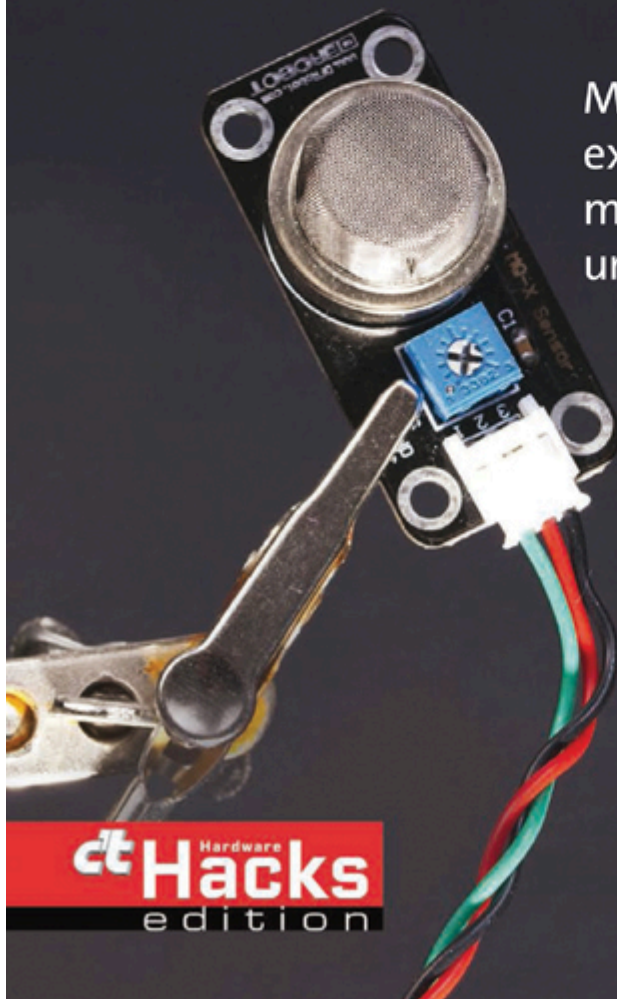



Kimmo Karvinen · Tero Karvinen · Ville Valtokari

# Sensoren

Messen und  
experimentieren  
mit **Arduino**  
und **Raspberry Pi**



**ct** Hardware  
**Hacks**  
edition

 dpunkt.verlag

# Inhaltsverzeichnis

## **Vorwort**

## **1 Raspberry Pi**

### 1.1 Raspberry Pi: Von null zum ersten Start

1.1.1 NOOBS\*.zip entpacken

1.1.2 Kabel anschließen

1.1.3 Raspbian hochfahren und installieren

1.1.4 Fehlersuche bei der Raspberry Pi-Installation

### 1.2 Willkommen bei Linux

1.2.1 Die allgegenwärtige Kommandozeile

1.2.2 Schauen Sie sich um

1.2.3 Textdateien für die Konfiguration

1.2.4 sudo mach mir ein Butterbrot!

### 1.3 Elektronische Bauteile an die Pins des Raspberry Pi anschließen

1.3.1 Hallo GPIO, lass eine LED blinken

1.3.2 Die Schaltung aufbauen

1.3.3 Zwei Nummerierungssysteme: Zweck und Ort

1.3.4 GPIO-Pins über die Kommandozeile steuern

1.3.5 Dateien ohne Editor bearbeiten

1.3.6 Die LED aufleuchten lassen

1.3.7 Fehlerbehebung

### 1.4 GPIO-Steuerung ohne Root-Berechtigungen

1.4.1 Fehlersuche bei der GPIO-Steuerung

### 1.5 GPIO in Python

1.5.1 Hello Python

### 1.6 Wie geht es weiter?

## **2 Arduino**

### 2.1 Grundinstallation des Arduino

- 2.1.1 Ubuntu Linux
- 2.1.2 Windows 7 und Windows 8
- 2.1.3 OS X
- 2.1.4 Hello World
- 2.1.5 Der Aufbau eines Arduino-Programms
- 2.1.6 Einfach und vielseitig dank Shields

## **3 Entfernung**

### 3.1 Experiment: Abstände mit Ultraschall messen (Ping)

- 3.1.1 Code und Schaltung für den Ping am Arduino
- 3.1.2 Code und Schaltung für den Ping am Raspberry Pi

### 3.2 Ultraschallsensor HC-SR04

- 3.2.1 Code und Schaltung für den HC-SR04 am Arduino
- 3.2.2 Code und Schaltung für den HC-SR04 am Raspberry Pi
- 3.2.3 Echoberechnungen
- 3.2.4 Praxisexperiment: Unsichtbare Objekte

### 3.3 Experiment: Hindernisse mit Infrarot erkennen (IR-Abstandssensor)

- 3.3.1 Code und Schaltung für den IR-Sensor am Arduino
- 3.3.2 Code und Schaltung für den IR-Sensor am Raspberry Pi

### 3.4 Praxisexperiment: Infrarotlicht sichtbar machen

### 3.5 Experiment: Bewegungen mit Infrarot verfolgen (IR-Facettenauge)

- 3.5.1 Code und Schaltung für das Facettenauge am Arduino
- 3.5.2 Code und Schaltung für das Facettenauge am Raspberry Pi
- 3.5.3 Bibliothek spidev installieren
- 3.5.4 Alternative Schaltungen für den Raspberry Pi

### 3.6 Testprojekt: Haltungswarner (Arduino)

- 3.6.1 Lernziele

- 3.6.2 Piezo-Summer
- 3.6.3 Alarm!
- 3.6.4 Piezo-Summer und IR-Sensor kombinieren
- 3.6.5 Eine elegante Verpackung für das Projekt

## **4 Rauch und Gas**

- 4.1 Experiment: Rauchmelder (analoger Gassensor)
  - 4.1.1 Code und Schaltung für den MQ-2 am Arduino
  - 4.1.2 Code und Schaltung für den MQ-2 am Raspberry Pi
  - 4.1.3 Praxisexperiment: Rauch steigt nach oben
  - 4.1.4 Experiment: Alkotest (Alkoholsensor MQ-303A)
  - 4.1.5 Praxisexperiment: Nüchtern bleiben beim Experimentieren
- 4.2 Testprojekt: Rauchalarm per E-Mail senden
  - 4.2.1 Lernziele
  - 4.2.2 Python für E-Mails und Social Media
  - 4.2.3 Das Projekt bauen
  - 4.2.4 Wie funktioniert E-Mail?
  - 4.2.5 Kann der Arduino E-Mails senden? – Nicht ohne Weiteres!
  - 4.2.6 Code für den Raspberry Pi
  - 4.2.7 Gehäuse

## **5 Berührung**

- 5.1 Experiment: Drucktasten
  - 5.1.1 Pullup-Widerstand
  - 5.1.2 Code und Schaltung am Arduino
  - 5.1.3 Code und Schaltung am Raspberry Pi
- 5.2 Experiment: Mikroschalter
  - 5.2.1 Code und Schaltung für den Mikroschalter am Arduino
  - 5.2.2 Code und Schaltung für den Mikroschalter am Raspberry Pi
- 5.3 Experiment: Potenziometer (regelbarer Widerstand, Poti)

- 5.3.1 Code und Schaltung für das Potenziometer am Arduino
- 5.3.2 Code und Schaltung für das Potenziometer am Raspberry Pi
- 5.4 Experiment: Berührungsfreier Berührungssensor (kapazitiver Berührungssensor QT113)
  - 5.4.1 Code und Schaltung für den QT113 am Arduino
  - 5.4.2 Code und Schaltung für den QT113 am Raspberry Pi
- 5.5 Praxisexperiment: Berührungen durch eine hölzerne Oberfläche hindurch erkennen
- 5.6 Experiment: Druck messen (FlexiForce)
  - 5.6.1 Code und Schaltung für den FlexiForce am Arduino
  - 5.6.2 Code und Schaltung für den FlexiForce am Raspberry Pi
- 5.7 Experiment: Berührungssensor im Eigenbau
  - 5.7.1 Code und Schaltung für den kapazitiven Sensor am Arduino
  - 5.7.2 Code und Schaltung für den kapazitiven Sensor am Raspberry Pi
- 5.8 Testprojekt: Geisterglocke
  - 5.8.1 Lernziele
  - 5.8.2 Servomotoren
  - 5.8.3 Code und Schaltung für die Geisterglocke am Arduino
  - 5.8.4 Servo an der Glocke montieren

## **6 Bewegung**

- 6.1 Experiment: Wo ist oben? (Tilt-Schalter)
  - 6.1.1 Code und Schaltung für den Kippsensor am Arduino
  - 6.1.2 Code und Schaltung für den Kippsensor am Raspberry Pi
- 6.2 Experiment: Good Vibrations mit Interrupts (digitaler Vibrationssensor)
  - 6.2.1 Code und Schaltung für den Vibrationssensor am Arduino
  - 6.2.2 Code und Schaltung für den Vibrationssensor am Raspberry Pi
- 6.3 Experiment: Am Regler drehen
  - 6.3.1 Code und Schaltung für den Drehgeber am Arduino

- 6.3.2 Code und Schaltung für den Drehgeber am Raspberry Pi
- 6.4 Experiment: Analoger Zweiachs-Daumen-Joystick
  - 6.4.1 Code und Schaltung für den Joystick am Arduino
  - 6.4.2 Code und Schaltung für den Joystick am Raspberry Pi
- 6.5 Praxisexperiment: Teile eines Xbox-Controllers wiederverwenden
- 6.6 Experiment: Alarmanlage (passiver IR-Sensor)
  - 6.6.1 Code und Schaltung für die Alarmanlage am Arduino
  - 6.6.2 Code und Schaltung für die Alarmanlage am Raspberry Pi
  - 6.6.3 Praxisexperiment: Eine Alarmanlage überlisten
- 6.7 Testprojekt: Pong
  - 6.7.1 Lernziele
  - 6.7.2 Tipps für ein ansprechendes Gehäuse
  - 6.7.3 Das Spiel beim Hochfahren des Raspberry Pi automatisch starten

## **7 Licht**

- 7.1 Experiment: Feuer erkennen (Flammensensor)
  - 7.1.1 Code und Schaltung für den Flammensensor am Arduino
  - 7.1.2 Code und Schaltung für den Flammensensor am Raspberry Pi
- 7.2 Praxisexperiment: Genauigkeit bei der Erkennung von Flammen
- 7.3 Experiment: Siehst du dieses Licht? (Fotowiderstand, LDR)
  - 7.3.1 Code und Schaltung für den Fotowiderstand am Arduino
  - 7.3.2 Code und Schaltung für den Fotowiderstand am Raspberry Pi
- 7.4 Praxisexperiment: Die Richtung erkennen
- 7.5 Experiment: Linien verfolgen
  - 7.5.1 Code und Schaltung für den Liniensensor am Arduino
  - 7.5.2 Code und Schaltung für den Liniensensor am Raspberry Pi
- 7.6 Praxisexperiment: Schwarz ist weiß
- 7.7 Experiment: Alle Farben des Regenbogens
  - 7.7.1 Code und Schaltung für den Farbsensor am Arduino

- 7.7.2 Code und Schaltung für den Farbsensor am Raspberry Pi
- 7.8 Testprojekt: Chamäleonkuppel
  - 7.8.1 Lernziele
  - 7.8.2 RGB-LEDs
  - 7.8.3 Code und Schaltung für RGB-LEDs am Arduino
  - 7.8.4 Gleitender Mittelwert
  - 7.8.5 Beliebige Farben mit RGB-LEDs erzeugen
  - 7.8.6 Eingänge in Ausgänge umwandeln
  - 7.8.7 Code für eine Kombination aus RGB-LED und Farbsensor
  - 7.8.8 Tipps zum Bau der Kuppel

