
Kimmo Karvinen · Tero Karvinen · Ville Valtokari
Sensoren
Messen und
experimentieren
mit Arduino
und Raspberry Pi
c't Hardware Hacks
edition
dpunkt.verlag

Inhaltsverzeichnis

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 (CO_2) liegen. Dieses Gas wird von allen Tieren (und Menschen) ausgeatmet. Sensoren im Lüftungssystem eines Gebäudes können einen erhöhten 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_4)
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).

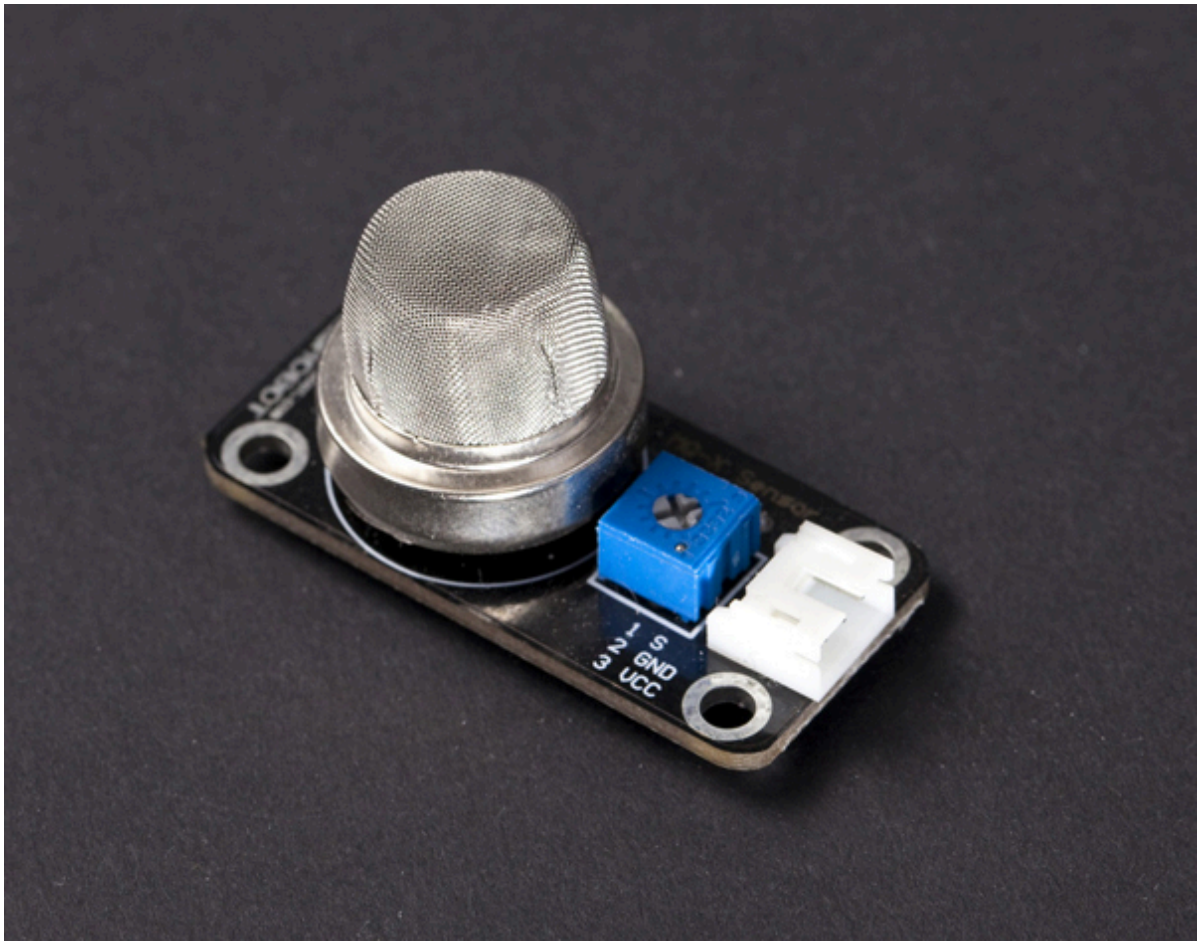

Abb. 4–1 *Analoger Gassensor*

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_2). Der menschliche Körper erzeugt bei der Energiegewinnung CO_2, 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_2 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

