SchrittWeise* PicoBello-07

PB-07 SchrittWeise MQTT + Node-RED auf RasPi B V20.odt

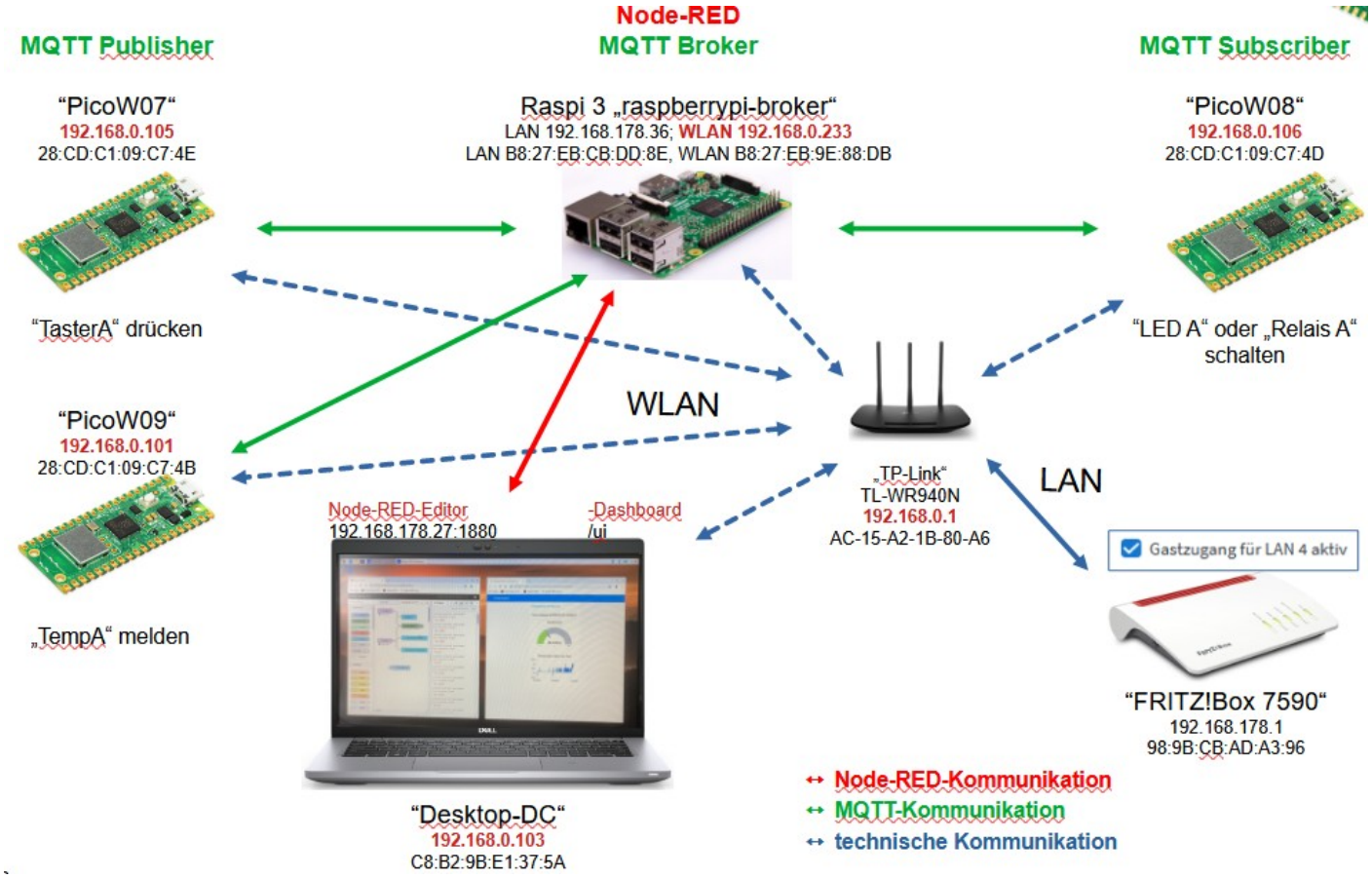
Inhaltsverzeichnis

(* Schritt für Schritt ein wenig weiser ...)

1) Intention

Aufbau einer Initial-Infrastruktur für sicheres und energiesparende Smart Home basierend auf MQTT und visualisiert und gesteuert über Node-RED als Basis für weiteren, individuellen Ausbau.

