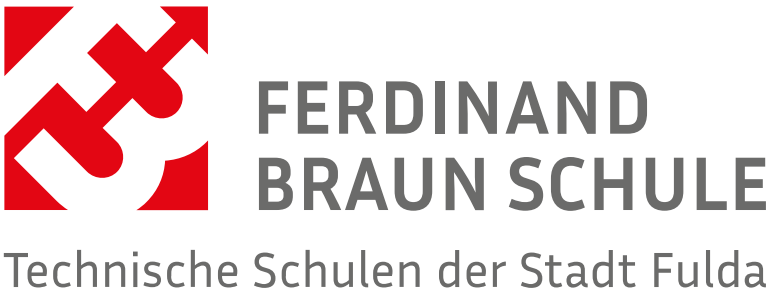
Dokumentation RGB-Sensor

Julius Hahl, Maximilian Trautwein und Sebastian Köhler, 11BG1



6. April 2022

Inhaltsverzeichnis

I Dokumentation **2**

1 Farberkennung **3**
1.1 Aufsetzen und Auslesung 4
1.1.1 Raspberry Pi aufsetzen: 4
1.1.2 Client-Seite (Computer oder Laptop) 4

II Code **5**

2 Quellcode **6**
2.1 Main.py 7
2.2 client.py 9

Kapitel 1

Farberkennung

1.1 Aufsetzen und Auslesung

1.1.1 Raspberry Pi aufsetzen:

Um den Raspberry Pi aufzusetzen, muss man diesen zuerst über ein HDMI-Kabel oder über SSH verbinden, sich einloggen (Benutzername: pi, Passwort:ilovecolors) und mit einem Netzwerk verbinden, wo der Laptop/Computer auch angeschlossen ist. Daraufhin kann man die Ausführung starten, indem man in der Console des Raspberry Pi's
"python main.py *hier die lokale IP-Adresse eingeben*"
schreibt und auf 'Enter' drückt.

```
$ python main.py *Adresse*
```

Jetzt sollte der Raspberry Pi aufgesetzt sein.

1.1.2 Client-Seite (Computer oder Laptop)

Wenn der PC im gleichen Netzwerk ist und sie den ersten Schritt vollendet haben, können sie die Python-Datei 'client.py' ausführen und dort die lokale IP-Adresse des Pi's eingeben und sich verbinden.

```
>python client.py
```

!!!Achtung, es ist wichtig, dass sie Python 3 auf ihrem PATH installiert haben und in dem Projektordner sind, damit die Datei ausgeführt werden kann!!!
Mit dem Knopf 'Verbinden zum Server' verbinden sie sich mit dem Raspberry Pi. Der Knopf 'Farberkennung' schickt eine Anfrage an den Raspberry Pi, der dann die Farbe erkennt und diese dann zurückschickt. Die erkannte Farbe sieht man dann in der Liste.

Kapitel 2

Quellcode

2.1 Main.py

Listing 2.1: Python-Code

```
1   import cv2 as cv
2   import socket
3   import sys
4
5   from null_preview import *
6   from picamera2 import *
7
8   #Setup des Sockets und der Kamera
9   currentColor = ""
10  lastColor = ""
11  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12  server_address = (str(sys.argv[1]), 18769)
13  print('starting up on {} port {}'.format(*server_address))
14  sock.bind(server_address)
15  picam2 = Picamera2()
16  preview = NullPreview(picam2)
17  picam2.configure(picam2.preview_configuration(main={"size":(640, 480)}))
18  picam2.start()
19  sock.listen(1000)
20  connection, client_address = sock.accept()
21
22
23  #Bild der Kamera wird als Numpy Array ausgelesen
24  def evaluate_current_frame():
25  img = picam2.capture_array()
26
27  img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
28
29  Z = img.reshape((-1,3))
30  # Konventiert zu np.float32
31  Z = np.float32(Z)
32  # Definiton der Kriterien der Farbdominanz, Anzahl an dominanten Farben(K
        ) und anschliessend wird der KMeans Algorithmus angewendet
33  criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 10, 1.0)
34  K = 1
35  ret,label,center=cv.kmeans(Z,K,None,criteria,10,cv.KMEANS_RANDOM_CENTERS)
36  # Zurueck zu unsigned int, damit das Buffer mit dem Bild wieder zu der
        urspruenglichen Form zurueckkehrt
37  center = np.uint8(center)
38  res = center[label.flatten()]
39  res2 = res.reshape((img.shape))
40
41
42  #Trennt Farbkanaele
43  (b, g, r) = cv.split(res2)
44
45  b_mean = np.mean(b)
46  g_mean = np.mean(g)
47  r_mean = np.mean(r)
48
49  # Bestimmt die prominenteste Farbe und setzt die Variable
50  if (b_mean > g_mean and b_mean > r_mean):
51  currentColor = "blue"
52  elif (g_mean > r_mean and g_mean > b_mean):
53  currentColor = "green"
54  else:
55  currentColor = "red"
56
57  #Sendet String an den Client zurueck
58
59  message = currentColor.encode()
60  connection.sendall(message)
61
62  def close_socket():
63  sock.close()
64
65  while True:
66  try:
67  data = connection.recv(16)
```

```
68      dataBuffer = data.decode('utf-8')
69      if(dataBuffer == "getcolor"):
70      evaluate_current_frame()
71      elif(dataBuffer == "closesocket"):
72      close_socket()
73
74
75      except OSError:
76      print("STOPPED")
77      sock.close()
78      break
79
80      except KeyboardInterrupt:
81      print("STOPPED")
82      sock.close()
83      break
```

2.2 client.py

Listing 2.2: Python-Code

```
import socket
import sys
import tkinter as tk

ip_was_false = True

i = 1
#Socket definieren
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

getColorFuncName = "getcolor"
sockCloseFuncName = "closesocket"

#Funktion zum Verbinden mit dem Raspberry Pi
def connect_to_server():
    try:
        print(ip_var.get())
        server_address = (ip_var.get(), 18769)
        print('connecting to {} port {}'.format(*server_address))
        sock.connect(server_address)
    except Exception:
        global ip_was_false
        if (ip_was_false == True):
            label2 = tk.Label(root, text = "Falsche IP-Adresse!", fg = '#ff0000')
            label2.pack()
            ip_was_false = False

#Abfrage der derzeitigen Farbe
def request_color():
    message = getColorFuncName.encode()
    sock.sendall(message)

    data = sock.recv(16)
    global i
    lb1.insert(i, data.decode('utf-8'))
    i = i + 1

#Schliessen des Sockets nach Schliessen des Programms
def close_socket():
    message = sockCloseFuncName.encode()
    sock.sendall(message)

#Definition des Fensters und des Inhalts
root = tk.Tk()
root.geometry("250x170")

ip_var = tk.StringVar()

label1 = tk.Label(root, text="RGB-Sensor-System")
label1.pack()

ip_feld = tk.Entry(root, textvariable = ip_var)
ip_feld.pack()

schaltf1 = tk.Button(root, text="Verbinde zum Server", command=connect_to_server)
schaltf1.pack()

schaltf2 = tk.Button(root, text="Farberkennung", command=request_color)
schaltf2.pack()

lb1 = tk.Listbox()

root.mainloop()
```