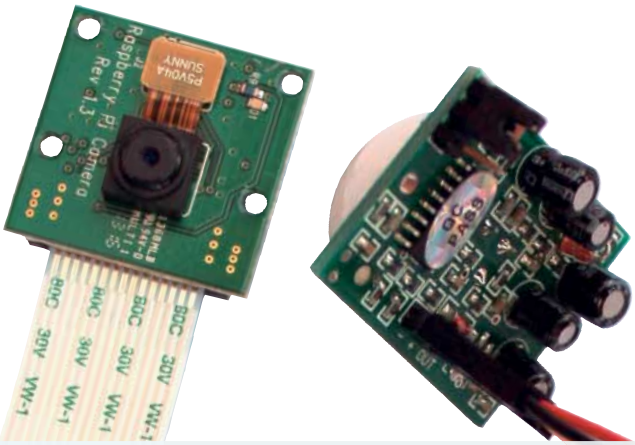
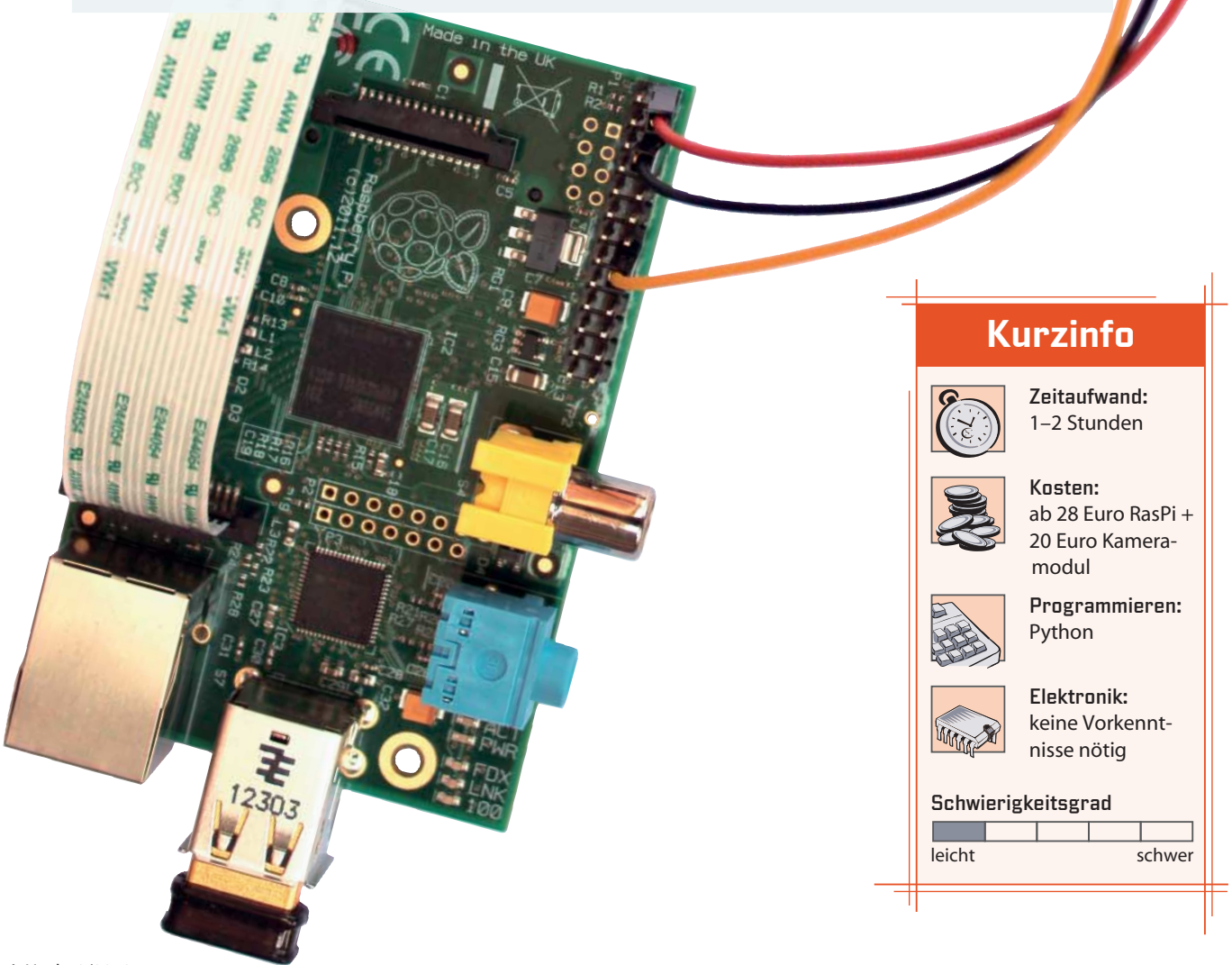


Maik Schmidt

Raspberry-Pi-Kameramodul

Mit seinem Kameramodul liefert der Raspberry Pi Videos im HD-Format und Bilder mit einer Auflösung von 5 Megapixel. Durch ein eigenes Interface auf dem Kleincomputer und der guten Software-Unterstützung lässt er sich prima in eigene Projekte integrieren.