## **8 Beschleunigung**

- 8.1 Beschleunigung und Winkelgeschwindigkeit
- 8.2 Experiment: Beschleunigung mit dem MX2125 messen
  - 8.2.1 Impulslängen des MX2125 entschlüsseln
  - 8.2.2 Code und Schaltung für den Beschleunigungsmesser am Arduino
  - 8.2.3 Code und Schaltung für den Beschleunigungsmesser am Raspberry Pi
- 8.3 Experiment: Beschleunigungsmesser und Gyroskop
  - 8.3.1 Code und Schaltung für den MPU 6050 am Arduino
  - 8.3.2 Code und Schaltung für den MPU 6050 am Raspberry Pi
  - 8.3.3 Hexadezimale, binäre und andere Zahlen
  - 8.3.4 Bitweise Operationen
- 8.4 Experiment: Den Wii Nunchuk zweckentfremden (über I2C)
  - 8.4.1 Code und Schaltung für den Nunchuk am Arduino
  - 8.4.2 Code und Schaltung für den Nunchuk am Raspberry Pi
- 8.5 Testprojekt: Steuerung einer Roboterhand mit dem Wii Nunchuk
  - 8.5.1 Lernziele
  - 8.5.2 Die Mechanik der Roboterhand

## **9 Identität**

### 9.1 Tastenfelder

9.1.1 Code und Schaltung für das Tastenfeld am Arduino

9.1.2 Code und Schaltung für das Tastenfeld am Raspberry Pi

### 9.2 Praxisexperiment: Fingerabdrücke erkennen

### 9.3 Fingerabdruckscanner GT-511C3

9.3.1 Code und Schaltung für den Fingerabdrucksensor am Arduino Mega

9.3.2 Code und Schaltung für den Fingerabdrucksensor am Raspberry Pi

### 9.4 RFID mit dem ELB149C5M

9.4.1 Code und Schaltung für das RFID-Lesegerät am Arduino Mega

9.4.2 Code und Schaltung für das RFID-Lesegerät am Raspberry Pi

### 9.5 Testprojekt: Futuristische Schatztruhe

9.5.1 Lernziele

9.5.2 Die Funktionsweise der Truhe

9.5.3 Die Truhe

9.5.4 Code und Schaltung für die futuristische Schatztruhe am Arduino

### 9.6 Menschen und Objekte erkennen

## **10 Elektrizität und Magnetismus**

### 10.1 Experiment: Spannung und Stromstärke

10.1.1 Code und Schaltung für den AttoPilot am Arduino

10.1.2 Code und Schaltung für den AttoPilot am Raspberry Pi

### 10.2 Experiment: Ist es magnetisch?

10.2.1 Code und Schaltung für den Hall-Effekt-Sensor am Arduino

10.2.2 Code und Schaltung für den Hall-Effekt-Sensor am Raspberry Pi

### 10.3 Experiment: Magnetischer Norden mit dem Kompass/Beschleunigungsmesser LSM303 bestimmen

10.3.1 Den Sensor kalibrieren

10.3.2 Code und Schaltung für den LSM303 am Arduino

- 10.3.3 Code und Schaltung für den LSM303 am Raspberry Pi
- 10.3.4 Das Protokoll für den LSM303
- 10.3.5 Berechnungen für die Kompassrichtung
- 10.4 Experiment: Hall-Schalter
  - 10.4.1 Code und Schaltung für den Hall-Schalter am Arduino
  - 10.4.2 Code und Schaltung für den Hall-Schalter am Raspberry Pi
- 10.5 Testprojekt: Webüberwachung von Solarzellen
  - 10.5.1 Lernziele
  - 10.5.2 Die Solarzellen anschließen
  - 10.5.3 Der Raspberry Pi als Webserver
  - 10.5.4 Die IP-Adresse ermitteln
  - 10.5.5 Eine Homepage auf dem Raspberry Pi einrichten
  - 10.5.6 Code und Schaltung für die Solarzellenüberwachung am Raspberry Pi
  - 10.5.7 Zeitliche Planung von Aufgaben mit cron
- 10.6 Wie geht es weiter?

## **11 Schall**

- 11.1 Experiment: Stimmen hören/Lautstärke
  - 11.1.1 Code und Schaltung für das Mikrofon am Arduino
  - 11.1.2 Code und Schaltung für das Mikrofon am Raspberry Pi
- 11.2 Praxisexperiment: Eine Stecknadel fallen hören
- 11.3 Testprojekt: Töne über HDMI sichtbar machen
  - 11.3.1 Lernziele
  - 11.3.2 Den seriellen Port des Raspberry Pi aktivieren
  - 11.3.3 Code und Schaltung für die Visualisierung am Raspberry Pi
  - 11.3.4 Schnelle Fourier-Transformation
- 11.4 Wie geht es weiter?

## **12 Wetter und Klima**

- 12.1 Experiment: Heiß hier drin, oder?
  - 12.1.1 Code und Schaltung für den LM35 am Arduino
  - 12.1.2 Code und Schaltung für den LM35 am Raspberry Pi
- 12.2 Praxisexperiment: Temperaturwechsel
- 12.3 Experiment: Wie feucht ist es?
  - 12.3.1 Wie feucht ist Ihr Atem?
  - 12.3.2 Code und Schaltung für den DHT11 am Arduino
  - 12.3.3 Code und Schaltung für den DHT11 am Raspberry Pi
  - 12.3.4 Kommunikation vom Raspberry Pi zum Arduino
- 12.4 Luftdrucksensor GY-65
  - 12.4.1 Code und Schaltung für den GY-65 am Arduino
  - 12.4.2 Arduino-Bibliotheken verwenden
  - 12.4.3 Die Arduino-Bibliothek für den GY-65
  - 12.4.4 Code und Schaltung für den GY-65 am Raspberry Pi
- 12.5 Experiment: Brauchen Ihre Pflanzen Wasser? (Sensor für Bodenfeuchtigkeit)
  - 12.5.1 Code und Schaltung für den Bodenfeuchtigkeitssensor am Arduino
  - 12.5.2 Code und Schaltung für den Bodenfeuchtigkeitssensor am Raspberry Pi
- 12.6 Testprojekt: Wettervorhersage auf E-Paper
  - 12.6.1 Lernziele
  - 12.6.2 Code und Schaltung für die Wetterstation am Arduino
- 12.7 Praxisexperiment: Schau mal, ganz ohne Stromanschluss!
- 12.8 Bilder in Headerdateien speichern
  - 12.8.1 Programm zur Umwandlung von BMP-Bildern in C-Dateien
  - 12.8.2 Tipps für das Gehäuse

## **A Linux-Befehle für den Raspberry Pi**

### **Index**

## 4 Rauch und Gas

*PIEP, PIEP, PIEP! Das laute Geräusch eines Rauchmelders hat schon viele Leben gerettet und Menschen geweckt, bevor das Kohlenmonoxid sie in ewigen Schlaf versetzen konnte. Eine andere Form von Gassensor, der Alkoholtester, hat viele Betrunkene daran gehindert, mit dem Auto loszufahren und tödliche Unfälle zu bauen.*

Wenn Sie bei der Arbeit oder bei einem eigentlich interessanten Kurs zu gähnen beginnen, kann das an Kohlendioxid ( $\text{CO}_2$ ) liegen. Dieses Gas wird von allen Tieren (und Menschen) ausgeatmet. Sensoren im Lüftungssystem eines Gebäudes können einen erhöhten  $\text{CO}_2$ -Pegel feststellen und dann für die erforderliche Frischluftzufuhr sorgen.

Die Feuerwehr kann messen, ob sich Kohlenwasserstoffe in der Luft befinden, um keine explosiven Überraschungen zu erleben. Auch in einem Automotor werden Gassensoren eingesetzt, um das Benzin-Luft-Verhältnis zu ermitteln. Nur wenn dieses Verhältnis korrekt ist, wird das gesamte Benzin im Zylinder verbrannt – es wäre nicht so gut, wenn Benzin aus dem Auspuff tropfen würde.

Es ist einfach und macht Spaß, Prototypen mit billigen Rauch- und Gasdetektoren zu entwickeln. Für echte Sicherheitsprodukte gibt es jedoch strenge Anforderungen, Testprotokolle und Zertifizierungsprogramme. Die Projekte in diesem Kapitel sind kein Ersatz für solche Produkte!

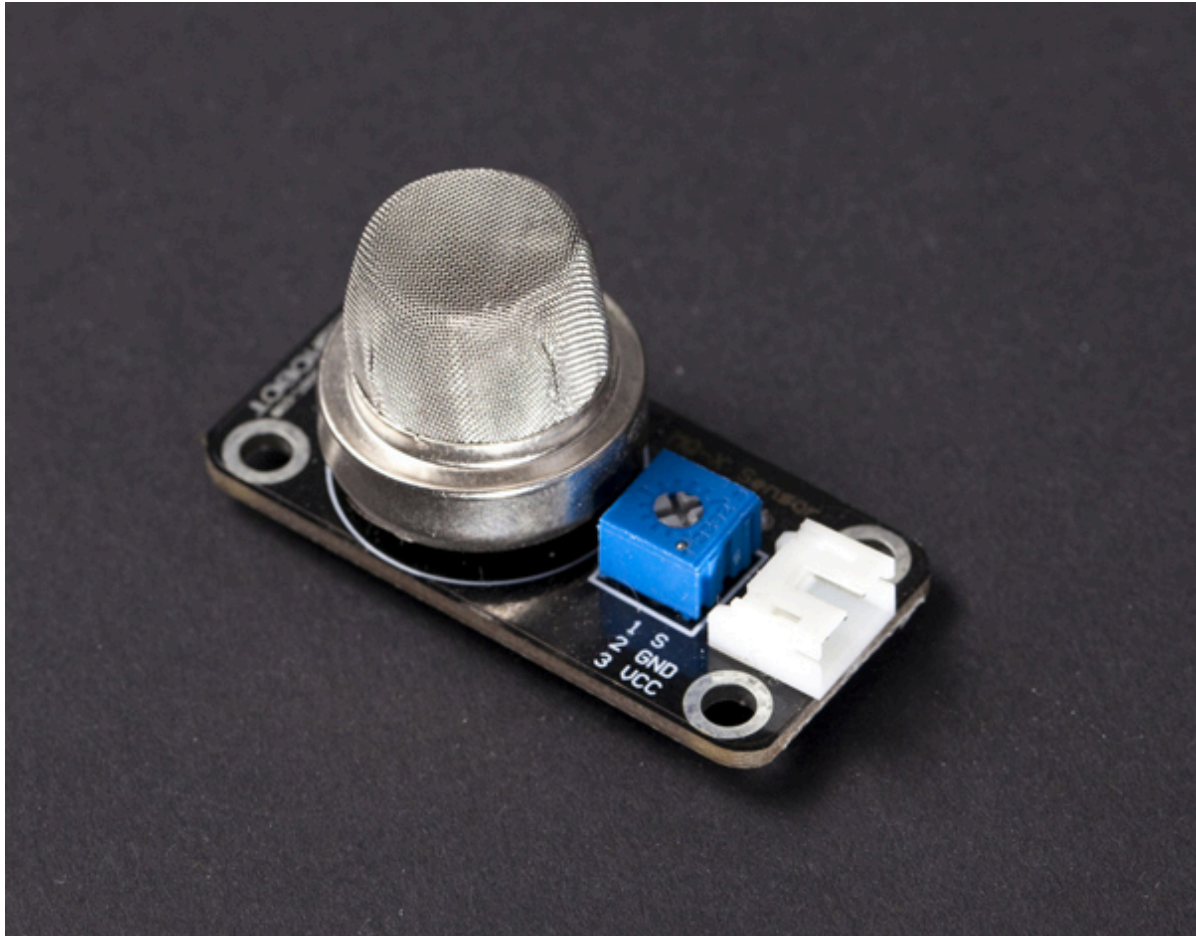
Die sogenannte MQ-Produktreihe umfasst preiswerte Sensoren für verschiedene Gase. Eine Auswahl davon finden Sie in Tabelle 4-1.

