Katzenklappe mit Katzenerkennung

von Julia Hellmann

Inhalt

1. Motivation und Zielsetzung

Meine Motivation liegt vor allem darin etwas Neues zu lernen und ein angefangenes Projekt fertigzustellen. Wobei dies, bei dem vorliegenden Projekt, auch für meine Familie einige Vorteile hat, da wir selbst zwei Katzen zu Hause haben, von denen zumindest eine, eine begeisterte Jägerin ist. So ist es meine Zielsetzung eine Katzenklappe zu entwickeln, welche mittels Gesichtserkennungssoftware gesteuert wird. Die Katze soll nur das Haus betreten dürfen, wenn sie keine Maus oder keinen Vogel im Maul trägt. Dies soll die Gesichtserkennungssoftware erkennen.

2. Kurzfassung

Bei meinem Projekt "Katzenklappe mit Katzenerkennung" wird mithilfe einer integrierten HD-Kamera eine Katzenklappe optimiert. Viele Katzen, vor allem Freigänger, sind dafür bekannt, dass sie kleine "Geschenke" in Form von Mäusen und Vögeln von ihren Beutezügen mitbringen. Um zu verhindern, dass Katzen diese mit ins Haus bringen, soll die Kombination aus HD-Kamera inkl. Gesichts-erkennungssoftware und Katzenklappe Abhilfe schaffen. Die Gesichtserkennung sorgt dafür, dass sich die Klappe nur öffnet, wenn die Katze keine Beute im Maul trägt. Die dabei verwendeten Hard- und Softwarekomponenten beruhen im Wesentlichen auf der Raspberry-Pi Familie. Das Erkennungsprogramm basiert auf OpenCV. Das Projekt wird dabei nicht nur für das oben genannte Problem eine Lösung bieten, sondern darüber hinaus zeigen, dass zukünftig die Steuerung der Katzenklappe in eine Haussteuerung integriert oder über selbige (oder über das Internet) erfolgen kann.

3. Stand der Technik

Zum jetzigen Zeitpunkt gibt es die sogenannten „normalen“ mechanischen Katzenklappen, welche wirklich nur als Durchgang für Katzen/Hunde dienen sowie die ersten mikrochipgesteuerten automatischen Katzenklappen, die bereits einige Möglichkeiten zur Festlegung einiger weniger Steuer- und Zugangsregelungen besitzen. Die Nachteile dieser heutigen automatischen mikrochipgesteuerten

Katzenklappen sind allgemein bekannt (siehe a.). So hat man, wenn man Freigänger besitzt, bisher nur drei Möglichkeiten.

a. Beschaffung einer mikrochipgesteuerten Katzenklappe, die anhand des in die Katze implantierten oder des im Halsband enthaltenen Chips erkennt, ob die Katze zur Familie gehört und ins Haus darf und nimmt dabei aber den Nachteil in Kauf, dass die Katze ab und zu ein Mitbringsel z.B. in Form von Mäusen mit ins Haus bringt,
b. Benutzung einer einfachen mechanischen Katzenklappe (Nachteile siehe a.), oder
c. Verzicht auf die Anschaffung einer solchen und man kümmert sich selbst um den Auslauf seiner Lieblinge.

Die von mir geplante Kombination von Katzenklappe und Gesichtserkennungs-software mit dem Programm OpenCV zum Schutz vor ungebetenen Mitbringsel wie Mäusen und Vögeln gibt es zurzeit noch nicht in dieser Form zu kaufen. Es gab aber bereits Versuche oben genannte Defizite mittels eines ähnlichen Projektes anzugehen. Auf dieses werde ich später noch einmal genauer eingehen (Absatz 7.2). Hier nur so viel, die Umsetzung beruht allein auf die Nutzung von Lichtschranken und ist aus meiner Bewertung heraus sehr fehleranfällig.

