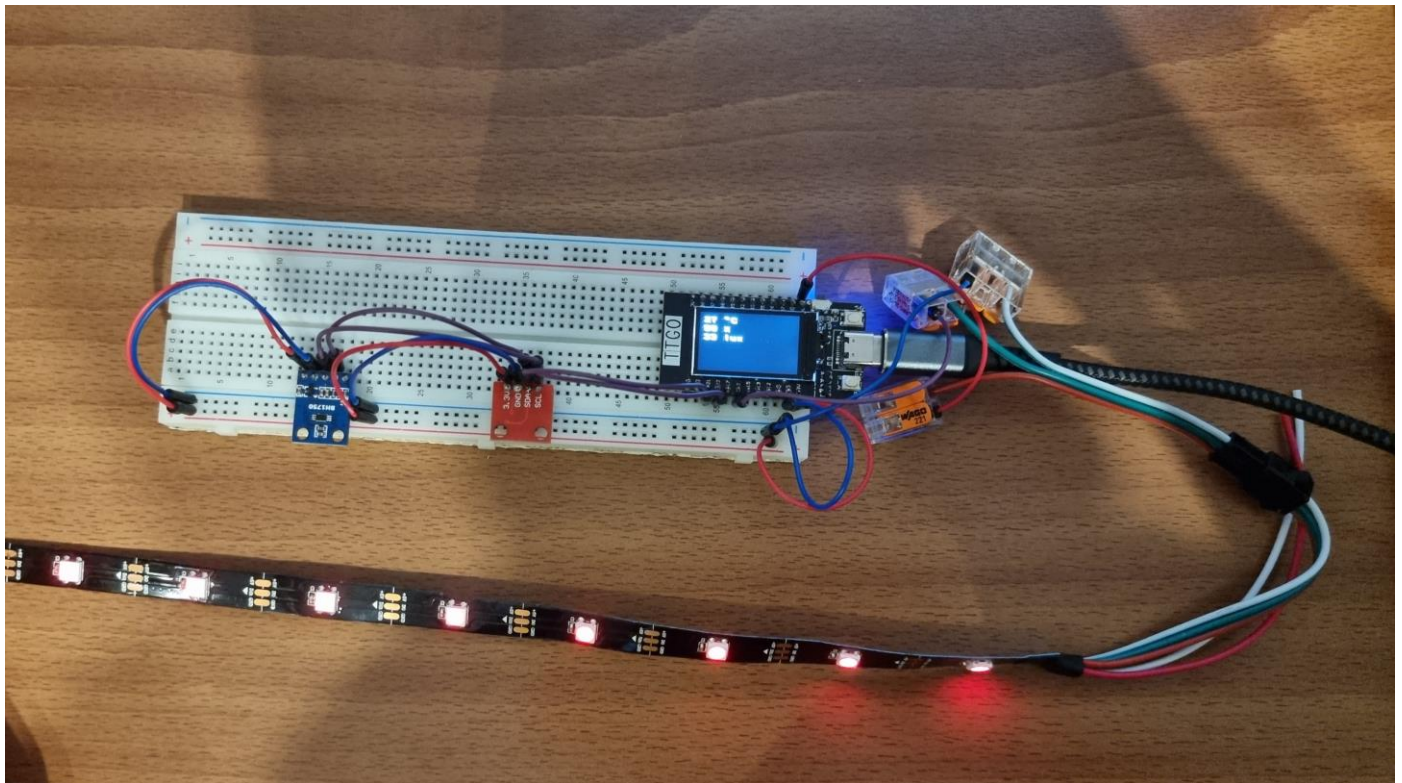


# WETTERSTATION

## MICROCONTROLLER PROJEKT



Matthias Grafe

ETS2021

20.06.2022

---

# Inhaltsverzeichnis

Kundenauftrag	1
Realisierung	1
Verwendete Hardware:	1
Blockschaltbild	2
Steckplan	3
Code	4
Node-RED	9
Node-RED UI	10
MySQL	11
Grafana	12
Fazit	13
Quellen	14

---

# WETTERSTATION

## Kundenauftrag

Die Meyer Holz GmbH hat ein modernes Designgehäuse aus Holz erstellt und möchte in das Gehäuse eine Wetterstation integrieren. Das Gehäuse soll in Massenproduktion gehen und an den Kunden verkauft werden.