MQ-Sensor	Nachgewiesenes Gas
MQ-2	Brennbare Gase und Rauch
MQ-3, MQ-303A	Alkohol (Ethanol)
MQ-4	Methan (CH <sub>4</sub> )
MQ-7	Kohlenmonoxid
MQ-8	Wasserstoff
MQ-9	Kohlenmonoxid, Methan, Kohlenwasserstoffe (Propan oder Butan)

**Tab. 4-1** MQ-Gassensoren

## 4.1 Experiment: Rauchmelder (analoger Gassensor)

Ein MQ-2-Rauchsensor meldet das Vorhandensein von Rauch anhand der ausgehenden Spannung. Der von uns verwendete MQ-2 verfügt über ein eingebautes Potenziometer, mit dem die Empfindlichkeit eingestellt werden kann (siehe Abb. 4-1).



**Abb. 4-1** Analoger Gassensor

Was ist das gefährlichste giftige Gas? Wenn Sie die Gefährlichkeit an der Anzahl der Opfer festmachen, dann ist es Kohlenmonoxid (CO). Bei einem Brand sterben die meisten Opfer an einer Rauchvergiftung, bevor die Flammen sie erreichen.

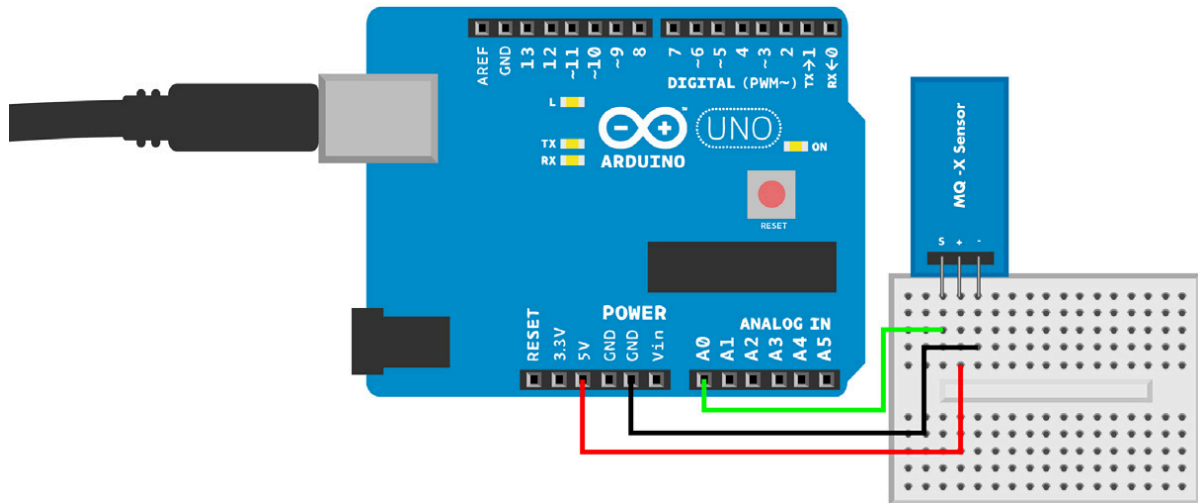
Das tödliche CO ist etwas anderes als das größtenteils harmlose Kohlendioxid (CO<sub>2</sub>). Der menschliche Körper erzeugt bei der Energiegewinnung CO<sub>2</sub>, und das Blut transportiert dieses Gas zu den Lungen, sodass es ausgeatmet werden kann.

CO ist tödlich, da es den Blutstrom nicht verlassen will. Es heftet sich an das Hämoglobin im Blut und verhindert für Stunden, dass das Blut Sauerstoff und CO<sub>2</sub> transportiert. Wenn das gesamte Hämoglobin mit CO besetzt ist, erhält das Gewebe keinen Sauerstoff mehr, woraufhin der Tod eintritt.

Da der MQ-2 drei Kontakte hat, braucht er keinen Pullup- oder Pulldown-Widerstand. Für den Arduino ist der MQ-2 ein Potenziometer mit drei Anschlüssen.

Der MQ-2 ist an Masse (schwarz, 0 V) und die +5-V-Stromversorgung (rot) angeschlossen. Wenn er Rauch erkennt, erhöht er die Spannung an seinem Pin S (näher in Richtung +5 V).

### 4.1.1 Code und Schaltung für den MQ-2 am Arduino



**Abb. 4-2** Schaltung für den MQ-2-Sensor am Arduino

```
// mq_x_smoke_sensor.ino - Gibt den Rauchgehalt am seriellen Monitor aus
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = A0;
int smoke_level = -1;      1

void setup() {
  Serial.begin(115200); // bit/s
  pinMode(sensorPin, INPUT);
}

void loop() {
  smoke_level = analogRead(sensorPin);      2
  Serial.println(smoke_level);
  if(smoke_level > 120) {                   3
    Serial.println("Smoke detected");
  }
  delay(100); // ms
}
```

**Listing 4-1** mq\_x\_smoke\_sensor.ino

- 1 Initialisiert den Rauchgehalt mit einem unrealistischen Wert, um das Debugging zu erleichtern. Wenn Sie in der Ausgabe -1 sehen, dann wissen Sie,

dass `analogRead()` den Initialisierungswert nie durch einen echten Wert ersetzt hat.

- 2 Der MQ-2 ist ein einfacher analoger Widerstandssensor, weshalb Sie den Wert mit `analogRead()` lesen können. Die Funktion gibt einen Wert zwischen 0 (0 V) und 1023 (+5 V) zurück.
- 3 Im Code ist zwar ein Grenzwert angegeben, aber es ist trotzdem besser, die Empfindlichkeit mit dem Potenziometer am Sensor selbst einzustellen.

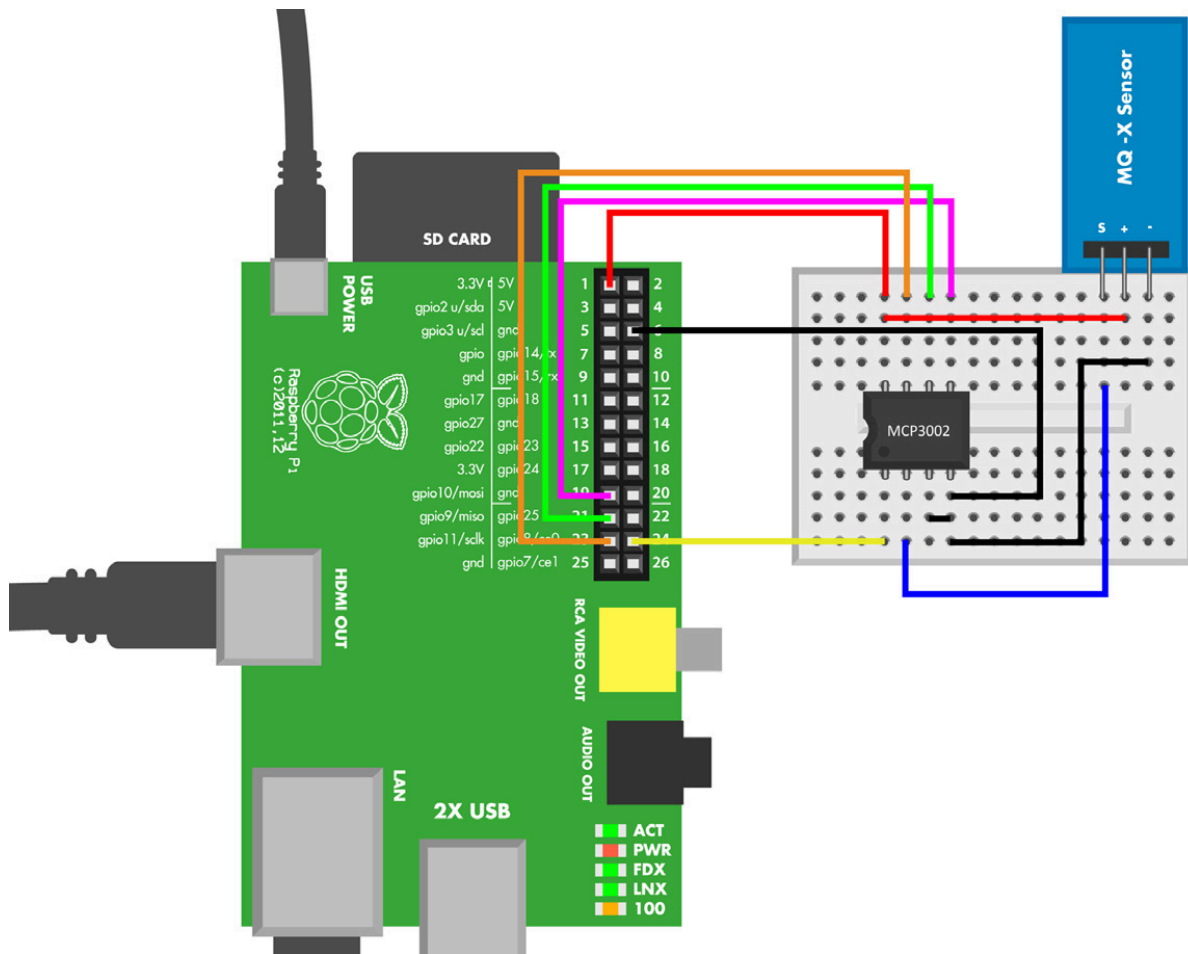
Ein Rauchmelder muss kein langweiliger Kasten sein, wie der von Paola Suhonen entworfene Lento beweist (siehe Abb. 4-3). Die Mottenform gibt diesem Alltagsgegenstand ein ganz neues Erscheinungsbild. Denken Sie beim Design Ihrer eigenen Geräte daran!



**Abb. 4-3** Der von Paola Suhonen gestaltete Rauchmelder Lento

#### 4.1.2 Code und Schaltung für den MQ-2 am Raspberry Pi

Um den MQ-2 auslesen zu können, braucht der Raspberry Pi einen externen Analog-Digital-Wandler (ADC). Ähnlich wie für andere analoge Widerstandssensoren können Sie auch hierfür den preiswerten MCP3002-Chip verwenden (siehe [Abschnitt 3.5.2](#)).



