HANSER

Leseprobe

zu

Mach was mit Python & Raspberry Pi!

von Jörn Donges

Print-ISBN: 978-3-446-46150-5
E-Book-ISBN: 978-3-446-46600-5
E-Pub-ISBN: 978-3-446-46601-2

Weitere Informationen und Bestellungen unter
http://www.hanser-fachbuch.de/978-3-446-46150-5
sowie im Buchhandel

Inhalt

1 Einführung

Mit dem Raspberry Pi (auch Pi oder RasPi genannt) bekommt man für wenig Geld ein vollwertiges Computersystem mitsamt reichhaltiger Softwarekollektion, die das Arbeiten vom spielerischen Einstieg bis hin zur Softwareentwicklung auf hohem Niveau ermöglicht.

Aufgrund der stetig zunehmenden Digitalisierung aller Lebensbereiche wird es immer wichtiger, die Konzepte der digitalen Informationsverarbeitung zu verstehen. Nur wer die Arbeitsweise von programmierbaren Systemen und Algorithmen versteht, kann auch realistisch einschätzen, wo deren Grenzen liegen. Darüber hinaus macht es einfach Spaß, eigene Ideen umzusetzen und zum Mitgestalter der digitalen Welt zu werden, anstatt sie nur zu konsumieren.

Dafür stellt der Raspberry Pi den idealen Einstieg dar. Der kleine Einplatinenrechner hat seit seiner Einführung im Jahr 2012 eine riesige Verbreitung gefunden. Über 20 Millionen Exemplare wurden bereits verkauft. Zu den Nutzern zählen nicht nur Bastler und Maker, sondern insbesondere auch Softwaretüftler und angehende Entwickler, denen der Raspberry Pi eine preiswerte Möglichkeit bietet, ein Linux-System einzurichten und damit zu experimentieren oder ein System lautlos und stromsparend im Dauerbetrieb laufen zu lassen, etwa als Homeserver oder als Medienzentrale.

Auch die Programmiersprache Python erfreut sich einer immer größer werdenden Community von Entwicklern und Anwendern, da die Sprache leicht erlernbar und verständlich ist, andererseits aber umfangreiche Bibliotheken für nahezu jeden erdenklichen Anwendungszweck bietet und daher universell einsetzbar ist. Was liegt also näher, als die beiden Welten miteinander zu verbinden?

Die Mission dieses Buches

Dieses Buch bietet eine Einführung in den Umgang mit dem Raspberry Pi und seinem bevorzugten Betriebssystem Linux (in der Raspbian-Distribution) sowie in die Programmiersprache Python 3.

Du brauchst keine Vorkenntnisse. Es wird alles von Grund auf erklärt. Dennoch habe ich versucht, einen breiten Bereich typischer Anwendungen abzudecken, wie etwa Spieleprogrammierung, GUI-Entwicklung oder Web Apps.

Bei der Auswahl der Beispielprojekte habe ich darauf geachtet, dass es sich um brauchbare kleine Anwendungen handelt, die du auch im Alltag benutzen kannst. Gleichzeitig ist der Code aber möglichst kurz gehalten, um für die Veröffentlichung in einem Buch geeignet zu sein. Daher handelt es sich immer um Basisversionen der entsprechenden Anwendungen, die jederzeit mit weiteren Features ausgebaut und erweitert werden können. Ich möchte dich genau dazu in die Lage versetzen. Die Beispielprojekte sind also nicht zum sturen Abtippen gedacht, sondern eher als Ideengeber, die dich dazu bringen sollen, selbst aktiv zu werden und an den Programmen weiterzuarbeiten.

Die Projekte sollen nicht in Stein gemeißelt sein, sondern zum Experimentieren und Probieren einladen. Die in diesem Buch angeschnittenen Themenbereiche sind in sich jeweils so komplex und umfangreich, dass sie nicht erschöpfend dargestellt werden können. Nimm also nicht nur dieses Buch als Anleitung zur Hand, sondern nutze bei Veränderungsideen und Verständnisschwierigkeiten auch Google oder die Coder-Hilfsplattform *https://stackoverflow.com*, um nach entsprechenden Begriffen zu suchen, denn du wirst schnell feststellen, dass eine Anwendung nie wirklich fertig ist. Es gibt immer Möglichkeiten der Weiterentwicklung. Vielleicht tauchen auch nach langem Einsatz noch Fehler auf, die behoben werden müssen.

An einigen Stellen im Buch setze ich voraus, dass du englischsprachige Texte verstehen kannst. Viele Begriffe in der Programmierung sind auf Englisch und bei fortgeschrittenen Themen wie API-Programmierung stehen oft auch keine deutschsprachigen Schnittstellen und Dokumentationen zur Verfügung. Daher war das nicht immer zu umgehen, doch wenn du tiefer in die IT-Welt einsteigen willst, wirst du wahrscheinlich schon wissen, dass man nicht darum herumkommt, Englisch zu lernen.

