```
"""
autor:   Rudolf, Christian
        Heikebrügge, Tim
datum: 09.04.2024
Funktionsbeschreibung :

Smart Home System
Unser Kunde wünscht sich ein Smarthome-System mit folgenden Komponenten:
Mit einem Helligkeitssensor die Helligkeit im Wohnzimmer feststellen um dann ggf. den virtuellen Raum abzudunkeln mit Rolläden.
Sauerstoffgehalt im Wohnzimmer ermitteln und dann ein virtuelles Fenster öffnen um zu lüften
Wenn die Anlage "scharf" geschaltet wird, soll ein Alarm ausgeben werden, wenn ein Bewegungsmelder ausgelöst wird.
Info für  Temperatur der Heizung und etc.
Visualisierung über Node-RED darstellen.
Zusätzlich werden die Temperatur und der Co 2-Gehalt in einer Datenbank abgelegt.


Versionsnummer: 1.0
Hardware :      Espressif ESP32-S3-WROOM-1
                1x Co2 Sensor (SDC30)
                1x Helligkeitssensor (BH1750)
                1x Temperatursensor (AHT10)
                1x Bewegungssensor (SEN-HC-SR501)
                5x Weiße LED´s 220 Ohm Widerstand (Jalousie höhe)
                3x LED´s 220 Ohm Widerstand (Alarmstatus)
                1x LED 220 Ohm Widerstand (Fenster Auf)
                1x Taster 10k Ohm Widerstand Pull-Down (Alarmanlage De-/Aktivieren)
Software:        MQTT
            Node-red
            Heidisql
            MariaDB
Bilbiothekenname : scd30.py (Co2 Sensor), bh1750.py (Helligkeitssensor), ahtx0.py (Temperatursensor)

Temperatursensor Anschluss :     VCC = 3.3V
                                GND = GND
                                SCL = GPIO39
                                SDA = GPIO38
Helligkeitssensor Anschluss :    VCC = 3.3V
                                GND = GND
                                SCL = GPIO39
                                SDA = GPIO38
Co2 Sensor Anschluss:                   VIN = 3.3V
                                GND = GND
                                SCL = GPIO39
                                SDA = GPIO38
Bewegungssensor Anschluss:              VCC = 3.3V
                                GND = GND
                                OUT = GPIO9

"""
#-----------------------------------------------------------------------------------------------
#Bibliotheken Importieren

from machine import Pin, SoftI2C
import time
from time import sleep

import json
from umqtt.simple import MQTTClient

#Netzwerk
```

```
61 import network
62 import ubinascii
63 import machine
64
65 #Import Temperatursensor Library
66 import ahtx0
67
68 #Import Co2 Library
69 from scd30 import SCD30
70
71 #Import Lichtstärke Library
72 from bh1750 import BH1750
73
74
75
76 #-----------------------------------------------------------------------------------------------
77 #Pins definieren
78 #Fenster
79 fensterAuf = Pin(3, Pin.OUT)
80
81 #Alarmanlage
82 bewegungsSensor = Pin(9, Pin.IN, Pin.PULL_DOWN)
83 tasterAlarm = Pin(8, Pin.IN)
84 ledRot = Pin(15, Pin.OUT)
85 ledGruen = Pin(16, Pin.OUT)
86 ledBlau = Pin(17, Pin.OUT)
87
88 #jalousie
89 ledjalousie100 = Pin(4, Pin.OUT)
90 ledjalousie75 = Pin(5, Pin.OUT)
91 ledjalousie50 = Pin(6, Pin.OUT)
92 ledjalousie25 = Pin(7, Pin.OUT)
93
94 #I2C-Bus definieren
95 i2c_sda = Pin(38)
96 i2c_scl = Pin(39)
97 I2C = SoftI2C(sda=i2c_sda, scl=i2c_scl)
98
99 #Luftfeuchtigkeit- und Temperatur- erfassung
100 sensorAhtx0 = ahtx0.AHT10(I2C)
101
102 #Helligkeitserfassung
103 helligkeit = BH1750(0x23, I2C)
104
105 #Co2 Datenerfassung
106 sauerstoff = SCD30(I2C, 0x61)
107
108 #-----------------------------------------------------------------------------------------------
109 #Variablen