**Abb. 4-4** Schaltung für den MQ-2-Sensor am Raspberry Pi

```

# mq_x_smoke_sensor.py - Gibt den Rauchgehalt aus
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp          1

smokeLevel = 0

def readSmokeLevel():
    global smokeLevel
    smokeLevel = mcp.readAnalog()      2

def main():
    while True:                        3
        readSmokeLevel()               4
        print("Current smoke level is %i " % smokeLevel)  5
        if smokeLevel > 120:
            print("Smoke detected")
        time.sleep(0.5) # s

```

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

#### **Listing 4-2** *mq\_x\_smoke\_sensor.py*

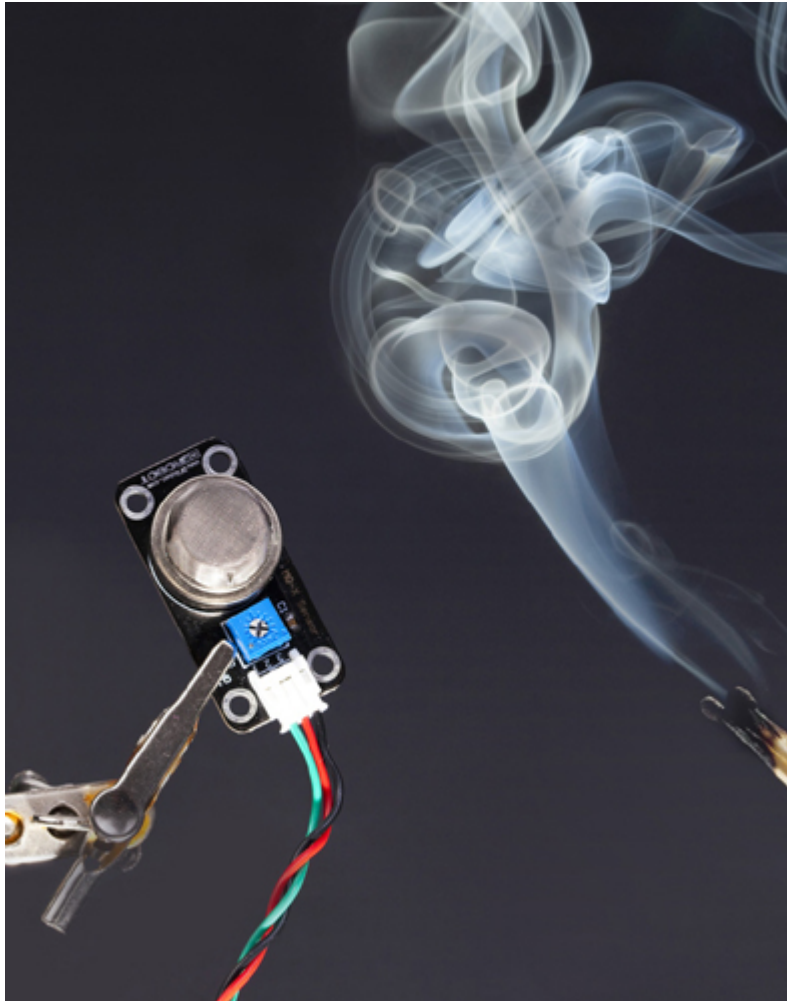
- 1 Die Bibliothek von *botbook.com* für den MCP3002 spart viel Programmierarbeit, da sie das Lesen analoger Sensorwerte vereinfacht. Die Bibliothek *botbook\_mcp3002.py* muss sich im selben Verzeichnis befinden wie das Programm *mq\_x\_smoke\_sensor.py*. Außerdem müssen Sie die Bibliothek *spidev* installieren, die von *botbook\_mcp3002.py* importiert wird. Mehr darüber erfahren Sie in den Kommentaren zu Beginn von *botbook\_mcp3002/botbook\_mcp3002.py* und in **Abschnitt 3.5.3**.
- 2 Liest die Spannung des ersten am MCP3002 angeschlossenen Geräts. Die Parameter in `readAnalog(device=0, channel=0)` sind Standardwerte, weshalb Sie auch auf die Angabe verzichten können.
- 3 Die permanente Wiederholung eines Programms ist bei eingebetteten Geräten üblich. Wenn Sie `while True` verwenden, sollten Sie am Ende der Schleife immer eine Verzögerung einfügen. Um das Programm abubrechen, drücken Sie **(Strg) + (C)**.
- 4 Es ist praktisch, die Hauptaufgabe des Programms in eine eigene Funktion zu packen. Der Zweck von `readSmokeLevel()` geht aus dem Namen hervor, was hilfreich ist, wenn Sie diese Funktion als Baustein für ein größeres Projekt mit vielen Sensoren und Ausgängen verwenden möchten.
- 5 Stellt den auszugebenden String mithilfe eines Formatierungsstrings zusammen. Der Wert von `smokeLevel` wird als Integer behandelt und ersetzt den Platzhalter `%i` im Formatierungsstring.

### **4.1.3 Praxisexperiment: Rauch steigt nach oben**

Es gibt einen guten Grund dafür, dass Rauchmelder an der Decke angebracht sind und nicht am Fußboden: Rauch wird gewöhnlich durch Brände erzeugt und ist daher wärmer als die umgebende Luft. Warme Gase aber sind weniger dicht, also leichter.

Warum ist warme Luft leichter als kalte? Wärme ist Bewegung: Moleküle wackeln und stoßen dabei gegeneinander. Je wärmer es wird, umso stärker wackeln und stoßen sich die Moleküle. Dabei drücken sie sich gegenseitig immer weiter weg, wodurch das Gas weniger dicht wird. Mit weniger

Molekülen in einem bestimmten Volumen wird das Gas leichter, und wenn es von schwererer Luft umgeben ist, steigt es nach oben.



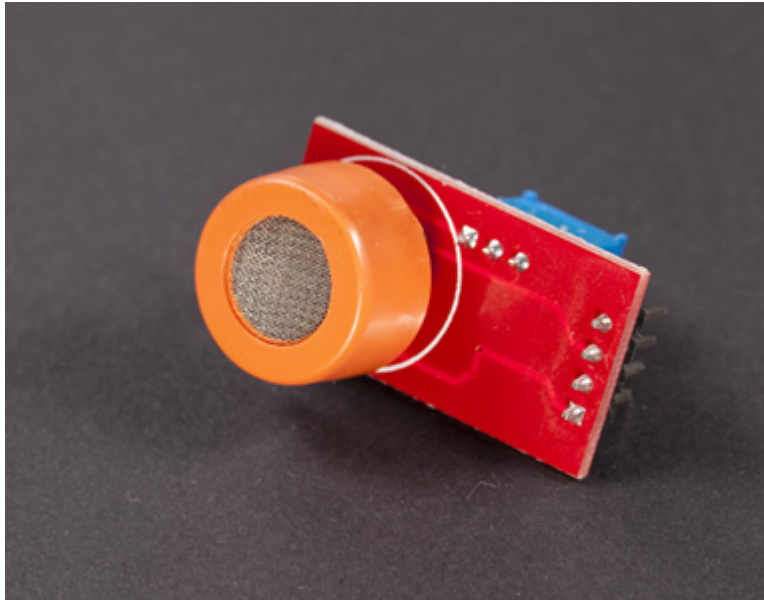
**Abb. 4-5** Rauch steigt nach oben.

Führen Sie das Rauchmelderprogramm aus und pusten Sie ein angezündetes Streichholz in der Nähe des Sensors aus. Wie hoch können Sie das ausgepustete Streichholz halten, sodass noch ein Messwert angezeigt wird. Wenn sich das Streichholz auf der Höhe des Sensors befindet, wird wahrscheinlich kein Unterschied mehr zu normaler Luft angezeigt. Wie der Titel des Experiments schon sagt, steigt Rauch nach oben. Darum kann er auch nicht gemessen werden, wenn Sie ihn oberhalb des Sensors aufsteigen lassen.

#### **4.1.4 Experiment: Alkotest (Alkoholsensor MQ-303A)**

Mit einem Alkotestgerät wird geprüft, ob jemand Alkohol in seinem Blut hat – genauer gesagt, Ethanol, die Art von Alkohol, die in Wein, Bier, Likör usw. enthalten ist.

Der Alkoholsensor MQ-3 sieht den anderen Gassensoren der MQ-Reihe sehr ähnlich (siehe Abb. 4–6).



**Abb. 4–6** Der Alkoholsensor MQ-3

Bevor Sie sich auf die Experimentierreihe freuen und sich schon mal ein Glas Castillo Montroy einschenken, sollten Sie zumindest einen Blick auf den [Abschnitt 4.1.5](#) werfen.

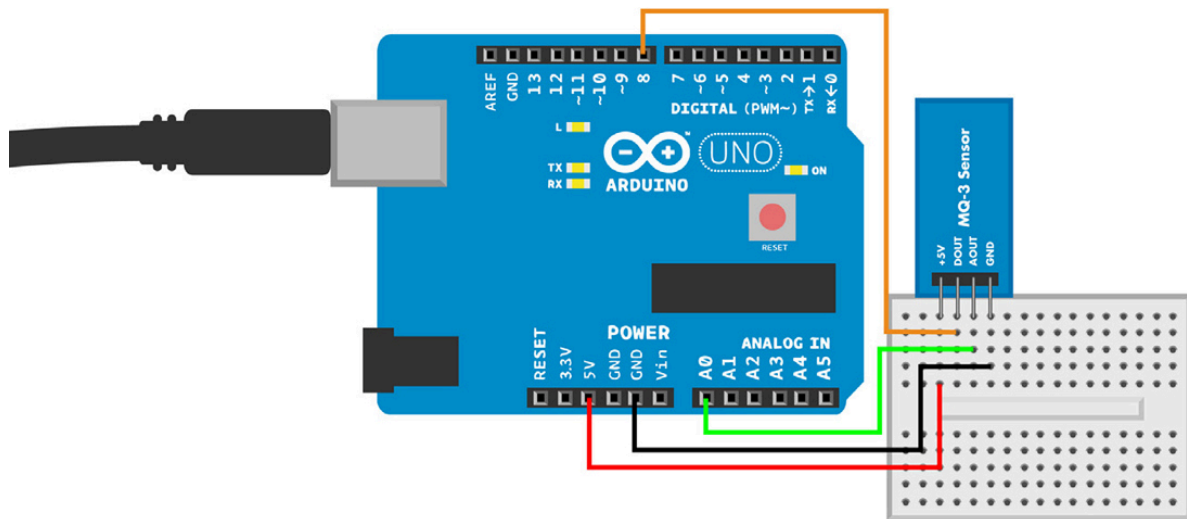
Beim Gasaustausch in den Lungen nimmt der Körper nicht nur Sauerstoff auf und wird Kohlendioxid los, sondern entlässt auch etwas von dem im Blut transportierten Alkohol in den Atem. Das ist der Alkohol, der von dem Gerät gemessen wird.

Je mehr Alkohol im Blut vorhanden ist, umso höher ist der Anteil an Alkohol in Ihrem Atem. Die Blutalkoholkonzentration ist ein gutes Maß dafür, wie betrunken jemand ist.

Zwar wird man mit steigender Menge von Alkohol im Blut immer betrunken, aber die Auswirkungen unterscheiden sich sehr stark von Person zu Person. Für die Gesetzgebung wird ein typischer Wert als Grenzwert verwendet. Der Grenzwert, von dem an man in Finnland für Fahren unter Alkoholeinfluss belangt werden kann, liegt beispielsweise bei 0,5 Promille.

Offiziell zugelassene Alkotestgeräte werden regelmäßig kalibriert, um genaue Messergebnisse zu garantieren. Solche Geräte können die Blutalkoholkonzentration anhand des Alkoholanteils in der Atemluft mittels einer eingebauten Formel berechnen.

Abbildung 4-7 zeigt die MQ-3-Schaltung für den Arduino, Abbildung 4-8 die für den Raspberry Pi. Den Arduino-Sketch finden Sie in Listing 4-3, den Python-Code in Listing 4-4.



