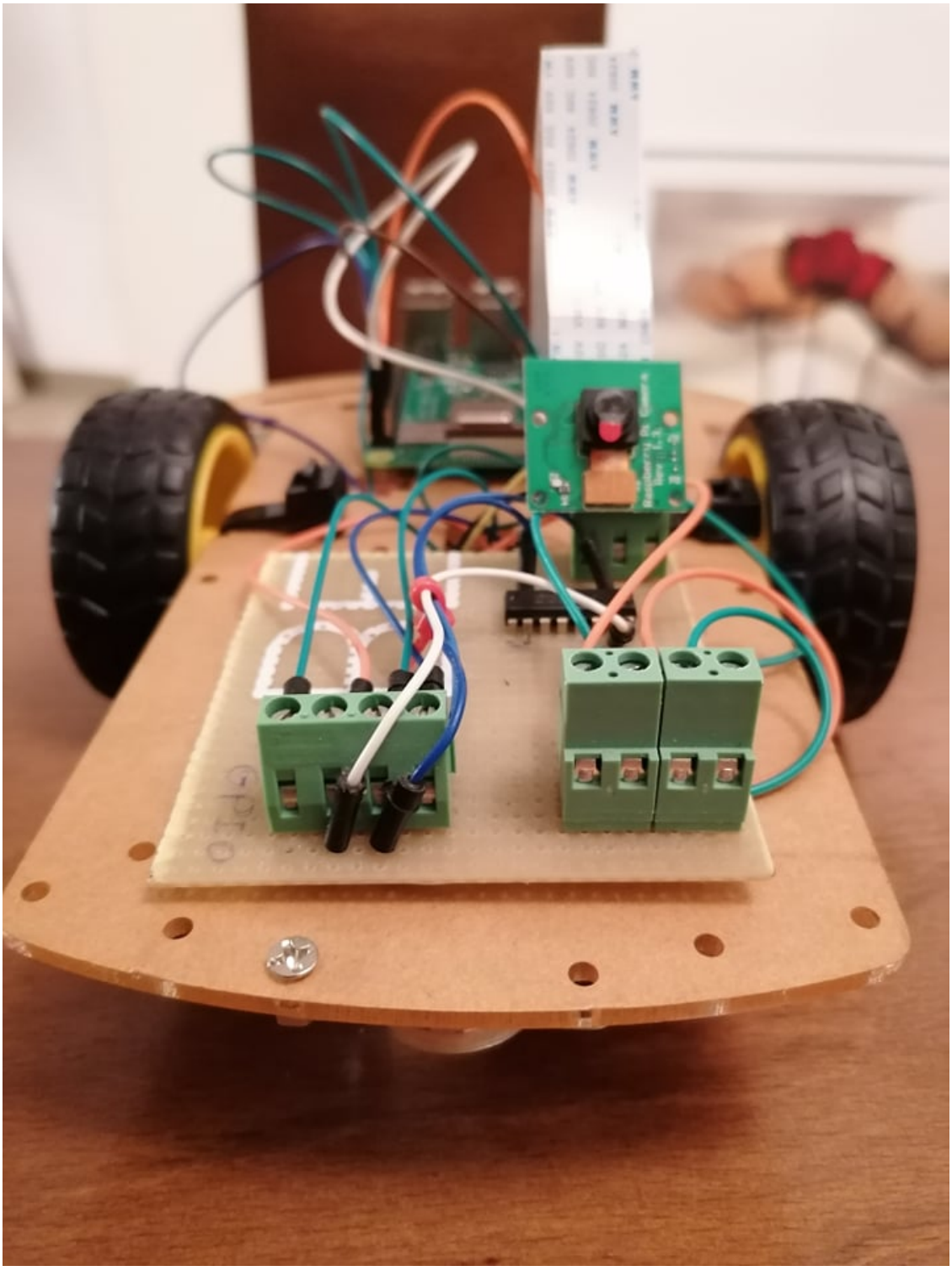
Ausarbeitung

June 30, 2020



IOT Raspberry PI und Python

Younes Labidi & Moez Rjiba

0.1 Inhalt :

- Framework installation
- Python motors code
- Kamera-Stream-Code mit Bewegungserkennung
- HTML code