Die MG GmbH bekommt den Auftrag die Wetterstation zu planen und zu erstellen.

## Realisierung

Luftfeuchte und Temperatur sollen für ein Display (ESP32 OLED) gemessen werden.

Zusätzlich soll die Luftfeuchte in drei Abstufungen mit LED's angezeigt werden. Rot für schlecht, gelb für grenzwertig und grün für gut.

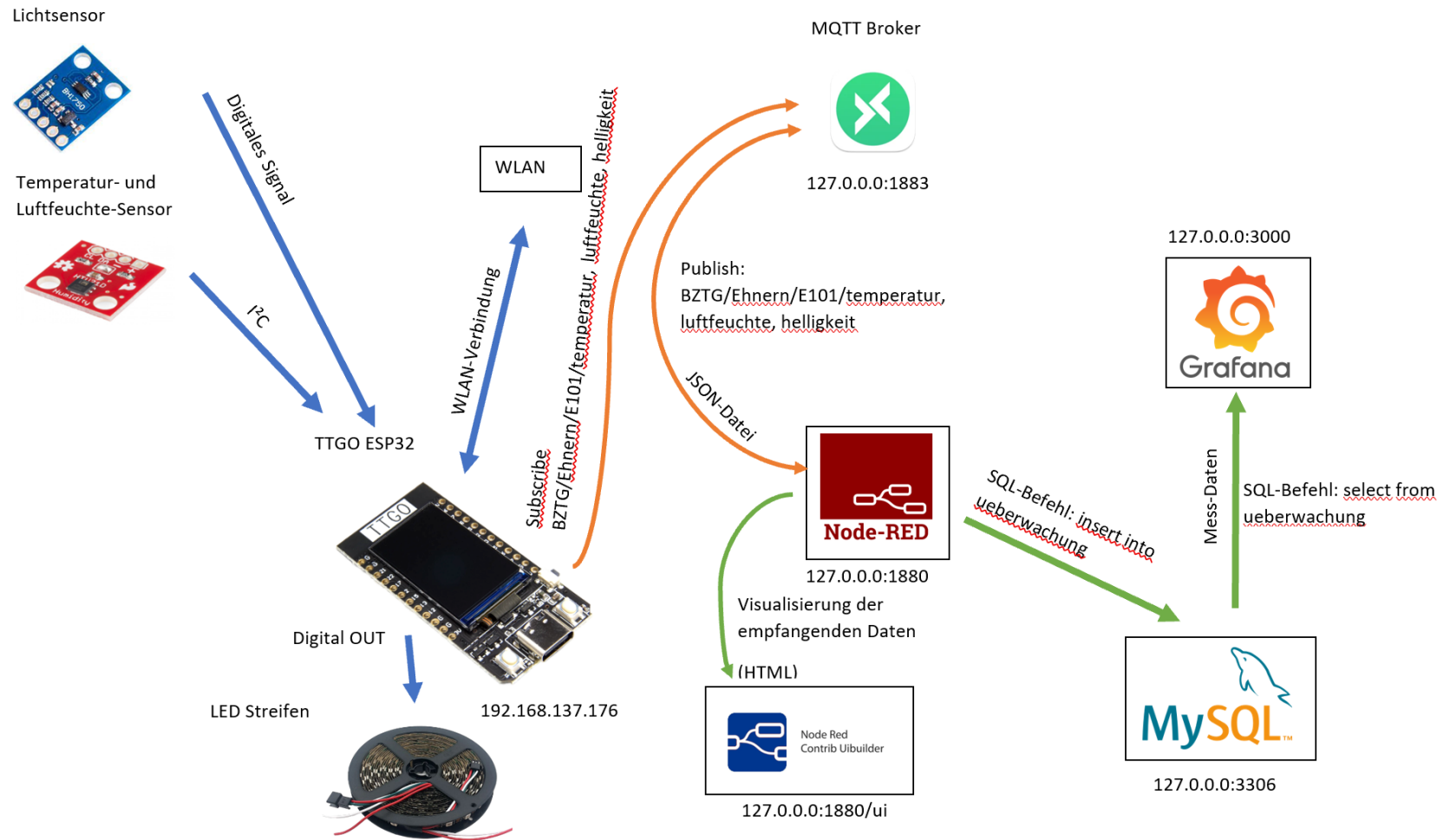
Ein BH1750 Sensor misst die Umgebungshelligkeit und je nach Lichteinfall, sollen die LED's mit voller Stärke oder gedimmt leuchten.

