellen Fourier-Transformation können Sie die einzelnen Frequenzen aus einer Schwingung entnehmen. Dies wird dazu genutzt, um in grafischen Equalizern, Spektrumanalysatoren und Oszillographen Frequenzdiagramme darzustellen.

Stellen Sie sich vor, dass Sie zwei Töne spielen, einen tiefen (5 Hz) und einen höheren (40 Hz). Hören können wir zwar nur Töne zwischen 20 Hz und 20.000 Hz, aber der Einfachheit halber verwenden wir in diesem Beispiel kleinere Zahlen.

In Abbildung 11–7 ist die Zeit auf der horizontalen Achse aufgetragen und die Amplitude, also in gewissem Sinne die Verdichtung und Verdünnung der Luft, auf der vertikalen Achse.

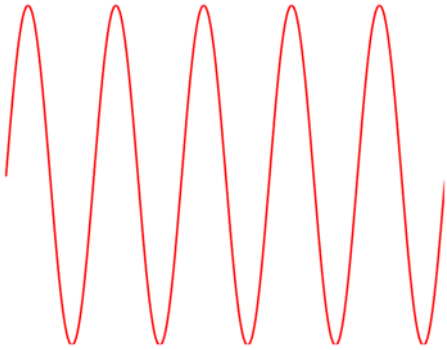

Abb. 11–7 *Die Sinusschwingung eines tiefen 5-Hz-Tons*

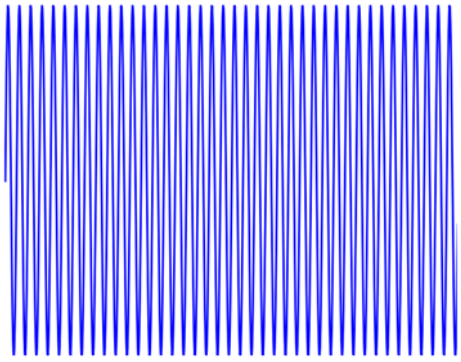

Abb. 11–8 *Die Sinusschwingung eines höheren 40-Hz-Tons*

Werden beide Töne gleichzeitig gespielt, überlagern sie sich zu einer gemeinsamen Schwingung. Das können Sie sich vorstellen wie kleine Kräuselungen auf einer großen Meereswelle. Der tiefe 5-Hz-Ton bildet dabei die »große Welle«, während die Kräuselungen durch die höhere Frequenz von 40 Hz zustande kommen. Die kombinierte Schwingung ist die Summe der beiden einzelnen Schwingungen.

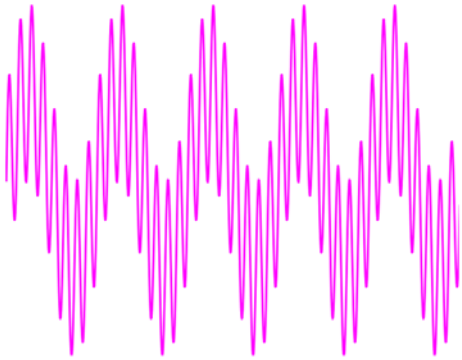

Abb. 11–9 *Beide Töne kombiniert*

Bei der Tonaufzeichnung erhalten Sie solche kombinierten Schwingungen. So ähnlich sieht das typische Ausgangsmaterial bei Tonmessungen aus, z. B. in den Formaten MP3, WAV oder OGG. Aber wie können Sie aus einer solchen kombinierten Schwingung wieder die beiden ursprünglichen Sinusschwingungen gewinnen? Dabei hilft Ihnen eine Fourier-Transformation.

Führen Sie eine schnelle Fourier-Transformation der kombinierten Schwingung durch. Wie Sie in dem Programmcode zu diesem Projekt gesehen haben, können Sie dazu einfach eine entsprechende Funktion aufrufen, der Sie die Daten der Schwingung als Parameter übergeben (siehe Abb. 11–10).

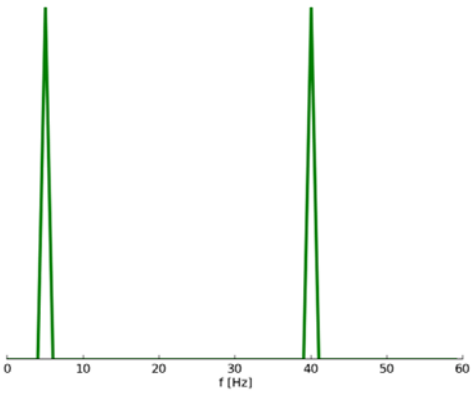

Abb. 11–10 *Eine schnelle Fourier-Transformation macht die Frequenzen der ursprünglichen Töne sichtbar.*

Die schnelle Fourier-Transformation berechnet die Frequenzen der ursprünglichen Sinusschwingungen, also 5 Hz und 40 Hz. Damit können Sie überlagerte Schwingungen in einzelne Frequenzen zerlegen. In Abbildung 11–11 sehen Sie ein Beispiel für die Equalizer-Anzeige.

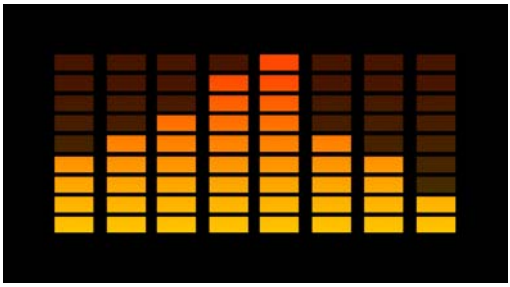

Abb. 11–11 *Der Equalizer*

11.4 Wie geht es weiter?

In diesem Kapitel haben Sie sich mit Schall beschäftigt. Mit dem Arduino haben Sie die Lautstärke gemessen und sich benachrichtigen lassen, wenn sie einen bestimmten Grenzwert überschreitet. Auf dem Raspberry Pi haben Sie Schall aufgezeichnet und in Echtzeit analysiert. In diesem Projekt haben Sie auch Ihre Kenntnisse in `pyGame` vertieft und eine schnelle Fourier-Transformation angewendet, um eine Schwingung in ihre Bestandteile zu zerlegen.

Auch im nächsten Kapitel geht es um »Luft«, aber in einem größeren Maßstab. Dort messen Sie das Wetter und das Klima, von der Temperatur und Luftfeuchtigkeit in einem Zimmer bis hin zur Wettervorhersage für Ihren Stadtteil.