0.2 Framework installation :

[15]:
```
! pip install Flask
```

```
Collecting Flask
  Using cached https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f4
0fba6a7841b47083260bf8bd8e737b0c6952df83f/Flask-1.1.2-py2.py3-none-any.whl
Collecting click>=5.1 (from Flask)
  Using cached https://files.pythonhosted.org/packages/d2/3d/fa76db83bf75c4f8d33
8c2fd15c8d33fdd7ad23a9b5e57eb6c5de26b430e/click-7.1.2-py2.py3-none-any.whl
Collecting Werkzeug>=0.15 (from Flask)
  Using cached https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef
8f1da638721e9da21e5bacee597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl
Collecting itsdangerous>=0.24 (from Flask)
  Using cached https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f
32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-
any.whl
Collecting Jinja2>=2.10.1 (from Flask)
  Using cached https://files.pythonhosted.org/packages/30/9e/f663a2aa66a09d83804
2ae1a2c5659828bb9b41ea3a6efa20a20fd92b121/Jinja2-2.11.2-py2.py3-none-any.whl
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->Flask)
  Using cached https://files.pythonhosted.org/packages/fb/40/f3adb7cf24a8012813c
5edb20329eb22d5d8e2a0ecf73d21d6b85865da11/MarkupSafe-1.1.1-cp27-cp27mu-
manylinux1_x86_64.whl
Installing collected packages: click, Werkzeug, itsdangerous, MarkupSafe,
Jinja2, Flask
Successfully installed Flask-1.1.2 Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1
click-7.1.2 itsdangerous-1.1.0
```

Flask ist ein Mikro-Web-Framework für unsere Web-Interface, mit dem wir das Auto steuern und den Live-Stream abrufen können.

[2]:
```
! pip install numpy
```

```
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: numpy in /usr/lib/python3/dist-packages (1.16.2)
```

Hier verwenden wir die numpy-Bibliothek zusätzlich mit opencv- und imutils-Bibliotheken, um die Kamera auf die Web-Interface zu streamen.

[4]:
```
! pip install opencv-python
```

```
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: opencv-python in
/home/pi/.local/lib/python3.7/site-packages (4.1.1.26)
Requirement already satisfied: numpy>=1.16.2 in /usr/lib/python3/dist-packages
(from opencv-python) (1.16.2)
```

[5]:
```
! pip install imutils
```

```
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: imutils in /home/pi/.local/lib/python3.7/site-
packages (0.5.3)
```

1 Python motors code :

Aufstellen von Pins an der Raspberry Pi zur Steuerung der Motoren.

[1]:
```
from flask import Flask, render_template, request, redirect, url_for,␣
 ↪make_response
import time
import RPi.GPIO as GPIO

mA1=18 #Motor1
mA2=23
mB1=24 #Motor2
mB2=25

GPIO.setmode(GPIO.BCM)    #Referring to the pins by the "Broadcom SOC channel"␣
 ↪number

GPIO.setup(mA1, GPIO.OUT) #Motors GPIO as OUTPUT
GPIO.setup(mA2, GPIO.OUT)
GPIO.setup(mB1, GPIO.OUT)
GPIO.setup(mB2, GPIO.OUT)

GPIO.output(mA1 , 0)      #Initial values 0
GPIO.output(mA2 , 0)
GPIO.output(mB1, 0)
GPIO.output(mB2, 0)

app = Flask(__name__)     #set up flask server
```

```
---------------------------------------------------------------------------

ModuleNotFoundError                       Traceback (most recent call last)

<ipython-input-1-ddaf1e51f673> in <module>
  1 from flask import Flask, render_template, request, redirect, url_for,␣
↪make_response
  2 import time
----> 3 import RPi.GPIO as GPIO
  4
  5 mA1=18 #Motor1


ModuleNotFoundError: No module named 'RPi'
```

Normaler Fehler, da er auf einer Raspberry-PI und nicht auf einem Desktop ausgeführt wird.