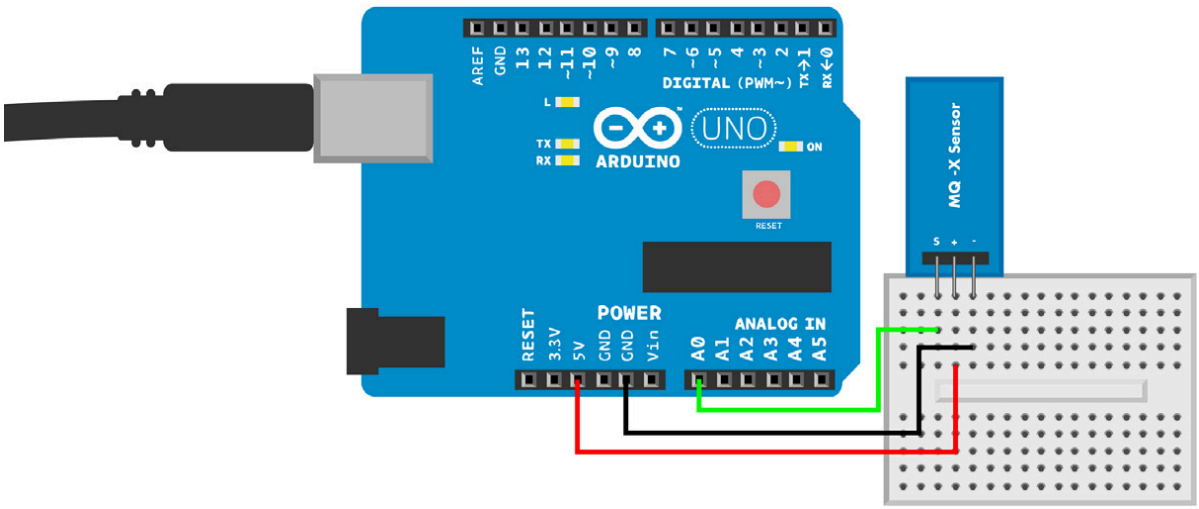

Abb. 4–2 *Schaltung für den MQ-2-Sensor am Arduino*

```
// 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 Alltagsgegen-stand 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).

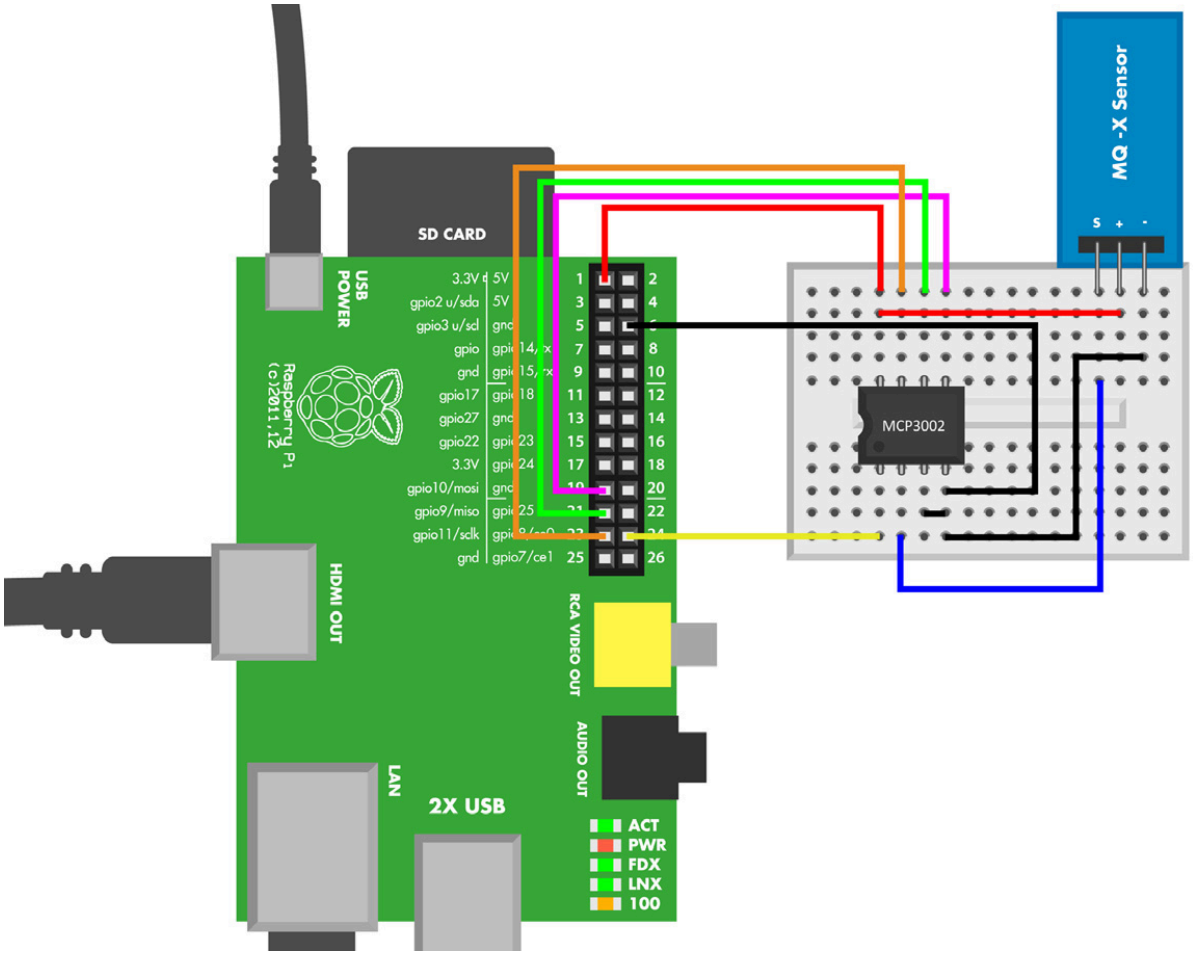

Abb. 4–4 *Schaltung für den MQ-2-Sensor am Raspberry Pi*

```
# 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 abzubrechen, 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.