110
111 #Alarmanlage
112 tasterStatus = False
113 tastergedrueckt = False
114 tasterwar_aus = False
115 alarmEin = False
116 alarmmanuell = ["Alarmanlage deaktiviert"]
117
118
119 #Jalousie
120 jalousiemanuell = ["0% Jalousie"]
121 jalousieAutomatik = ["Jalousie Automatik Aus"]
122 jalousieAutomatik_Wert = 4000
123
```

```
124 #Fenstert
125 fenstermanuell = ["geschlossen"]
126 fensterAutomatik = ["Fenster Automatik Aus"]
127 fensterAutomatik_Wert = 40
128 fensterAutomatik_co2_Wert = 3000
129 temperatur_AHTX = 0
130 luftfeuchte_AHTX = 0
131 fensterStatus = False
132
133 #Zeitdifferenz Anfangswerte für Zeitabfrage ohne sleep()
134 zeit_alt_temp = 0
135 zeit_alt_rel = 0
136 zeit_alt_hell = 0
137 zeit_alt_co2 = 0
138 zeit_alt_taster = 0
139 zeit_alt_mqtt = 0
140 zeit_alt_read = 0
141
142
143 #-----------------------------------------------------------------------------------------------
144 # Wifi Verbindung
145
146 wlan_ssid = "BZTG-IoT"
147 wlan_passwort = "WerderBremen24"
148
149
150
151 #Mit Wlan verbinden
152 wlan = network.WLAN(network.STA_IF)
153 wlan.active(True)
154 wlan.connect(wlan_ssid, wlan_passwort)
155
156 #Auf Wlan Verbindung warten
157 while not wlan.isconnected():
158     pass
159
160 #SSID und IP ausgeben in Konsole
161 print("-----------------------------------------------------------------")
162 print("Verbunden mit", wlan_ssid)
163 print("IP-Adresse vom ESP:", wlan.ifconfig()[0])
164 print("-----------------------------------------------------------------")
165
166
167 #-----------------------------------------------------------------------------------------------
168 # MQTT Broker Verbindung
169 client_id = ubinascii.hexlify(machine.unique_id()) #Eindeutigen namen vergeben
170 server = "192.168.1.108"  #MQTT Broker IP
171 topic = "SMARTHOME_T&C"  #Topic einstellen
172
173 c = MQTTClient(client_id, server) #MQTT Funktion in Variable "c" schreiben
174 c.connect()      #Verbindung herstellen
175 print("-----------------------------------------------------------------")
176 print(f"Verbunden mit {server}")
177 print("-----------------------------------------------------------------")
178
179
180 # Nachrichten aus dem MQTT Broker empfangen und in Variablen schreiben
181 def mqtt_nachricht(topic, message):
182     global jalousiemanuell, alarmmanuell, fenstermanuell, fensterAutomatik, fensterAutomatik_Wert,
jalousieAutomatik, jalousieAutomatik_Wert, fensterAutomatik_co2_Wert
183     try:
184         msg_neu = json.loads(message.decode())
185         if "Status" in msg_neu:                                #Alarmanlage über Website Ein-/
```

```
Ausschalten
            alarmmanuell = msg_neu["Status"]
        if "Befehl" in msg_neu:                         #Jalousie über Website steuern
            jalousiemanuell = msg_neu["Befehl"]
        if "Fenster" in msg_neu:                        #Fenster über Website Öffnen/Schließen
            fenstermanuell = msg_neu["Fenster"]
        if "Fenster_Auto" in msg_neu:                   #Fenstersteuerung auf Automatik
            fensterAutomatik = msg_neu["Fenster_Auto"]
        if "Fenster_Auto_Wert" in msg_neu:              #Fenstersteuerung ab welchem Tempwert soll
geöffnet/geschlossen werden
            fensterAutomatik_Wert = msg_neu["Fenster_Auto_Wert"]
        if "Fenster_Auto_co2_Wert" in msg_neu:   #Fenstersteuerung ab welchem Co2wert soll
geöffnet/geschlossen werden
            fensterAutomatik_co2_Wert = msg_neu["Fenster_Auto_co2_Wert"]
        if "Jalousie_Auto" in msg_neu:                  #Jalousiesteuerung auf Automatik