So ist dieses Buch aufgebaut

In Kapitel 2 zeige ich dir, wie du deinen Raspberry Pi einrichtest und welche Tools für die Python-Programmierung benötigt werden. Du lernst auch die Grundlagen der Linux-Distribution Raspbian kennen, die standardmäßig auf dem RasPi installiert wird.

Mit einer Raspbian-Installation steht dir ein riesiges Softwarepaket kostenlos zur Verfügung, und du kannst gleich mit der Python-Programmierung loslegen. Für fortgeschrittene Projekte musst du manchmal ein externes Modul oder ein neues Programm installieren. Auch das erkläre ich dir alles genau. Du erfährst, wie du über das Paketverwaltungssystem apt ganze Softwarepakete nachinstallieren kannst, aber auch, wie du mit dem Python Package Index gezielt eines der unzähligen Python-Module finden, herunterladen und einbinden kannst.

Es gibt verschiedene Möglichkeiten, auf dem kleinen Rechner zu arbeiten. Du kannst entweder klassisch Tastatur, Monitor und Maus anschließen und wie an einem herkömm-

lichen Desktop-PC arbeiten. Es ist aber auch möglich, über eine Netzverbindung auf den RasPi zuzugreifen und die Benutzeroberfläche auf einen anderen Computer zu holen. Auch das wird ausführlich erklärt.

In Kapitel 3 führe ich dich in die Grundlagen der Python-Programmierung ein. Anhand eines kleinen Zahlenratespiels stelle ich dir die grundlegenden Programmierkonzepte wie Abfragen, Schleifen, Funktionen und Variablen vor. Außerdem werfen wir einen Blick auf den Debugger, ein wichtiges Werkzeug, um Programmierfehlern auf die Spur zu kommen. Die nächste Stufe ist dann die objektorientierte Programmierung. Diese ist für die Entwicklung moderner Apps notwendig. Das erste Projekt wird ein Vokabeltrainer sein, der dich in einer beliebigen Sprache abfragen kann, für die du eine Textdatei mit den Vokabeln hast.

In Kapitel 4 werfen wir einen Blick auf das weite Feld der Spieleprogrammierung. Mit Pygame ist im Standardumfang von Python auf dem RasPi eine Multimedia-Bibliothek vorhanden, die dir ermöglicht, Grafik und Sound in deine Projekte einzubinden. Das ist ein Thema, das ganze Bücher füllen könnte, deshalb wird im Rahmen dieses Buches nur ein kurzer Einblick möglich sein. Ähnliches gilt für alle weiteren Projekte. Die Projekte können immer nur eine erste Annäherung ans Thema sein. Dennoch wirst du in diesem Kapitel lernen, einfache Grafiken und Animationen zu programmieren und Audiodateien abzuspielen. Am Ende wirst du einen Klassiker unter den Videospielen nachprogrammiert haben, wobei ich die Gelegenheit nutze und die Einführung in die objektorientierte Programmierung nochmals vertiefe.

Zentrales Thema von Kapitel 5 ist die Entwicklung grafischer Benutzeroberflächen. Mit Tkinter stellt Python eine Bibliothek zur Verfügung, die das Erstellen grafischer Benutzerschnittstellen mit Textfeldern, Menüs und Schaltflächen stark vereinfacht. In Kapitel 5 wirst du dem Vokabeltrainer aus Kapitel 3 eine grafische Oberfläche verpassen, um die Anwendung komfortabler zu gestalten und eine Taschenrechner-App für den Desktop entwickeln. Nach dem Durcharbeiten des Kapitels solltest du in der Lage sein, eigene grafische Benutzeroberflächen zu gestalten und umzusetzen und damit deinen eigenen Projektideen eine Oberfläche zu geben.

Neben grafischen Oberflächen sind Web Apps heutzutage eine wichtige Schnittstelle, um Anwendungen an den Benutzer zu bringen. Der Vorteil: Jedes internetfähige Gerät mit einem Webbrowser wird zum Client. Mit Python und dem Modul Flask wirst du in Kapitel 6 lernen, Web Apps zu erstellen. Als praktisches Beispiel entwickelst du eine To-do-App für deinen Raspberry Pi, die auch zur Einkaufsliste oder App für kurze Notizen erweitert werden kann.

Eine Anwendung davon bietet dann Kapitel 7: Hier bauen wir ein Webradio. Dieses wird sowohl als Web App als auch als GUI umgesetzt, sodass du den Radiostream nicht nur vom Raspberry Pi aus steuern kannst, sondern auch per Browser von anderen Geräten in deinem Heimnetzwerk. Dann musst du nur noch einen Lautsprecher anschließen und die Reise in die Welt der weltweiten Audiostreams kann losgehen.