### Verwendete Hardware:

- Microcontroller: ESP32 (TTGO)
- Sensoren:
  - o HTU2X, I2C Temperatur- und Luftfeuchtigkeits-Sensor
  - o BH1750, Helligkeitssensor
- Aktoren:
  - o LED Stripe (WS2812)
  - o Display (ESP32)

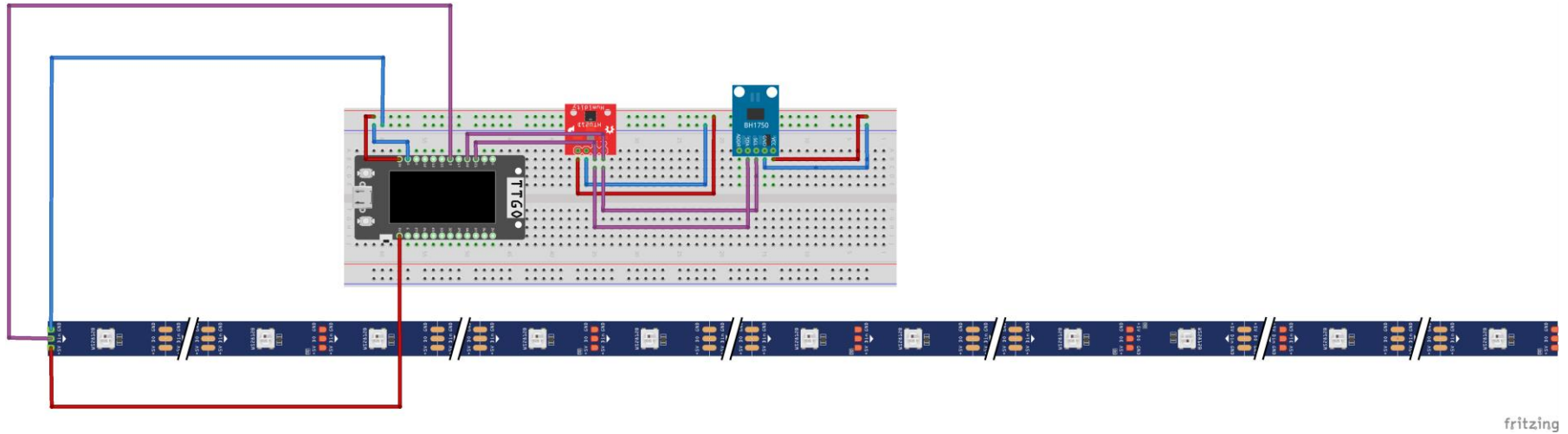
# WETTERSTATION

## Blockschaltbild



# WETTERSTATION

## Steckplan



# WETTERSTATION

## Code

```
hauptprogramm.py > ...
1  # Matthias Grafe
2  # ETS2021
3  # µController Projekt
4  # 20.06.2022
5  # Versionsnummer 2.5 vom 20.06.2022
6
7
8
9  """
10  Was macht das Programm?
11  Ein HTU2X Sensor misst die Temperatur und Luftfeuchtigkeit. Ein BH1750 Helligkeitssensor misst die Helligkeit.
12  Die drei Werte werden auf dem TFT Display vom ESP32 ausgegeben.
13  Außerdem leuchten die LED's von einem LED Stripe abhängig von der Luftfeuchtigkeit:
14  rot bei schlechten Werten, gelb bei erhöhten Werten und grün bei guten Werten
15  Die LEDs leuchten in Abhängigkeit von der Helligkeit entweder mit voller Lichtstärke oder gedimmt.
16  Die drei Werte werden außerdem als JSON File an MQTTX gesendet und von dort aus weiter verarbeitet.
17  """
18
19
20  # Verwendete Hardware:
21  """
22  |   ESP32
23  |   HTU2X
24  |   BH1750
25  |   WS2812b
26  """
27
28  # BUS-Systeme
29  """
30  i2c:
31  |   BH1750  (Lichtsstärke)
32  |   HTU21   (Temperatur / Luftfeuchtigkeit)
33  -----
34  SPI:
35  |   ST7789  (Onboard Display)
36  -----
37  neopixel:
38  |   WS2812b
39  """
40
```

# WETTERSTATION

```
41 # Spannungen:
42 """
43     BH1750 an 3,3V DC
44     HTU2X an 3,3V DC
45     WS2812b an 5V DC
46 """
47
48 #####
49 # Die verwendeten Bibliotheken stehen in der Readme #
50 #####
51
52
53
54
55
56 #-----Bibliotheken aufrufen-----
57
58 import json                                #
59 from main import MQTT_TOPIC, mqtt_MG      #
60 from time import sleep, time              #
61 from neopixel import NeoPixel             #
62 from htu2x import HTU21D                   #
63 from bh1750 import BH1750                 #
64 from machine import Pin, SoftSPI, SoftI2C  # Pin(BMP180 und TFT), SoftSPI(TFT), SoftI2C(BMP180)
65 import st7789py as st7789                 # TFT-Display
66 from fonts import vga2_16x16 as font      # Schriftart laden
67
68
69 i2c = SoftI2C(scl=Pin(22), sda=Pin(21))     # Pins für den i2c Bus
70 htu = HTU21D(22,21)                       # i2c Bus Anschluss Temperatur und Luftfeuchte