Kurzinfo

Zeitaufwand:
1–2 Stunden

Kosten:
ab 28 Euro RasPi + 20 Euro Kameramodul

Programmieren:
Python

Elektronik:
keine Vorkenntnisse nötig

Schwierigkeitsgrad

leicht — schwer

© Copyright by Heise Zeitschriften Verlag
Persönliches PDF für alle Leser des Make Magazins aus 30625 Hannover

Beinahe zeitgleich mit der Veröffentlichung des Raspberry Pi wurde auch eine eigene Kamera für den Mini-Computer angekündigt. Das ist schon einige Zeit her, aber seit wenigen Wochen ist das RasPi-Kamera-Board offiziell für rund 20 Euro im Handel verfügbar. Lange Lieferzeiten trüben bislang die Freude, aber so nach und nach trudelt das Zubehör bei den Vorbestellern ein.

Die Kamera eröffnet dem RasPi völlig neue Anwendungsfelder. Unter anderem lassen sich mit der Kamera auf einfache Weise Zeitraffer-Videos erstellen. Damit kann man zum Beispiel die eigene Geburtstagsparty für Freunde und Verwandte zusammenfassen oder aber Naturereignisse dokumentieren.

Raspberry-Pi-Kamera

Wie der Raspberry Pi versprüht auch die Kamera eher robusten Charme und kommt als nackte Platine mit einem Flachband-Kabel daher. Insgesamt sieht ein RasPi mit Kamera wie ein zerlegtes Smartphone aus. Die Kamera wurde speziell für den RasPi entwickelt und während der Entwicklungsphase mehrfach überarbeitet. Die aktuelle Version heißt Revision 1.3. Sie ist mit 24 mm × 25 mm annähernd quadratisch, knapp 9 mm hoch und wiegt nur etwa drei Gramm.

Die inneren Werte entsprechen dem, was gängige Mobiltelefone bieten. Der Fokus ist fest und der Sensor liefert 5 Megapixel. Videos nimmt die Kamera in 1080p, 720p oder 640 × 480 mit maximal 30 Bildern pro Sekunde auf. Ton erfasst das Gerät nicht und Fotos haben eine maximale Auflösung von 2592 × 1944 Pixeln.

Anschluss finden

Die Kamera schließt man an die CSI-Schnittstelle (Camera Serial Interface) des RasPi an. Diese liegt beim Modell B hinter der Ethernet-Buchse und beim Modell A hinter der Stelle, an der sich die Ethernet-Buchse befinden würde.

Der CSI-Anschluss muss erst einmal geöffnet werden. Dazu nimmt man am besten die beiden äußeren Enden zwischen Daumen und Zeigefinger und zieht das Ganze wenige Millimeter nach oben. Danach schiebt man das Kabel in die CSI-Buchse, wobei die Kontakte in Richtung des HDMI-Anschlusses zeigen müssen. Das Kabel lässt sich nur wenige Millimeter tief in die Buchse schieben und es ist wichtig, dass es einigermaßen gerade sitzt. Abschließend wird die Buchse mit leichtem Druck von oben wieder verschlossen.

Wenn die Kamera angeschlossen ist, lässt das Kabel knapp 15 Zentimeter Spiel. Für die meisten Zwecke dürfte das ausreichen, insbesondere weil es sehr flexibel ist. Im Bedarfsfall lässt sich das Kabel auch austauschen, darf aber nicht beliebig lang werden. In den RasPi-Foren findet man Erfolgsmeldungen für bis zu 30 Zentimeter Kabellänge.