Gesichtserkennung wird hauptsächlich im Bereich Sicherheit verwendet. So gibt es neben den professionellen Anwendungsprogrammen der Sicherheitsorgane wie für die Polizei verschiedene frei verfügbare Lösungsansätze. Da ist zum Beispiel die frei verfügbare Programmbibliothek OpenCV in der Programmiersprache Python oder auch C++, das unter anderem die Haar-Kaskaden Klassifizierung verwendet. Bei der Haar-Kaskaden-Klassifizierung handelt es sich um eine der häufiger genutzten Arten der Objekterkennung. Dabei wird das zu untersuchende Bild nach und nach in kleinere Abschnitte unterteilt, in denen der Algorithmus einfache Merkmale wie Farbverlauf, Ecken, Kanten oder Linien heraussucht und anhand eines vorher trainierten Modells des Objektes eine Wahrscheinlichkeit für jeden Abschnitt des Bildes aufstellt, ob das Objekt im Bild vorhanden ist. Aber das Programm OpenCV ist natürlich nicht die einzige Möglichkeit zur Erkennung von Bildern/Texten oder Objekten.

Weitere Methoden, auf die ich aber nicht näher eingehe, sind beispielsweise:

- Neuronale Netzwerke
- FindFace
- Tesseract

- OCR
- Blink

4. Aufbau und Material

Bevor ich zum Aufbau und Material komme, muss ich eine Tatsache besonders hervorheben: Bei dem gesamten Prototyp-Modell muss berücksichtigt werden, dass der Versuchsaufbau bisher ausschließlich darauf aufgebaut ist, bei genügend (Tages)Lichteinfall verwendet zu werden. Ich habe bewusst auf die Nutzung von Infrarotversuchen und einen entsprechenden Versuchsaufbau verzichtet. Ich verweise hierzu auf Absatz 7.1.

Das Modell für den Wettbewerb besteht aus einem Raspberry Pi 3b, einer PiNoiR V2 Kamera (beides, siehe Abbildung 3), einer umfunktionierten mikrochipgesteuerten Katzenklappe von SUREFLAP, einem Steckernetzteil von Voltcraft, einem 4-Kanal-Relaisboard und einer Holzplatte. Auf dem Raspberry Pi befindet sich das Erkennungsprogramm auf einer 4 GB SD-Card. Als Ergänzung habe ich noch ein sieben Zoll Touchpad hinzugefügt, um auch später noch Einstellungen vornehmen zu können. Das gesamte Material, mit Ausnahme der Holzbretter, habe ich bei Amazon bestellt und ist frei verfügbar. Angefangen habe ich dabei mit dem Raspberry Pi, der HD-Kamera und einem Handbuch für den Raspberry Pi. Den Computer schloss ich für die Arbeitszeit an Maus, Tastatur und Bildschirm an. Mit Hilfe einer sehr guten Bekannten und entsprechenden Informationen im Internet (siehe Links) entstanden dann die ersten Programme. Zuallererst ging es um ein paar allgemeine Grundprinzipien von Python, welche ich von Anfang an verwendet habe. Da Python häufig genutzt wird und nicht besonders schwierig zu erlernen ist, lud ich eine OpenCV Modelldatei herunter. Um diese zu testen, lud ich ein paar Fotos aus dem Internet auf meinen Computer. Für die Simulation einer elektrisch gesteuerten Verriegelung (z.B. Magnetschalter) nutzte ich eine LED (rot), die bei positivem Ergebnis (Katzenklappe öffnet) aufleuchten sollte.

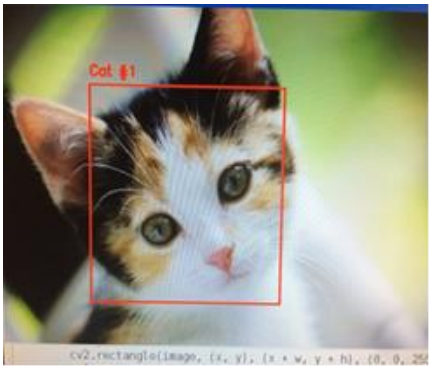

Abb.1:

Bild einer Katze, welches von dem OpenCV Programm als solche erkannt wird

Da sich aber nach einigen Versuchen herausstellte, dass sich diese Modelldatei nicht mit den General Purpose Input/Output (GPIO)-Pins des Raspberry Pi verbinden lies, musste ich eine andere Version der Modelldatei herunterladen, das war die Version OpenCV 3.3.0. Diese Version habe ich anschließend unter Nutzung eines selbst geschriebenen Programms mit der Kamera kombiniert. Wobei ich einen Teil des Programmquellcodes aus einem ähnlichen Programm aus dem Internet (Link 9) mit meinen Programmanteilen ergänzt habe. Die ersten Versuche zeigten, dass das Programm zwar anfänglich eine gewisse Fehlerquote, da manche Katzen zum Beispiel aufgrund der Kopfform nicht sofort erkannt wurden, in der Gesamtheit aber war diese Kombination recht zuverlässig. Die Frage welche Katzenklappe ich verwenden sollte, beschäftigte mich eine ganze Weile. Zunächst war geplant, dass ich eine „normale“ mechanische Katzenlappe verwende und an dieser einen Verriegelungsmechanismus befestige. Schlussendlich benutze ich aber eine modifizierte mikrochipgesteuerte Katzenklappe, bei welcher ich die ursprüngliche Steuerplatine entfernte. Anschließend schloss ich den schon eingebauten Riegel (ein elektromagnetischer Schalter) über das Relaisboard, einem Stecker-Netzteil und den GPIO-Pins an den Raspberry Pi an. Mein Programm veränderte ich daraufhin soweit, dass es mir gelang, die Katzenklappe mit dem ursprünglichen Verriegelungs-mechanismus ohne Probleme zu öffnen und zu schließen. Als dies fertig war, fing ich mit dem eigentlichen Aufbau an. Für den Türersatz hatte ich mir ein Holzbrett gekauft, in welche die Katzenklappe eingesetzt worden ist. Darüber ist in einem kleinen Kasten der Raspberry Pi mit Kamera, Relaisboard und Zubehör angebracht.

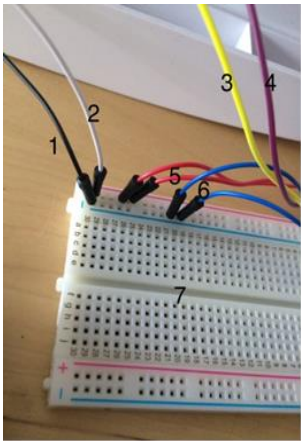

Abb.2: Anschlussübersicht Steckbrett (7), GND (**2**) +(**3+4**), Anschlüsse an die Katzenklappe, **5.** + Anschlüsse an das Relaisboard, **6.** - Anschlüsse an das Relaisboard,

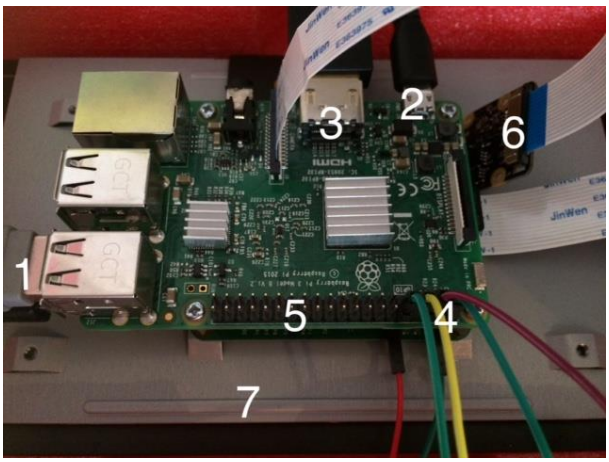

Abb.3: Raspberry Pi 3b, **1.** USB-Anschlüsse Maus u. Tastatur, **2.** Stromanschlusskabel, **3.** HDMI Anschluss Bildschirm, **4.** Verbindungen zum Relaisboard, **5.** GPIO-Pins, **6.** Kamera Anschluss + Kamera (PiNoiR V2)