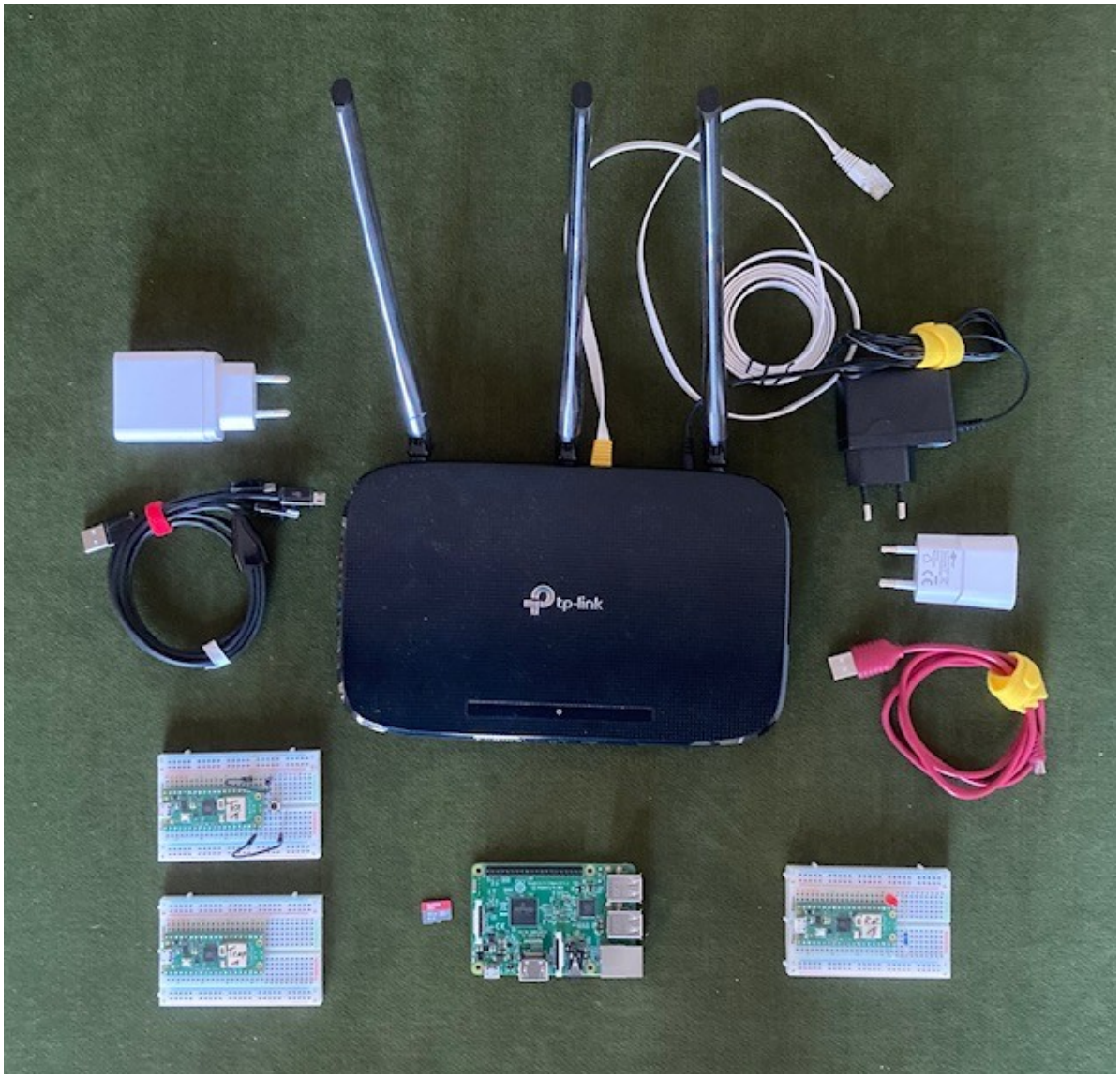
2) Struktur

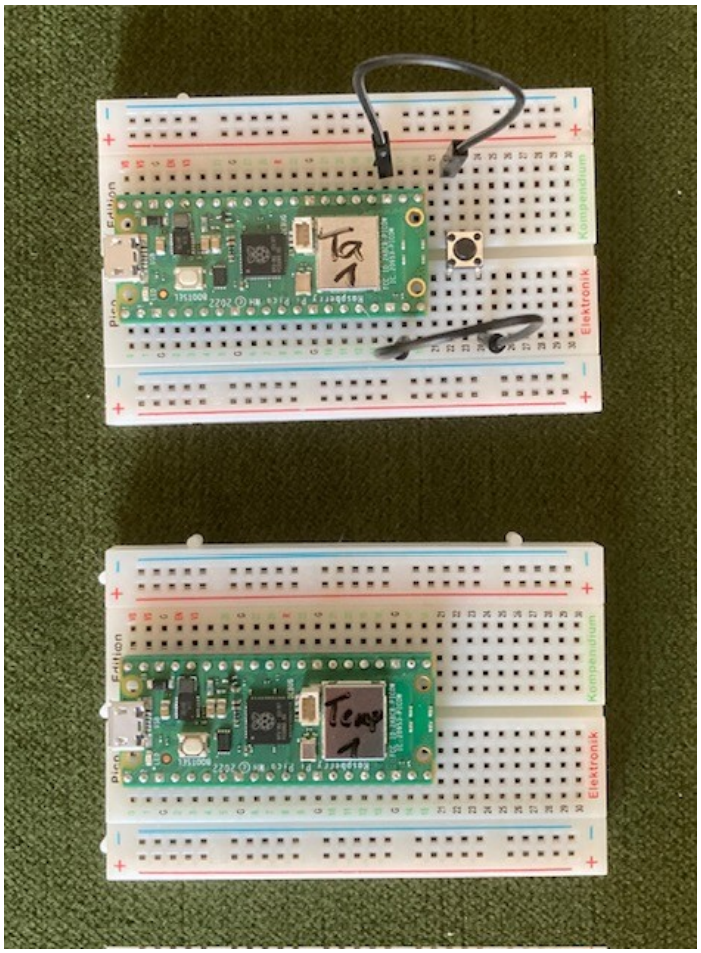


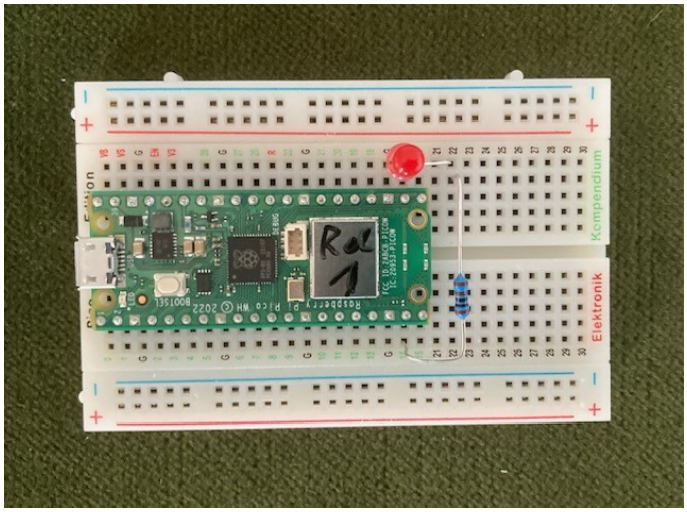
3) Benötigte Hard- und Software

Hardware	Software	Grund, Zweck
1 Raspberry Pi Pico W, 1 Raspberry Pi Pico W	MicroPython, Taster-Programm, MicroPython, Temperatur-Programm	senden Schalter- und Temperaturdaten als MQTT-Publisher an MQTT-Broker
1 Raspberry Pi Pico W	MicroPython, LED- oder Relais-Programm	empfängt Schalt-Daten als MQTT-Subscriber vom MQTT-Broker
1 Raspberry Pi 3 (oder 4 oder 5)	Raspberry Pi OS (auf SD), Mosquitto (MQTT Software für Raspi)	Betriebssystem auf Raspi 3/4; SSH-Zugriff von PC ist MQTT-Broker und vermittelt Status-, Mess- und Schalt-Daten zwischen MQTT-

	Node-RED	Publisher und MQTT-Subscriber; bietet Node-RED Editor zur Programmierung und Node-RED Dashboard zur Anzeige und Steuerung
1 WLAN-Router (z.B. FRITZ!Box)		über ihn kommunizieren die Pico Ws mit dem Raspi 3/4
1 PC (mit Internet-Anschluss)	Thonny; LAN zu Raspi 3 und PuTTY; Browser	als MicroPython Editor; SSH-Zugriff auf Raspi 3/4 für Software-Update und -Download; Steuerung von Node-RED Editor und Node-RED Dashboard auf Raspi 3/4
1 SmartPhone	Browser OpenVPN und Browser	zu Hause: Direkt-Zugriff auf Node-RED Dashboard unterwegs: sicherer VPN-Zugriff auf Node-RED Dashboard







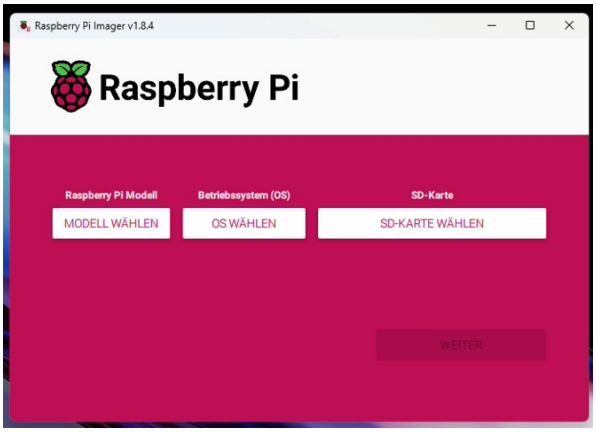
4) Betriebssystem Raspberry Pi OS für Raspberry Pi 3 auf SD-Karte

4.1 „Raspberry Pi Imager“ herunterladen von https://www.raspberrypi.com/software/

4.2 „Raspberry Pi Imager“ installieren.