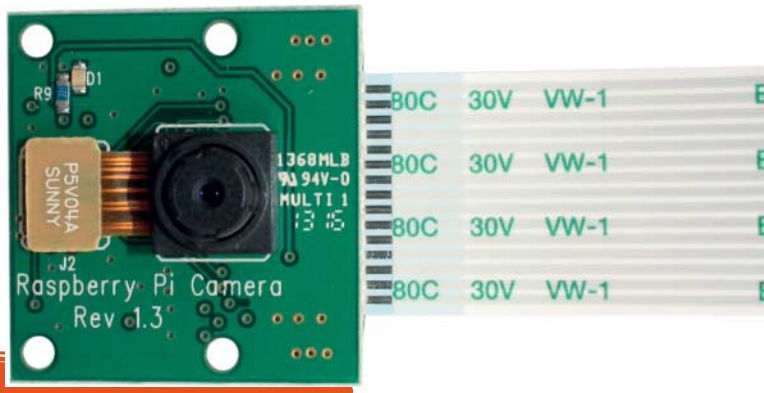
Nachdem die Hardware korrekt angeschlossen ist, fehlt noch ein wenig Software, um die Kamera zum Leben zu erwecken. Wer mit Raspbian arbeitet, muss auf der Software-Seite für die Integration der Kamera nicht viel tun. Die folgenden Kommandos bringen das System auf den neuesten Stand:

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get install git-core -y
sudo wget http://goo.gl/1BOfJ -O /usr/bin/rpi-update
sudo chmod +x /usr/bin/rpi-update
sudo rpi-update
```

Diese Anweisungen aktualisieren alle installierten Pakete, und installieren das Programm rpi-update. rpi-update aktualisiert die Firmware für den Raspberry Pi und installiert auch die Treiber für die Kamera.

Anschließend muss die Kamera-Unterstützung mittels des Konfigurationsprogramms raspi-config noch aktiviert werden:

```
sudo raspi-config
```



Über ein Flachbandkabel verbindet man die Pi-Kamera mit dem CSI-Port. Ton erfasst das Modul nicht. Fotos haben eine maximale Auflösung von 2592 × 1944 Pixel.

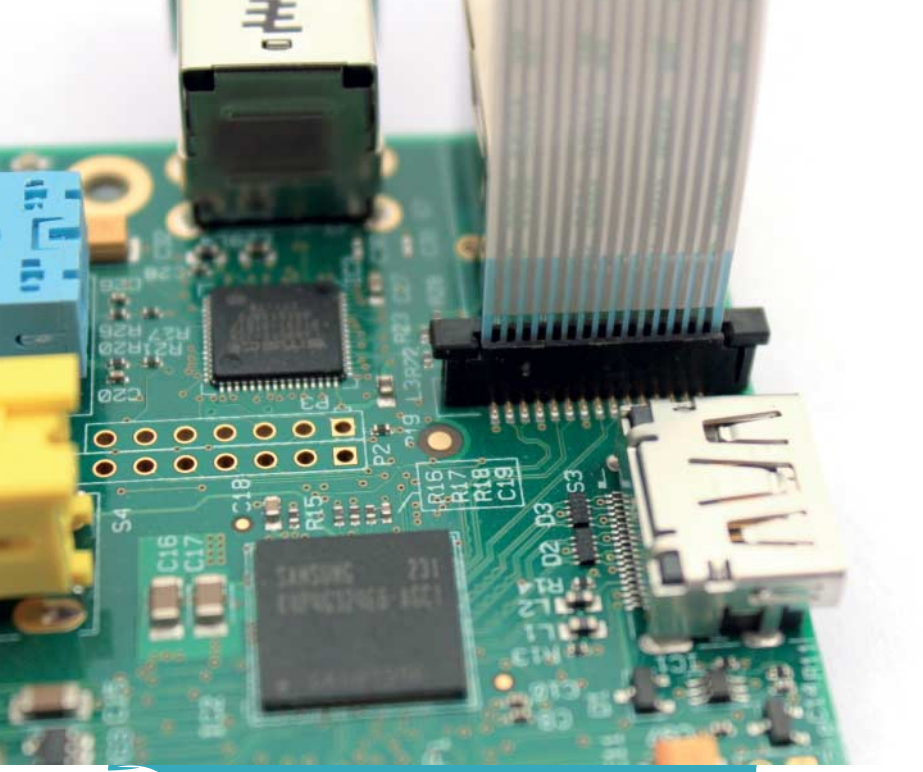
Zutaten

- Raspberry Pi (Model A oder Model B)
- Steckplatine
- PIR-Sensor
- diverse Jumper-Kabel (female/female)
- Raspberry-Pi-Kamera-modul optional
- externer USB-Akku
- WLAN-Adapter
- use für Raspberry Pi

Listing 1