71 bh = BH1750(i2c)                          # i2c Bus Lichtsensor
72
73 #-----Display Initialisierung-----
74 spi = SoftSPI(                             # Objekt spi instanzieren
75     baudrate=2000000,                     # TFT Kommunikationsgeschwindigkeit
76     polarity=1,                          #
77     phase=0,                             #
78     sck=Pin(18),                          #
79     mosi=Pin(19),                        #
```

# WETTERSTATION

```
80 |     miso=Pin(13))                                #
81 |
82 | tft = st7789.ST7789(                               # Objekt tft instanzieren
83 |     spi,                                           # Schnittstelle
84 |     135,                                           # Pixel x-Achse(hochkant)
85 |     240,                                           # Pixel y-Achse
86 |     reset=Pin(23, Pin.OUT),                       #
87 |     cs=Pin(5, Pin.OUT),                           #
88 |     dc=Pin(16, Pin.OUT),                           # TEST!
89 |     backlight=Pin(4, Pin.OUT),                     #
90 |     rotation=1)                                    # rotation 90°, 2 180°
91 | #----- Display Initialisierung Ende -----
92 |
93 | #----- Neo Pixel Initialisierung -----
94 | NUM_OF_LED = 11                                   # 11 LED's von den WS2812 LED's verwendet
95 | np = NeoPixel(Pin(2), NUM_OF_LED)                 # LED Band an Pin 2 angeschlossen
96 |
97 | r = 0                                              #
98 | g = 0                                              #
99 | b = 0                                              #
100 |
101 | #-----| Funktionen |-----
102 | def clearStripe():                               #
103 |     for i in range(NUM_OF_LED):                   # Liste für die 11 LED's
104 |         np[i] = (0,0,0)                           #
105 |         np.write()                                #
106 |
107 | def neoSetzten(r, g, b):                           #
108 |     for i in range(NUM_OF_LED):                   #
109 |         np[i] = (r,g,b)                           #
110 |         np.write()                                #
111 |
112 | def convertHelligkeit(x):
113 |     wert = (x - 0) * (254 - 1) // (3000 - 0) + 1
114 |     if wert > 254: wert=254
115 |     return wert
116 |
117 |
118 | mqttSenden = 3                                    # alle 3 sekunden
```