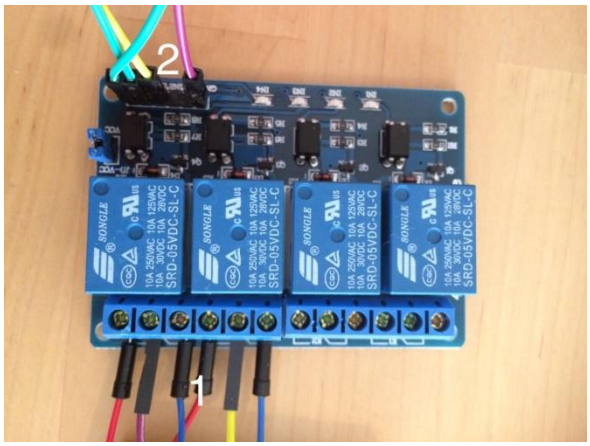

Abb. 4: Kanal-Relaisboard, **1.** Anschlüsse zur elektromagnetischen Verriegelung u. über ein Steckbrett zum Steckernetzteil, **2.** Verbindungen zum Raspberry Pi

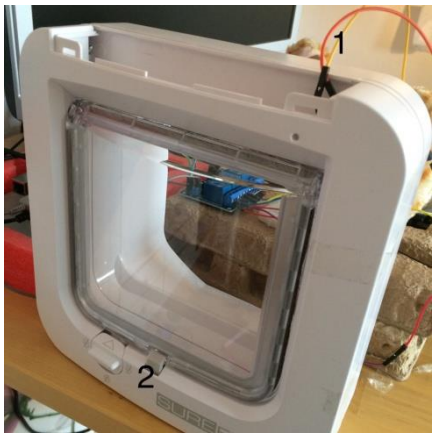

Abb.5: Katzenklappe, **1.** Verbindungskabel zum Relaisboard, **2.** Verriegelung

5. Erste Programme für den Raspberry Pi 3 mit Kameramodul

Um meine Idee in die Tat umzusetzen, habe ich mich für einen Raspberry Pi entschieden. Für die Raspberrys gibt es verschiedene Kameras. Zum einen die PiNoiR, die keinen Infrarotfilter besitzt, das Pi Camera Board und seit 2016 auch zu beiden eine überarbeitete Version, welche mit V2 gekennzeichnet sind. Auch beim

Raspberry selbst gibt es verschiedene Versionen. Da gibt es die Raspberry Pi's 1, 2 und 3, in Modellen A/A+ und B/B+, den Raspberry Pi Zero W, welcher der Kleinste bisher verfügbare ist und das Raspberry Pi 3 Compute Module. Auf die einzelnen Unterschiede gehe ich nicht ein.

Wie bereits erwähnt (Absatz 2.1), habe ich mich für den Raspberry Pi 3b entschieden, da dieser einer der leistungsfähigsten Raspberry Pi's ist und vor allem über genügend Anschlüsse verfügt. Zwar ist dieses Modell nicht das neueste, der anschließend entwickelte Raspberry Pi Zero W ist allerdings viel kleiner, verfügt dadurch nur über die minimalste Zahl an Anschlüssen und besitzt nur halb so viel Speicherplatz wie das Modell 3b.

Mein erstes Programm kombiniert die Gesichtserkennungssoftware OpenCV mit dem Kameramodul. Vor dem Anschluss an die Katzenklappe griff mein Programm auf eine LED zu, die bei Erkennung eines gesuchten Objekts aufleuchtete. Diese wurde später durch den Verriegelungsmechanismus der Klappe ersetzt. Die Aufnahme der Kamera wird währenddessen auf dem Bildschirm angezeigt.

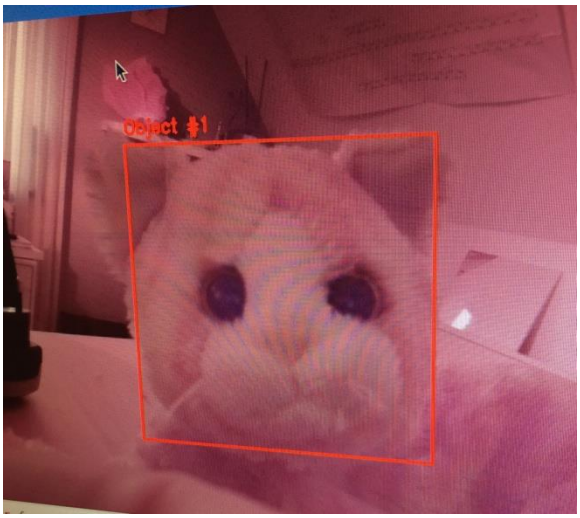

Abb.6:
Programm erkennt Plüschkatze mit der HD-Kamera

Die dabei Integrierte OpenCV Modelldatei, welche für die Erkennung zuständig ist, habe ich dabei vollständig aus dem Internet heruntergeladen. Diese ist in Python geschrieben. Das gesamte Programm stellt sich in Teilen aus Kopien aus dem Internet zusammen, ist aber mit eigens adaptierten Programmanteilen ergänzt. Später fügte ich die Programmzeilen hinzu, die den Zugriff/Steuerung des Relais ermöglichen, um den elektromagnetischen Schalter automatisch zu öffnen und zu schließen.