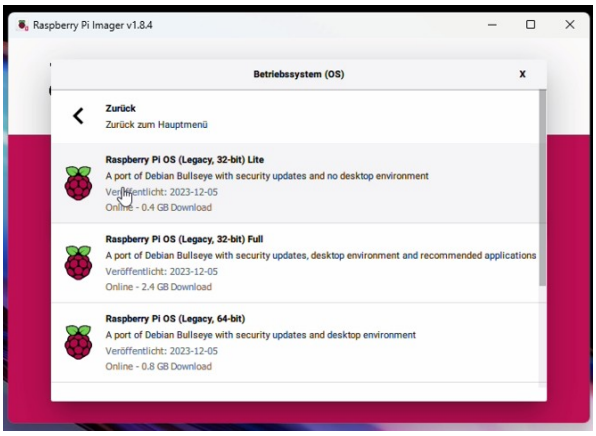
4.3 32 GB SD-Karte über Card-Adapter mit PC verbinden.

4.4 „Raspberry Pi Imager“ aufrufen:



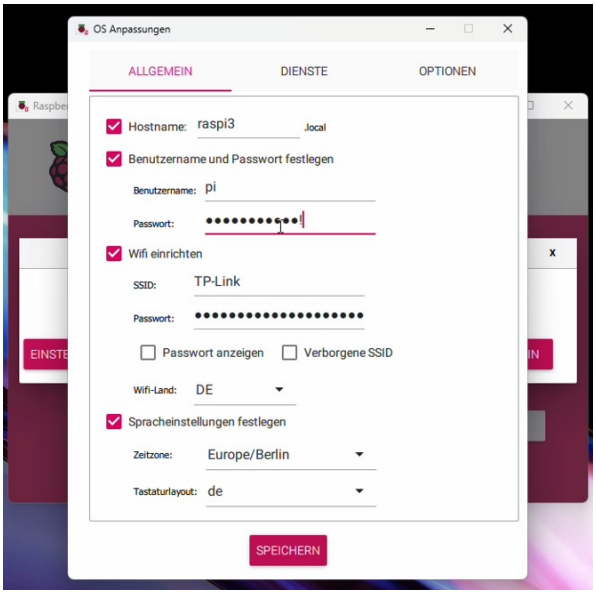
4.5 “MODELL WÄHLEN” anklicken

4.6



32er Lite Version und SD-Adapter-Laufwerk auswählen

4.7



Sprechenden Hostname eingeben

z.B. **raspi3**

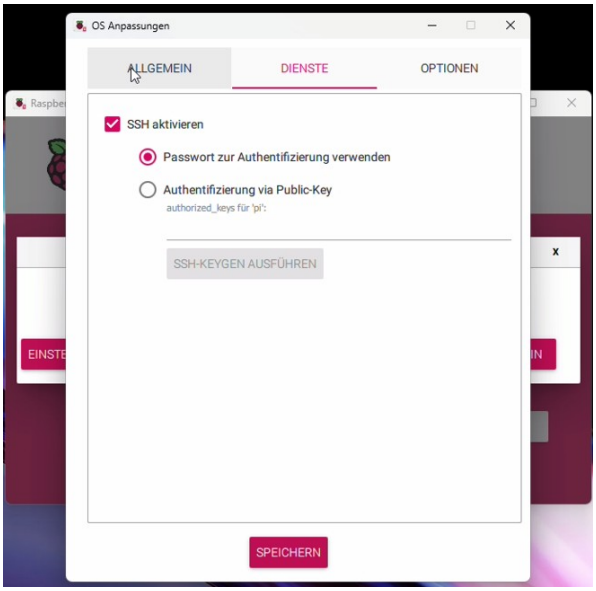
SSH aktivieren (damit von externem PC auf Raspi zugegriffen werden kann)

Benutzername und Passwort setzen.

Die SSID desjenigen WLAN-Accesspoints eingeben, über den auch die Pico Ws als MQTT-Clients mit dem MQTT-Broker kommunizieren

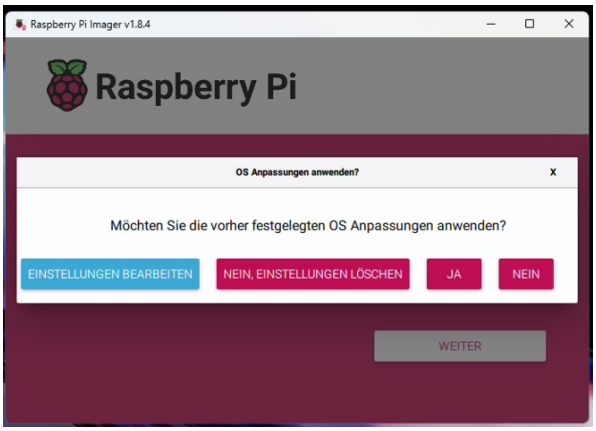
Achtung: die **Pico Ws können nur im 2,4 GHz Band** arbeiten.

4.8



SSH aktivieren (damit von externem PC auf Raspi zugegriffen werden kann).

4.9



Dann SD-Betankung über „JA“ starten.
und Warnung mit “JA” zustimmen.

