Auslesen einer Hanazeder SH-8 Steuerung und per MQTT veröffentlichen

(Solar Heizungssteuerung SH Hersteller: Hanazeder)

die Hanazeder SH-8 Steuerung besitzt eine serielle Schnittstelle mit der man die Daten auslesen kann.

Danke der Hilfe eines Freundes kann ich inzwischen die Steuerung auslesen und die Werte als JSON per MQTT versenden. Das Projekte war meine erste Berührung zu Python3.

Diese Anleitung ist für den Versierten Bastler gedacht. Ich lese die Serielle Schnittstelle mit einem USB-3.3V Seriell Adapter aus. Es darf nur TX, RX und GND verbunden werden.

Es handelt sich um ein TTL Serielle Schnittstelle mit 3,3V Angeschlossen wird nur:

- TX
- RX
- GND

| |HanazederSH8_2_MQTT.py

```
#!/usr/bin/env python3
#
# V0.9
#
#
#Dieses Script liest die Serielle Schnittstelle einer Hanazeder SH8
Steuerung regelmÃ¤ssig aus
```

```
#und Sendet ein JSON per MQTT an den Broker
#V0.1 - devzero hat den JSON Teil und das GrundgerÃ¼st zu verfÃ¼gung
gestellt
#V0.2 - gezielte Abfrage der Seriellen Schnittstelle hinzugefÃ¼gt.
#V0.3 - JSON um Solar Leistung erweitert
#V0.4 - MQTT impelemntierung (offen)
#V0.5 - Optimirungen der Initalisirungen
#V0.6 - Individueller Name fÃ¼r FÃ¼hler und AusgÃ¤nge (devzero)
#V0.7 - Optimierung Abfrage der Seriellen Schnittstelle (devzero & lsd)
#V0.8 - Abfrage der Seriellen Schnittstelle komplett neu implementiert
#V0.9 - Status Infos zum System per MQTT

import netifaces as ni
import serial
import time
import json
import paho.mqtt.client as mqttclient
import re

DEBUG=False
DISPLAY=True

#MQTT Parameter
#https://www.emqx.io/blog/how-to-use-mqtt-in-python
MQTTBROKER = "iobrokerv2.home.logout.de"
MQTTPORT = 1883
MQTTUSER = "iobroker"
MQTTPASS = "brokerio"
MQTTTOPIC = "Hanazeder/SH8"
MQTTID = 'HZ-Solar-1234'
MQTTSTATUS = MQTTTOPIC+"/status"
MQTTSYSTEM = MQTTTOPIC+"/system"
MQTTSTATUSSERIAL = MQTTTOPIC+"/statusseriell"
MQTTDATA = MQTTTOPIC+"/json"

#Serielle Schnittstell
SERPORT="/dev/hzsh8solar"
SERBAUD=9600
SERTIME=1
SLEEPSEC=300
SERREAD_BYTE=500

#Es kÃ¶nnen auch Werte weggelssen werden dann wird A[1-8] oder F[1-14]
behalte
#z.B. ist F14 nicht angelegt

AUTOMATIK_AUSGANG_MAPPING = {
    "A1": "A1-Solar to Boiler",
    "A2": "A2-Solar to Puffer",
    "A3": "A3-freigabe BHKW",
    "A4": "A4-Puffer to Boiler",
```

```
    "A5": "A5-NN",
    "A6": "A6-Heiz-Pumpe",
    "A7": "A7-Mischer-Auf",
    "A8": "A8-Mischer-Zu"
}

FUEHLER_MAPPING = {
    "F1": "F1-Kollektor",
    "F2": "F2-nn",
    "F3": "F3-VL_Solar",
    "F4": "F4-RL_Solar",
    "F5": "F5-P_unten",
    "F6": "F6-P_mitte",
    "F7": "F7-P_oben",
    "F8": "F8-B_unten",
    "F9": "F9-B_oben",
    "F10": "F10-NN",
    "F11": "F11-NN",
    "F12": "F12-Heizkreis",
    "F13": "F13-Aussentemp"
}

###################### Parameter ENDE ####################

def Host_name_IP_2_MQTT():
    try:
        ret = dict()
        gws = ni.gateways()
        net['DGW'] = gws['default'][ni.AF_INET][0]
        net['IP'] = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
        net['Netmask'] =
ni.ifaddresses('eth0')[ni.AF_INET][0]['netmask']
        net['MAC'] = ni.ifaddresses('eth0')[ni.AF_LINK][0]['addr']
    except:
        if DEBUG: print("Unable to get Hostname and IP")

    return ret

def parse(data_str):
    def _get_dict_from_key_value_string_list(key_value_string_list,
name_mapping_dict=None, type_cast_function=None):
        ret = dict()
        for key_value_string in key_value_string_list:
            key, value = key_value_string.split(':')
            if name_mapping_dict and key in name_mapping_dict:
                key = name_mapping_dict[key]
            if type_cast_function is not None:
                value = type_cast_function(value)
            ret[key] = value
        return ret
```

```
    solarleistung = None
    automatik = dict()
    ausgang = dict()
    fuehler = dict()

    for line in data_str.decode().split('\n\r'):
        if 'Mom.Solar-Leist' in line:
            match = re.search(r'\d+\.\d+', line)
            if match:
                solarleistung = float(match.group(0))
            continue

        pattern = re.compile(r'A\d:\d')
        if pattern.search(line):
            if 'Automatik' in line:
                automatik = _get_dict_from_key_value_string_list(
                    pattern.findall(line), AUTOMATIK_AUSGANG_MAPPING,
lambda x: bool(int(x)))
            elif 'Ausgang' in line:
                ausgang = _get_dict_from_key_value_string_list(
                    pattern.findall(line), AUTOMATIK_AUSGANG_MAPPING,
lambda x: bool(int(x)))
            continue

        pattern = re.compile(r'F\d+:\d+')
        if pattern.search(line):
            fuehler.update(
_get_dict_from_key_value_string_list(pattern.findall(line),
FUEHLER_MAPPING, lambda x: int(x)))
            continue

    ret = dict()
    ret['Solarleistung'] = solarleistung
    ret['Automatik'] = automatik
    ret['Ausgang'] = ausgang
    ret['Fuehler'] = fuehler
    return ret

def mqtt_connect():
#MQTT Client initalisiren
    def MQTT_ONLINE(MQTTCLIENT, userdata, flags, rc):
     MQTTCLIENT.publish(MQTTSTATUS,"online",0,retain=False)
     if rc == 0:
      print("Connected to MQTT Broker! Return Code: "+str(rc))
     else:
      print("Failed to connect, Return Code: "+str(rc))
    MQTTCLIENT = mqttclient.Client(MQTTID)
    MQTTCLIENT.username_pw_set(MQTTUSER, MQTTPASS)
    MQTTCLIENT.on_connect = MQTT_ONLINE
    MQTTCLIENT.will_set(MQTTSTATUS,"offline",0,retain=True)
    MQTTCLIENT.connect(MQTTBROKER, MQTTPORT)
```

```
    return MQTTCLIENT

def warten_auf_bytes(erwartetes_byte, serial_port):
    antwort=False
    while True:
        read_serial = serial_port.read(1)
        if read_serial == erwartetes_byte:
            antwort=True
            break

        if not read_serial:
            break

    return antwort

if __name__ == '__main__':

#MQTT Client Verbinden
    MQTTCLIENT = mqtt_connect()
    MQTTCLIENT.loop_start()
    JSON = json.dumps(Host_name_IP_2_MQTT(), indent=4)
    MQTTCLIENT.publish(MQTTSYSTEM,JSON,0,retain=True)


    #Initalisirung der Seriellen Schnittstell
    ser = serial.Serial(
        SERPORT,
        baudrate=SERBAUD,
        bytesize=serial.EIGHTBITS,
        parity=serial.PARITY_NONE,
        stopbits=serial.STOPBITS_ONE,
        timeout=SERTIME,
        xonxoff=False,
        rtscts=False,
        write_timeout=None,
        dsrdtr=False,
        inter_byte_timeout=None,
        exclusive=None
    )

    if ser.is_open:
        if DEBUG: print("Ser. Interface ist ONLINE")
        MQTTCLIENT.publish(MQTTSTATUSSERIAL,"CONNECT",0,retain=False)
        #time.sleep(0.2)
        while True:
            #ser.flushInput()
            ser.reset_input_buffer()
            ser.write(b'\x55')
            if DEBUG: print('DEBUG_Main: 1-2 Befehlsmodus aktivieren')
            if warten_auf_bytes(b'\x55', ser):
```