5.1 Erste Erfolge

Das OpenCV Programm (Version 3) auf Basis Haar-Kaskaden-Klassifizierung habe ich auf dem Raspberry Pi installiert und durch ein weiteres Programm mit der Kamera verbunden. Es erkannte schon nach den ersten Versuchen ziemlich zuverlässig Katzen. Die Verbindung des Raspberrys mit der leicht umfunktionierten elektrischen Katzenklappe funktionierte ebenfalls. Mit Hilfe einiger Vorlagen aus dem Internet stellte ich das jetzige Programm (Absatz 3.2) fertig. Die Tests mit der Klappe funktionierten bereits beim ersten Mal.

5.2 Das vollständige Programm

Im Anhang befindet sich der Quellcode des verwendeten Erkennungsprogramms mit beistehender Erläuterung.

6. Herausforderungen und Problemstellungen

Anfängliche Probleme waren:

- die zunächst unscharfen Bilder der Kamera, insbesondere bei zu geringem Lichteinfall
- das erste OpenCV Programm, bei dem sich die GPIO-Pins des Raspberry Pis nicht ansteuern ließen
- nicht genügend Arbeitsspeicher auf dem Raspberry Pi
- Programm erkennt bestimmte Katzen nicht oder nur sporadisch

Des Weiteren hatte ich die Idee, zur Erkennung ein Programm zu verwenden, welches eine oder mehrere bestimmte Katzen individuell erkennen kann. In dieser Hinsicht bestehen die Grundlagen, aus Ordnern mit den benötigten Bildern, sowie die grundlegenden Befehle/Kommandos. Allerdings dauert das Training/der Lernvorgang dieser Programme zur Gesichtserkennung sehr lange. Für diesen Vorgang benötigt man einen leistungsfähigen Computer. Anfänglich war mein Computer überlastet und stürzte mehrfach ab. Um ein möglichst genaues und

fehlerfreies Programm zu erhalten, muss man sich auf eine lange Wartezeit einstellen und geduldig sein.
Eine andere Herausforderung besteht darin, dass das jetzige Programm von Zeit zu Zeit Personen bzw. Gesichter von Personen als „Katzen“ erkennt. Diese Eigenschaft des Programms ist zwar bedauerlich, stellt aber aus meiner Sicht kein sehr großes Problem dar, da im normalen/alltäglichen Gebrauch gar keine Gesichter in den Erkennungsbereich der Kamera kommen.

7. Weiterentwicklung, bisherige Projektideen und Vorteile meines Projektes

7.1 Ideen für die Weiterentwicklung meines Projektes

Als nächster Schritt der Weiterentwicklung könnte man die Katzenklappe mittels individueller Erkennung für eine bestimmte Katze verändern. Außerdem besteht die Möglichkeit einen Versuch mit einem anderen Gesichtserkennungsprogramm durchzuführen, um es mit dem OpenCV zu vergleichen.
Das von mir genutzte Projekt ist ausschließlich auf den Gebrauch bei Tageslicht bzw. Vorhandensein einer ausreichenden (Tages)Lichtquelle ausgerichtet. Daher könnte man versuchen die Klappe so zu überarbeiten, dass diese unter Verwendung einer Infrarot-Kamera ergänzt wird.
Da sprachgesteuerte oder durch eine Applikation (APP) steuerbare Geräte immer häufiger Anwendung finden, wäre in Punkto Ansteuerung weiter voraus zu denken. So kommt es immer häufiger vor, dass sich die Menschen mehr und mehr Technik zulegen, welche sie per Mobiltelefon, Smartphone oder über das Internet steuern können, statt sich mit konventionellen Schaltern zu begnügen. Das kann man sich hier auch zunutze machen. Es ist eine App denkbar, mit welcher man die Katzenklappe direkt steuern würde. Diese App wäre mit einem Web-Service gekoppelt, der die Bilder der eigenen Katze prozessiert, bearbeitet und in die eigene Katzenklappe einspeist. Zu guter Letzt wäre auf Basis der Bluetooth oder Wi-Fi Technologie eine Integration der Katzenklappe in die Haustechnik jederzeit möglich.