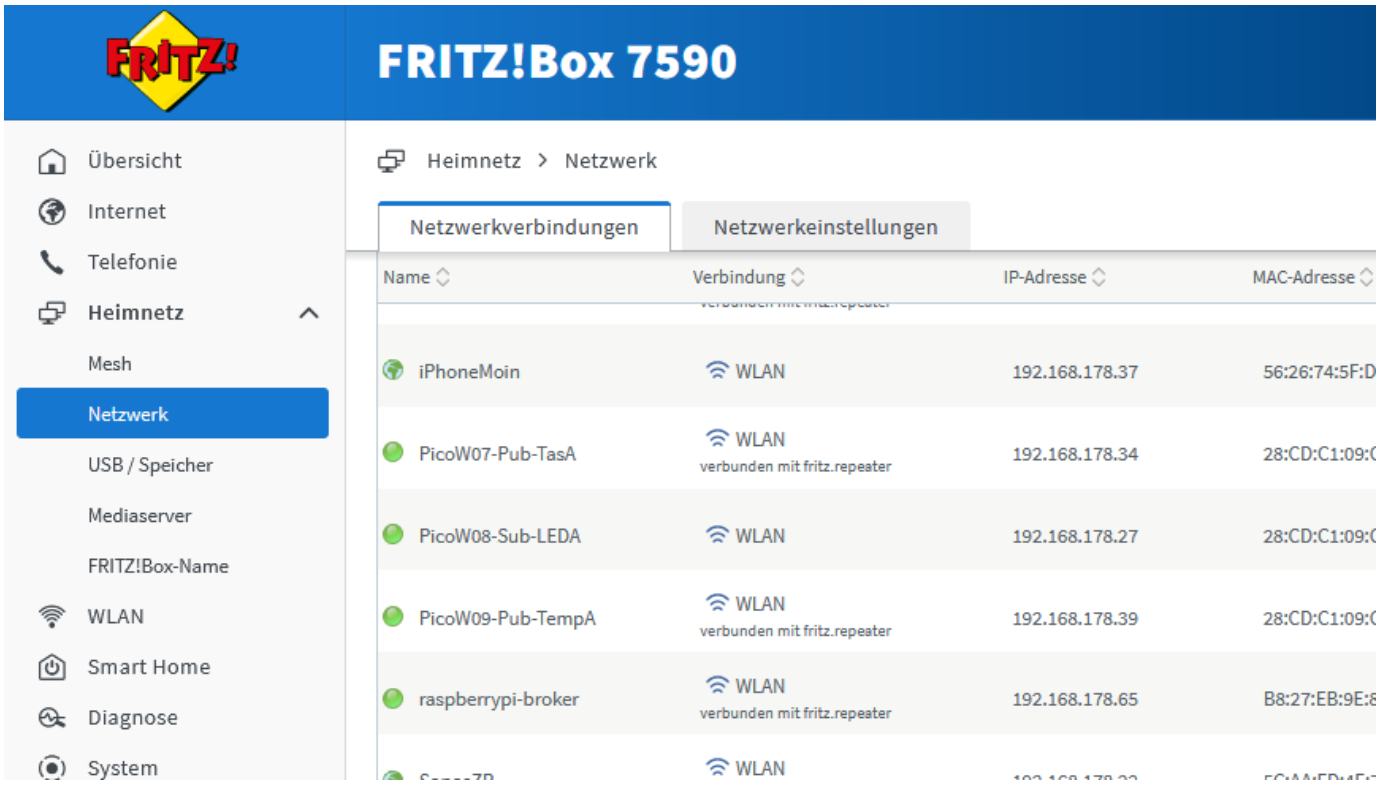
Das anschließende Schreiben und Verifizieren dauert jeweils ca. 4 Minuten.

4.10 SD-Karte in Raspi stecken und Raspi an Stromversorgung anschließen.

5) IP-Adresse von Raspberry Pi 3 ermitteln

5.1 Das Hochfahren (also bis der Raspi im WLAN-Router z.B. FRITZ!Box oder TP-Link auftaucht) kann 4-5 Minuten dauern.

5.2 Beispiel: FRITZ!Box



5.3 Der Raspi 3 arbeitet unter dem unter 4.7 eingegebenen Hostnamen (im Beispiel „raspberrypi.broker“ und IP-Adresse: 192.168.178.65

Diese IP-Adresse wird **benötigt beim PuTTY-Zugriff.**

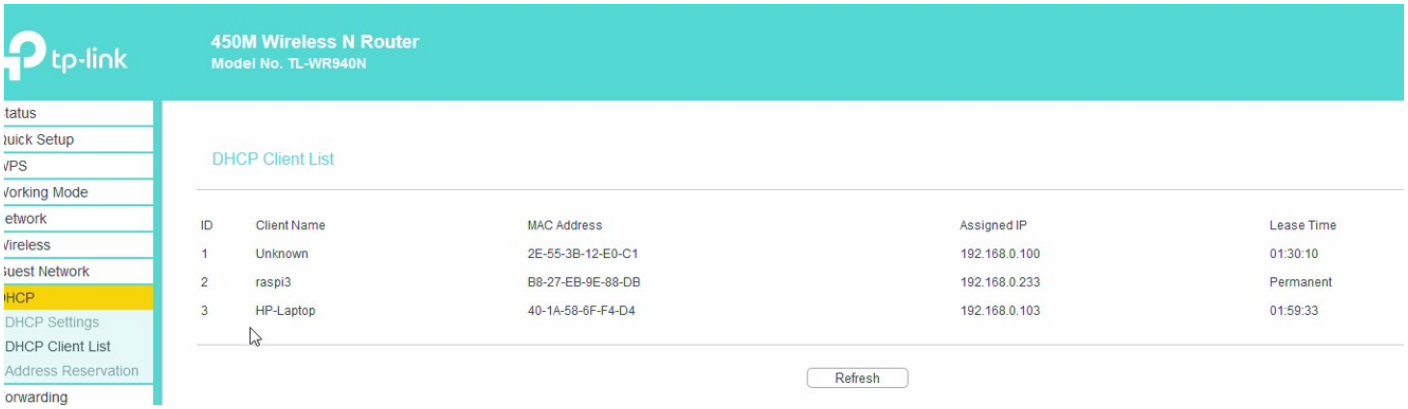
5.4

5.5 Im **Internet-Router sollte dem Raspi (als MQTT-Broker) immer die gleiche IP-Adresse zugeordnet werden,** da die Pico Ws, als MQTT-Publisher und -Subscriber, in ihren Micropython-Programmen den MQTT-Broker unter seiner IP-Adresse ansprechen.

5.6 Beispiel: TP-Link

5.7 Der Raspi 3 arbeitet im Beispiel unter dem Hostnamen „**raspi3**" und mit der IP-Adresse: 192.168.0.233

Diese IP-Adresse wird benötigt beim PuTTY-Zugriff.:



5.8 Im **Internet-Router sollte dem Raspi (als MQTT-Broker) immer die gleiche IP-Adresse zugeordnet werden**, da die Pico Ws, als MQTT-Publisher und -Subscriber, in ihren Micropython-Programmen den MQTT-Broker unter seiner IP-Adresse ansprechen.

6) PuTTY-Zugriff von PC auf Raspberry Pi 3

Nach Chat-GPT:

SSH steht für "Secure Shell" und ist ein Netzwerkprotokoll, das verwendet wird, um eine sichere Verbindung zu einem entfernten Computer herzustellen und eine verschlüsselte Kommunikation zwischen den beiden Systemen zu ermöglichen.