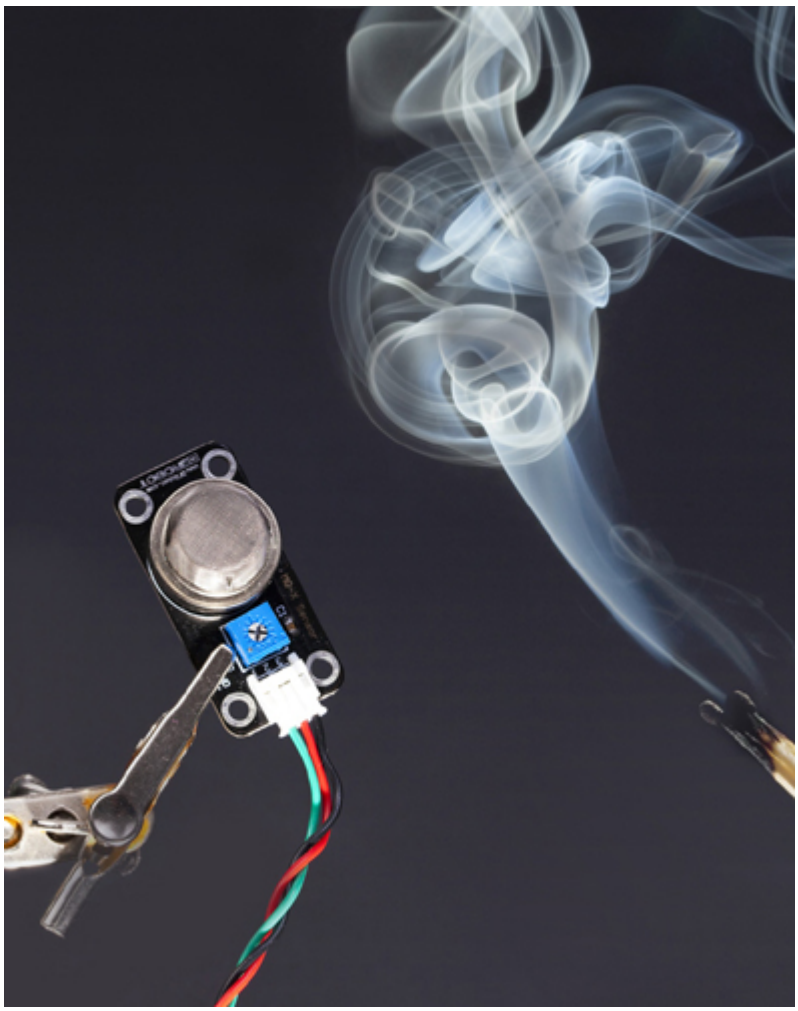

Abb. 4–5 *Rauch steigt nach oben.*

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).

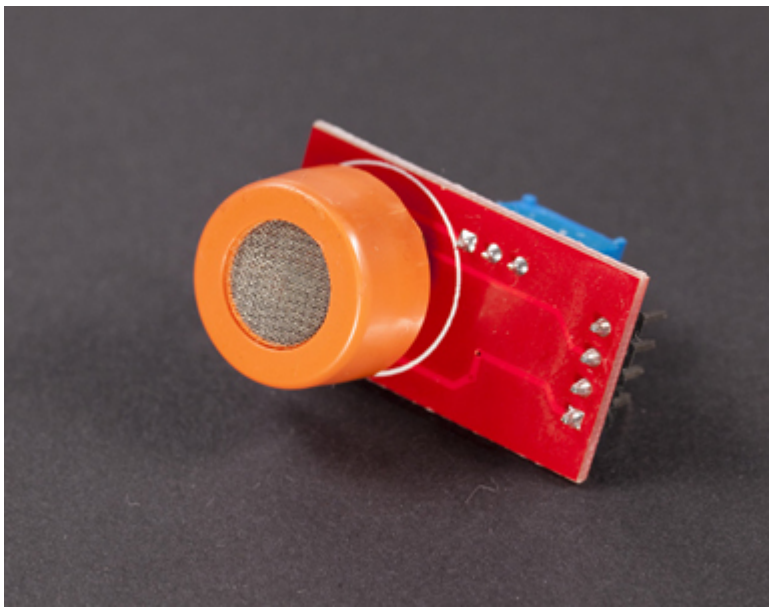

Abb. 4–6 *Der Alkoholsensor MQ-3*

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 betrunkener, 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.