7.2 Bisherige Projektideen

Wie in Absatz 2 schon angedeutet, gab es bereits andere Versuche die Katzen/Hundeklappen zu optimieren. Hier zähle ich ein paar dieser Ideen auf, welche meiner Entwicklung im Kerngedanken besonders nahekommen, also zu verhindern, dass sich eine Katzenklappe öffnet, wenn das betreffende Tier seine Beute, z.B. eine Maus, mitbringt.

So wurde 2009, mit dem Projekt „Katzenklappe mit Mauserkennung“, von drei Auszubildenden, der Sonderpreis der Internationalen Fachmesse "Ideen-Erfindungen-Neuheiten" (iENA) beim Regionalentscheid "Jugend forscht" in der Oberpfalz gewonnen.

Bei diesem Projekt wurden, zur Erkennung des Umrisses des Kopfes einer Katze, Lichtschranken verwendet.

Eine weitere Erfindung (theoretisch) wurde beim Deutschen Patent- und Markenamt gesichert. Es ist dort seit dem 30.01.2014 eine „Steuervorrichtung für eine Zugangsvorrichtung für Tiere, insbesondere für eine Katzenklappe“ mit der Veröffentlichungsnummer De102014101167A1 zum Patent angemeldet. Dabei handelt es sich, wie der Name schon zum Teil verrät ebenfalls um eine elektronische Vorrichtung für Katzenklappen. Es wird bei dieser Arbeit, mithilfe einer Kamera und den Umrissen/Konturen der Katze, festgestellt, ob das betreffende Tier etwas im Maul trägt oder nicht, worauf die Klappe bei letzterem geöffnet wird. Aufgrund der vorliegenden Patentanmeldung wurde meinerseits auf eine Patentanmeldung verzichtet.

7.3 Vorteile meines Projektes

Im Vergleich zu anderen bisher markverfügbaren automatischen Katzenklappen ist meine Kombination aus computergestützter Gesichtserkennung, HD-Kamera und Magnetschalter in der Lage, ohne großen Aufwand, individuell angepasst zu werden. Außerdem hat es die bereits oben genannten Vorteile (Absatz 2). Es verbindet viele verschiedene Komponenten und bietet hervorragende Möglichkeiten zur Weiterentwicklung.

8. Literatur- und Linkliste

Literatur:
1. Raspberry Pi - Das umfassende Handbuch von Koffer Kühnast Scherbeck, ISBN 978-3-8362-5859-3
2. Foto und Video mit Raspberry Pi von E. F. Engelhaft, ISBN 978-3-645-60314-0

Links:
1.http://www.pyimagesearch.com
2.http://docs.opencv.org
3.http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html
4.http://stackoverflow.com
5.http://GitHub.com/sonots/tutorial-haartraining
6.http://Wikipedia.de(OpenCV, Gesichtserkennung)
7.https://www.katzenklappen-infos.de/katzenklappe-mit-mauserkennung/
8.http://www.neumarktonline.de/art.php?newsid=59507
9.https://www.pyimagesearch.com/2016/06/20/detecting-cats-in-images-with-opencv/
10.https://depatisnet.dpma.de/DepatisNet/depatisnet?window=1&space=main&content=treffer&action=treffer&atreffer=1
11.https://pjreddie.com/darknet/yolo/
12.https://www.datenschutz-notizen.de/findface-verbrecherjagd-mit-neuester-gesichtserkennungssoftware-0118560/
13.https://opensource.google.com/projects/tesseract
14.http://thema.giga.de/s/gesichtserkennung-software-freeware-download/

Alle Links zuletzt am 10.01.18 verwendet.
Alle verwendeten Fotos sind selbst gemacht.
Hilfe hatte ich bei dem ganzen Projekt vor allem von meinem Vater und einer guten Bekannten.