**Abb. 4-7** Schaltung für den MQ-3-Sensor am Arduino

```
// mq_3_alcohol_sensor.ino - Gibt Alkoholwert und Angaben
// zum Grenzwert digital aus.
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

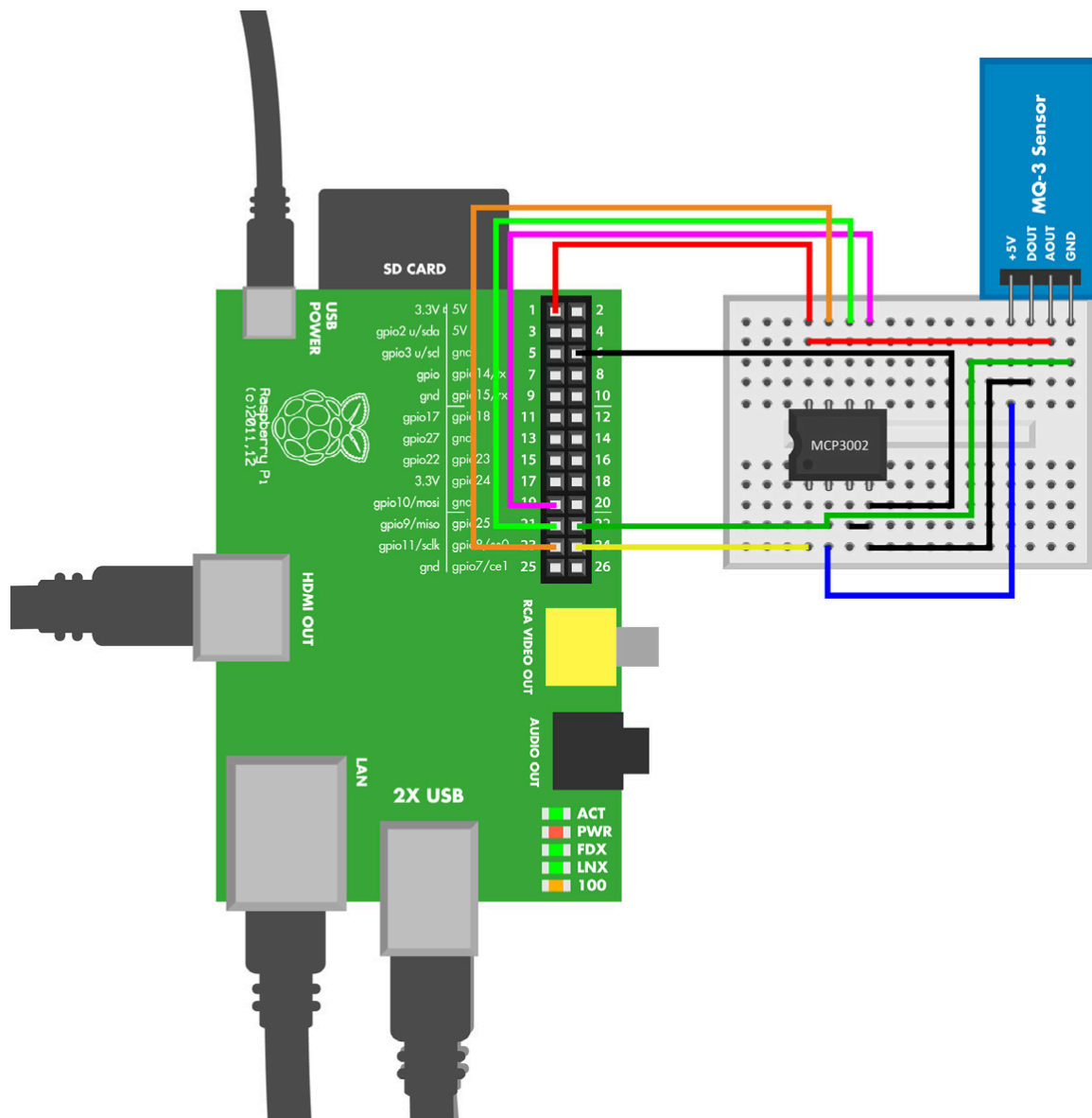
const int analogPin = A0;
const int digitalPin = 8;

int limit = -1;
int value = 0;

void setup() {
  Serial.begin(115200);
  pinMode(digitalPin, INPUT);
}

void loop()
{
  // Liest den analogen Wert
  value = analogRead(analogPin);
  // Prüft, ob der Alkoholgrenzwert überschritten ist
  limit = digitalRead(digitalPin);
  Serial.print("Alcohol value: ");
  Serial.print(value);
  Serial.print(" Limit: ");
  Serial.println(limit);
  delay(100);
}
```

**Listing 4-3** mq\_3\_alcohol\_sensor.ino



**Abb. 4-8** Schaltung für den MQ-3-Sensor am Raspberry Pi

```
# mq_3_alcohol_sensor.py -
  Liest den digitalen Ausgang des Alkoholsensors
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio

def readLimit():
    limitPin = 23
    gpio.setMode(limitPin,"in")
    return gpio.read(limitPin)

def main():
    while True:
        if readLimit() == gpio.HIGH:
```

```
print("Limit breached!")
time.sleep(0.5)

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

*Listing 4-4* mq\_3\_alcohol\_sensor.py

### 4.1.5 Praxisexperiment: Nüchtern bleiben beim Experimentieren

Bei elektronischen Experimenten Alkohol zu sich zu nehmen, ist eher kontraproduktiv. Zum Glück müssen Sie keine Flasche Whisky trinken, um den Sensor zu testen. Auch mit anderen Produkten, die Alkohol enthalten, können Sie schon messbare Werte erzielen. Mit Mundwasser und Schnapspralinen haben wir gute Erfahrungen gemacht.



**Abb. 4-9** Schnapspralinen und Mundwasser für Alkoholtest

Führen Sie den Alkotestcode aus und öffnen Sie den seriellen Monitor. Kauen Sie einige Schnapspralinen oder spülen Sie den Mund mit etwas Mundwasser (nicht schlucken!). Pusten Sie dann gegen den Sensor. Die Messwerte sollten sofort in die Höhe schnellen. Selbst mehrere Minuten später können Sie noch einen höheren Alkoholgehalt feststellen als üblich.

## 4.2 Testprojekt: Rauchalarm per E-Mail senden

Wenn Ihr Apparat Rauch erkennt, erhalten Sie eine E-Mail.



**Abb. 4-10** Der fertige E-Mail-Rauchmelder

### 4.2.1 Lernziele

In diesem Projekt lernen Sie Folgendes:

- Mit Aktionen auf Änderungen in der Umgebung reagieren: Bei Rauchentwicklung erhalten Sie eine Warnung per E-Mail.
- E-Mails automatisch vom Raspberry Pi senden lassen
- Grundlagen zum Verschicken von E-Mails

### 4.2.2 Python für E-Mails und Social Media

Dies ist ein Beispiel für ein Projekt, das sich mit dem Raspberry Pi leicht verwirklichen lässt. E-Mails können Sie vom Raspberry Pi aus auf die gleiche Weise automatisch senden lassen wie von jedem anderen Linux-System.

Python ist bekannt dafür, dass »Batterien enthalten sind«, das heißt, es gibt für alles eine Bibliothek. Mithilfe der vorhandenen Bibliotheken ist das Senden von E-Mails eine triviale Aufgabe. Außerdem existieren noch ähnliche Bibliotheken, um Daten mithilfe des Protokolls HTTP über das Web zu senden und zu empfangen.

Was ist mit Social Media? Sie können das Programm so anpassen, dass es Nachrichten an Twitter, Facebook und andere »soziale« Dienste sendet. Alle diese Dienste verwenden jedoch Protokolle, die zwar auf offenen Technologien

aufbauen, aber unternehmenseigen sind und Sie an den Dienst binden. Diese Riesenunternehmen legen selbst die Regeln fest, nach denen ihre Dienste funktionieren – und können diese Regeln von heute auf morgen von sich aus ändern. Beispielsweise könnte Twitter die Anzahl der zulässigen Anfragen pro Tag ändern oder irgendeine Funktion abschaffen, auf die Sie sich in Ihrem eigenen Programm verlassen. Bauen Sie Ihre Sandburgen nicht auf einem Privatstrand, der jemand anderem gehört! Zumindest sollten Sie sich gut informieren, bevor Sie diese Entscheidung treffen.

### 4.2.3 Das Projekt bauen

Verbinden Sie den Rauchsensor MQ-2 wie in [Abschnitt 4.1.2](#) gezeigt an den Raspberry Pi an. Am besten ist es, den Rauchsensor zunächst nur mit dem Sensorcode (aus [Listing 4-2](#)) zu testen.

Wenn Sie geprüft haben, dass die Schaltung funktioniert, können Sie dazu übergehen, über Ihren Code E-Mails zu senden. Testen Sie Ihre E-Mail-Anmeldeinformationen, bevor Sie sie in diesem Programm verwenden. Dazu können Sie sie in Ihrem üblichen E-Mail-Programm wie Thunderbird, KMail, Claws oder schlimmstenfalls Outlook einsehen. In dem Beispiel wird zwar ein Gmail-Konto verwendet, aber jedes beliebige Konto ist geeignet.

Verwenden Sie für solche Tests nicht Ihr persönliches E-Mail-Konto. Wenn Sie in dem Code einen Fehler machen und zu viele E-Mails senden, kann der E-Mail-Server das als einen Versuch werten, Spam zu senden. Am besten richten Sie ein E-Mail-Konto nur für Testzwecke ein. Dienste wie SendGrid (<http://sendgrid.com/>) und Amazon SES (<http://aws.amazon.com/ses/>) sind eigens auf solche Situationen zugeschnitten, weshalb Sie auch sie nutzen können, wenn Sie viele Nachrichten senden müssen.

### 4.2.4 Wie funktioniert E-Mail?

Wissen Sie noch, wie E-Mail funktioniert? Im Zeitalter von Gmail und anderen Webmail-Clients vergisst man die Grundlagen nur allzu leicht.

Im Folgenden beschreiben wir die Funktionsweise von E-Mail aus der Sicht eines E-Mail-Clients, der lokal auf Ihrem Computer läuft, also nicht über Webmail. Der Einfachheit halber zeigen wir nur die wichtigsten Punkte anhand eines Beispiels.

Sie müssen sich darüber im Klaren sein, dass zum Senden und Empfangen von E-Mails zwei verschiedene Server erforderlich sind (der des Absenders und der des Empfängers). Gewöhnlich befinden sich der Sende- und der Empfangsserver in verschiedenen Netzwerken, oft sogar auf verschiedenen Kontinenten.

Wenn Sie in Ihrem E-Mail-Client auf *Senden* klicken, nimmt Ihr Computer Kontakt mit Ihrem *SMTP-Server* auf. Dieser SMTP-Server kann sich in Ihrem Netzwerk befinden, von Ihrem Internetprovider betrieben werden (z. B. [smtp.verizon.net](http://smtp.verizon.net) oder [smtp.comcast.net](http://smtp.comcast.net)) und alle Verbindungen akzeptieren, die ihren Ursprung innerhalb des Netzwerks haben, ohne ein Passwort oder eine Anmeldung zu verlangen. Es kann aber auch sein, dass er von Ihrem E-Mail-Provider betrieben wird (z. B. [smtp.gmail.com](http://smtp.gmail.com) oder [mail.gmx.com](http://mail.gmx.com)) und eine TLS/SSL-Verschlüsselung und eine Anmeldung verlangt. Wird die E-Mail akzeptiert, stellt der SMTP-Server sie dem Empfänger zu.

Wenn der Empfänger seinen Posteingang auf neue Mails prüft, nimmt sein E-Mail-Client Verbindung mit einem *IMAP-Server* auf, der von seinem E-Mail-Provider (wie Gmail oder GMX) oder seinem Internetprovider betrieben wird. Um private E-Mails lesen zu können, sind natürlich eine Anmeldung und ein Passwort erforderlich, und der gesunde Menschenverstand sollte einem schon sagen, dass zur Verbindungsaufnahme eine Verschlüsselung (SSL/TLS) verwendet werden sollte (die meisten E-Mail-Provider verlangen eine Verschlüsselung). Der E-Mail-Client lädt anschließend die Nachrichtenheader und eine Kopie der gesamten Nachricht vom IMAP-Server herunter. Wenn jemand Ihnen eine E-Mail sendet, läuft der Vorgang umgekehrt ab: Der E-Mail-Client des Absenders nimmt Kontakt mit *dessen* SMTP-Server auf, der die Nachricht an Ihren Mailserver sendet, und wenn Sie Ihren Posteingang überprüfen, rufen Sie die Kopie der Mail von Ihrem IMAP-Server ab.

Nachrichten verbleiben gewöhnlich auf dem IMAP-Server. Wenn Sie Ihren Posteingang von Ihrem Computer und Ihrem Handy aus überprüfen, sehen Sie daher auf beiden Geräten die gleiche Liste.

#### **4.2.5 Kann der Arduino E-Mails senden? – Nicht ohne Weiteres!**

Um ein ähnliches Projekt mit dem Arduino zu bauen, müssten Sie einen externen Computer heranziehen. Der Arduino würde dann den Rauchsensor lesen und den Wert seriell über USB an den Computer weiterleiten. Dort könnte

dann ein Python-Programm die seriellen Daten lesen (mithilfe der API `pySerial`) und wie in unserem nächsten Beispiel eine E-Mail senden.

Selbst mit einem Ethernet- oder einem WLAN-Shield wäre es schwierig, E-Mails direkt vom Arduino zu senden. Oberflächlich gesehen, ist das E-Mail-Programm zwar ziemlich einfach, doch können beim Senden von E-Mails trotzdem viele Überraschungen auftreten, was es sehr schwierig macht, eine zuverlässige SMTP-Bibliothek für den Arduino zu schreiben. (Außerdem verlangen viele Server eine SSL-Verbindung, was der Arduino Uno und ähnliche Modelle nicht unterstützen.)

Es ist offensichtlich einfacher, dieses Projekt mit einem Raspberry Pi oder einer Mikrocontroller-Platine mit ähnlichen Eigenschaften zu bauen, z. B. mit dem BeagleBone (oder dem Arduino/Linux-Hybriden Arduino Yún).

## 4.2.6 Code für den Raspberry Pi