# WETTERSTATION

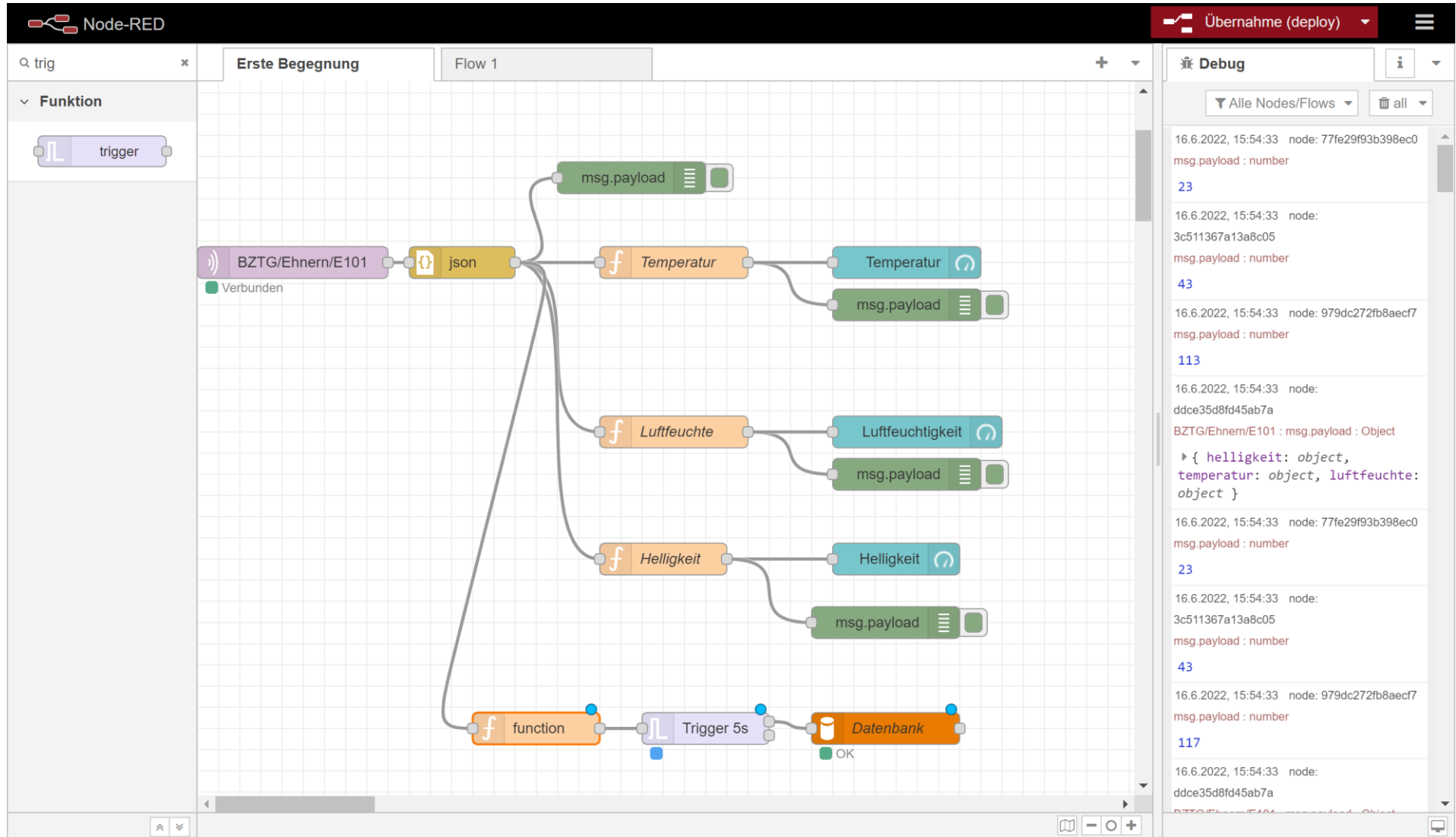
```
119 sensorenAuswerten = 1          # jede Sekunde
120 tftAnzeigen = 5                 # alle 5 Sekunden
121
122 zeitMqtt = time() + mqttSenden
123 zeitSensoren = time() + sensorenAuswerten
124 zeitTft = time() + tftAnzeigen  #
125
126 mqtt_MG.connect()               #
127
128 #-----Werte senden-----
129 while True:
130     if time() >= zeitSensoren:
131         try:
132             tempa = round(htu.temperature)    #
133             luft = round(htu.humidity)        #
134         except:
135             print("Temp / Hum sensor kaputt")
136             temp = str(0)
137             luft = 0
138
139         try:
140             helligkeit = round(bh.luminance(BH1750.CONT_LOWRES)) #
141         except:
142             print("Helligkeit Sensor defekt") #
143             helligkeit = 5000                #
144
145         npHell = convertHelligkeit(helligkeit) #
146
147         if luft < 30:                        #
148             r = 0
149             g = 1
150             b = 0
151
152         if (luft >= 30 and luft <= 50):      #
153             r = 1
154             g = 1
155             b = 0
156
```