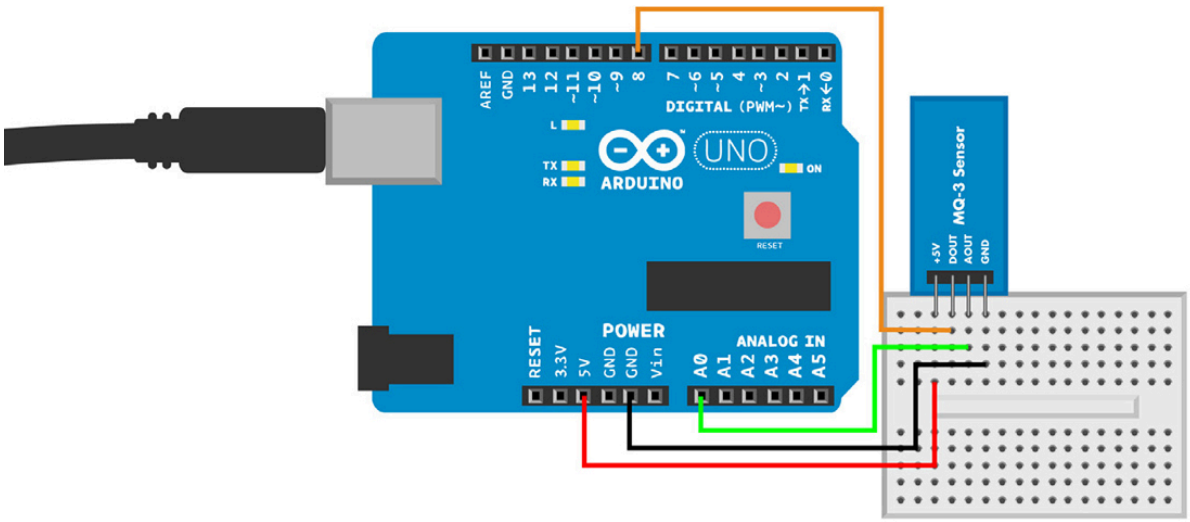

Abb. 4–7 *Schaltung für den MQ-3-Sensor am Arduino*

```
// 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*

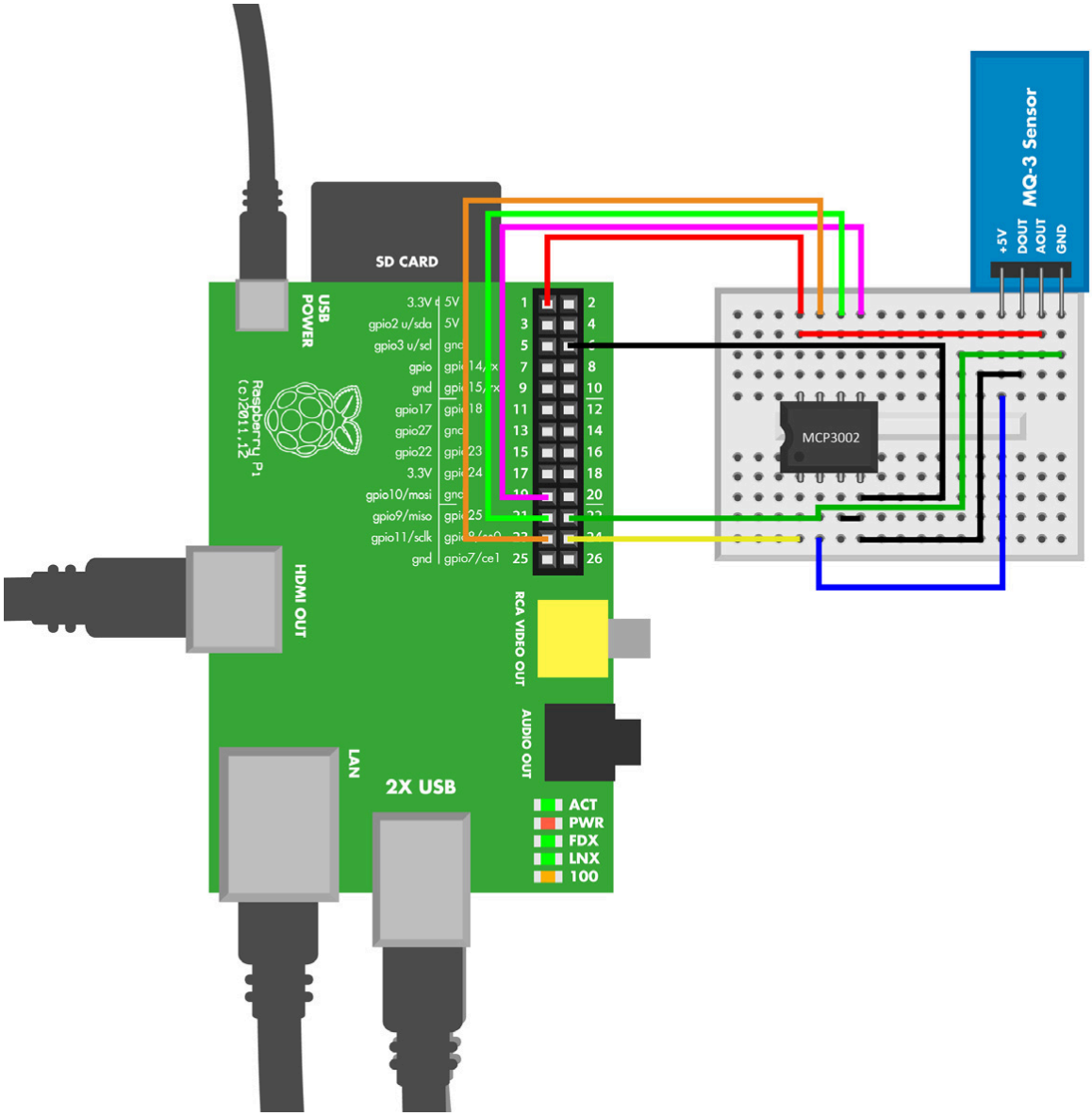

Abb. 4–8 *Schaltung für den MQ-3-Sensor am Raspberry Pi*