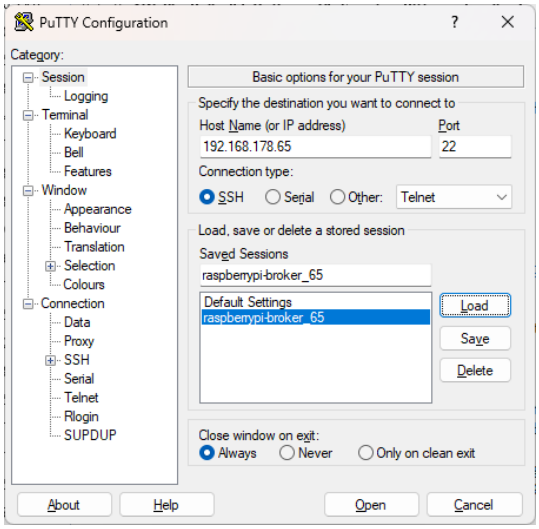
6.1 Mit SSH können Sie einen entfernten Computer fernsteuern, als ob Sie direkt davor sitzen würden. Es bietet eine verschlüsselte Übertragung von Befehlen, Dateien und anderen Daten zwischen Ihrem lokalen Computer und dem Remote-Computer.

Die Funktionsweise von SSH beruht auf dem Client-Server-Modell. Der Remote-Computer (in unserem Fall: der Raspi 3), zu dem Sie eine Verbindung herstellen möchten, muss einen SSH-Server ausgeführt haben (initiiert im Raspberry Pi Imager). Auf Ihrem lokalen Computer (in unserem Fall: ein PC) verwenden Sie einen SSH-Client (in unserem Fall: PuTTY), um eine Verbindung zum Server herzustellen.

6.2 Den SSH-Client “PuTTY” herunterladen von: https://putty.org/

6.3 PuTTY installieren und aufrufen.

6.4



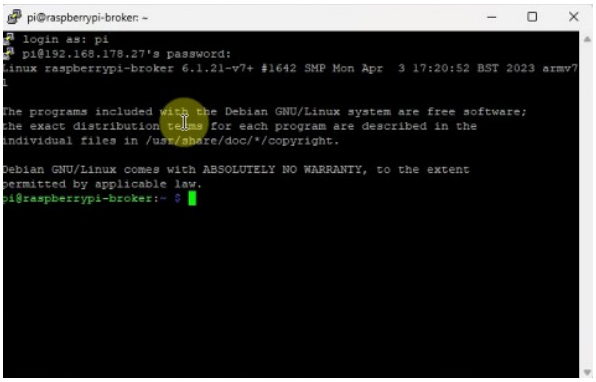
Bei „Host Name“ die über die FRITZ!Box ermittelte IP-Adresse des Raspi 3/4 eintragen,
... im Beispiel die IP-Adresse: 192.168.178.65, oder
... im Beispiel die IP-Adresse: 192.168.0.233

Als Port eintagen: 22

Als „Connection type“ anklicken: „SSH“

IP-Adresse kann per „Save“ gespeichert und per „Load“ aufgerufen werden.

6.5



Wie unter 4.8 angelegt:
z.B. „Pi“ als User eingeben.

Achtung:
bei der Passwort-Eingabe bewegt sich der Cursor nicht ....
dennoch vollständig eingeben und danach Return drücken.

Nach korrekter Eingabe meldet sich „der Raspi“, z.B. mit:
pi@raspberrypi-broker

Vorschlag: kopieren der folgenden Shell-Befehle aus dem PDF mit CTRL+C, und einfügen in (PuTTY-) Kommando-zeile mit Klick auf rechte Maustaste.

Wenn die SD-Karte gewechselt werden soll, sollte der Raspi vorher korrekt heruntergefahren werden:

6.6	`sudo shutdown -h now`	Fährt den Raspberry Pi herunter; "-h" steht für "halt" (Anhalten) und gibt an, dass der Raspberry Pi heruntergefahren und anschließend angehalten werden soll. Zur Inbetriebnahme muss er manuell neu gestartet werden.

7) Mosqitto installieren & testen (ohne Authentifizierung)

7.1	`sudo apt update`	Raspberry Pi OS auf den neusten Stand bringen
7.2	`sudo apt upgrade`	Raspberry Pi OS auf den neusten Stand bringen; bei Abfrage: „y“ bestätigen.
7.3	`sudo reboot`	Raspberry Pi OS neu starten Bei dem Runter-/Rauf-Fahren geht die SSH-Verbindung verloren und muss per PuTTY neu aufgesetzt werden. Das erneute Hochfahren (also bis der Raspi in der FRITZ! Box auftaucht) kann wieder 4-5 Minuten dauern.
7.4	`sudo apt install mosquitto mosquitto-clients`	Mosquitto und MQTT-Clients installieren; Achtung: zwischen den „mosquittos“ ist 1 Leerzeichen; bei Abfrage: „y“ bestätigen.
7.5	`sudo systemctl enable mosquitto`	Bedeutet, dass MQTT-Broker "Mosquitto" automatisch gestartet wird, wenn das System hochgefahren wird. Nach System-Neustart ist Mosquitto automatisch verfügbar.
7.6	`sudo systemctl start mosquitto` `sudo systemctl stop mosquitto`	(ggf. Ausnahme: Mosquitto starten und stoppen)
7.7	`sudo systemctl status mosquitto`	Status vom MQTT-Broker Mosquitto anzeigen lassen; hinter „Active“ sollte „active (running)“ stehen.
7.8	mosquitto_sub -t test	(ggf. zusätzlich: In 2. PuTTY-Fenster aufrufen: starten des MQTT-Subscriber-Client und „subscriben“ am Topic „test“. Subscriber-Client beenden mit STRG + C.
7.9	mosquitto_pub -t test -m "Hallo Welt"	(ggf. zusätzlich: In 3. PuTTY-Fenster aufrufen: starten des MQTT-Publisher-Client und „publishen“ zu Topic „test“ die Nachricht “Hallo Welt”.
7.10	`sudo nano /etc/mosquitto/conf.d/local.conf`	Konfiguration: Kommunikation ohne Authentifizierung; Öffnen der Mosquitto-Konfigurationsdatei ...
7.11	`listener 1883` `allow_anonymous true`	diese Zeilen in Datei einfügen; Datei speichern und schließen: Strg + O, Return, Strg + X.
7.12	`sudo systemctl restart mosquitto`	Mosquitto neu starten, damit Konfiguration wirksam wird.
7.13	mosquitto_sub -t "#" -v	(ggf. zusätzlich: In PuTTY-Fenster aufrufen: starten des MQTT-Subscriber-Client und „subscriben“ an alle Topics: meldet sobald neue Daten empfangen wurden als Logging; Subscriber-Client beenden mit STRG + C.

