Beachten Sie, dass das Erkennen von Flanken nur bei Pins funktioniert, die das auch unterstützen. In der Dokumentation zur Bibliothek steht zudem, dass die Information, die zu beliebigen Zeitpunkten aus dem Status gelesen wurde, abhängig von der Hardware nicht garantiert korrekt ist.

Projekt: ein einfaches Soundboard

Nachdem Sie nun wissen, wie Sie die Eingangs-Pins eines Raspberry Pi auslesen, können Sie die Soundfunktionen des Python-Moduls Pygame nutzen, um ein Soundboard zu basteln. Mit einem Soundboard können Sie kurze Audioaufzeichnungen abspielen, wenn Sie seine Taster drücken. Um Ihr eigenes Soundboard zu bauen, benötigen Sie neben Ihrem Raspberry Pi noch folgende Bauteile:

- 3 Drucktaster
- Jumper-Kabel Stecker auf Buchse
- Standard-Jumper-Kabel oder passend gekürzter Schaltdraht
- Steckplatine
- Computer-Lautsprecher oder einen HDMI-Monitor mit eingebauten Lautsprechern. Sie können auch ein einfaches Paar Kopfhörer verwenden – irgendetwas, das sich am analogen Audio-OUT-Port des Pi anschließen lässt.

Dazu brauchen Sie noch ein paar nicht komprimierte Sounddateien im *.wav*-Format. Zu Testzwecken gibt es auf dem Raspberry Pi schon ein paar solcher Dateien, die Sie einsetzen können. Funktioniert das Soundboard erst einmal, ist es ein Leichtes, diese Dateien durch die eigentlich gewünschten Dateien zu ersetzen, auch wenn Sie sie vorher vielleicht erst ins *.wav*-Format umwandeln müssen. Fangen Sie aber zunächst mit dem Schaltkreis an:

1. Verbinden Sie den Masse-Pin des Raspberry Pi über ein Jumper-Kabel mit Stecker und Buchse mit der negativen Leiste auf der Steckplatine.
2. Stecken Sie die drei Drucktaster so auf die Platine, dass sie den mittleren Bereich überspannen.

3. Verbinden Sie durch Standard-Jumper-Kabel oder kurzen Schaltdraht die Masseleiste der Steckplatine mit dem oberen Pin jedes Tasters.
4. Verbinden Sie durch Jumper-Kabel mit Stecker und Buchse den unteren Pin jedes Tasters mit den GPIO-Pins des Raspberry Pi. Für dieses Projekt haben wir die Pins 4, 14 und 25 verwendet.

Abb. 7–2 zeigt den vollständigen Aufbau.

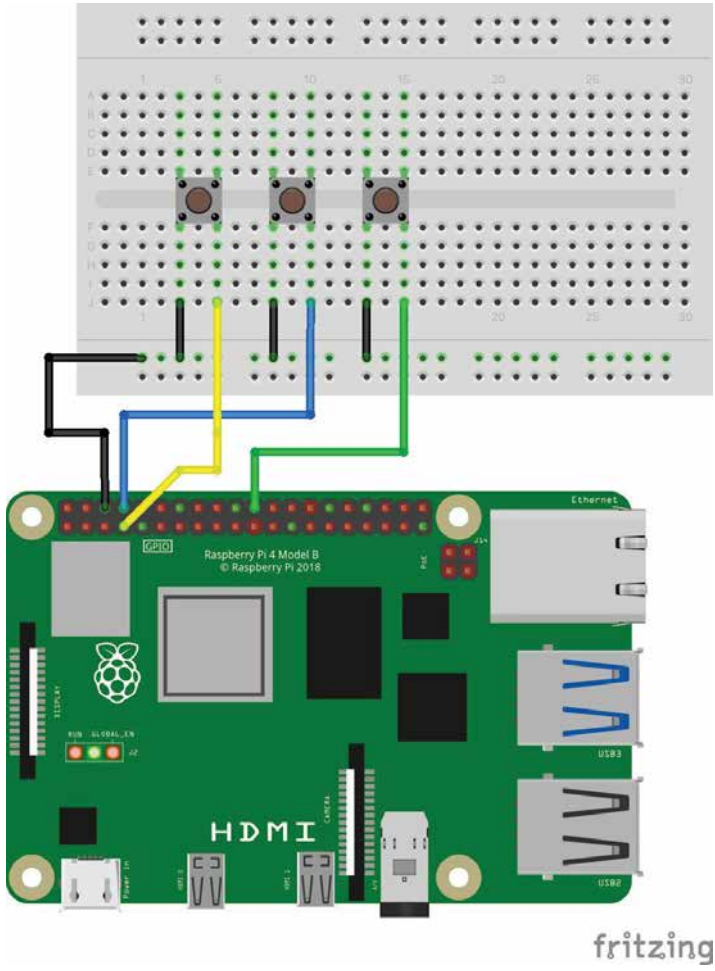

Abb. 7–2 Vollständiger Schaltkreis für das Soundboard-Projekt

Nachdem Sie die Steckplatine aufgebaut haben, müssen wir uns noch um den Code kümmern:

1. Erstellen Sie in Ihrem Home-Verzeichnis ein neues Verzeichnis namens *soundboard*.
2. Öffnen Sie diesen Ordner und erstellen Sie dort eine Datei mit dem Namen *soundboard.py*.
3. Öffnen Sie *soundboard.py* und geben Sie folgenden Code ein:

```
import pygame.mixer
from time import sleep
from gpiozero import Button
from sys import exit

button1 = Button(4)
button2 = Button(14)
button3 = Button(25) Again, BCM 4,14 and 25 match up with pins
4,14, and 25??

pygame.mixer.init(48000, -16, 1, 1024)  ❶
soundA = pygame.mixer.Sound(  ❷
    "/usr/share/sounds/alsa/Front_Center.wav")
soundB = pygame.mixer.Sound(
    "/usr/share/sounds/alsa/Front_Left.wav")
soundC = pygame.mixer.Sound(
    "/usr/share/sounds/alsa/Front_Right.wav")

soundChannelA = pygame.mixer.Channel(1)  ❸
soundChannelB = pygame.mixer.Channel(2)
soundChannelC = pygame.mixer.Channel(3)

print "Soundboard bereit."  ❹

while True:
    try:
       button1.wait_for_press()  ❺
       soundChannelA.play(soundA)  ❻
       button2.wait_for_press()
       soundChannelB.play(soundB)
       button3.wait_for_press()
       soundChannelC.play(soundC)
       sleep(.01)  ❼

except KeyboardInterrupt:  ❽
exit()
```

❶ Den Mixer von Pygame initialisieren.

❷ Die Klänge laden.

❸ Drei Kanäle einrichten – einen für jeden Klang –, sodass wir parallel drei verschiedene Klänge abspielen können.

❹ Den Anwender informieren, dass das Soundboard bereit ist.

❺ Den Taster abfragen. Wurde er gedrückt, die folgende Zeile ausführen.

❻ Den Klang abspielen.

❼ Den Prozessor nicht komplett in Beschlag nehmen, indem die Taster nicht häufiger als nötig abgefragt werden.

❽ Das Skript sauber verlassen, wenn der Anwender Strg-C drückt, ohne die Traceback Message angezeigt zu bekommen.

4. Wechseln Sie zur Befehlszeile und navigieren Sie zu dem Ordner, in dem Sie *soundboard.py* gespeichert haben. Führen Sie nun das Skript aus:

```
pi@raspberrypi ~/soundboard $ python3 soundboard.py
```

5. Nach der Anzeige von »Soundboard bereit« drücken Sie auf die Taster, um die Audio-Samples abzuspielen.

Abhängig davon, wie Ihr Raspberry Pi eingerichtet ist, werden die Audioinformationen über HDMI an Ihren Monitor übermittelt oder über die 3,5-mm-Audiobuchse auf dem Board ausgegeben. Um das zu ändern, verlassen Sie das Skript durch Strg-C und führen die folgende Befehlszeile aus, um die analoge Audioausgabe zu verwenden:

```
pi@raspberrypi ~/soundboard $ sudo amixer cset numid=31
```

Um die Audiodaten über HDMI an den Monitor zu schicken, verwenden Sie:

```
pi@raspberrypi ~/soundboard $ sudo amixer cset numid=32
```

Die mitgelieferten Klänge sind natürlich nicht sehr spannend, aber Sie können sie durch beliebige eigene Audiodaten ersetzen: Applaus, Gelächter, Summer und Plings. Kopieren Sie sie in das Verzeichnis *soundboard*

und passen Sie den Code so an, dass er auf sie zugreift. Wollen Sie mehr Klänge für Ihr Soundboard nutzen, fügen Sie weitere Taster hinzu und ändern Sie den Code entsprechend.

Weitere Informationen

Dokumentation zu gpiozero *(gpiozero.readthedocs.io/en/stable/)*
Die gpiozero-Bibliothek wird nicht nur immer noch aktiv entwickelt, sie enthält auch bergeweise Funktionalität, von der wir hier nur einen winzigen Teil eingesetzt haben. Sie sollten auf der Homepage des Projekts in der Dokumentation stöbern, ein paar Komponenten auf der Steckplatine verbinden und schauen, was Sie damit anstellen können.