

Dokumentation des RBG-Sensors

Fach: ITEC

Julius Hahl, Maximilian Trautwein und Sebastian Köhler, 11BG1



**FERDINAND
BRAUN SCHULE**

Technische Schulen der Stadt Fulda

7. April 2022

Inhaltsverzeichnis

1	Zweck und Einsatzmöglichkeiten	3
1.1	Zweck	3
1.2	Einsatzmöglichkeiten	3
2	Technische Daten	4
2.1	Hardware	4
2.2	Software	4
3	Funktionsweise und Beispiel	5
3.1	Funktionsweise(Quellcode)	5
3.2	Beispiel	8
3.2.1	Raspberry Pi aufsetzen:	8
3.2.2	Client-Seite (Computer oder Laptop)	8
4	Projektphase	9
5	Quellen	9

1 Zweck und Einsatzmöglichkeiten

1.1 Zweck

Der RGB-Sensor soll Farben erkennen und diese über das lokale/globale Netzwerk schicken. Geräte, die diese Farbinformation brauchen, können eine Funktion aufrufen (auch über ein Event) und somit die Farbe für den derzeitigen Farbwürfel zurückgeben, woraufhin der Roboter seine Tätigkeit fortsetzen kann (z.B.: Würfel in das richtige Lager legen).

1.2 Einsatzmöglichkeiten

Wie im vorherigen Teil schon erwähnt, kann der RGB-Sensor dazu verwendet werden, um einem Roboter beim Einordnen verschiedenfarbiger Gegenstände einzuordnen. Der Sensor fungiert als Auge:



Abbildung 1: Beispiel eines Roboterarms beim Sortieren.

2 Technische Daten

2.1 Hardware

- Raspberry Pi 4 B 4GB RAM
- Joy-IT Armor Case
- RPi4 Kamera
- Standard Raspberry Pi 4 Netzteil

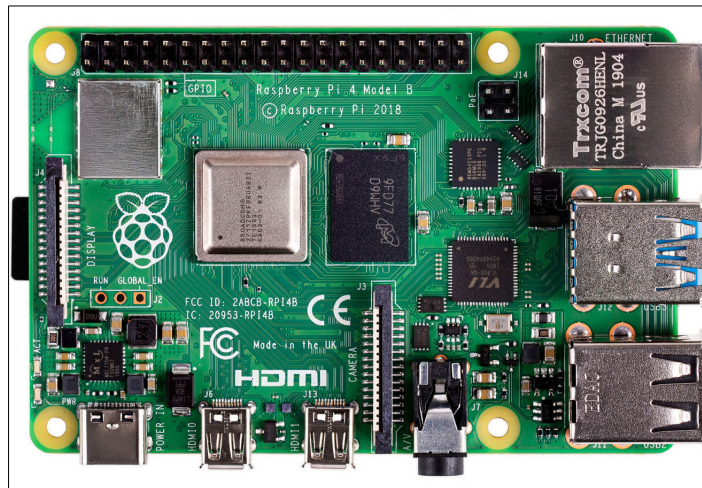


Abbildung 2: Das Herzstück des Projekts.

2.2 Software

- OpenCV 4.5.5 (Quellcode für 64-Bit kompiliert, <https://opencv.org/>)
- libcamera Python-Sprachanbindung (nur auf 64-Bit Systemen ausführbar)
- Python 3

3 Funktionsweise und Beispiel

3.1 Funktionsweise(Quellcode)

In dieser Sektion wird der Quellcode nochmals kommentiert dargelegt.