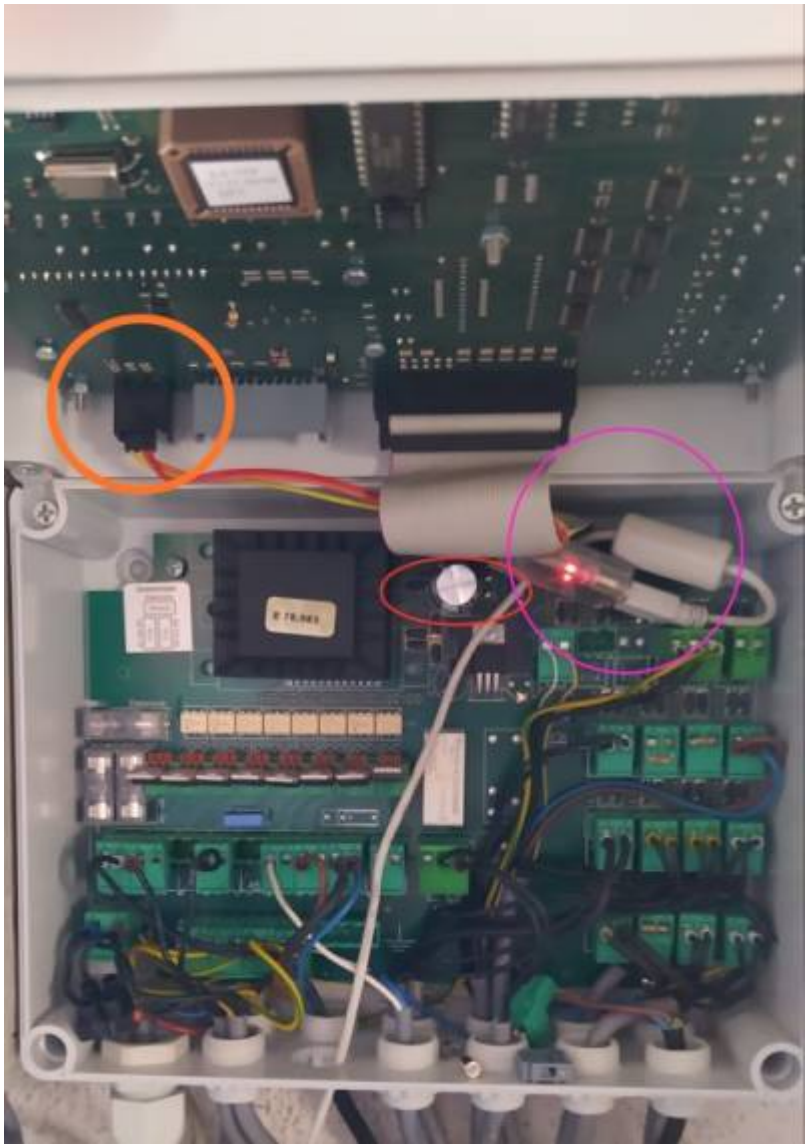
```
                if DEBUG: print('DEBUG_Main: 2-2 Befehlsmodus
aktiviert')
                if DEBUG: print('DEBUG_Main: Sende DWORT 0x0000')
                ser.write(b'\x00\x00\x00\x00')
                if DEBUG: print('DEBUG_Main: Sende anzahl (1) der zu
Empfangenen Byts 0x01')
                ser.write(b'\x01')
                if warten_auf_bytes(b'\x02', ser):
                    if DEBUG: print('DEBUG_Main: Nutzdaten kÃ¶nne
empfangen werden')
                    reply = ser.read(size=SERREAD_BYTE)
                    if DEBUG: print ("Daten die Empfangen wurden ",
reply)
                    if DEBUG: print ("2 DEBUG-Rufe >> JSON << auf")
                    if DEBUG: print(json.dumps(parse(reply), indent=4))
                    JSON = json.dumps(parse(reply), indent=4)
                    if (DISPLAY):
                        print ("_____________ JSON Ergebniss
__________________")
                        print (JSON)
                        print
("____________",(time.ctime()),"__________________\n")
                    MQTTCLIENT.publish(MQTTDATA,JSON,0,retain=True)
                else:
                    print('doof, timeout')
            else:
                print('doof, timeout')

            if DEBUG: print ("Sleep: ",SLEEPSEC)
            time.sleep(SLEEPSEC)
            if DEBUG: print ("Sleep: Ende")
    else:
        if DEBUG: print("Ser. Interface ist OFFLINE")
MQTTCLIENT.will_set(MQTTSTATUSSERIAL,"DISCONNECT",0,retain=True)
ser.close()
```

Unterlagen die mir vom Hersteller zu Verfügung gestellt wurden

protokoll.txt.zip

Display flackert und/oder die Steuerung friert ein

Falls ihr auch das Problem habt, daß die Steuerung einfriert oder das Display leicht flackert. Dann liegt dies am Kondensator der defekt ist. Einfach den ROT maskierten Kondensator austauschen.



- ORANGE ⇐ Serielle Schnittstelle
- MAGENTA ⇐ USB-Seriell (3,3V) Adapter
- ROT ⇐ Kondensator