8) Node-RED installieren

8.1

```
bash <(curl -sL
https://raw.githubusercontent.com/no
de-red/linux-installers/master/deb/
update-nodejs-and-nodered)
```

Holt Installationsskript und führt es aus.
Zu Beginn bittet das Skript um eine Bestätigung des Installationsprozesses und fragt, ob die Raspberry-Pi-spezifischen Nodes installiert werden sollen. Fragen mit (y) beantworten. Danach lädt das Skript die notwendigen Pakete herunter und installiert sie.

Die einzelnen Schritte lassen sich im Terminal verfolgen … und können einige Minuten dauern:

```
Stop Node-RED                       ✓
Remove old version of Node-RED      ✓
Remove old version of Node.js       ✓
Install Node 18.19.0-1nodesource1   ✓   v18.19.0   Npm 10.2.3
Clean npm cache                     ✓
Install Node-RED core               ✓   3.1.3
Move global nodes to local          -
Npm rebuild existing nodes          ✓
Install extra Pi nodes              ✓
Add shortcut commands               ✓
Update systemd script               ✓
```

8.2

```
Node-RED Settings File initialisation
=====================================
This tool will help you create a Node-RED settings file.

✓ Settings file · /home/pi/.node-red/settings.js

User Security
=============
✓ Do you want to setup user security? · No

Projects
========
The Projects feature allows you to version control your flow using a local git repository.

✓ Do you want to enable the Projects feature? · No

Flow File settings
==================
✓ Enter a name for your flows file · flows.json
✓ Provide a passphrase to encrypt your credentials file · *********

Editor settings
===============
✓ Select a theme for the editor. To use any theme other than "default", you will need to install
ib-themes/theme-collection in your Node-RED user directory. · default

✓ Select the text editor component to use in the Node-RED Editor · monaco (default)

Node settings
=============
✓ Allow Function nodes to load external modules? (functionExternalModules) · Yes


Settings file written to /home/pi/.node-red/settings.js
pi@raspi3:~ $ sudo systemctl start nodered
pi@raspi3:~ $
```

8.3	`sudo systemctl start nodered`	Node-RED als Systemprozess starten.
8.4	`sudo systemctl status nodered`	Status von Node-RED anzeigen lassen; hinter „Active“ sollte „active (running)“ stehen.
8.5	`sudo systemctl enable nodered`	Bedeutet, dass Node-RED automatisch gestartet wird, wenn das System hochgefahren wird. Nach System-Neustart ist Node-RED automatisch verfügbar.
8.6	sudo systemctl stop nodered	(Ausnahme: Node-RED stoppen)

9) Zugriff von PC-Browser auf Node-RED Editor auf Raspberry Pi 3

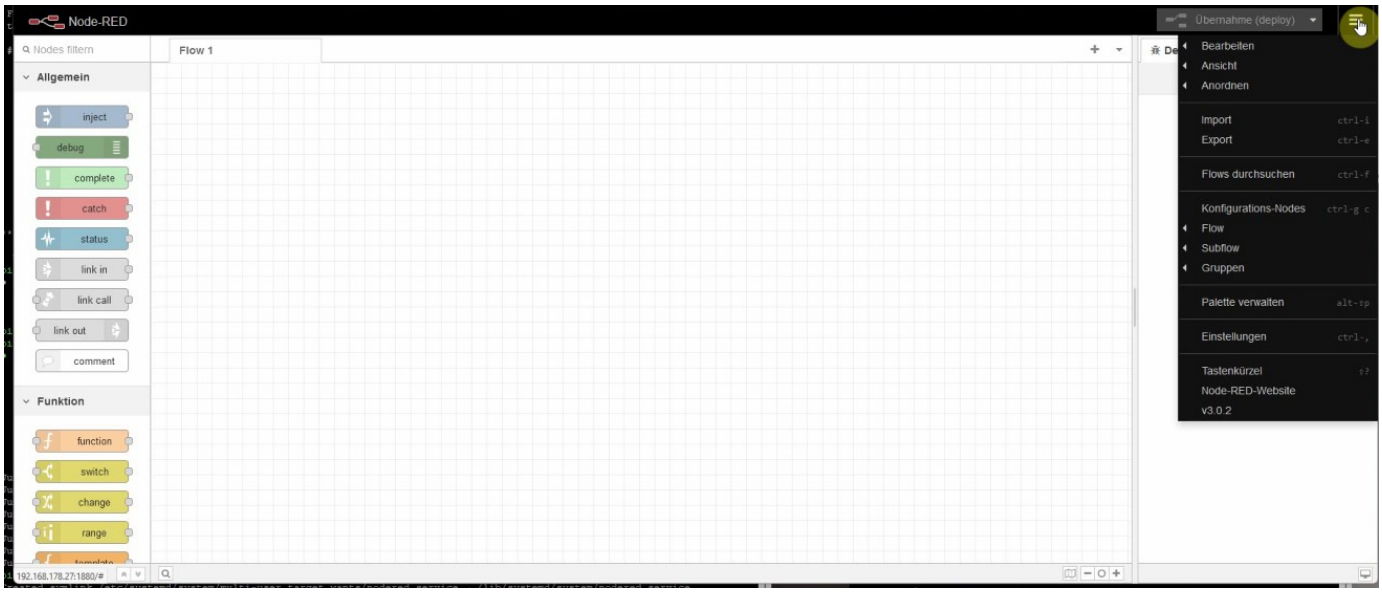
9.1 Node-RED-Aufruf “auf” Raspi-IP-Adresse