Kapitel 8 greift schließlich ein Thema auf, das du in wenigen anderen Python-Büchern finden wirst. Doch gerade für den Raspberry Pi ist es relevant, weil dieser kleine Rechner gerne im Dauerbetrieb benutzt wird, um Ereignisse im Internet zu protokollieren und Daten zu sammeln, wie z. B. Sportergebnisse, Börsenkurse, Wetter usw. Du wirst lernen, wie man mittels Web Scraping interessante Daten automatisch aus Webseiten extrahiert und wie man mithilfe einer API einen Server kontaktiert, um bestimmte Daten herunterzuladen und sie weiter auszuwerten.

Die Beispielprojekte aus dem Buch in der Übersicht	
Kapitel 3	▪ Zahlenratespiel ▪ Vokabeltrainer (Konsole)
Kapitel 4	▪ Analoguhr ▪ Pong
Kapitel 5	▪ Vokabeltrainer (GUI) ▪ Taschenrechner
Kapitel 6	▪ To-do-App
Kapitel 7	▪ Webradio (Web) ▪ Webradio (GUI)
Kapitel 8	▪ Wetter-App (Konsole) ▪ Wetter-App (Web)

Sämtliche Quellcodes zu den Beispielprojekten stehen in ungekürzter Form auf *plus.hanser-fachbuch.de* zur Verwendung bereit.

Ansonsten bleibt mir an dieser Stelle nur, dir viel Spaß auf der spannenden Reise durch die Möglichkeiten der Sprache Python und des kleinen Raspberry Pi zu wünschen.

Hamburg, August 2020

Jörn Donges

Die Funktion `items` liefert schließlich Zweiertupel mit Schlüssel und Wert:

```
>>> print(list(einwohner.items()))
[('München', 1471500), ('Köln', 1085700)]
```

3.5 Wörterbücher in Aktion – ein Vokabeltrainer

Nun wird es Zeit, das Gelernte wieder in einem kleinen Projekt umzusetzen. Listen und Dictionaries eignen sich wie gesagt ganz gut, um Daten abzuspeichern. Dictionary bedeutet wörtlich übersetzt Wörterbuch, denn es erlaubt, Informationen unter einem bestimmten Schlüsselwort abzuspeichern. Eine Liste speichert Dinge der Reihe nach, und um auf ein Element der Liste zuzugreifen, benötigst du den Index des Elements, also die Nummer seines Listenplatzes. Beim Dictionary brauchst du den Index des Elements aber gar nicht zu kennen, es genügt, das Schlüsselwort zu haben, um auf das Element zuzugreifen.

Das nutzen wir jetzt aus mit einem kleinen Vokabeltrainer. Das Programm soll dich beim Lernen von Vokabeln unterstützen, indem es dich interaktiv abfragt. Die Vokabeln speichern wir in einer Textdatei separat ab. Das hat den Vorteil, dass du die Liste jederzeit erweitern kannst. Du kannst dir damit natürlich auch verschiedene Vokabellisten anlegen, eine für Deutsch-Englisch und eine für Deutsch-Spanisch oder eine andere Fremdsprache, die du gerne lernen möchtest.

Der Vokabeltrainer soll nach dem Start die Vokabeln aus der Textdatei einlesen und sie im Dictionary abspeichern. Danach fragt er die Vokabeln ab, indem er ein Element der Liste zufällig auswählt, dir das englische Wort nennt und du dann die deutsche Übersetzung angeben musst. Wir implementieren zunächst eine vereinfachte Variante, die bei einer falschen Eingabe einfach so lange weiterfragt, bis du das richtige deutsche Wort eingegeben hast. Hast du die richtige Antwort gegeben, dann entfernen wir das Wort aus dem Wörterbuch, damit nicht immer wieder die Wörter abgefragt werden, die du schon kennst.

Diese Abfrage läuft so lange, bis du jedes Wort richtig eingegeben hast. Im Folgenden ist das komplette Listing dargestellt (siehe Listing 3.3).

Listing 3.3 Vokabeltrainer

```
1  #Vokabeltrainer
2  import random
3  Sprache = "Englisch"
4  # Vokabelliste aus der Datei ins Woerterbuch einlesen
5  VokabelFile = open("wortliste.txt")
6  Zeilen = VokabelFile.readlines()
7  Woerterbuch = {}
```

```
for Zeile in Zeilen:
    if Zeile[0]!='#' and Zeile[0]!='\n':
      VokabelDeutsch = Zeile.split(",")[0]
      VokabelFremdsprache = Zeile.split(",")[1].strip('\n')
      Woerterbuch[VokabelDeutsch]=VokabelFremdsprache


# Abfrage von Vokabeln
Eingabe  = ""
while Woerterbuch != {}:
  #Wähle ein Element zufällig aus dem Wörterbuch
  Zufallswahl = random.choice(list(Woerterbuch.keys()))

  Eingabe  = ""
  VokabelDeutsch = Zufallswahl
  VokabelFremdsprache = Woerterbuch[Zufallswahl]
  #Eingabe mit Vokabel vergleichen
  while Eingabe != VokabelFremdsprache:
    Eingabe =  input("{} in {}: ".format(VokabelDeutsch, Sprache))
    if Eingabe == VokabelFremdsprache:
        print("========== Richtig!")
        Woerterbuch.pop(VokabelDeutsch)
print("Alles richtig eingegeben. Gut gemacht.  :-)")
```