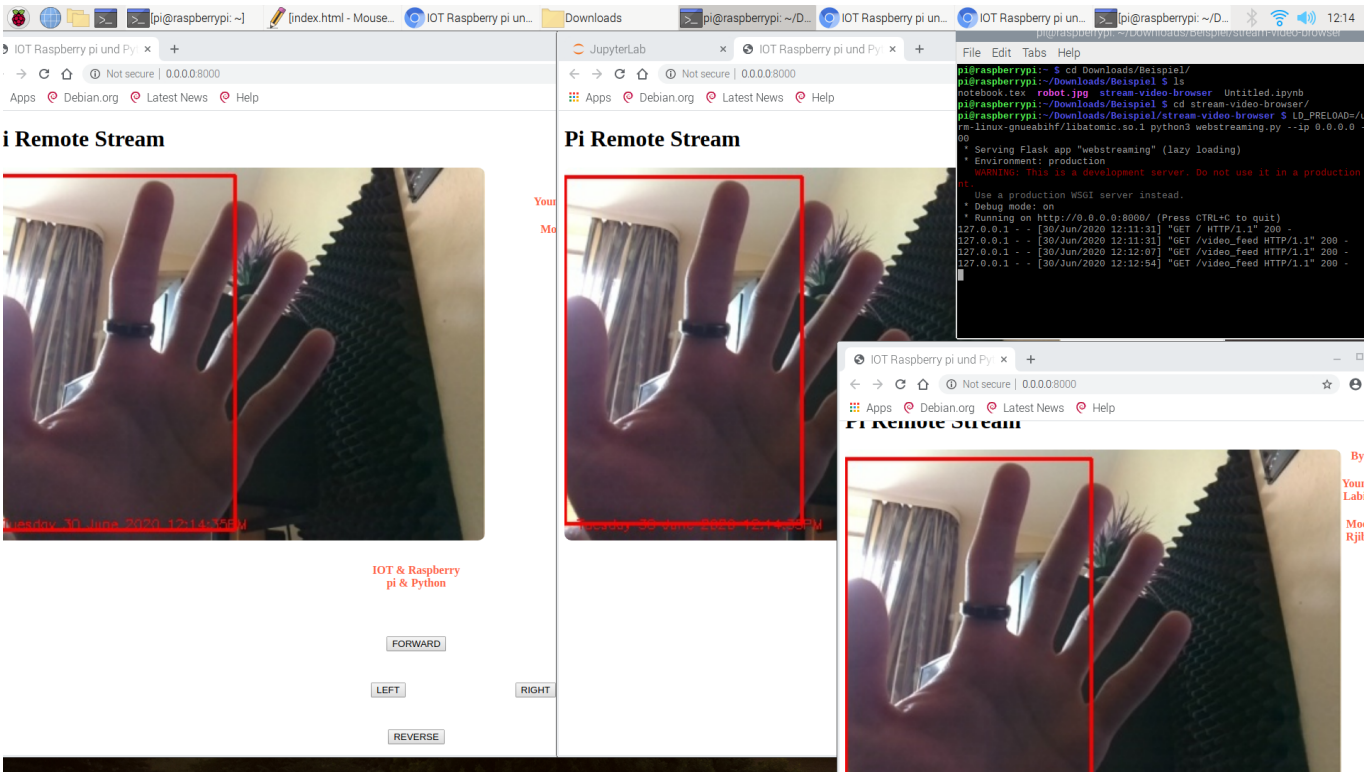
[ ]:
```
@app.route('/')
def index():
    return render_template('index.html')

#Recieve which pin to change from the button press on index.html
#Each button returns a number that triggers a command in this function
```