# WETTERSTATION

```
157     if luft >= 50:                                     #
158         r = 1
159         g = 0
160         b = 0
161
162     neoSetzten(r*npHell, g*npHell, b*npHell)           #
163
164     ZeitSensoren = time() + sensorenAuswerten          #
165
166
167     if time() >= zeitMqtt:                               #
168         datatemp = {                                     #
169             "temperatur":{                               #
170                 "HTU21Dtemp": tempa                     #
171             },
172             "luftfeuchte":{                               #
173                 "HTU21Dluft": luft                       #
174             },
175             "helligkeit":{                               #
176                 "BH1750": helligkeit                     #
177             }
178         }
179
180     print("MQTT verbunden!")                             #
181     print(datatemp)                                     #
182     #mqtt_MG.connect()
183     #sleep(0.5)
184     mqtt_MG.publish(MQTT_TOPIC,json.dumps(datatemp))    #
185     #sleep(0.5)
186     #mqtt_MG.disconnect()
187
188     zeitMqtt = time() + mqttSenden                      #
189
190
191 #-----Werte auf Display-----
192     if time() >= zeitTft:                                 #
193         tft.fill(st7789.BLACK)                          #
194         tft.text(font, str(tempa) + ' °C', 10, 10, st7789.WHITE, st7789.BLACK) #
195
196         tft.text(font, str(luft) + ' %', 10, 30, st7789.WHITE, st7789.BLACK) #
197
198         tft.text(font, str(helligkeit) + ' lux', 10, 50, st7789.WHITE, st7789.BLACK)#
199
200     zeitTft = time() + tftAnzeigen                      #
```

# WETTERSTATION

## Node-RED



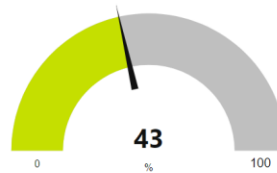
# WETTERSTATION

Node-RED UI

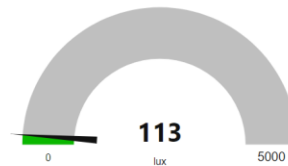
Home

Status

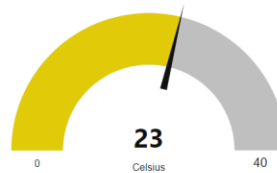
Luftfeuchtigkeit



Helligkeit



Temperatur



# WETTERSTATION

## MySQL

Kundenprojekt\kundenauftragmg\ueberwachung\ - HeidiSQL 11.3.0.6295

Datei Bearbeiten Suchen Abfrage Werkzeuge Gehe zu Hilfe

Datenbankfilter Tabellenfilter Host: 127.0.0.1 Datenbank: kundenauftragmg Tabelle: ueberwachung Daten Abfrage

**Kundenprojekt**

- information\_schema
- kundenauftragmg** 16,0 KiB
  - ueberwachung** 16,0 KiB
  - mysql
  - performance\_schema
  - sys

kundenauftragmg.ueberwachung: 334 Zeilen gesamt (ungefähr)