Außerdem benötigst du noch die Textdatei mit dem Wörterbuch. Auch diese kannst du in Thonny anlegen oder auch einen beliebigen anderen Text-Editor verwenden.

3.5.1 Die Textdatei in Thonny anlegen

Ein Python-Quelltext ist nichts anderes als eine Textdatei. Der Unterschied ist lediglich die Dateinamensendung *.py* statt wie für Textdateien allgemein üblich *.txt*. An der Endung erkennen der Python-Interpreter und alle anderen Programme wie Editoren und IDEs, dass es sich um einen Quelltext in Python handelt. Bei Erstellen einer Datei kannst du natürlich beliebige andere Namensendungen vergeben.

Öffne einen neuen Editor-Bereich mit der Schaltfläche New (grünes Kreuz) und gib den folgenden oder einen entsprechend abgewandelten Text ein, wie in Bild 3.3 zu sehen.

```
Schule,school
Buch,book
Auto,car
Sonne,sun
Mond,moon
Stern,star
Hund,dog
Katze,cat
Maus,mouse
```

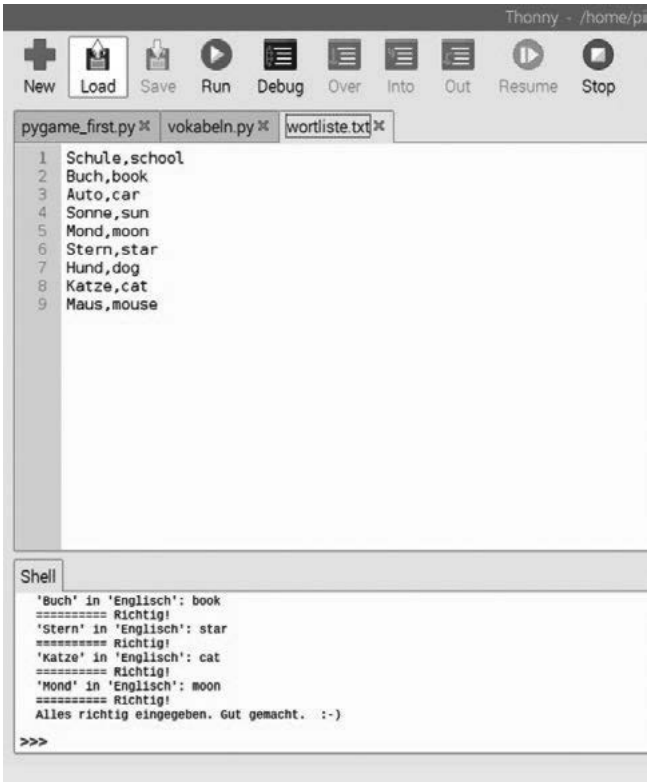

Bild 3.3 Anlegen der Wörterbuchdatei in Thonny

Gehe auf die SPEICHERN-Schaltfläche und speichere die Datei unter dem Namen „wortliste.txt“ in dem Ordner, in dem sich auch das Python-Programm für den Vokabeltrainer befindet.

Du kannst die Eingabe natürlich beliebig erweitern oder abwandeln, solange du das folgende Format einhältst:

Pro Zeile ein Wort in der Form `Deutsches_Wort, Wort_ in_Fremdsprache`

Auch andere Sprachen sind möglich. Achte aber darauf, dass der Vokabeltrainer in seiner einfachsten Form, wie er hier vorgestellt wird, immer die gesamte Wörterbuchdatei am Anfang einliest. Das könnte bei sehr umfangreichen Wörterbüchern zu Speicherproblemen führen.

Auswahl der angezeigten Dateiendungen im Dateidialog von Thonny

Wenn du später die Textdatei wieder in Thonny öffnen willst, achte darauf, im Dateidialog rechts unten die Auswahl von *Python files* auf *text files* oder *all files* umzustellen, um auch Dateien mit der Endung *.txt* angezeigt zu bekommen (siehe Bild 3.4).

Bild 3.4
Auswahl der Dateiendung für die angezeigten Dateien

3.5.2 Der Vokabeltrainer – Schritt für Schritt erklärt

Gehen wir nun das Listing zum Vokabeltrainer noch einmal Schritt für Schritt durch (siehe Listing 3.4).