[ ]:
```
#Uses methods from motors.py to send commands to the GPIO to operate the motors
@app.route('/<changepin>', methods=['POST'])
def reroute(changepin):
    changePin = int(changepin) #cast changepin to an int
    if changePin == 1:
        print ("Vorne") # drive forward
        GPIO.output(mA1 , 0)
        GPIO.output(mA2 , 1)
        GPIO.output(mB1 , 1)
        GPIO.output(mB2 , 0)
    elif changePin == 2:
        print ("Links") # turn left
        GPIO.output(mA1 , 0)
        GPIO.output(mA2 , 0)
        GPIO.output(mB1 , 1)
        GPIO.output(mB2 , 0)
    elif changePin == 3:
        print ("Stop") # stop the car
        GPIO.output(mA1 , 0)
        GPIO.output(mA2 , 0)
        GPIO.output(mB1 , 0)
        GPIO.output(mB2 , 0)
```

```
    elif changePin == 4:
        print ("Rechts") # turn right
        GPIO.output(mA1 , 0)
        GPIO.output(mA2 , 1)
        GPIO.output(mB1 , 0)
        GPIO.output(mB2 , 0)
    else:
        print("Reverse") # drive backwards
        GPIO.output(mA1 , 1)
        GPIO.output(mA2 , 0)
        GPIO.output(mB1 , 0)
        GPIO.output(mB2 , 1)
    response = make_response(redirect(url_for('index')))
    return(response)
app.run(debug=True, host='0.0.0.0', port=8000) #set up the server in debug mode␣
 ↪to the port 8000
```

2 Kamera-Stream-Code mit Bewegungserkennung :



Bewegungserkennungsalgorithmus, der Numpy für die numerische Verarbeitung, Imutils für unsere Komfortfunktionen und cv2 für OpenCV-Bindungen verwendet.

Unser Bewegungsmelder-Algorithmus erkennt Bewegung durch eine Form der Hintergrundsubtraktion.

Der Subtraktionsalgorithmus arbeitet nach:

```
1- Akkumulieren des gewichteten Mittelwertes der vorherigen N Bilder.
2- Nehmen Sie den aktuellen Rahmen und subtrahieren Sie ihn vom gewichteten Durchschnitt
der Rahmen.
3- Schwellenwert für die Ausgabe der Subtraktion, um die Regionen mit erheblichen
Unterschieden in den Pixelwerten hervorzuheben
("weiß" für den Vordergrund und "schwarz" für den Hintergrund).
4- Anwendung grundlegender Bildverarbeitungstechniken wie Erosionen und Dilatationen,
um Rauschen zu entfernen.
5- Verwendung der Konturerkennung zur Extraktion der Regionen, die eine Bewegung enthalten.
```

Unsere Bewegungserkennungs-Implementierung wird innerhalb der SingleMotionDetector-Klasse leben, die in singleMotionDetector.py zu finden ist.

Sie wird als "Einzelbewegungsmelder" bezeichnet, da der Algorithmus selbst nur daran interessiert ist, die größte einzelne Bewegungsregion zu finden.

Wir können diese Methode leicht erweitern, um auch mehrere Bewegungsregionen zu behandeln.

[8]:
```
# import the necessary packages
import numpy as np
import imutils
import cv2
```

```
---------------------------------------------------------------------------

ImportError                               Traceback (most recent call last)

<ipython-input-8-ea8c565de04b> in <module>
  1 # import the necessary packages
  2 import numpy as np
----> 3 import imutils
  4 import cv2


~/.local/lib/python3.7/site-packages/imutils/__init__.py in <module>
  6
  7 # import the necessary packages
----> 8 from .convenience import translate
  9 from .convenience import rotate
 10 from .convenience import rotate_bound


~/.local/lib/python3.7/site-packages/imutils/convenience.py in <module>
```

```
      4 # import the necessary packages
      5 import numpy as np
----> 6 import cv2
      7 import sys
      8


~/.local/lib/python3.7/site-packages/cv2/__init__.py in <module>
      1 import importlib
      2
----> 3 from .cv2 import *
      4 from .data import *
      5


ImportError: /home/pi/.local/lib/python3.7/site-packages/cv2/cv2.
↪cpython-37m-arm-linux-gnueabihf.so: undefined symbol: __atomic_fetch_add_8
```