9.2 im Beispiel FRITZ!Box: http://192.168.178.65:1880

9.3 im Beispiel TP-Link: http://192.168.0.233:1880

10)Dashboard Nodes importieren

10.1 Oben rechts auf “Drei Striche” klicken ...

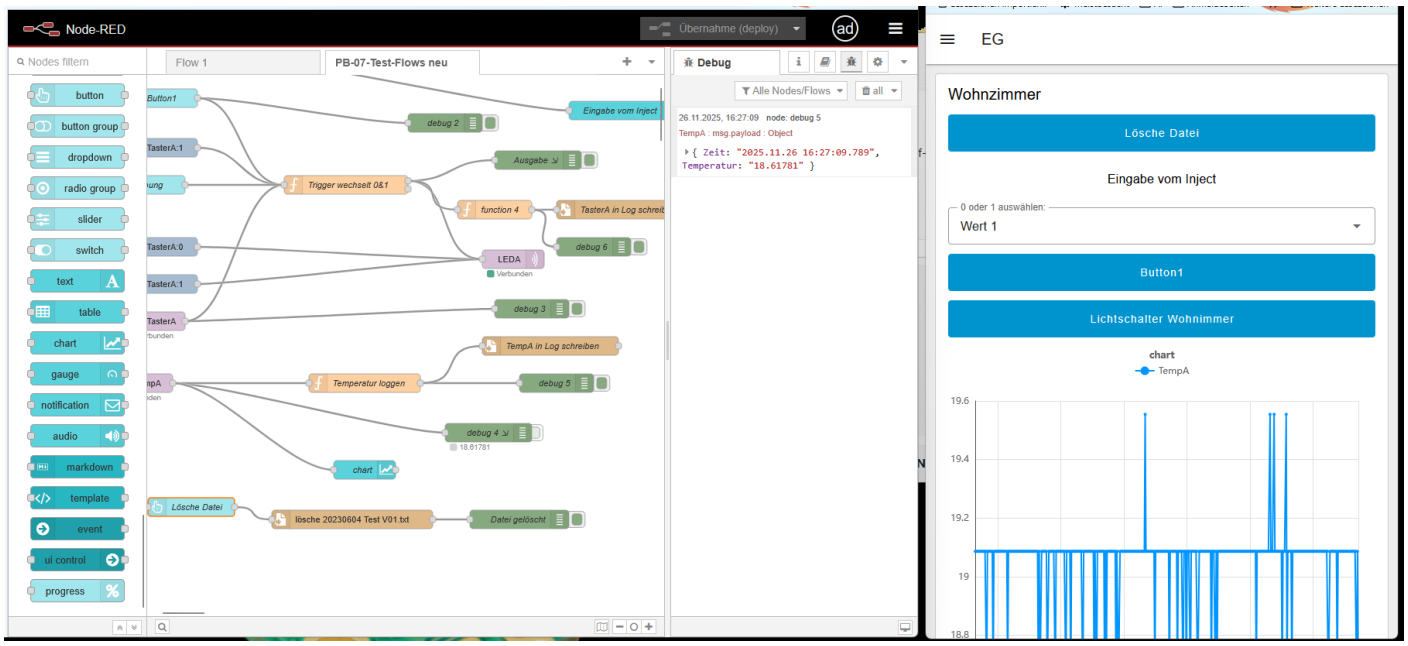


10.2 „Palette verwalten“ aufrufen; dort unter „Installation“ nach „dashbord“ suchen.

10.3 „**@flowfuse/node-red-dashboard**“ installieren.

10.4 Dadurch taucht links die neue Knotengruppe „Dashboard“ mit 21 neuen Knoten (in blauer Farbe) auf.

11)Demo-Flow mit Dashboard Nodes



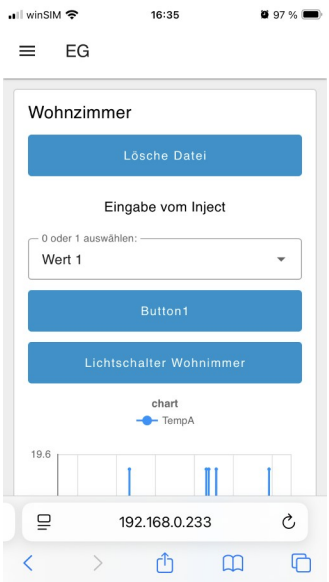
11.1 “button” ist ein Dashboard-Eingabe-Knoten, mit dem ein Vorgang oder eine Eingabe getriggert werden kann, im Beispiel die Nutztlast: “Moin vom Button”.

11.2 “text” ist ein Dashboard-Ausgabe-Knoten, mit dem eine Ausgabe gezeigt werden kann, im Beispiel die Nutztlast timestamp 1685622337617, was die aktuelle Sekunden-Anzahl seit 01.01.1970 bedeutet.

12)Zugriff von PC- oder iPhone-Browser auf Node-RED Dashboard auf Raspberry Pi 3

12.1 Aufruf auf Raspi-IP-Adresse, im Beispiel: http://192.168.0.233:1880/dashboard

12.2



13) MQTT-Publisher-Funktion (Taster; Temperaturmeldung) auf Pico W laden und betreiben

13.1 Für „Taster“ siehe Pico-Programm (aus Datei „**PB-07 Programme gezippt.zip**“):

PB-07-3-10-PicoW07_Pub_Top_TasterA.pyw

```
#=============================================================================
#
# PB-07-3-10-PicoW07_Pub_Top_TasterA.pyw
#
#  1 Taster (TasterA)
#
#  onboard LED:
#  Z47  *  =  WLAN Verbindung erfolgreich hergestellt
#  Z53  **  =  MQTT-Client erfolgreich erstellt
#  Z61  ***  =  "on" ge-publisht
#
#  Z28  MQTT_TOPIC = "TasterA"
#  Z50  client = MQTTClient("PicoW11", MQTT_BROKER)
#
#=============================================================================
#
# Bibliotheken laden
import machine
import network
from time import sleep
from simple import MQTTClient
from Zugang_DC import wlanSSID, wlanPW, IP_MQTT_broker
```

13.2

```
# WLAN-Zugangsdaten und MQTT-Broker-Details
WIFI_SSID = wlanSSID()
WIFI_PASSWORD = wlanPW()
MQTT_BROKER = IP_MQTT_broker()
MQTT_TOPIC = "TasterA"

# Initialisieren des Taster-Pins und LED-Anzeige
button = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP)
LED = machine.Pin("LED", machine.Pin.OUT)

def LED_blinkt(Zahl):
   for Nummer in range(Zahl):
      LED.value(1);sleep(0.3);LED.value(0);sleep(0.2)
   return

# Verbindung zum WLAN herstellen
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_SSID, WIFI_PASSWORD)
while not wifi.isconnected():
   sleep(0.1)

Blinki = 1 ;LED_blinkt(Blinki); print("LED_blinkt ", Blinki);sleep(1)
#  *  =  WLAN Verbindung erfolgreich hergestellt

# MQTT-Client erstellen
client = MQTTClient("PicoW11", MQTT_BROKER)
```

```
Blinki = 2 ;LED_blinkt(Blinki); print("LED_blinkt ", Blinki);sleep(1)
#  **  =  MQTT-Client erfolgreich erstellt

# Hauptprogramm
while True:
   if button.value() == 0:
      client.connect()
      client.publish(MQTT_TOPIC, "Taster A on")
      Blinki = 3 ;LED_blinkt(Blinki); print("LED_blinkt ", Blinki);sleep(1)
      #  ***  =  "on" ge-publisht
      client.disconnect()
      sleep(0.5)
   else:
      sleep(0.1)
```

13.3

Für „Temperaturmeldung“ siehe Pico-Programm (aus Datei „**PB-07 Programme gezippt.zip**“):