            jalousieAutomatik = msg_neu["Jalousie_Auto"]
        if "Jalousie_Auto_Wert" in msg_neu:             #Jalousiesteuerung ab welchem Lumenwert
soll geöffnet/geschlossen werden
            jalousieAutomatik_Wert = msg_neu["Jalousie_Auto_Wert"]


        #Werte zur Prüfung in Konsole ausgeben
        print("-----------------------------------------------------------------")
        print("Jalousie Status:", jalousiemanuell)
        print("Alarm Status:", alarmmanuell)
        print("Fenster Status:",fenstermanuell)
        print("Fenster Automatik:", fensterAutomatik)
        print("Fenster Automatik Temperatur Wert:", fensterAutomatik_Wert)
        print("Fenster Automatik Co2 Wert:", fensterAutomatik_co2_Wert)
        print("Jalousie Automatik:", jalousieAutomatik)
        print("Jalousie Automatik Wert:", jalousieAutomatik_Wert)
        print("-----------------------------------------------------------------")
    except Exception as e:
        print("Fehler beim Parsen der Nachricht:", e)



#-------------------------------------------------------------------------------------------------

# Aufrufen der Funktion mqtt_nachricht
c.set_callback(mqtt_nachricht)
# Topic abonnieren
c.subscribe(topic)

while True:
#-------------------------------------------------------------------------------------------------
# Auf Nachrichten im MQTT-Broker prüfen
    zeit_now_read = time.ticks_ms()
    if(time.ticks_diff(zeit_now_read, zeit_alt_read) > 1000):
        c.check_msg()
        zeit_alt_read = time.ticks_ms()
#-------------------------------------------------------------------------------------------------
# Fenstersteuerung
    if fensterAutomatik == "Fenster Automatik Aus":
        if fenstermanuell == "geoeffnet":
            fensterAuf.on()
            fensterStatus = True
        if fenstermanuell == "geschlossen":
            fensterAuf.off()
            fensterStatus = False

    if fensterAutomatik == "Fenster Automatik Ein":
        if temperatur_AHTX >= fensterAutomatik_Wert or co2 >= fensterAutomatik_co2_Wert:
```

```
            fensterAuf.on()
            fensterStatus = True
        else:
            fensterAuf.off()
            fensterStatus = False

#-----------------------------------------------------------------------------------------------
# Jalousie
    if jalousieAutomatik == "Jalousie Automatik Aus":
        if jalousiemanuell == "25% Jalousie":
            ledjalousie100.on()
            ledjalousie75.off()
            ledjalousie50.off()
            ledjalousie25.off()

        if jalousiemanuell == "50% Jalousie":
            ledjalousie100.on()
            ledjalousie75.on()
            ledjalousie50.off()
            ledjalousie25.off()

        if jalousiemanuell == "75% Jalousie":
            ledjalousie100.on()
            ledjalousie75.on()
            ledjalousie50.on()
            ledjalousie25.off()

        if jalousiemanuell == "100% Jalousie":
            ledjalousie100.on()
            ledjalousie75.on()
            ledjalousie50.on()
            ledjalousie25.on()

        if jalousiemanuell == "0% Jalousie":
            ledjalousie100.off()
            ledjalousie75.off()
            ledjalousie50.off()
            ledjalousie25.off()

    if jalousieAutomatik == "Jalousie Automatik Ein":
        jalousiemanuell = ""
        if helligkeit_BH1750 >= jalousieAutomatik_Wert:
            ledjalousie100.on()
            ledjalousie75.on()
            ledjalousie50.off()