Wenn Sie diesen Importfehler erhalten, führen Sie Ihr Programm einfach mit diesem Befehl aus:

LD_PRELOAD=/usr/lib/arm-linux-gnueabihf/libatomic.so.1 python3 webstreaming.py –ip 0.0.0.0 –port 8000

[9]:
```python
class SingleMotionDetector:
    def __init__(self, accumWeight=0.5):
        # store the accumulated weight factor
        self.accumWeight = accumWeight

        # initialize the background model
        self.bg = None
```

Je größer accumWeight ist, desto weniger wird der Hintergrund (bg) bei der Akkumulation des gewichteten Durchschnitts berücksichtigt.

[ ]:
```python
    def update(self, image):
        # if the background model is None, initialize it
        if self.bg is None:
            self.bg = image.copy().astype("float")
            return

        # update the background model by accumulating the weighted
        # average
        cv2.accumulateWeighted(image, self.bg, self.accumWeight)
```

Definiert eine Aktualisierungsmethode, die ein Eingabefenster akzeptiert und den gewichteten Durchschnitt berechnet.

[ ]:
```
    def detect(self, image, tVal=25):
        # compute the absolute difference between the background model
        # and the image passed in, then threshold the delta image
        delta = cv2.absdiff(self.bg.astype("uint8"), image)
        thresh = cv2.threshold(delta, tVal, 255, cv2.THRESH_BINARY)[1]

        # perform a series of erosions and dilations to remove small
        # blobs
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)
```

Vor unserem Hintergrund (bg) können wir nun die Bewegungserkennung mittels Detektionsmethode anwenden.

Die Methode erfordert einen einzigen Parameter mit einem optionalen Parameter:

```
-image: Eingabebild
-tval : Schwellenwert
```

[ ]:
```
        # find contours in the thresholded image and initialize the
        # minimum and maximum bounding box regions for motion
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        (minX, minY) = (np.inf, np.inf)
        (maxX, maxY) = (-np.inf, -np.inf)
```

Anwendung der Konturerkennung zum Extrahieren beliebiger Bewegungen.

[ ]:
```
 # if no contours were found, return None
        if len(cnts) == 0:
            return None

        # otherwise, loop over the contours
        for c in cnts:
            # compute the bounding box of the contour and use it to
            # update the minimum and maximum bounding box regions
            (x, y, w, h) = cv2.boundingRect(c)
            (minX, minY) = (min(minX, x), min(minY, y))
            (maxX, maxY) = (max(maxX, x + w), max(maxY, y + h))

        # otherwise, return a tuple of the thresholded image along
        # with bounding box
        return (thresh, (minX, minY, maxX, maxY))
```

Wir prüfen, ob unsere Konturenliste leer ist.

Wenn das der Fall ist, dann wurde keine Bewegung im Rahmen gefunden, und wir können sie getrost ignorieren.

3 Init

[ ]:
```
# import the necessary packages
from .singlemotiondetector import SingleMotionDetector
```

4 Kombination von OpenCV mit Flask

Import unseres Bewegungserkennungsalgorithmus, Flask zum Rendern unserer index.html-Vorlage und Threading, damit wir Gleichzeitigkeit unterstützen können.

[ ]:
```
# USAGE
# python webstreaming.py --ip 0.0.0.0 --port 8000

# import the necessary packages
from pyimagesearch.motion_detection import SingleMotionDetector
from imutils.video import VideoStream
from flask import Response
from flask import Flask
from flask import render_template
import threading
import argparse
import datetime
import imutils
import time
import cv2
```

Kombination von OpenCV und Flask zur Bereitstellung von Einzelbildern aus einem Videostream an einen Webbrowser.