Listing 3.4 Vokabeltrainer

```
1  #Vokabeltrainer
2  import random
3  Sprache = "Englisch"
```

Zeile 3 ist vorausschauend programmiert. Wir werden den Vokabeltrainer vielleicht auch für andere Sprachen nutzen wollen. Daher definieren wir die verwendete Sprache in einer String-Variablen. Überall, wo wir später im Benutzerdialog die Sprache erwähnen, werden wir diese Variable benutzen. Auf diese Weise lässt sich das Programm einfach auch anderen Sprachen anpassen. Im Unterschied zu den anderen Variablen ändert sich die Sprache im weiteren Verlauf jedoch nicht. Sie wird am Anfang festgelegt und bleibt dann unverändert bestehen. Das nennt man eine Konstante. Konstanten sind wichtige Teile des Codes, die meist für die Konfiguration der Software verwendet werden.

```
4  # Vokabelliste aus der Datei ins Woerterbuch einlesen
5  VokabelFile = open("wortliste.txt")
```

Hier öffnen wir eine Textdatei mit dem Befehl `open` und weisen der Variablen `VokabelFile` diese Datei zu.

Beachte, dass die Datei dafür bereits existieren und im gleichen Ordner wie der Vokabeltrainer liegen muss, da du sonst eine Fehlermeldung erhalten würdest.

Es handelt sich hier um einen Zugriffstypen auf einen sogenannten Stream. Streams sind sehr wichtig für den Datenaustausch mit externen Quellen. Immer wenn ein Programm Daten einliest oder ausgibt, kontrolliert man dies auf der Programmierebene über Daten-Streams. Dabei spielt es keine Rolle, ob die Daten von der Tastatureingabe, einer Datei oder aus dem Internet stammen und ob die Ausgabe auf die Konsole oder einen Drucker erfolgt. Für all diese Kanäle gibt es einen Stream-Datentyp, der dir immer die gleichen Methoden zum Einlesen und Schreiben von Daten anbietet. Das macht es einfacher, mit verschiedenen Ein- und Ausgabegeräten zu arbeiten, ohne dass du dich um die konkrete Hardwareebene kümmern musst.

4.1.4 Eine Analoguhr mit Pygame

Mit dem bisherigen Wissen über die Möglichkeiten von Pygame kannst du schon einiges anfangen, z. B. einfache grafische Animationen programmieren und darauf aufbauende Spielideen umsetzen. Wie wäre es z. B. mit einer Analoguhr? Dazu brauchen wir nur einen Kreis zu zeichnen sowie einige Zahlen als Grafikobjekte in die Zeichenfläche einzufügen (siehe Bild 4.3). Die Zeiger können als Linien vom Mittelpunkt des Kreises aus gezeichnet werden. Das sind alles Dinge, die in den vorangegangenen Beispielen schon einmal enthalten waren. Wir benötigen noch ein klein wenig Mathe, um aus der aktuellen Uhrzeit, die wir aus dem Python-Modul `datetime` auslesen können, die richtigen Winkel für die Uhrenzeiger auszurechnen.

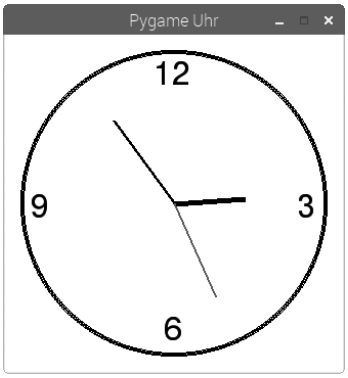

Bild 4.3
Eine Analoguhr mit Pygame

Der Code sieht so wie in Listing 4.5 aus.