Nächste Zeile Alle Zeilen Sortierung Spalten (4/4) Filter

Zeit	Temperatur	Luftfeuchtigkeit	Helligkeit
2022-06-16 16:05:48	23	43	120
2022-06-16 16:05:54	23	43	117
2022-06-16 16:06:00	23	43	117
2022-06-16 16:06:06	23	43	117
2022-06-16 16:06:12	23	43	117
2022-06-16 16:06:18	23	43	117
2022-06-16 16:06:24	23	43	117
2022-06-16 16:06:30	23	43	117
2022-06-16 16:06:36	23	43	103
2022-06-16 16:06:42	23	43	100
2022-06-16 16:06:48	23	43	100
2022-06-16 16:06:54	23	43	100
2022-06-16 16:07:00	23	42	100
2022-06-16 16:07:06	23	42	103
2022-06-16 16:07:12	23	42	103
2022-06-16 16:07:18	23	42	100
2022-06-16 16:07:24	23	42	100
2022-06-16 16:07:30	23	42	100
2022-06-16 16:07:36	23	42	97
2022-06-16 16:07:42	23	43	97
2022-06-16 16:07:48	23	43	93
2022-06-16 16:07:54	23	42	93
2022-06-16 16:08:00	23	42	317
2022-06-16 16:08:06	23	43	753
2022-06-16 16:08:12	23	43	97
2022-06-16 16:08:18	23	42	97
2022-06-16 16:08:24	23	43	97
2022-06-16 16:08:30	23	43	97
2022-06-16 16:08:36	23	43	97

Filter: Regulärer Ausdruck

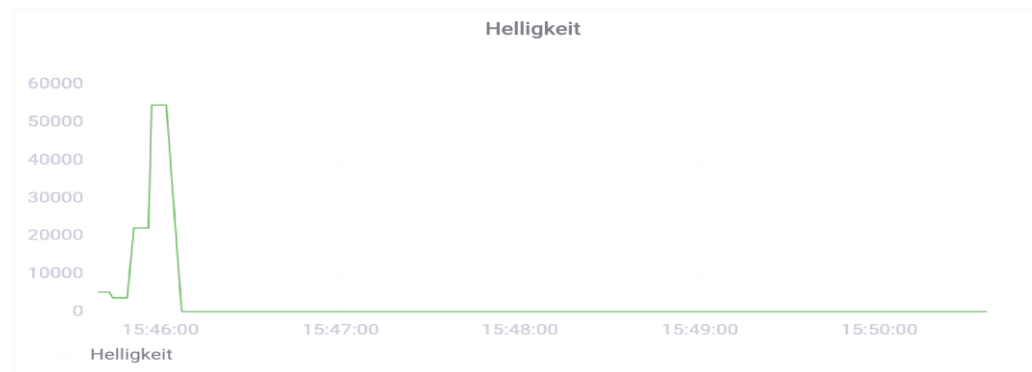
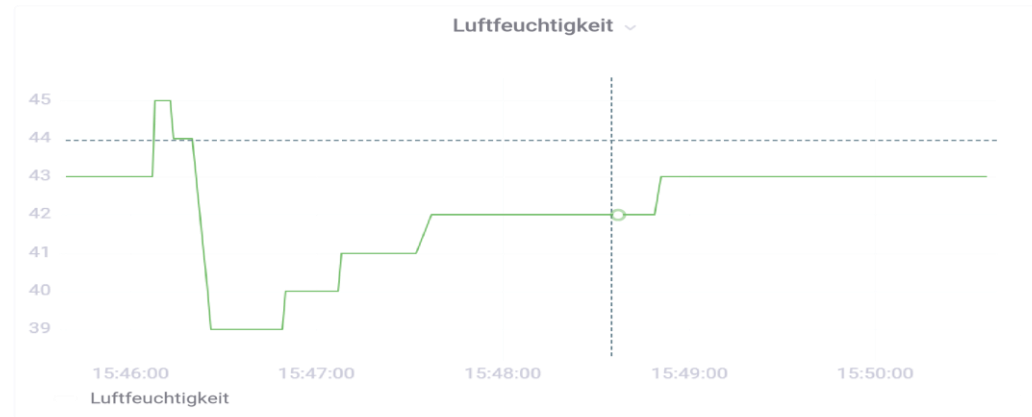
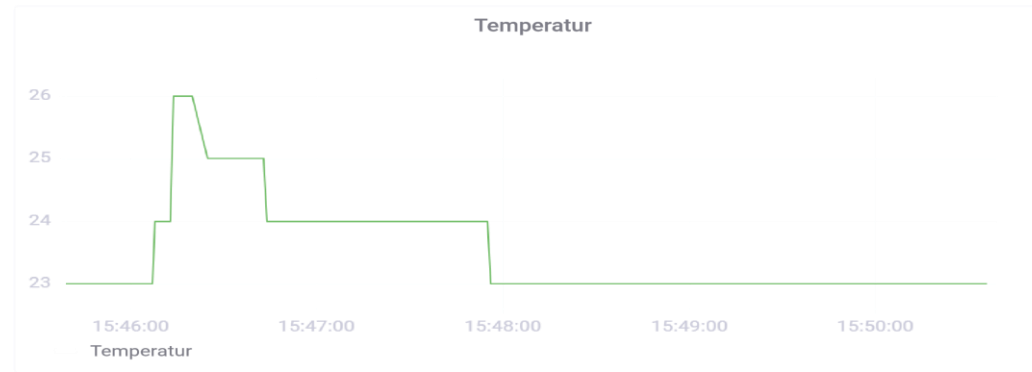
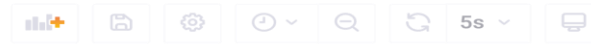
```
26 SHOW ENGINES;
27 SHOW COLLATION;
28 SHOW CREATE TABLE `kundenauftragmg`.`ueberwachung`;
29 SELECT CONSTRAINT_NAME, CHECK_CLAUSE FROM `information_schema`.`CHECK_CONSTRAINTS` WHERE CONSTRAINT_SCHEMA='kundenauftragmg' AND TABLE_NAME='ueberwachung';
30 SELECT * FROM `kundenauftragmg`.`ueberwachung` LIMIT 1000;
```