[ ]:
```
# initialize the output frame and a lock used to ensure thread-safe
# exchanges of the output frames (useful for multiple browsers/tabs
# are viewing tthe stream)
outputFrame = None
lock = threading.Lock()

# initialize a flask object
app = Flask(__name__)

# initialize the video stream and allow the camera sensor to
# warmup
vs = VideoStream(usePiCamera=1).start()
time.sleep(2.0)
```

Initialisierung von outputFrame, derselbe Rahmen, der auch den Kunden zur Verfügung gestellt wird.

Erstellen einer Sperre, um thread-sicheres Verhalten bei der Aktualisierung des outputFrame zu gewährleisten.

Initialisierung der Flask-Anwendung beim Zugriff auf den Videostream.

[ ]:
```
@app.route("/")
def index():
    # return the rendered template
    return render_template("index.html")
```

Rendern der index.html-Vorlage und Bereitstellen des Ausgabe-Videostroms.

[ ]:
```
def detect_motion(frameCount):
    # grab global references to the video stream, output frame, and
    # lock variables
    global vs, outputFrame, lock

    # initialize the motion detector and the total number of frames
    # read thus far
    md = SingleMotionDetector(accumWeight=0.1)
    total = 0
```

Diese Funktion wird verantwortlich sein:

-Schleifen über Einzelbilder aus dem Videostrom
-Anwendung der Bewegungserkennung
-Zeichnungsergebnisse auf dem Ausgabeframe

[ ]:
```
    # loop over frames from the video stream
    while True:
        # read the next frame from the video stream, resize it,
        # convert the frame to grayscale, and blur it
        frame = vs.read()
        frame = imutils.resize(frame, width=400)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (7, 7), 0)

        # grab the current timestamp and draw it on the frame
        timestamp = datetime.datetime.now()
        cv2.putText(frame, timestamp.strftime(
            "%A %d %B %Y %I:%M:%S%p"), (10, frame.shape[0] - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)
```

Wir können jetzt eine Hintergrundsubtraktion durchführen und mit dem Looping über die Bilder der Kamera beginnen.

```
[ ]:        # if the total number of frames has reached a sufficient
            # number to construct a reasonable background model, then
            # continue to process the frame
            if total > frameCount:
                # detect motion in the image
                motion = md.detect(gray)

                # cehck to see if motion was found in the frame
                if motion is not None:
                    # unpack the tuple and draw the box surrounding the
                    # "motion area" on the output frame
                    (thresh, (minX, minY, maxX, maxY)) = motion
                    cv2.rectangle(frame, (minX, minY), (maxX, maxY),
                        (0, 0, 255), 2)

            # update the background model and increment the total number
            # of frames read thus far
            md.update(gray)
            total += 1

            # acquire the lock, set the output frame, and release the
            # lock
            with lock:
                outputFrame = frame.copy()
```

Mit der Endkontrolle können wir nun die Bewegungserkennung durchführen.

```
[ ]: def generate():
        # grab global references to the output frame and lock variables
        global outputFrame, lock

        # loop over frames from the output stream
        while True:
            # wait until the lock is acquired
            with lock:
                # check if the output frame is available, otherwise skip
                # the iteration of the loop
                if outputFrame is None:
                    continue

                # encode the frame in JPEG format
                (flag, encodedImage) = cv2.imencode(".jpg", outputFrame)

                # ensure the frame was successfully encoded
                if not flag:
                    continue
```

```
        # yield the output frame in the byte format
        yield(b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' +
            bytearray(encodedImage) + b'\r\n')
```

Die Funktion generate, ist ein Generator, der verwendet wird, um das OutputFrame als JPEG-Daten zu kodieren.

[ ]:
```
@app.route("/video_feed")
def video_feed():
    # return the response generated along with the specific media
    # type (mime type)
    return Response(generate(),
        mimetype = "multipart/x-mixed-replace; boundary=frame")
```

Die app.route-Signatur teilt Flask mit, dass es sich bei dieser Funktion um einen URL-Endpunkt handelt und dass die Daten von http://your_ip_address/video_feed/ serviert werden.