Listing 4.5 Der Code von *analoguhr.py*

```
1  import pygame, datetime, math
2
3  weiss=(255,255,255)
4  schwarz=(0,0,0)
5  rot=(250,20,20)
6  schwarz=(0,0,0)
7  mitte=(165,165)# Mittelpunkt des Zeichenbereichs
8
9  pygame.init()
10 pygame.display.set_caption("Pygame Uhr")
11 uhr=pygame.display.set_mode((330,330))
12 font = pygame.font.SysFont("arial", 32)
13 label12 = font.render("12", True, schwarz)
14 label3 = font.render("3", True, schwarz)
15 label6 = font.render("6", True, schwarz)
16 label9 = font.render("9", True, schwarz)
17 clock = pygame.time.Clock()
18
19 laenge_kleinerZeiger=70
```

```
20 laenge_grosserZeiger=100
21 laenge_sekundenZeiger=100
22
23 laeuft=True
24 while laeuft:
25     for event in pygame.event.get(): #
26         if event.type == pygame.QUIT: #
27               laeuft = False
28         elif event.type==pygame.KEYDOWN:
29                 if event.key==pygame.K_x: #
30                     laeuft=False
31     aktuelle_zeit=datetime.datetime.now()
32     uhr.fill(weiss)
33     uhr.blit(label12, (145,20))
34     uhr.blit(label3, (285,150))
35     uhr.blit(label6, (155,270))
36     uhr.blit(label9, (25,150))
37     pygame.draw.circle(uhr,schwarz,mitte,150,4)
38
39     stunden=aktuelle_zeit.hour%12+aktuelle_zeit.minute/60.
40     minuten=aktuelle_zeit.minute
41     sekunden=aktuelle_zeit.second
42     koordStunden=(mitte[0]+math.sin(stunden*math.pi/6)*laenge_
   kleinerZeiger,mitte[1]-math.cos(stunden*math.pi/6)*laenge_kleinerZeiger)
43     koordMinuten=(mitte[0]+math.sin(minuten*math.pi/30)*laenge_
   grosserZeiger,mitte[1]-math.cos(minuten*math.pi/30)*laenge_grosserZeiger)
44     koordSekunden=(mitte[0]+math.sin(sekunden*math.pi/30)*laenge_
   sekundenZeiger,mitte[1]-math.cos(sekunden*math.pi/30)*laenge_sekundenZeiger)
45
46     pygame.draw.line(uhr,schwarz,mitte,koordStunden,5)
47     pygame.draw.line(uhr,schwarz,mitte,koordMinuten,3)
48     pygame.draw.line(uhr,rot,mitte,koordSekunden,2)
49     pygame.display.flip()
50     clock.tick(1)
```

In **Zeile 1** importieren wir neben dem obligatorischen `pygame` auch die Module `datetime`, um die aktuelle Uhrzeit auszulesen, und `math`, für die Umrechnung der Zeit in die Zeigerstellung.

Die **Zeilen 3 bis 6** definieren die verwendeten Farbcodes.

In **Zeile 7** wird der Mittelpunkt der Uhr festgelegt. Dies ist der Startpunkt für das Zeichnen der Zeiger und natürlich der Mittelpunkt des Kreises für das Ziffernblatt.

Zeilen 9 bis 12 kennst du bereits: Wir öffnen und initialisieren das Pygame-Fenster und laden eine Systemschrift für das Ziffernblatt.

In den **Zeilen 13 bis 16** werden die Beschriftungen des Ziffernblattes definiert. So wie bei der Laufschrift erzeugen wir auch hier Image-Objekte aus der Systemschrift, die dann per `blit`-Funktion in die Uhr eingefügt werden.

Zeile 17 definiert die Pygame-`clock`, die wir zum Timing der Animation benötigen.

In den **Zeilen 19 bis 21** werden die Längen der Zeiger festgelegt. Wie üblich ist der kleine Zeiger etwas kürzer, aber du kannst hier natürlich mit anderen Werten experimentieren.

Die **Zeilen 23 bis 30** kennst du bereits aus Abschnitt 4.1.2 – sie beschreiben eine Schleife, die so lange läuft, bis eines der Ereignisse „Schließen des Fensters“ oder „Drücken der X-Taste“ die Bool-Variable `laeuft` negiert und das Programm daraufhin beendet.

Ab **Zeile 31** kommt das Herzstück der Analoguhr: Zunächst wird die aktuelle Uhrzeit aus dem Modul `datetime` mit der Methode `datetime.now()` ausgelesen. Das Modul `datetime` enthält Funktionen, um mit Datum und Zeit umzugehen, und ist ein wichtiges Python-Standardmodul. Wir werden im Rahmen der Analoguhr lediglich die Attribute `hour`, `minute` und `second` nutzen, die, wie du sicher schon vermutet hast, die aktuelle Stunde, Minute und Sekunde als ganze Zahl zwischen 0 und 23 bzw. 0 und 59 enthalten.

Die **Zeilen 32 bis 37** zeichnen den Hintergrund der Uhr – die Ziffernblatteinträge und den umrundenden Kreis als Begrenzung des Ziffernblattes.

In den **Zeilen 39 bis 46** werden die Endpunkte für die Zeiger aus den Werten der aktuellen Uhrzeit berechnet. Dazu nutzen wir Winkelfunktionen Sinus und Cosinus aus dem Mathematikmodul von Python (`math`). Wenn du dazu Näheres erfahren willst, dann schau dir den Einschub im Infokasten an. Die berechneten Werte werden in Zweiertupeln gespeichert, die an die Methode `draw.line` übergeben werden können.

Die **Zeilen 46 bis 48** zeichnen die Zeiger über die Methode `draw.line`. Dabei wird der kleine Zeiger etwas dicker dargestellt und der Sekundenzeiger am dünnsten und in roter Farbe.

In **Zeile 49** wird das gezeichnete Uhrenbild zur Anzeige gebracht.