```
import RPi.GPIO as GPIO
from subprocess import call

CameraLedPin = 5

class Camera:
  def __init__(self):
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(CameraLedPin, GPIO.OUT, initial = False)

  def led_on(self):
    GPIO.output(CameraLedPin, True)

  def led_off(self):
    GPIO.output(CameraLedPin, False)

  def take_photo(self, filename):
    call(['raspistill -n -t 0 -hf -o {0}'.format(filename)], shell = True)

  def timelapse(self, filenames, interval, duration):
    call(['raspistill -n -o {0} -tl {1} -t {2}'.format(filenames, interval, duration)], shell =
True)
```

© Copyright by Heise Zeitschriften Verlag
Persönliches PDF für alle Leser des Make Magazins aus 30625 Hannover



Das Kameramodul wird an der CSI-Schnittstelle (Camera Serial Interface) befestigt, die sich zwischen dem Ethernet- und HDMI-Anschluss des Raspberry Pi befindet. Der Vorteil der CSI-Schnittstelle gegenüber USB besteht in der direkten Verbindung zwischen dem Kameramodul und dem ARM System-on-a-Chip (SoC).

Die Option „Enable Camera“ muss aktiviert werden. Ferner sollte man den Grafikspeicher des Pi auf 128 MB setzen. Das geschieht im Menü „Advanced Options“ und dem Untermenü „Memory Split“. Nach einem Neustart ist die Kamera bereit.

Licht aus …

Immer wenn die Kamera Aufnahmen macht, schaltet sie automatisch eine recht grelle rote LED ein. Das ist in vielen Situationen unerwünscht und dankenswerterweise lässt sich die LED komplett abschalten.

Dazu öffnet man die Datei /boot/config.txt mit einem Text-Editor wie zum Beispiel nano:

```
sudo nano /boot/config.txt
```

Am Ende der Datei ist die folgende Zeile einzufügen:

```
disable_camera_led=1
```

Anschließend muss der RasPi mit sudo reboot neu gestartet werden. Ab sofort leuchtet die LED nicht mehr, wenn die Kamera Aufnahmen machten. Sie lässt sich bei Bedarf jedoch weiterhin per Software ein- und ausschalten.

… Spot an

Apropos Software: Ein ausgereiftes Python-Modul zur Steuerung der Kamera gibt es noch nicht. Dafür gibt es aber die nützlichen Kommandozeilen-Werkzeuge raspistill und raspivid, die sich leicht in Python einbetten lassen.

Mit raspistill lassen sich auf die Schnelle Fotos machen. Im einfachsten Fall gibt man nur den Namen der Datei an, in der die Fotodaten gespeichert werden sollen:

```
raspistill -o foto.jpg
```

Wenn der Pi an einen Monitor angeschlossen ist, zeigt er beim Aufruf dieses Kommandos für fünf Sekunden das aktuelle Kamerabild in einer Vorschau an. Nach Ablauf dieser Frist wird ein Foto in der Datei foto.jpg gespeichert.

Das ist schon mal ganz praktisch, hat aber noch ein paar Haken. Für die meisten Anwendungen wird die Vorschau nicht benötigt und die Wartezeit vorm Auslösen ist auch nicht immer erwünscht. Darüber hinaus nimmt die Kamera die Fotos per Voreinstellung seitenverkehrt auf. All das lässt sich mit ein paar Optionen beheben:

```
raspistill -n -t 0 -hf -o foto.jpg
```

Die Option -n unterdrückt die Vorschau und -hf spiegelt das Bild an der Horizontalachse. Schließlich setzt die Option -t die Auslöseverzögerung auf 0 Millisekunden.

raspistill verfügt über eine ganze Menge weiterer Optionen und bietet prinzipiell die typischen Einstellungen einer digitalen Kompakt-Kamera. Eine Übersicht über alle Möglichkeiten bietet der bloße Aufruf des Programms ohne Argumente.

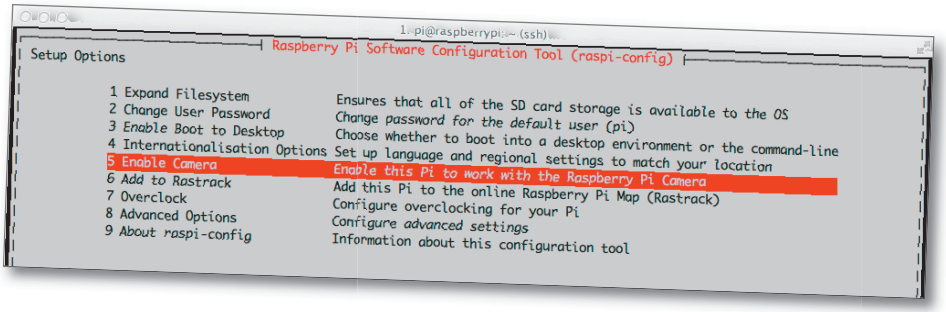
Auch Zeitraffer-Videos lassen sich ohne Mühe erstellen. Der folgende Befehl erzeugt eine Stunde lang alle drei Sekunden ein Foto im Verzeichnis fotos:

```
raspistill -n -t 3000 -tl 3600000 -o fotos/foto%03d.jpg
```

Die erzeugten Dateien gehorchen dabei alle einem strikten Namensschema, das heißt, die Dateien beginnen alle mit dem Präfix „foto“ und darauf folgt eine dreistellige Zahl, die gegebenenfalls mit führenden Nullen aufgefüllt wird. Im Verzeichnis fotos finden sich am Ende also Dateien mit Namen wie foto001.jpg, foto002.jpg und so weiter.

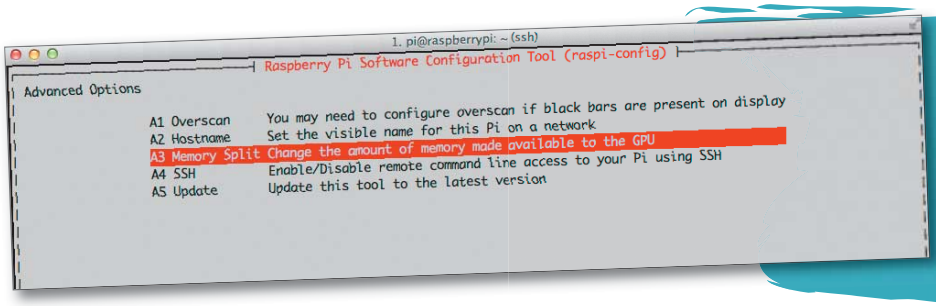
Jetzt mit Python

Ein direkter Zugriff auf die Hardware der Kamera ist ein kompliziertes Unterfangen, aber Programme



Das Kameramodul lässt sich über das Menü komfortabel einrichten.

© Copyright by Heise Zeitschriften Verlag
Persönliches PDF für alle Leser des Make Magazins aus 30625 Hannover



Mit dem Kameramodul steigt der Hunger nach Grafikspeicher. Mindestens 128 MByte sollte man reservieren.

wie raspistill lassen sich leicht in Python einbetten. Wie's geht, zeigt die Klasse Camera in camera.py.

Die Klasse enthält nur wenige Funktionen, bietet aber alles, was man für die meisten Anwendungen benötigt. Der Konstruktor __init__() setzt die Pin-Nummerierung auf den Broadcom-Standard und versetzt den Pin GPIO #5, der mit der Kamera-LED verbunden ist, in den Ausgabemodus. Somit können die Funktionen led_on() und led_off() die LED der Kamera ein- beziehungsweise ausschalten. Das funktioniert übrigens unabhängig von Konfigurationsparameter disable_camera_led.

Mit der Funktion take_photo() kann man ein Foto schießen und in einer Datei speichern. Die Funktion timelapse() dient zur Erstellung von Zeitrafferaufnahmen und erzeugt für eine gewisse Dauer in festgelegten Intervallen Foto-Dateien. Sämtliche Zeitangaben müssen in Millisekunden erfolgen.

Das folgende Programm erzeugt alle zwei Sekunden eine neue Aufnahme und bricht nach zwanzig Sekunden ab:

```
from camera import *
camera = Camera()
camera.timelapse('photos/photo%04d.jpg', 2000, 20000)
```

Die erzeugten Dateien landen im Verzeichnis photos und haben die Namen photo0001.jpg, photo0002.jpg und so fort.

Keine Bewegung!

Erweitert man den vorangegangenen Aufbau zusätzlich noch um einen Bewegungssensor, wird aus dem RasPi schnell ein günstiges und vollautomatisches Überwachungssystem, das unter anderem als Einbrecher-Alarm oder zur Beobachtung von Nistkästen dienen kann.

Um Bewegungen mit dem RasPi zu erkennen, bietet sich ein passiver Infrarotsensor (PIR) an. Solche Sensoren sind günstig zu haben und lassen sich leicht auswerten.

Der Infrarot-Sensor verfügt über drei Pins: Stromversorgung, Masse und Signal. Um diese mit dem RasPi zu verbinden, sind Jumper-Kabel mit weiblichem Anschluss an beiden Enden besonders praktisch. Die Stromversorgung geschieht über den 5V-Pin des RasPi, zusätzlich verbindet man beide Masse-Leitungen miteinander. Die Signalleitung erfolgt mit einem der GPIO-Pins, in diesem Fall ist es die Nummer 23.

Tipp

Auf dem RasPi gibt es zwei Arten, einen Pin zu adressieren: BCM (Broadcom) und BOARD. Bei der ersten werden die Pins so angesprochen, wie sie der Hersteller Broadcom im „System on a Chip" (SoC) des RasPi definiert hat. Die zweite Art verwendet die Nummerierung der Pins auf dem RasPi-Board. Alle Pins in der oberen Reihe haben gerade Nummern, die in der unteren Reihe haben ungerade. Beispielsweise entspricht der BCM-Pin GPIO #23 dem Board-Pin 16.

Listing 2

```
import RPi.GPIO as GPIO

class PassiveInfraredSensor:
  def __init__(self, pin):
    self.pin = pin
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(self.pin, GPIO.IN)

  def motion_detected(self):
    return GPIO.input(self.pin)
```

Sensor-Software

Die Verkabelung des PIR-Sensors geht leicht von der Hand und auch auf der Software-Seite ist nicht viel zu tun. Zur Umsetzung der Beispiele dieses Artikels nutzen wir die Programmiersprache Python. Sie ist verhältnismäßig einfach zu erlernen und in der Raspberry-Pi-Gemeinde sehr populär. Mittlerweile existiert auch eine ganze Reihe an Python-Bibliotheken, welche die Programmierung der RasPi-Hardware vereinfachen. Eine davon ist RPi:

```
sudo apt-get update
sudo apt-get install python-dev python-rpi.gpio
```

Mehr Voraussetzungen benötigt das Beispiel-Programm pir.py nicht. Darin ist eine Klasse namens PassiveInfraredSensor definiert, mit der sich der PIR-Sensor kontrollieren lässt.

Die Klasse verfügt über zwei Funktionen, nämlich __init__() und motion_detected(). Die erste Funktion ist der sogenannte Konstruktor. Sie wird immer dann aufgerufen, wenn ein neues Objekt der Klasse erzeugt wird. Als Parameter bekommt die Funktion einen Verweis auf das zu erzeugende Objekt self und die Nummer des Pins pin, an den der PIR-Sensor angeschlossen wurde.

Der Konstruktor speichert zuerst die Pin-Nummer im privaten Attribut self.pin und nutzt dann die Funktion GPIO.setmode(), um die Pin-Zählweise der RPi-Bibliothek festzulegen.

Abschließend versetzt die __init__()-Funktion den Sensor-Pin mittels GPIO.setup() in den Eingabemodus.

Die Funktion motion_detected() liefert den aktuellen Wert, der am Sensor-Pin anliegt. Ist der Wert 0, wurde keine Bewegung erkannt. Andernfalls hat sich im Sichtfeld des Sensors etwas bewegt.

Kurze Demo

Das Beispiel-Programm pir_demo.py demonstriert die Verwendung der neuen PassiveInfraredSensor-Klasse. Es muss, wie alle Programme, die auf die GPIO-Pins zugreifen, mit Administrator-Rechten gestartet werden: sudo python pir_demo.py

© Copyright by Heise Zeitschriften Verlag
Persönliches PDF für alle Leser des Make Magazins aus 30625 Hannover

Bei gutem Tageslicht gelingen mit dem Kameramodul recht gute Fotos. Bei schlechten Lichtverhältnissen entsteht Bildrauschen und auch Hintergrunddetails sind nur schwer erkennbar.

Die Logik des Programms ist denkbar einfach. Es merkt sich den letzten Sensor-Zustand und gibt den Text „Wer ist da?“ aus, wenn eine Bewegung erkannt wurde. Wurde keine Bewegung erkannt, wird die Zeile „Alles ruhig.“ ausgegeben und das Programm wartet zwei Sekunden, bis die Überwachung erneut gestartet wird.

Mit derselben Logik lässt sich auch ein Programm entwickeln, das ein Foto schießt, wenn sich etwas bewegt. Dazu muss allerdings erst einmal die RasPi-Kamera angeschlossen und installiert werden.

Alles spielt zusammen

Mit den Klassen PassiveInfraredSensor und Camera lässt sich ein Überwachungssystem bauen, das automatisch die Kamera aktiviert, wenn eine Bewegung erkannt wurde. Das Programm observer.py zeigt eine minimale Implementierung.

Im Kern entspricht das Ganze der ersten Demonstration des PIR-Sensors. Statt jedoch nur Texte in der Konsole auszugeben, nutzt es die neue Camera-Klasse, um Fotos zu schießen.

Mit minimalem Hard- und Software-Einsatz lässt sich so ein flexibles und leistungsstarkes Überwachungssystem bauen. Selbstverständlich können statt eines PIR-Sensors auch ganz andere Sensoren, wie zum Beispiel Temperaturfühler oder Feuchtigkeitssensoren zum Einsatz kommen.

Listing 3