            ledjalousie25.off()
        else:
            ledjalousie100.off()
            ledjalousie75.off()
            ledjalousie50.off()
            ledjalousie25.off()

#-----------------------------------------------------------------------------------------------
# Alarmanlage

# Tasterabfrage
    tastergedrueckt = tasterAlarm.value()
    zeit_now_taster = time.ticks_ms()
    bewegung = bewegungsSensor.value()
    bewegMeldung = False

    if tastergedrueckt or alarmmanuell == "Alarmanlage aktiviert" or (alarmmanuell == "Alarmanlage
deaktiviert" and tasterwar_aus == True):
```

```
307         if(time.ticks_diff(zeit_now_taster, zeit_alt_taster) > 300):
308             tasterStatus = not tasterStatus
309             tasterwar_aus = False
310             zeit_alt_taster = time.ticks_ms()
311     if not tastergedrueckt:
312         tasterwar_aus = True
313 
314 # Steuerung
315     if alarmmanuell == "Alarmanlage aktiviert" or tasterStatus == True:
316         alarmEin = True
317         alarmmanuell = []
318         ledGruen.off()
319         ledRot.on()
320     if alarmmanuell == "Alarmanlage deaktiviert" or tasterStatus == False:
321         alarmEin = False
322         alarmmanuell = []
323         ledRot.off()
324         ledGruen.on()
325         ledBlau.off()
326 
327     if alarmEin and bewegung == True:
328         ledBlau.on()
329         bewegMeldung = True
330     if alarmEin and bewegung == False:
331         ledBlau.off()
332         bewegMeldung = False
333 
334 
335 
336 #-------------------------------------------------------------------------------------------------
337 # Temperatur, Luftfeucht, Helligkeit messen
338     zeit_now_temp = time.ticks_ms()
339     if(time.ticks_diff(zeit_now_temp, zeit_alt_temp) > 5000):
340         temperatur_AHTX = round(sensorAhtx0.temperature *2) / 2          #Runden
341         luftfeuchte_AHTX = round(sensorAhtx0.relative_humidity)
342         helligkeit_BH1750 = round(helligkeit.measurement)
343         zeit_alt_temp = time.ticks_ms()
344 
345 
346 # Sauerstoffgehalt messen und richtig ausgeben
347     while sauerstoff.get_status_ready() != 1:
348         sleep(0.2)
349 
350     zeit_now_co2 = time.ticks_ms()
351     if(time.ticks_diff(zeit_now_co2, zeit_alt_co2) > 5000):
352         sauerstoffTuple = sauerstoff.read_measurement()
353         co2 = round(sauerstoffTuple[0])
354         zeit_alt_co2 = time.ticks_ms()
355 
356 # Werte in JSON format verpacken
357     x_json = {
358         "Ort": "Haus",
359         "Raum": "Wohnzimmer",
360         "Sensorwerte": [
361              {"Temperatur": temperatur_AHTX},
362              {"Luftfeuchte": luftfeuchte_AHTX},
363              {"CO2-Gehalt": co2},
364              {"Helligkeit": helligkeit_BH1750},
365              {"Alarm": bewegMeldung},
366              {"Alarm Status": alarmEin},
367              {"Fenster Status": fensterStatus},
368             ]
369     }
```

```
    sorted_string = json.dumps(x_json)
#-------------------------------------------------------------------------------------------------
# Nachricht senden an MQTT Broker
    zeit_now_mqtt = time.ticks_ms()
    if(time.ticks_diff(zeit_now_mqtt, zeit_alt_mqtt) > 5000):
        c.publish(topic, sorted_string)
        print("-----------------------------------------------------------------")
        print("Nachricht an MQTT-Broker gesendet.")
        print("-----------------------------------------------------------------")
        zeit_alt_mqtt = time.ticks_ms()


```