Schließlich können wir in **Zeile 50** die Frame-Rate auf 1 fps begrenzen, das heißt, die Aktualisierung der Anzeige erfolgt nur einmal pro Sekunde (fps = frames per second). Doch das reicht ja, da sich die Anzeige auch nur im Sekundentakt ändert.

Einschub: Berechnung der Zeigerkoordinaten

Um die Zeiger der Uhr korrekt zu zeichnen, müssen wir eine Linie vom Mittelpunkt bis zu der x- und y-Koordinate der aktuellen Minute, Stunde oder Sekunde zeichnen. Doch wie werden diese berechnet? Dazu braucht man ein wenig Mathematik. Wir haben es bei der Uhr mit einem Instrument zu tun, das die Uhrzeit über verschiedene Winkel der Zeiger anzeigt. Die Zeigerstellung 12 entspricht dabei einer Linie, die vom Mittelpunkt aus senkrecht nach oben geht. Wenn wir diese als einen Winkel von 0° festlegen, dann liegt die 3 bei 90°, die 6 bei 180° und die 9 bei 270°. Dies gilt entsprechend für die Minuten, 15 Minuten liegen bei 90°, 30 Minuten bei 180° usw.

Wir müssen also für die Stunden die Zahlen 0 bis 12 auf einem Winkel abbilden und für die Minuten die Zahlen 0 bis 60. Die Winkel kann man also ganz einfach über einen Dreisatz berechnen:

Stundenwinkel = Stunden * 360 / 12 = Stunden * 30

Minutenwinkel = Minuten * 360 / 60 = Minuten * 6

Den Stundenwinkel müssen wir allerdings noch etwas korrigieren. Wenn der Stundenwert z. B. 14 Uhr beträgt, dann müssen wir für die Ausgabe auf der Analoguhr den Wert 2 nehmen. Diese Umrechnung machen wir alle im Alltag ständig. Man macht sich aber leicht

6 Willkommen in der Cloud: Web Apps mit dem Raspberry Pi

Bisher hast du zwei generelle Möglichkeiten kennengelernt, um eine Anwendung oder eine App zum Laufen zu bringen: als Konsolenprogramm mit einfacher Textein- und -ausgabe oder als grafische Benutzeroberfläche mit einem GUI-Framework, wie z. B. Tkinter. In diesem Kapitel schauen wir uns eine weitere Art an, die immer wichtiger wird: Webapplikationen.

Bestimmt nutzt du in deinem Alltag irgendein Programm, auf das du mit dem Webbrowser zugreifst – und wenn es nur ein Terminkalender oder eine To-do-Liste ist oder ähnliche Organisationstools. Auch Spiele, auf die du mit dem Browser zugreifst, gehören in diese Kategorie. Diese Programme haben alle gemeinsam, dass sie auf einem Server ausgeführt werden, der sich von dem Gerät, an dem du das Programm aufrufst, im Allgemeinen unterscheidet. Du kannst Webprogramme ja auch von verschiedenen Geräten aus aufrufen. Egal, ob Smartphone, Laptop oder Tablet, du gibst einfach die richtige Adresse ein, meldest dich an und kannst genau an der Stelle weiterarbeiten, an der du letztes Mal oder auf einem anderen Gerät aufgehört hast. Diese Art, eine Anwendung an den Nutzer zu bringen, hat viele Vorteile und wird daher auch in der Softwareindustrie immer häufiger eingesetzt. Große Applikationen mit vielen tausend Nutzern laufen in Hochleistungsrechenzentren, aber das Ganze geht auch in klein.

Der Raspberry Pi ist eine ideale Plattform für einen privaten Webserver. Er ist klein und sparsam, er kann ohne Weiteres rund um die Uhr in Betrieb sein und das, ohne Lüftergeräusche zu machen oder die Umgebung merklich aufzuheizen, wie ein großer Server das tun würde. Mit einer Webschnittstelle löst du das Zugangsproblem zum Pi, wenn du keinen Monitor anschließen willst und der in Kapitel 2 beschriebene SSH-Zugang dir zu aufwendig ist, denn wenn du auf dem RasPi eine Webapplikation implementierst, kannst du innerhalb deines Heimnetzes von allen Geräten darauf zugreifen, die über einen Internetbrowser verfügen. Daraus ergeben sich viele spannende Projektideen. Wie wäre es mit einem Messenger-Service für Nachrichten in der Familie oder deine eigene To-do-Listen-App? Oder du programmierst eine Schnittstelle für deine Hausautomatisierung …

6.1 Erste Schritte mit Flask

Flask ist ein Python-Modul zur einfachen Programmierung von Web Apps, also von Anwendungen, deren Benutzerschnittstelle in einem Webbrowser läuft. Der Vorteil dieser Vorgehensweise liegt auf der Hand. Wenn du dein Programm als Web App umsetzt, dann ist es für jede Plattform lauffähig und natürlich automatisch netzwerkfähig. Gerade für den Raspberry Pi eignet sich diese Benutzerschnittstelle sehr gut, da viele Anwendungen wie Server oder Datenlogger die meiste Zeit unbeaufsichtigt im Hintergrund laufen. Für die seltenen Benutzerangriffe jedes Mal Tastatur und Bildschirm anzuschließen ist zwar möglich, einfacher und bequemer ist es aber, mit dem Smartphone oder Laptop einen Browser zu öffnen und damit dann die Einstellungen vorzunehmen.