```
from pir import *
import time

PIR_PIN = 23
pir = PassiveInfraredSensor(PIR_PIN)

last_state = False
while True:
    if (pir.motion_detected() ==
True):
        if (last_state == False):
            print "Wer ist da?"
            last_state = True
    else:
        if (last_state == True):
            print "Alles ruhig."
            time.sleep(2)
            last_state = False;
```

Daten speichern

Bei längeren Überwachungsaktionen muss man sich Gedanken darüber machen, wo das erzeugte Bildmaterial gespeichert wird. Im einfachsten Fall schreibt man die Daten auf eine SD-Karte und eine Kapazität von 32 GB reicht locker für Zeitraffer-Aufnahmen über mehrere Stunden hinweg.

Für längere Laufzeiten oder höhere Bildfrequenzen bieten sich eine externe Platte oder aber Netzwerk-Dateisystem wie zum Beispiel Samba an. Damit überträgt man die Dateien automatisch vom RasPi auf einen Rechner mit mehr Speicherkapazität.

Netzwerk-Anbindung

Auch um das System von außen zu kontrollieren, empfiehlt sich eine Netzwerk-Anbindung, vorzugsweise per WLAN. Der RasPi unterstützt mittlerweile eine große Auswahl an WLAN-Sticks. Viele dieser Geräte, wie zum Beispiel der EDIMAX EW-7811UN, kosten nur knapp 10 Euro und verbrauchen wenig Strom.

© Copyright by Heise Zeitschriften Verlag
Persönliches PDF für alle Leser des Make Magazins aus 30625 Hannover

Für die Konfiguration eines solchen Sticks muss man lediglich zwei Dateien editieren. Die erste Datei ist /etc/network/interfaces. Nach dem Editieren muss die Datei wie folgt aussehen:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface home inet static
address 192.168.1.105
netmask 255.255.255.0
network 192.168.1.0
gateway 192.168.1.1
```

Damit erhält der RasPi, wenn er per Ethernet mit einem Netzwerk verbunden ist, per DHCP automatisch eine IP-Adresse. Ist er aber mit einem WLAN verbunden, bekommt er die statische IP-Adresse 192.168.1.105. Die Netzwerk-Parameter müssen an die lokalen Gegebenheiten angepasst werden.