```
# 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` oder `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` oder `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 Postein-gang 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                                          1
from email.mime.text import MIMEText       2

# E-Mail-Adressen
email_to = 'example@gmail.com'             3
email_from = 'example@gmail.com'           4

# Einstellungen für den SMTP-E-Mail-Server
server = 'smtp.gmail.com'                          5
mail_port = 587
user = 'example@gmail.com'                       6
password = 'password'                                       7
gracePeriod = 5 * 60 # Sekunden               8

def sendEmail(subject, msg):       9

      msg = MIMEText(msg)
            10

      msg['Subject'] = subject               11
      msg['To'] = email_to,        12

      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 Anmeldenamen (*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).

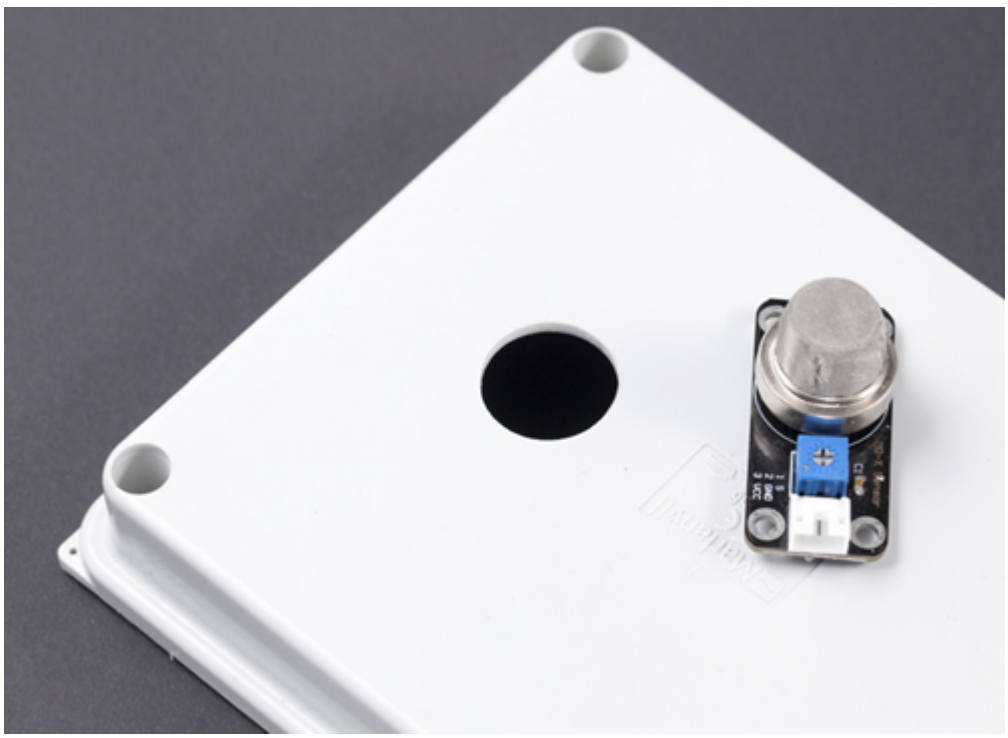

Abb. 4–12 *Das Loch für den Rauchsensor*

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.