9. Anhang:

Adaptierter Quellcode für das Erkennungsprogramm inkl. Magnetschaltersteuerung

```
#Benötigte Bibliotheken importieren

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

GPIO.setup(14,GPIO.OUT)
GPIO.setup(15,GPIO.OUT)

def motor_close():              #Riegel aus/zu:
    GPIO.output(14,GPIO.LOW)      #Pin 14 Strom
    GPIO.output(15,GPIO.HIGH)     #Pin 15 kein Strom
    time.sleep(0.1)             #Warten (0.1)
    motor_stop()                #Motor stop/aus

def motor_open():               #Riegel offen:
    GPIO.output(14,GPIO.HIGH)     #Pin 14  kein Strom
    GPIO.output(15,GPIO.LOW)      #Pin 15 Strom
    time.sleep(0.1)             #Warten (0.1)
    motor_stop()                #Motor aus/stop

def motor_stop():               #Motor aus/stop:
    GPIO.output(14,GPIO.HIGH)     #Pin 14 kein Strom
    GPIO.output(15,GPIO.HIGH)     #Pin 15 kein Strom

#Für das Programm weitere benötigte Bibliotheken (Kamera, Raspberry Pi Camera)
importieren

import cv2                                          #Modelldatei OpenCV importieren
from picamera.array import PiRGBArray
from picamera import PiCamera

detector = cv2.CascadeClassifier("haarcascade_frontalcatface_extended.xml")
#Die Modelldatei, die das zu erkenne Objekt beschreibt, laden (Katzenerkennung)
                                                    #detector =
cv2.CascadeClassifier("BuchErweitert.xml") #Hier das gleiche (Bucherkennung)

camera = PiCamera()                                 #Kamera benutzen
camera.resolution = (640, 480)      #Auflösung der Kamera auf 640*480 Pixel setzen
camera.framerate = 32         #Bildrate der Kamera auf 32 Bilder pro Sekunde setzen
rawCapture = PiRGBArray(camera, size = (640, 480))          #Format der
später aufgezeichneten Bilder festlegen
motor_state = 0
```

```
time.sleep(0.1)                 #Einen Moment warten, damit die Kamera "starten" kann
motor_close()

for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):     #Bilder von der Kamera aufzeichnen und in die Variable
"frame" speichern
    image = frame.array     #Die Pixel des aufgezeichneten Bildes werden in einem für
OpenCV verständlichen Format in die Variable "image" gespeichert
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)   #Das Bild in schwarz-weiß
umwandeln und ion der Variable "gray" speichern

rects = detector.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=20,
#Objekte im schwarz-weiß Bild erkennen und als Liste von Rechtecken in die
Variable "rects" speichern
minSize=(75, 75))              #Genauigkeit, min. Größe

A    for (i, (x, y, w, h)) in enumerate(rects):            #Alle Rechtecke (also alle
erkannten Objekte) durchlaufen
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)    #Das Rechteck in das
Kamerabild einzeichnen
        cv2.putText(image, "Object #{}".format(i + 1), (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.55, (0, 0, 255), 2)  #Nummer des Objekts in das
kamerabild einzeichnen

    if len(rects) > 0:             #Wurde min. ein Objekt erkannt?  (Anzahl der Rechtecke,
also der erkannten Objekte)
        if motor_state != 1:
            motor_open()        #Dann Klappe auf
            motor_state = 1
            time.sleep(10)
    else:           #Andernfalls
        if motor_state != 0:
            motor_close()       #Klappe zu
            motor_state = 0


    cv2.imshow("Object Detector", image)        #Kamerabild mit eingesetzten
Rechtecken auf dem Monitor anzeigen
    cv2.waitKey(1)            #Das Programm NICHT pausieren (bei cv2.waitKey(0)
musste man zuerst eine Taste drücken, damit das Programm weiter läuft)

    rawCapture.truncate(0)  #Es kann passieren, dass während dem Erkennen von
Objekten die Kamera schon neue Bilder aufzeichnet. Diese werden hiermit
verworfen, damit das Programm immer das aktuelle Bild bearbeitet
```