r1: c1 Verbinden: 00:00 h MariaDB 10.7.3 Betriebszeit: 6 Tage, 02:46 h Serverzeit: 18:32 Leerlauf.

# WETTERSTATION

Grafana

☰ New dashboard



## Fazit

Da ich ja leider eine sehr lange Zeit ausgefallen bin und Prüfungen, Roboter Projekt und Präsentationen anstanden, ist mir das Projekt sehr schwergefallen. Doch dank der Hilfe von Klassenkameraden, konnte ich mein Projekt fertig stellen.

Auch konnte ich durch dieses Projekt alles was durch meine Corona Erkrankung gefehlt hat, sehr gut nachholen.

Allerdings sind die Kommentare zum Programm durch die kurze Zeit leider viel zu kurz gekommen und schlecht nachvollziehbar.

## Quellen

Bildquellen (Deckblatt, Aufbau, Screenshots): Privat

HTU21: [HTU21D Temperatur- und Luftfeuchtigkeitssensor I2C - Bastelgarage Elektronik Online Shop](#)

BH1750: [HALJIA gy-302 bh1750 Digital Lichtintensität Sensor Beleuchtung Detektor Modul 3 V - 5 V Kompatibel mit Arduino GY302](#)

[BH1750FVI: Amazon.de: Computer & Zubehör](#)

ESP32: [LILYGO TTGO T-Display ESP32 Development Board \(16 MB\) CP2104 | Elektor](#)

WS2812b: [5m WS2812b LED Pixel Streifen 60LEDs/m IP65 \(5V\) | pixel-imperium.de](#)

Bibliotheken: Siehe readme

Dokumentation, Kundenauftrag und alles weiter sind auf Github hinterlegt:



Letzter Zugriff aller Quellen: 12.06.2022 um 15 Uhr