```
# smoke_alarm.py -
# Sendet alle 5 Minuten eine E-Mail, wenn Rauch erkannt wurde
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp
import smtplib
from email.mime.text import MIMEText

# E-Mail-Adressen
email_to = 'example@gmail.com'
email_from = 'example@gmail.com'

# Einstellungen für den SMTP-E-Mail-Server
server = 'smtp.gmail.com'
mail_port = 587
user = 'example@gmail.com'
password = 'password'
gracePeriod = 5 * 60 # Sekunden

def sendEmail(subject, msg):

    msg = MIMEText(msg)

    msg['Subject'] = subject
    msg['To'] = email_to,

    msg['From'] = email_from
```

```

smtp = smtplib.SMTP(server,mail_port) 13

smtp.starttls()

smtp.login(user, password) 14

smtp.sendmail(email_from, email_to, msg.as_string()) & 15

smtp.quit() 16

def main():

    while True:

        smokeLevel = mcp.readAnalog()

        print("Current smoke level is %i " % smokeLevel)

        if smokeLevel > 120:

            print("Smoke detected")

            sendEmail("Smoke","Smoke level was %i" % smokeLevel) 17

            time.sleep(gracePeriod) 18

        time.sleep(0.5) # s 19

if __name__ == "__main__":
    main()

```

#### **Listing 4-5** *smoke\_alarm.py*

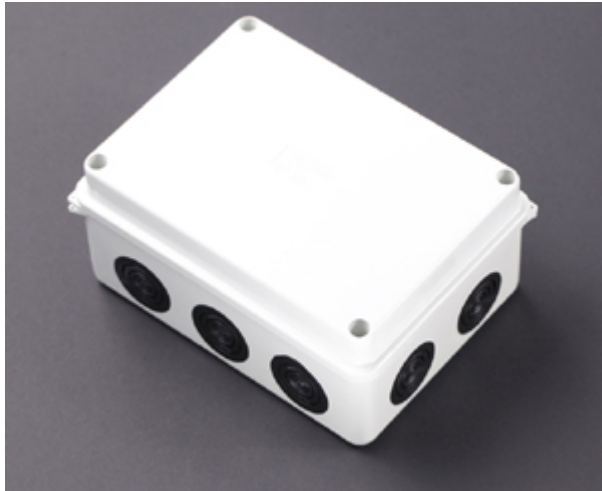
- 1 »Batterien enthalten«: Python verfügt über eine eingebaute Bibliothek für die SMTP-Kommunikation. Sie müssen sich nicht selbst um die Einzelheiten der Socket-Programmierung kümmern.
- 2 Früher wurden E-Mails als einfacher Text gesendet: ein Header pro Zeile, eine leere Zeile und dann der Textrumpf. Heutzutage wird jedoch erwartet, dass auch Nicht-ASCII-Zeichen aus anderen Sprachen funktionieren. Daher wird der E-Mail-Text oft ebenso MIME-codiert wie Anhänge. Das geht schon mit den Umlauten los – nicht nur im Deutschen, sondern auch in Finnisch: Päivää (*Tag*)!
- 3 Die **To**-Adresse (*An*) ist der Empfänger der E-Mail. Um die Nachrichten des Rauchmelders zu empfangen, sollten Sie hier Ihre eigene Adresse angeben. Diese Variablen sind global und somit für alle Funktionen sichtbar.

- 4 Die `From`-Adresse der E-Mail (*Von*) kann theoretisch beliebig sein. Im Zeitalter von Spamfiltern kann eine absurde E-Mail-Adresse jedoch die Chancen dafür beeinträchtigen, dass die Nachricht durchkommt. Verwenden Sie hier am besten Ihre eigene E-Mail-Adresse, da der Domänenname dann gewöhnlich mit demjenigen des verwendeten SMTP-Servers übereinstimmt.
- 5 Der SMTP-Server ist der sendende Server. Da dieses Programm E-Mails nur sendet und nicht empfängt, brauchen Sie keinen weiteren Server.
- 6 Ihr SMTP-Anmeldename. Hierbei kann es sich um Ihre E-Mail-Adresse (*example@gmail.com*) oder um einen kürzeren Anmeldnamen (*jdoe*) handeln.
- 7 Das SMTP-Passwort sollte das gleiche sein, das Sie zur Webmail-Anmeldung oder zur Verbindungsaufnahme von einem E-Mail-Client aus verwenden.
- 8 Wenn Rauch erkannt wird, wollen Sie sicherlich nicht 60 E-Mails pro Sekunde erhalten. E-Mails in so rascher Folge zu senden kann dazu führen, dass der Server den Absender auf die schwarze Liste setzt. Aus diesem Grund sollten Sie eine Toleranzfrist einräumen. Der Zeitraum von 5 Minuten wird hier im Code durch Multiplikation mit 60 in Sekunden umgerechnet. (Schreiben Sie so einfache Berechnungen in den Code, anstatt solche »magischen Zahlen« auf Papier auszurechnen. Widerstehen Sie auch der Versuchung, den Wert in Kommentaren zu wiederholen.)
- 9 Der Klarheit halber steht der Code zum Senden von E-Mails in einer eigenen Funktion. Betreffzeile (`subject`) und Text (`msg`) stammen aus Funktionsparametern.
- 10 Der Text der Nachricht ist MIME-codiert, um auch Nicht-ASCII-Zeichen zuverlässig handhaben zu können. Wie Sie an dem großen Anfangsbuchstaben erkennen können, handelt es sich bei `MIMEText` um eine Klasse. Der Konstruktor `MIMEText()` gibt ein neues Objekt zurück, das in der Variablen `msg` gespeichert wird. `MIMEText` dient nur dazu, die Nachricht zu erstellen, hat aber nichts damit zu tun, irgendwelche Verbindungen aufzubauen oder etwas zu senden.
- 11 Um die E-Mail-Header zu bearbeiten, können Sie das `MIMEText`-Objekt `msg` wie einen *Dictionary-Datentyp* verwenden.
- 12 Empfänger und Sender werden aus den globalen Variablen bezogen.

- 13 Erstellt ein neues Objekt der Klasse `SMTP`, das für die Verbindungsaufnahme mit dem SMTP-Server verwendet wird. Der erste Teil (`smtplib`) kennzeichnet den *Namensraum*, der automatisch durch `import smtplib` erstellt wird.
- 14 Stellt die Verbindung zum SMTP-Server mit dem Benutzernamen und dem Passwort aus den globalen Variablen her.
- 15 Sendet die E-Mail mit der Methode `sendmail()` des Objekts `smtp` der Klasse `smtplib.SMTP`. Die Methode `smtp.sendmail()` erwartet einen String, weshalb das Objekt `msg` mit der eingebauten Methode in einem String abgelegt wird.
- 16 Schließt die Verbindung und räumt auf. Die Funktion `quit()` ist eine Methode des Objekts `smtp`.
- 17 Verwendet einen Formatierungsstring, um den Nachrichtentext `msg` aufzubauen. Dabei wird der Platzhalter `%i` durch den Wert von `smokeLevel` ersetzt.
- 18 Wartet nach dem Senden der E-Mail fünf Minuten lang, bevor die nächste Mail gesendet wird.
- 19 Wartet eine halbe Sekunde, um zu vermeiden, dass die Schleife mit Höchstgeschwindigkeit durchlaufen wird.

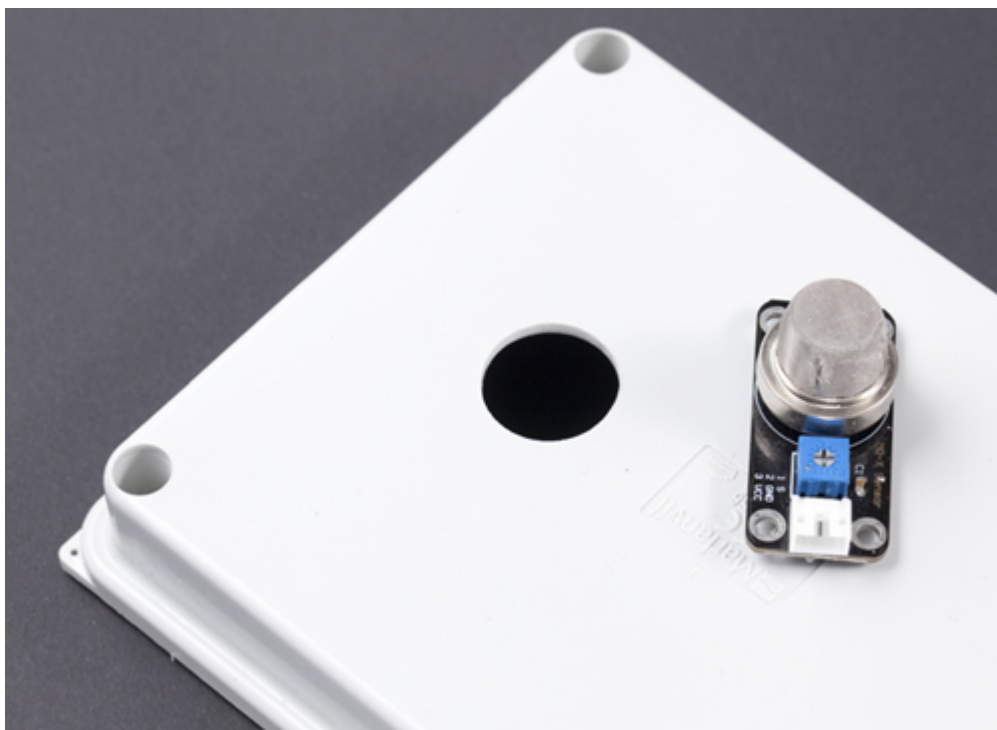
## 4.2.7 Gehäuse

Um unseren E-Mail-Rauchmelder zu verkleiden, haben wir eine Verteilerdose verwendet (siehe Abb. 4–11). Solche Kästen sehen zwar ziemlich schmucklos aus, sind aber überall erhältlich, und durch die mit Gummi abgedeckten Öffnungen lassen sich Anschlüsse schnell und einfach herstellen.



**Abb. 4-11** Verteilerdose

Die einzige Öffnung, die Sie noch hineinbohren müssen, ist die für den Rauchsensor (siehe Abb. 4-12). Für die anderen müssen Sie lediglich die Gummiabdeckungen der vorhandenen Löcher mit einem scharfen Messer ausschneiden. Im Grunde genommen könnten Sie auch eine dieser Öffnungen für den Sensor verwenden, aber da sie gewöhnlich ziemlich grob wirken, benutzen wir sie nur für die Kabel. Ein 19-mm-Bohrer erwies sich als genau passend für den Sensor. Wir brauchten nicht einmal Klebeband, um ihn an Ort und Stelle festzumachen (siehe Abb. 4-13).