Listing 1: main.py

```
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     # Konvertiert zu np.float32
31     Z = np.float32(Z)
32     # Definition der Kriterien der Farbdominanz, Anzahl an dominanten Farben(K) und anschliessend
33     # wird der KMeans Algorithmus angewendet
34     criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 10, 1.0)
35     K = 1
36     ret, label, center=cv.kmeans(Z,K,None,criteria,10,cv.KMEANS_RANDOM_CENTERS)
37     # Zurueck zu unsigned int, damit das Buffer mit dem Bild wieder zu der urspruenglichen Form
38     # zurueckkehrt
39     center = np.uint8(center)
40     res = center[label.flatten()]
41     res2 = res.reshape((img.shape))
42
43 #Trennt Farbkanaele
44 (b, g, r) = cv.split(res2)
45
46 b_mean = np.mean(b)
47 g_mean = np.mean(g)
48 r_mean = np.mean(r)
49
50 # Bestimmt die prominenteste Farbe und setzt die Variable
51 if (b_mean > g_mean and b_mean > r_mean):
52     currentColor = "blue"
53 elif (g_mean > r_mean and g_mean > b_mean):
54     currentColor = "green"
55 else:
56     currentColor = "red"
57
58 #Sendet String an den Client zurueck
59
60 message = currentColor.encode()
61 connection.sendall(message)
62
63 def close_socket():
64     sock.close()
```

```
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
```

Listing 2: client.py

```

1  import socket
2  import sys
3  import tkinter as tk
4
5  ip_was_false = True
6
7  i = 1
8  #Socket definieren
9  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10
11  getColorFuncName = "getcolor"
12  sockCloseFuncName = "closesocket"
13
14  #Funktion zum Verbinden mit dem Raspberry Pi
15  def connect_to_server():
16      try:
17          print(ip_var.get())
18          server_address = (ip_var.get(), 18769)
19          print('connecting to {} port {}'.format(*server_address))
20          sock.connect(server_address)
21      except Exception:
22          global ip_was_false
23          if (ip_was_false == True):
24              label2 = tk.Label(root, text = "Falsche IP-Adresse!", fg = 'ff0000')
25              label2.pack()
26              ip_was_false = False
27
28  #Abfrage der derzeitigen Farbe
29  def request_color():
30      message = getColorFuncName.encode()
31      sock.sendall(message)
32
33      data = sock.recv(16)
34      global i
35      lbl.insert(i, data.decode('utf-8'))
36      i = i + 1
37
38  #Schliessen des Sockets nach Schliessen des Programms
39  def close_socket():
40      message = sockCloseFuncName.encode()
41      sock.sendall(message)
42
43  #Definition des Fensters und des Inhalts
44  root = tk.Tk()
45  root.geometry("250x170")
46
47  ip_var = tk.StringVar()
48
49  label1 = tk.Label(root, text="RGB-Sensor-System")
50  label1.pack()
51
52  ip_feld = tk.Entry(root, textvariable = ip_var)
53  ip_feld.pack()
54
55  schaltf1 = tk.Button(root, text="Verbinde zum Server", command=connect_to_server)
56  schaltf1.pack()
57
58  schaltf2 = tk.Button(root, text="Farberkennung", command=request_color)
59  schaltf2.pack()
60
61  lbl = tk.Listbox()
62
63  root.mainloop()

```

3.2 Beispiel

3.2.1 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.

3.2.2 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.

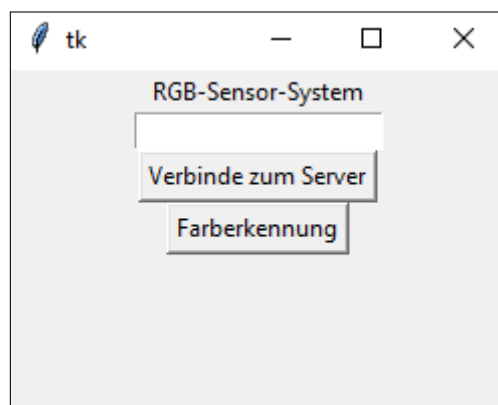


Abbildung 3: GUI des Programms

4 Projektphase

Wir würden die Projektphase als etwas holprig bezeichnen. Wir sind die einzige Gruppe, die sich für ein 64-Bit-System entschieden haben, um die Effizienz zu steigern. Jedoch mussten wir die neue libcamera-Bibliothek verwenden, sowie OpenCV kompilieren, da die vor-kompilierten Versionen nur für 32-Bit-Systeme vorgesehen sind.

5 Quellen

Roboter: <https://www.youtube.com/watch?v=AWFFJYYCF44>

Raspberry Pi 4: <https://www.amazon.de/Raspberry-Pi-ARM-Cortex-A72-Bluetooth-Micro-HDMI/dp/B07TC2BK1X>