Um dies zu realisieren, musst du nicht in die Tiefen der Netzwerkprogrammierung hinabsteigen und dich um Protokolle und Verbindungen kümmern. Dies kannst du stattdessen dem Framework überlassen. Die bekanntesten Frameworks für Web Apps in Python sind Django und Flask. Wir werden uns Flask etwas näher ansehen, da es leichter zu erlernen ist und trotzdem ermöglicht, beliebig umfangreiche Web App-Projekte zu ermöglichen.

Wie immer beginnen wir mit einer Hallo-Welt-Applikation. Wie diese in Flask aussieht, zeigt Listing 6.1.

Listing 6.1 Die erste Begegnung mit Flask

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route("/")
5  def hello():
6    return "Hallo Welt"
7
8  if __name__ == "__main__":
9    app.run(host='0.0.0.0', port=5000)
```

Wenn du das Programm startest, gibt Python in der Konsole folgende Meldung aus:

```
Running on http://localhost:5000/ (Press CTRL+C to quit
```

Das bedeutet, dass ein Service gestartet wurde, der unter der angegebenen IP-Adresse auf Verbindungen wartet.

`localhost` ist ein festgelegter Name für das eigene System. Immer wenn du irgendwo eine Webadresse oder URL eingeben kannst, steht dort `localhost` oder äquivalent dazu die numerische Adresse 127.0.0.1 für das eigene System. Wenn du aber auf einem normalen Office-PC „localhost“ in die Adresszeile eines Browsers eingibst, wird in dem meisten Fällen eine Fehlermeldung kommen. Das liegt einfach daran, dass auf dem System dann kein Webserver läuft, der auf Anfragen wartet und diese beantwortet.

Probieren wir aus, ob es beim RasPi anders ist. Starte dazu einen beliebigen Browser und gib in der URL-Zeile Folgendes ein, nachdem du Listing 6.1 ausgeführt hast:

```
http://localhost:5000
```

Der Browser antwortet mit den berühmten Worten „Hallo Welt“. Das sieht vielleicht simpel aus, hat es aber in sich, denn du hast gerade einen kompletten Webserver implementiert und gestartet. Natürlich liegen dort noch nicht viele Inhalte, aber du hast die technischen Voraussetzungen geschaffen, um den RasPi über ein Webbrowser-Interface zu steuern. Die Zahl 5000 hinter der Adresse ist übrigens die Portnummer. Sie dient dazu, unterschiedliche Dienste unter derselben IP-Adresse laufen zu lassen. Wenn du die IP-Adresse mit einer Adresse eines Hauses vergleichst, dann ist die Portnummer ein bestimmtes Zimmer in diesem Haus. Es ist möglich, verschiedene Dienste gleichzeitig laufen zu lassen, indem man diesen einfach unterschiedliche Portnummern gibt. Ein normaler Webseitenaufruf geht übrigens standardmäßig an Port 80, solange du keine andere Zahl angibst. Dies hat sich im Laufe der Zeit als Konvention durchgesetzt.

Zugriff von außen

In der Standardkonfiguration ist der Raspberry Pi in einem lokalen Netzwerk unter dem Hostnamen „raspberrypi“ sichtbar. Den Namen kannst du im Konfigurationstool des RasPi bearbeiten. Wenn dein RasPi unter der Standardadresse „raspberrypi“ im lokalen Netzwerk angemeldet ist, dann kannst du jetzt irgendein Gerät nehmen, das ebenfalls im lokalen Netz angemeldet ist, z. B. über einen W-LAN-Router. Öffne einen Webbrowser und gib als Adresse *http://raspberrypi:5000* ein. Schon meldet sich der von Python und Flask betriebene Webservice auf dem Tablet oder dem Smartphone und du hast somit eine Verbindung zu deinem RasPi aufgebaut (siehe Bild 6.1).

Bild 6.1 Der RasPi meldet sich als Webserver auch in einem externen Browser (hier in Chrome auf einem Windows-PC).

Zeile 1 dürfte klar sein, das Modul Flask wird hier importiert.

Zeile 2 erzeugt nun eine Instanz eines Flask-Objekts mit dem Namen `app`. Der Parameter `__name__` ist eine von Python reservierte Variable, die den Namen des Moduls enthält, zu

Index