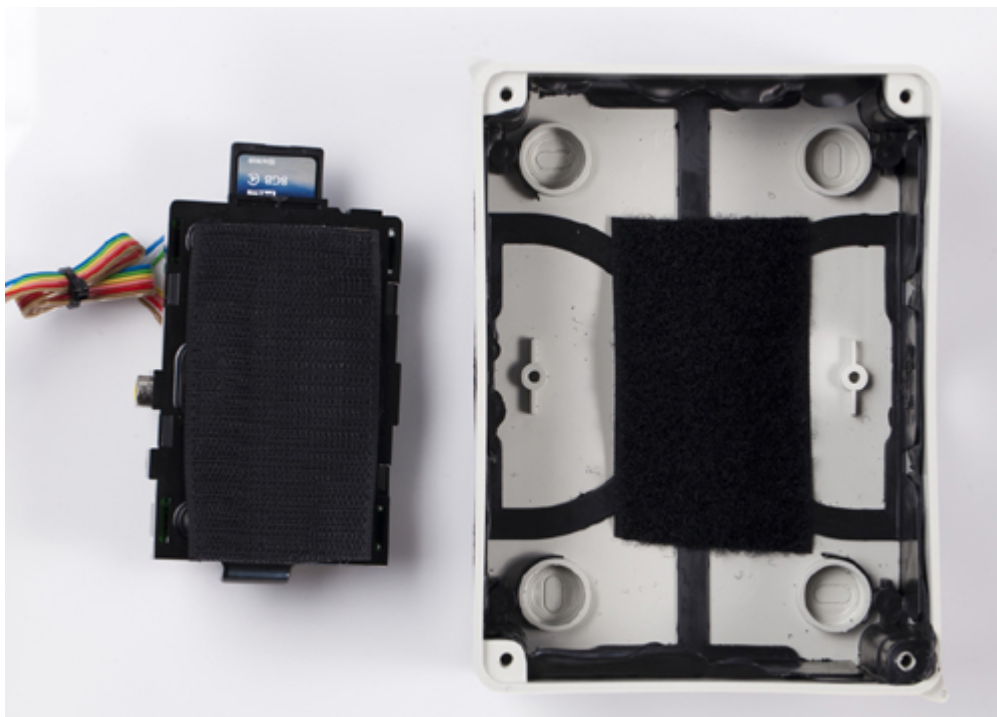


**Abb. 4-12** Das Loch für den Rauchsensor



**Abb. 4-13** Der montierte Sensor

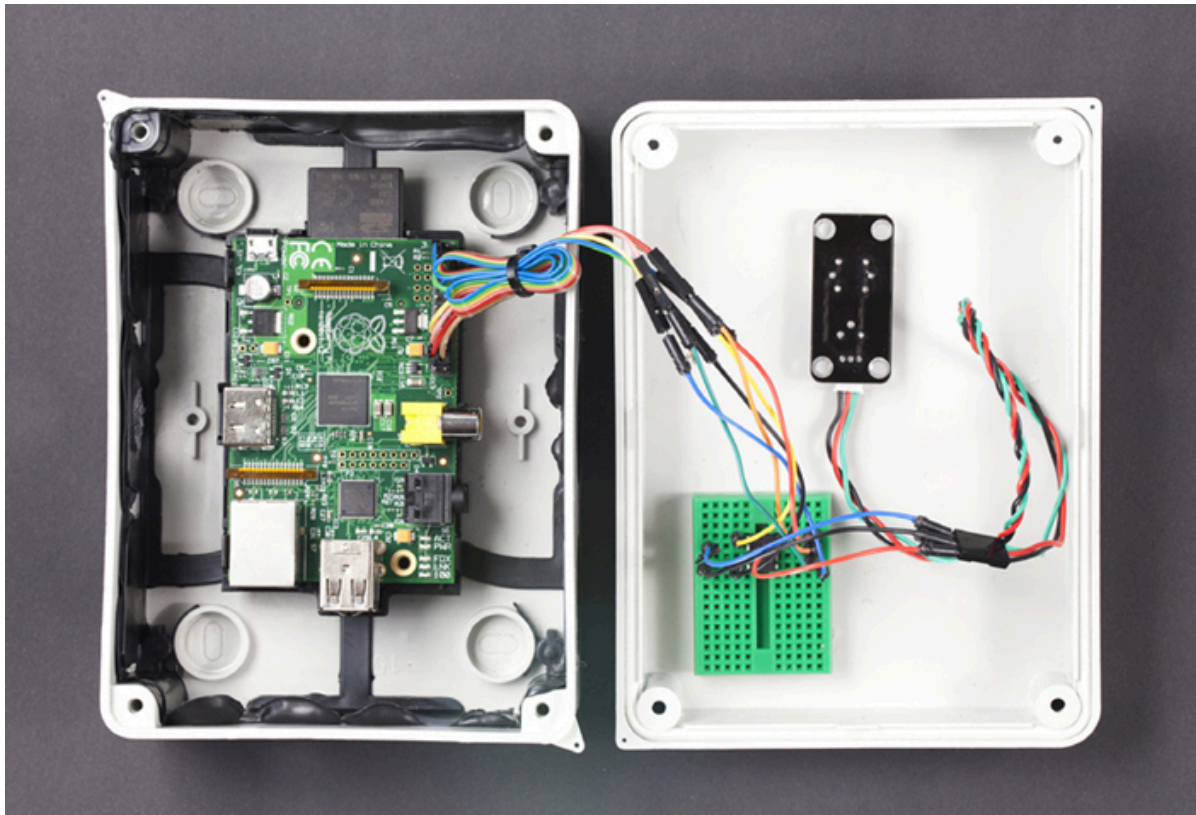
Damit der Raspberry Pi innerhalb der Dose nicht hin und her rutscht, haben wir ihn mit Klettband am Boden befestigt, wie Sie in [Abbildung 4-14](#) sehen. Dadurch können Sie ihn zur Fehlersuche oder zur Verwendung in anderen Projekten wieder leicht herausnehmen.



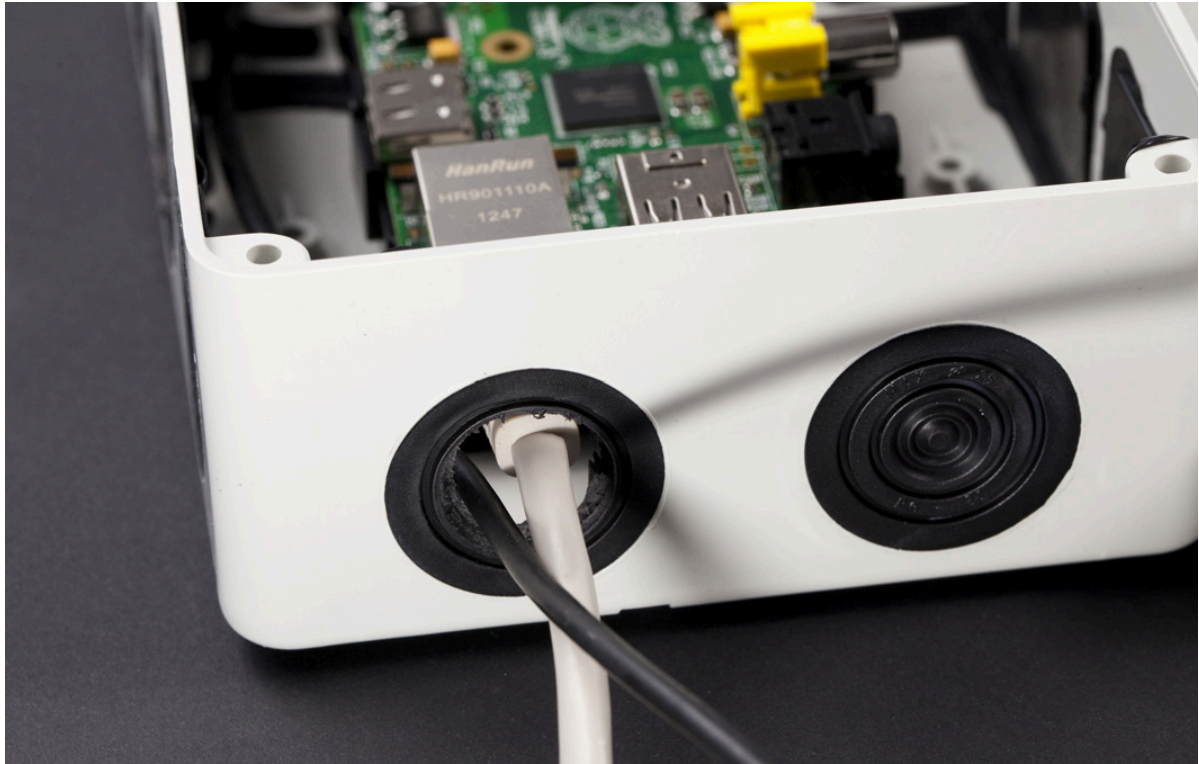
**Abb. 4-14** Der Raspberry Pi wird mit Klettband festgehalten.

In den Deckel der Dose haben wir ein Mini-Steckbrett geklebt (siehe [Abb. 4-15](#)) und die Jumperkabel mit Kabelbinder sauber gebündelt.

Für das LAN- und das Stromkabel haben wir an der Seite ein großes Loch (ca. 2 cm) geöffnet (Abb. 4-16). Da der fertige Apparat nur möglichst kleine Öffnungen aufweisen sollte, hat unsere Version keinen Zugang für andere Kabel (siehe Abb. 4-17).



**Abb. 4-15** Mini-Steckbrett mit durch Kabelbinder fixierte Kabel



**Abb. 4-16** Das Loch für die Kabel



**Abb. 4-17** Die angeschlossenen Kabel

Jetzt müssen Sie nur noch den Deckel schließen, und schon ist das Gerät wie in **Abbildung 4–10** gezeigt bereit für die ersten Tests! Denken Sie bei der Montage des Rauchmelders daran, in welche Richtung sich Gase bewegen.

Ihr Apparat hat nun eine »elektronische Nase«. Mithilfe der Sensoren aus der MQ-Reihe können Sie Gase aufspüren, die giftig (CO), brennbar (Kohlenwasserstoffe) oder explosiv sind (Wasserstoff) oder die auf eine Vergiftung hindeuten (Ethanol im Atem).

In diesem Projekt haben Sie ein Python-Programm geschrieben, das auf Änderungen in der Umgebung reagiert. Dieses Programm können Sie so anpassen, dass es auf Ereignisse von anderen Sensoren reagiert oder eine andere Reaktion zeigt, also anstatt eine E-Mail zu senden etwa Kontakt mit einem Webdienst aufnimmt oder einen Ausgang aktiviert.

Gassensoren können Dinge wahrnehmen, die mit menschlichen Sinnen nur schwer oder gar nicht zu erkennen sind. Im nächsten Kapitel sehen wir uns dagegen ein spürbares Phänomen an, nämlich Berührungen.

# Index

## A

Aktiver Infrarot-Entfernungssensor 47

Alarm

    Analoger 2-Achs-Daumen-Joystick 149

    Haltungswarner 77

Annäherung erkennen 48

Arbeitsverzeichnis

    ausgeben 20

    Dateien auflisten 20

Arduino

    Alarmanlage 156

    Blinktest 42

    Bodenfeuchtigkeitssensor 369

    Bootloader 40

    DHT11-Feuchtigkeitssensor 351

    Drehgeber 146

    Farbsensoren 191

    Flammensensor 176

    FlexiForce-Sensor 123

    Fotowiderstand 181

    Grundeinstellungen 40

    GY65 barometrischer Drucksensor 358

    Hall-Schalter 319

    Haltungswarner-Projekt 74

    HC-SR04-Sensor 54

IR-Abstandssensor 61  
IR-Facettenauge 67  
Joysticks 150  
Kapazitätssensor 127  
Kommunikation mit Raspberry Pi 355  
Liniensensor 186  
LM35-Temperatursensor 346  
Mikrofon 334  
Neigungssensoren 140  
OS X-Einrichtung 40  
Ping-Sensor 49  
Programmanatomie 43  
QT113-Sensor 120  
Servomotoren 134  
Shields 44  
Vibrationssensoren 142  
Vorzüge 39

Aufgaben zeitgesteuert ausführen 329

## **B**

Barometrische Drucksensoren 357

BCM-Zahlen 26

Befehlszeilen-Interface (CLI)

Aufruf im Raspberry Pi 19

GPIO-Pins steuern 27

Linux 18

Skriptautomatisierung 19

sudo-Befehl 20

Benutzerrechte 20

## Berührung

Druckmessung 123

Klingel-Projekt 130

Messung durch massive Objekte hindurch 122

## Beschleunigung 213

Beschleunigungsmesser und Gyroskop im Vergleich 214

Beschleunigungsmesser, Verwendung 213

MX215-Sensor 214

## Bewegung

Daumenjoystick 150

digitale Vibrationssensoren 142

Drehgeber 146

Erkennungsmethoden 139

Neigungsschalter 139

passiver Infrarot-Sensor 155

Pong-Spielprojekt 162

Verfolgen mit dem IR-Facettenauge 65

## Bilder 380

BMP-Grafiken 380

Bootloader 40

## **C**

Cat-Befehl 27

Chamäleonkuppel-Projekt 197

C-Header-Dateien 380

Cron 329

## **D**

Daemon

Installation 21

## Dateien

- auflisten 19

- erstellen 20

- Inhalt anzeigen 27

- Linux-Konfigurationsdateien 20

- ohne Editor bearbeiten 27

- speichern 20

- Sticky-Dateien 31

Datenvisualisierung 21, 337

Daumen-Joystick 149, 150

DHT11-Feuchtigkeitssensor 349

Digitale Vibrationssensoren 142

Drehgeber 146

Druckmessung 123

## **E**

Echo-Berechnungen 57

Einbruchsalarm

- Adaptionsdauer 156

- Arduino Code/Schaltung 156

- Neigungssensoren 139

- Raspberry Pi Code/Schaltung 158

- testen 159

Electret-Mikrofon 333

Elektrizität/Magnetismus Erkennungsmethoden

- Hall-Schalter 318

- Webüberwachung von Solarzellen 321

Empfindlichkeitseinstellung 179

Entfernung

- Bewegungen folgen 65
- Echo-Berechnungen 57
- Erkennungsmethoden 47
- HC-SR04-Ultraschallsensor 54
- Hinderniserkennung 60
- Infrarotlicht erkennen 64
- Testprojekt 74
- Ultraschalltöne mit Ping messen 48
- weiche Objekte 59

Entfernungsmesser 48

E-Paper-Wetterstation

- Arduino 372

- Bildanzeige/-speicherung 380

E-Paper-Wettervorhersage Projekt

- Gehäuse 383

- Strombedarf 380

- Übersicht 371

## **F**

Farbsensoren

- Arduino Code/Schaltung 191

- Raspberry Pi Code/Schaltung 194

- Übersicht 191

Fehlersuche

- elektronische Verbindungen 29

- GPIO 35

- GPIO in Phyton 36

- GPIO-Überspannung 24

- Raspberry Pi-Installation 16

sudo-Befehl 20

Feuchtigkeitssensoren 349

FFT (schnelle Fourier-Transformation) 341

Fingerabdruckscanner 261

Flammensensoren 176

FlexiForce

    Protokolle 123

    Raspberry Pi Code/Schaltung 124

Flipperautomaten 139

Fotowiderstände 180

## **G**

Geisterglocke

    Arduino Code/Schaltung 134

    Servo an Glocke anschließen 137

    Servo-Motoren 130

    Übersicht 129