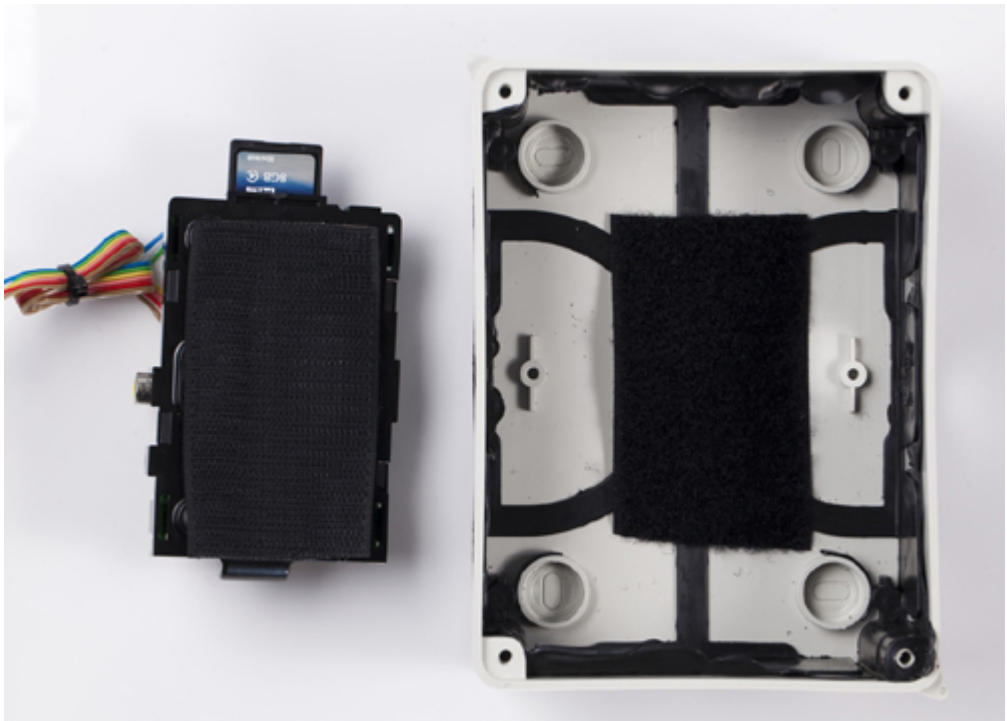

Abb. 4–14 *Der Raspberry Pi wird mit Klettband festgehalten.*

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).