[ ]:
```
# check to see if this is the main thread of execution
if __name__ == '__main__':
    # construct the argument parser and parse command line arguments
    ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--ip", type=str, required=True,
        help="ip address of the device")
    ap.add_argument("-o", "--port", type=int, required=True,
        help="ephemeral port number of the server (1024 to 65535)")
    ap.add_argument("-f", "--frame-count", type=int, default=32,
        help="# of frames used to construct the background model")
    args = vars(ap.parse_args())

    # start a thread that will perform motion detection
    t = threading.Thread(target=detect_motion, args=(
        args["frame_count"],))
    t.daemon = True
    t.start()

    # start the flask app
    app.run(host=args["ip"], port=args["port"], debug=True,
        threaded=True, use_reloader=False)

# release the video stream pointer
vs.stop()
```

Diese letzte Funktion behandelt das Parsen von Befehlszeilenargumenten und das Starten der Flask-Anwendung.

5 HTML Code

Wie wir in den vorherigen Codes (Motoren, Stream) gesehen haben, rendern wir eine HTML-Vorlage mit dem Namen index.html.

Die Vorlage selbst wird durch das Web-Framework Flask gefüllt und dann dem Web-Server zur Verfügung gestellt.

img src="{{{ url_for('video_feed') }}": Flask wird angewiesen, die URL unseres Video-Feeds dynamisch zu rendern.

button id="FWD" class="robot">FORWARD</button: Einrichten der Steuerknöpfe.

[ ]:
```
<!DOCTYPE html>
<html>
            <head>
                        <title>IOT Raspberry pi und Python</title>
                        <meta name="viewport" content="width=device-width,␣
 ↪initial-scale=1">
            </head>
<style>
table ,td, tr {
                        width: 30%;
}
</style>
            <body>
                        <h1>Pi Remote Stream</h1>
                        <img src="{{ url_for('video_feed') }}" style="float:left;
 ↪ width:100%; max-width: 700px; max-height:600px; border-radius:10px"></img>
                <table style="float:right; width:100%; max-width: 400px; height:
 ↪300px;">
                 <tr style="text-align:center">
                    <td >
                         <h4 style="text-align: center; color:Tomato">IOT &␣
 ↪Raspberry pi & Python</h4>
                             </td>
                 <tr>
                 <tr>
                             <td style="text-align: center">
                                    <form action="/1" method="POST">
                                 <button id="FWD" class="robot">FORWARD</button>
                                 </br>
                               </form>
                             </td>
                 <tr>
             <tr>
                             <td style="text-align: left">
                    <form action="/2" method="POST">
                                     <button id="LFT" class="robot">LEFT</button>
```

```
                                    </br>
                                </form>
                            </td>

                            <td style="text-align: right">
                                    <form action="/4" method="POST">
                                     <button id="RGT" class="robot">RIGHT</
 ↪button>
                                 </br>
                                    </form>
                            </td>
                            <td >
                                    <form action="/3" method="POST">
                                     <button id="STP" class="robot">STOP</button>
                                      </br>
                                    </form>
                      </td>
                    <tr>
                    <tr>
                                <td style="text-align: center">
                                    <form action="/5" method="POST">
                                     <button id="REV" class="robot">REVERSE</
 ↪button>
                                      </br>
                                </form>
                            </td>
                    </tr>

                         </table>
<h4 style="text-align: center; color:Tomato">By:</h4>
<h4 style="text-align: center; color:Tomato; ">Younes Labidi</h4>
<h4 style="text-align: center; color:Tomato; ">Moez Rjiba</h4>

        </body>
</html>
```

6 Fazit

Das Projekt gibt Ihnen eine Vorstellung davon, was wir mit Raspberry Pi und all den Frameworks, die Python liefern kann, tun können, um ein IOT-System zu erstellen.

Das Projekt selbst kann durch die Implementierung von Sensoren optimiert und verbessert werden und es autonom machen. Wir können den Stream auch von überall her verfügbar machen, indem wir ein VPN oder eine API erstellen.