GPIO-Pins (General Purpose Input / Output)

    Belegungsdiagramm 26

    Fehlersuche 35

    Gefahr durch Überspannung 24

    in Python 34

    Nummerierungssystem 25

    ohne Root-Rechte 31

    Vielseitigkeit 22

    von der CLI steuern 28

Grafik-Equalizer 337

Graphen 327

GY65 barometrischer Drucksensor 358

## **H**

Hall-Schalter 318

Haltungswarner

Alarm 77

Gehäuse 80

Piezo- und IR-Sensor kombinieren 78

Piezosummer 75

Übersicht 74

HC-SR04-Ultraschallsensor

Arduino Code/Schaltung 54

Echo-Berechnungen 57

Raspberry Pi Code/Schaltung 56

Vorteile 54

weiche Objekte 59

HDMI

Töne sichtbar machen über 337

Header-Dateien 380

Hello World (Blinktest) 42

Hinderniserkennung 60

Homepage 326

## **I**

Infrarot-Entfernungsmesser (siehe IR-Entfernungsmesser) 47

Infrarotlicht 48

Infrarotsensoren 64

Interrupt 142

IP-Adresse 326

ipython 21

IR-Entfernungsmesser

- Arduino Code/Schaltung 61
- Raspberry Pi Code/Schaltung 63
- Vorteile 60

## IR-Facettenauge

- Arduino Code/Schaltung 67
- Bewegungen verfolgen 65
- Kalibrierung 67
- Raspberry Pi Code/Schaltung 69
- Raspberry Pi, alternative ICs 73
- SpiDev-Installation 72

## **J**

- Joysticks 150

## **K**

### Kapazitive Berührungssensoren

- Arten 120
- eigene bauen 126
- Erkennung durch massive Objekte hindurch 122
- QT113 120

- Kernel-GPIO-Treiber 27

### Kompassrichtung

- Berechnung 316

- Konfigurationsdateien 20

- KY-026 Flammensensor 175

## **L**

- Lautstärke (siehe auch Ton) 333

### LEDs

- Ein- und ausschalten 28

### Licht

- Erkennungsmethoden 175
- Farbsensoren 191
- Flammensensoren 175
- Fotowiderstände 180
- Liniendetektoren 185
- Richtungserkennung 184
- Lichtabhängiger Widerstand (LDR) 180
- Liniendetektoren 185
- Linux
  - Arduino-Einrichtung 40
  - Befehlszeilen-Interface (CLI) 19
  - Dateibearbeitung 20
  - Installation auf Raspberry Pi 12
  - Konfigurationsdateien 20
  - Nützliche Befehle 387
  - sudo-Befehl 20
  - systemweite Konfiguration 20
  - wichtige Verzeichnisse 388
- LM35-Temperatursensor 345
- LSM303-Kompass-Beschleunigungsmesser
  - Kompasskurs-Berechnung 316
  - Protokoll 315
- Luftdruck 357
- LXTerminal 27
- M**
- Master/Slave-Geräte 316
- Mathematische Graphen 327
- matplotlib-Bibliothek 327

Mikrofone 333

MX2125-Beschleunigungssensor 214

## **N**

Nano-Texteditor 27

Neigungsschalter 139

NOOBS\*.zip 17

## **P**

Parallax PIR 156

passive Infrarotsensoren 47

Penetrationstester 159

Physische Pin-Nummern 26

Piezo-Summer 75

Ping-Sensor

- Arduino Code/Schaltung 49

- Entfernungen messen 48

- Raspberry Pi Code/Schaltung 51

Polling 142

Pong (Spiel)

- Automatische Anmeldung 172

- Auto-Start 170

- beim Anmelden ausführen 171

- Gehäuse 167

- Raspberry Pi Code/Schaltung 164

- Übersicht 162

Potenziometer

- in Joysticks 150

Programme

- Eingabeaufforderungen erkennen 21

installieren 21

## Protokolle

für FlexiForce-Sensor 123

für LSM303-Kompass-Beschleunigungsmesser 315

SPI 72

pwd-Befehl 18

pyGame-Bibliothek 162

## Python

GPIO 34

ipython-Werkzeug 21

matplotlib-Bibliothek 327

## Q

QT113 kapazitiver Berührungssensor

Alternativen 126

Arduino Code/Schaltung 120

Raspberry Pi Code/Schaltung 121

## R

Raspberry Pi

als Webserver verwenden 325

Anschlusskabel 13

Bodenfeuchtigkeitssensor 370

Booten 14

DHT11-Feuchtigkeitssensor 353

Drehgeber 148

Einbruchsalarm 158

Elektronik anschließen 22

Farbsensoren 194

Fehlersuche nach der Installation 16

Flammensensoren 177  
FlexiForce-Sensor 124  
Fotowiderstand 182  
GPIO-Pin-Nummern 25  
Grundlegende Einrichtung 13  
Hall-Schalter 320  
HC-SR04-Sensor 56  
Herunterfahren 22  
Homepage erstellen 326  
IR-Entfernungssensor 63  
IR-Facettenauge 69  
Joysticks 152  
Kapazitätssensor 128  
Kommunikation mit dem Arduino 355  
Liniendetektoren 187  
Linux-Betriebssystem verwenden 19  
Linux-Installation 13  
LM35-Temperatursensor 347  
Mikrofon 335  
Modellempfehlungen 11  
Neigungssensor 141  
Ping-Sensor 51  
Pong-Projekt 162  
QT113-Sensor 121  
SD-Karte formatieren 16  
seriellen Port aktivieren 338  
Solarzellenmonitor 327  
Tonvisualisierung 338

Vibrationssensor 144

Raspbian

Installation 14

Root-Benutzer 20

## **S**

Schall

Analyse und Messung 333

Lautstärke erkennen 333

Sensorempfindlichkeit 336

Töne über HDMI visualisieren 337

ScrewShield 44

SD-Karte formatieren 16

Sensoren

aktive Infrarotsensoren 47

analoger 2-Achs-Daumen-Joystick 150

barometrischer Drucksensor 357

Bodenfeuchtigkeit 368

Farbsensoren 191

Feuchtigkeit 349

FlexiForce 123

IR-Abstand 60

KY-026 Flammensensor 175

Linienverfolgung 185

MX2125 214

Neigungssensoren 139

Parallax PIR 156

passive Infrarotsensoren 47

Ping 48

QT113 120  
Schall 336  
Temperatur 345  
Ultraschall 54  
Vibrationssensoren 142  
Wiederverwerten von Spielkonsolen 154

## Servomotoren

grundlegender Betrieb 130  
Servobereich herausfinden 131  
Servo.h-Bibliothek 135  
Vorteile 130

Shell 19

Shell-Befehle 21

Shields 44

Skripts 19

Solarzellen-Webmonitor-Projekt 321

SpiDev-Bibliothek 73

SPI-Protokoll 72

Sticky-Dateien 31

Stimmerkennung 333

Strg-X 20

sudo-Befehl 20, 28

Super User 20

## **T**

Tabulator-Taste 22

Temperatursensoren 345

Text ausgeben auf dem Terminal 27

Trennen von Benutzerrechten 20

TTL-Level 57

## **U**

Ubuntu 40

Udev 31

Ultraschall (PING) 48

Ultraschall-Entfernungsmesser 54

Umleitungsoperator (>) 27

## **V**

Vibrationsmotoren 155

Vibrationssensoren 142

Videospiel-Konsolen 149

## **W**

Wetter/Klima

atmosphärische Drucksensoren 357

Bodenfeuchtigkeitssensoren 368

E-Paper-Wettervorhersage 371

Feuchtigkeitssensoren 349

Temperatursensoren 345

Widerstände

Empfindlichkeit 179

Fotowiderstände 180

Kennung 29

Windows 40

Windows-Taste 40

## **Z**

Zeitgesteuerte Aufgaben 329

Zweckorientierte Pin-Nummerrierung 26

## **Sonderzeichen**

>-Operator (Umleitung) 27 \$-Eingabeaufforderung 18