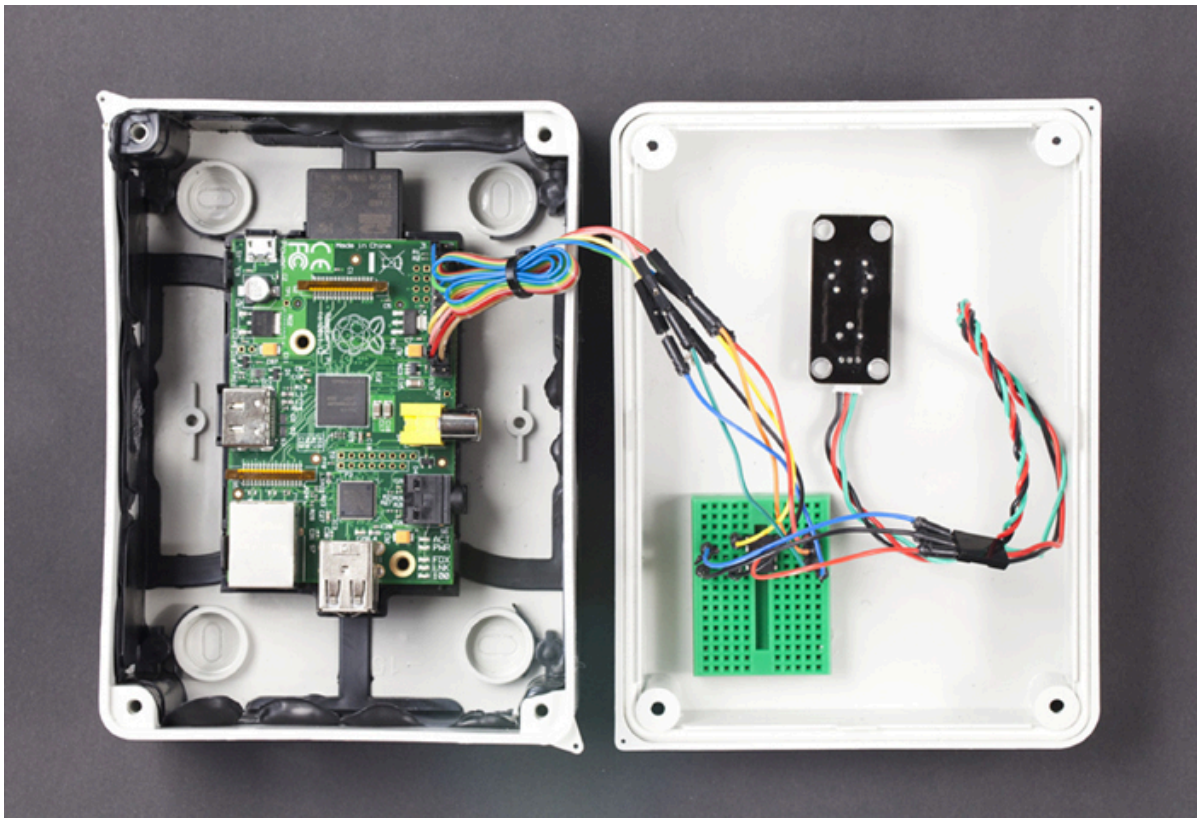

Abb. 4–15 *Mini-Steckbrett mit durch Kabelbinder fixierte Kabel*

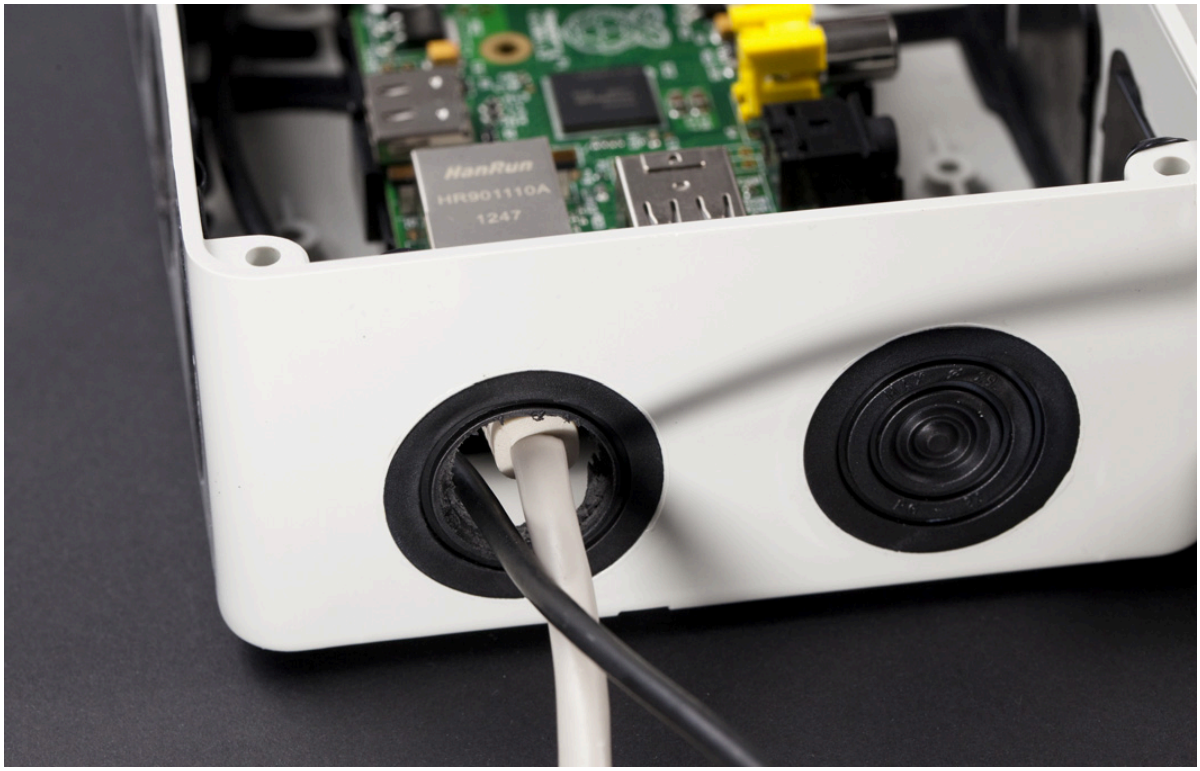

Abb. 4–16 *Das Loch für die Kabel*

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

B

C

D

E

F

G

M

N

P

Q

R

S

T