Ferner müssen in der Datei /etc/wpa_supplicant/wpa_supplicant.conf noch die Parameter der WLAN-Verbindung namens „home“ definiert werden:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="darknet"
    psk="t0p$ecret"
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=TKIP
    auth_alg=OPEN
    id_str="home"
}
```

Hier sind die Parameter ssid und psk auf die tatsächliche SSID und das tatsächliche Passwort zu setzen. Nach der Neukonfiguration sollte der RasPi neu gestartet werden. Ab sofort kann man sich per WLAN und Ethernet beim Überwachungssystem anmelden. Packt man den RasPi in ein Gehäuse und schließt ihn via USB an einen externen Akku an, hat man so schon ein mobiles Gerät, das man zu Überwachungszwecken im Garten deponieren kann.

Tipp

Unter Hobbyisten ist der PIR-Sensor von Parallax recht beliebt. Dieser arbeitet ausgezeichnet mit dem RasPi zusammen. Es gibt ihn in den Ausführungen Rev A und Rev B. Die neuere Version Rev B kommt mit ein paar Detailverbesserungen daher, liefert aber ein höheres Ausgangssignal als Rev A. Für den Einsatz am RasPi scheidet diese Version daher aus, weil dieser Spannungen oberhalb von 3,3 V nicht verträgt. Rev A tut’s für die meisten Anwendungen aber auch und ist obendrein noch günstiger. Wer einen anderen Sensor einsetzen möchte, muss darauf achten, dass die Ausgangsspannung des Sensors nicht mehr als 3,3V beträgt.

Fazit

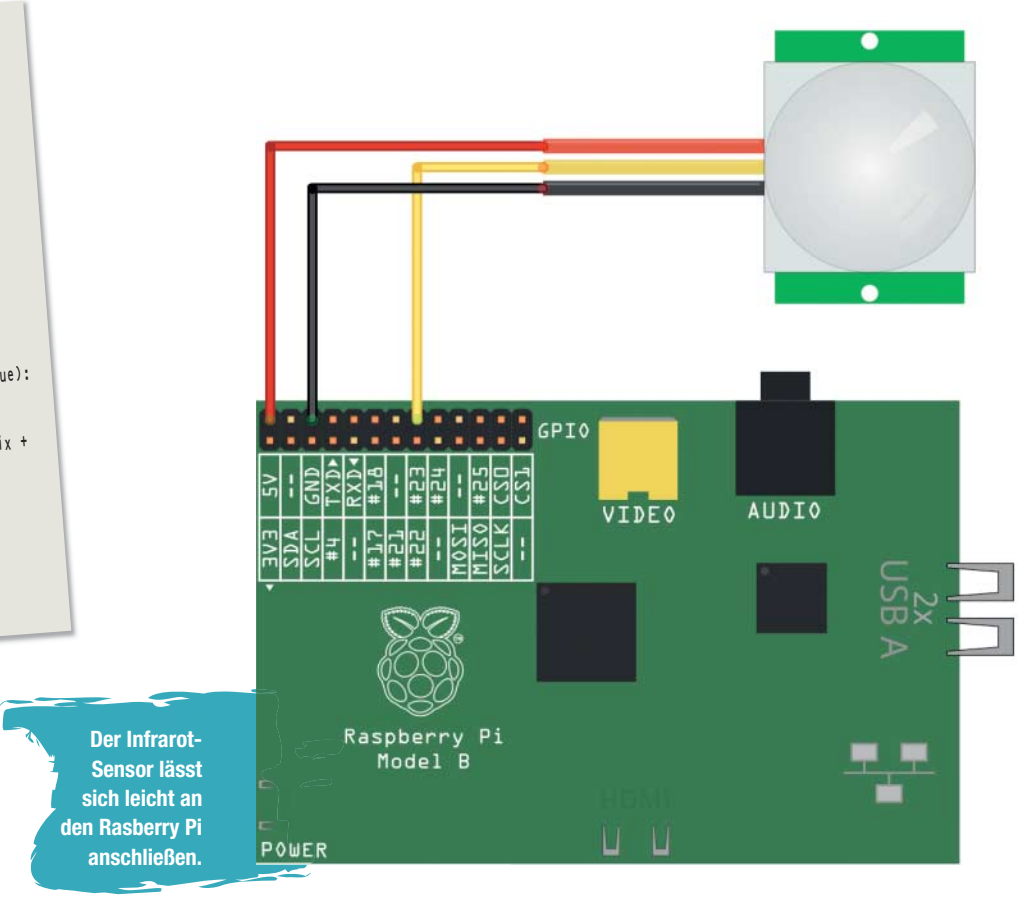
Die Arbeit mit der RasPi-Kamera macht Spaß. Zwar ist die Qualität der Bilder nicht überragend, aber das Gerät erweitert den Raspberry Pi sinnvoll und reicht für viele Einsatzzwecke vollkommen aus.

Darüber hinaus wiegt der Winzling nur drei Gramm und hängt an einem halbwegs flexiblen Kabel. Es ist also kein Problem, die Kamera mit einem Motor zu bewegen und auszurichten. Auch ein automatischer Blitz, der zum Beispiel von einem Lichtsensor gesteuert wird, ließe sich ohne großen Aufwand nachrüsten. (ogo)

Links und Foren
www.ct.de/ch1303024

Listing 4

```
from pir import *
from camera import *
import time

PIR_PIN = 23
pir = PassiveInfraredSensor(PIR_PIN)
camera = Camera()

prefix = 'photos/test'
i = 0
last_state = False
while True:
    if (pir.motion_detected() == True):
        if (last_state == False):
            print "Foto!"
            camera.take_photo(prefix +
'{0}.jpg'.format(i))
            i = i + 1
            last_state = True
    else:
        if (last_state == True):
            time.sleep(1)
            last_state = False;
```



Der Infrarot-Sensor lässt sich leicht an den Rasberry Pi anschließen.

© Copyright by Heise Zeitschriften Verlag
Persönliches PDF für alle Leser des Make Magazins aus 30625 Hannover