PB-07-3-11-PicoW09_Pub_Top_TempA.pyw

13.4

```
#=============================================================================
#
# PB-07-3-11-PicoW09_Pub_Top_TempA.pyw
#
#  1 Taster (TasterA)
#
#  onboard LED:
#  Z47  *  =  WLAN Verbindung erfolgreich hergestellt
#  Z53  **  =  MQTT-Client erfolgreich erstellt
#  Z61  ***  =  "on" ge-publisht
#
#  Z28  MQTT_TOPIC = "TempA"
#  Z50  client = MQTTClient("PicoW11", MQTT_BROKER)
#
#=============================================================================
#
# Bibliotheken laden
import machine
import network
from time import sleep
from simple import MQTTClient
from Zugang_DC import wlanSSID, wlanPW, IP_MQTT_broker
from machine import ADC
# WLAN-Zugangsdaten und MQTT-Broker-Details
WIFI_SSID = wlanSSID()
WIFI_PASSWORD = wlanPW()
MQTT_BROKER = IP_MQTT_broker()
MQTT_TOPIC = "TempA"

# Initialisieren des Tem-ADCs  und LED-Anzeige
Temperatursensor = ADC(4);Umrechnungsfaktor = 3.3 / (65535)
LED = machine.Pin("LED", machine.Pin.OUT)

def LED_blinkt(Zahl):
   for Nummer in range(Zahl):
      LED.value(1);sleep(0.3);LED.value(0);sleep(0.2)
   return
```

```
# Verbindung zum WLAN herstellen
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_SSID, WIFI_PASSWORD)
while not wifi.isconnected():
   sleep(0.1)

Blinki = 1 ;LED_blinkt(Blinki); print("LED_blinkt ", Blinki);sleep(1)
#  *  =  WLAN Verbindung erfolgreich hergestellt

# MQTT-Client erstellen
client = MQTTClient("PicoW09", MQTT_BROKER)

Blinki = 2 ;LED_blinkt(Blinki); print("LED_blinkt ", Blinki);sleep(1)
#  **  =  MQTT-Client erfolgreich erstellt

# Hauptprogramm
while True:
   EinlesewertDigi = Temperatursensor.read_u16()
   Spannung = EinlesewertDigi * Umrechnungsfaktor
   temperatur = 27 - (Spannung - 0.706) / 0.001721
   print("Temperatur (°C): ", temperatur)
   TempStr=str(temperatur)
   client.connect()
   client.publish(MQTT_TOPIC, TempStr)

   Blinki = 3 ;LED_blinkt(Blinki); print("LED_blinkt ", Blinki);sleep(1)
   #  ***  =  "on" ge-publisht
   client.disconnect()
   sleep(5)
```

14)MQTT-Subscriber-Funktion (LED, Relais) auf Pico W laden und betreiben

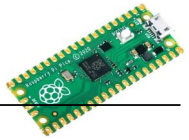
14.1 Für „LED / Relais A“ siehe Pico-Programm (aus Datei „**PB-07 Programme gezippt.zip**“):

PB-07-3-20-PicoW08_Sub_Top_TasterA_RelaisA_Kom.pyw

```
#=============================================================================
#
# PB-07-3-20-PicoW08_Sub_Top_TasterA_RelaisA_Kom.pyw
#
#  1 LED (Relais)
#
#  onboard LED:
#  Z47  *  =  WLAN Verbindung erfolgreich hergestellt
#  Z53  **  =  MQTT-Client erfolgreich erstellt
```

14.2

```
#  Z69  ***  =  mit MQTT verbunden und warten im Hauptprogramm
#
#  Z28  MQTT_TOPIC = "TasterA"
#  Z50  client = MQTTClient("PicoW21", MQTT_BROKER)
#
#=============================================================================
#
# Bibliotheken laden
import machine
import network
```



```
from time import sleep
from simple import MQTTClient
from Zugang_DC import wlanSSID, wlanPW, IP_MQTT_broker

# WLAN-Zugangsdaten und MQTT-Broker-Details
WIFI_SSID = wlanSSID()
WIFI_PASSWORD = wlanPW()
MQTT_BROKER = IP_MQTT_broker()
MQTT_TOPIC = "LEDA"

# Initialisieren des Relais-Pins
relay1 = machine.Pin(14, machine.Pin.OUT)
LED = machine.Pin("LED", machine.Pin.OUT)

def LED_blinkt(Zahl):
   for Nummer in range(Zahl):
      LED.value(1);sleep(0.3);LED.value(0);sleep(0.2)
   return

# Verbindung zum WLAN herstellen
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_SSID, WIFI_PASSWORD)
while not wifi.isconnected():
   sleep(0.1)

Blinki = 1 ;LED_blinkt(Blinki); print("LED_blinkt ", Blinki);sleep(1)
#  *  =  WLAN Verbindung erfolgreich hergestellt

# MQTT-Client erstellen
client = MQTTClient("PicoW08", MQTT_BROKER)

Blinki = 2 ;LED_blinkt(Blinki); print("LED_blinkt ", Blinki);sleep(1)
#  **  =  MQTT-Client erfolgreich erstellt


# Funktion zum Umschalten des Relais
def toggle_relay(topic, msg):
   if msg == b"1":
      relay1.value(1)
   elif msg == b"0":
      relay1.value(0)


# MQTT-Abonnent erstellen und verbinden
client.set_callback(toggle_relay)
client.connect()
client.subscribe(MQTT_TOPIC)

Blinki = 3 ;LED_blinkt(Blinki); print("LED_blinkt ", Blinki);sleep(1)
 #  ***  =  mit MQTT verbunden und warten im Hauptprogramm

# Hauptprogramm - auf Nachrichten warten
while True:
   client.wait_msg()
```

15)Importieren weiterer Node-RED Test-Flows

15.1 In Node-RED importieren (aus Datei „**PB-07 Programme gezippt.zip**“):

20230604 2113 PB-07-Test-Flows.json

16)Anzeigen und Löschen einer Text-Datei auf dem Raspi 3

16.1	ls	Listet Dateien im aktuellen Verzeichnis auf, z.B. '20230604 Test V01.txt'
16.2	sudo nano '20230604 Test V01.txt'	Öffnet Editor und zeigt Datei-Inhalte an. Strg X ist Ausstieg
16.3	rm '20230604 Test V01.txt'	Löscht Datei

17)Raspi 3 neustarten und herunterfahren

17.1	sudo reboot	Alle laufenden Prozesse werden beendet, das Betriebssystem wird heruntergefahren und dann automatisch neu gestartet.
17.2	sudo shutdown -h now	Fährt den Raspberry Pi herunter; "-h" steht für "halt" (Anhalten) und gibt an, dass der Raspberry Pi heruntergefahren und anschließend angehalten werden soll. Zur Inbetriebnahme muss er manuell neu gestartet werden.

18)VPN-Zugriff für iPhone auf FRITZ!Box einrichten

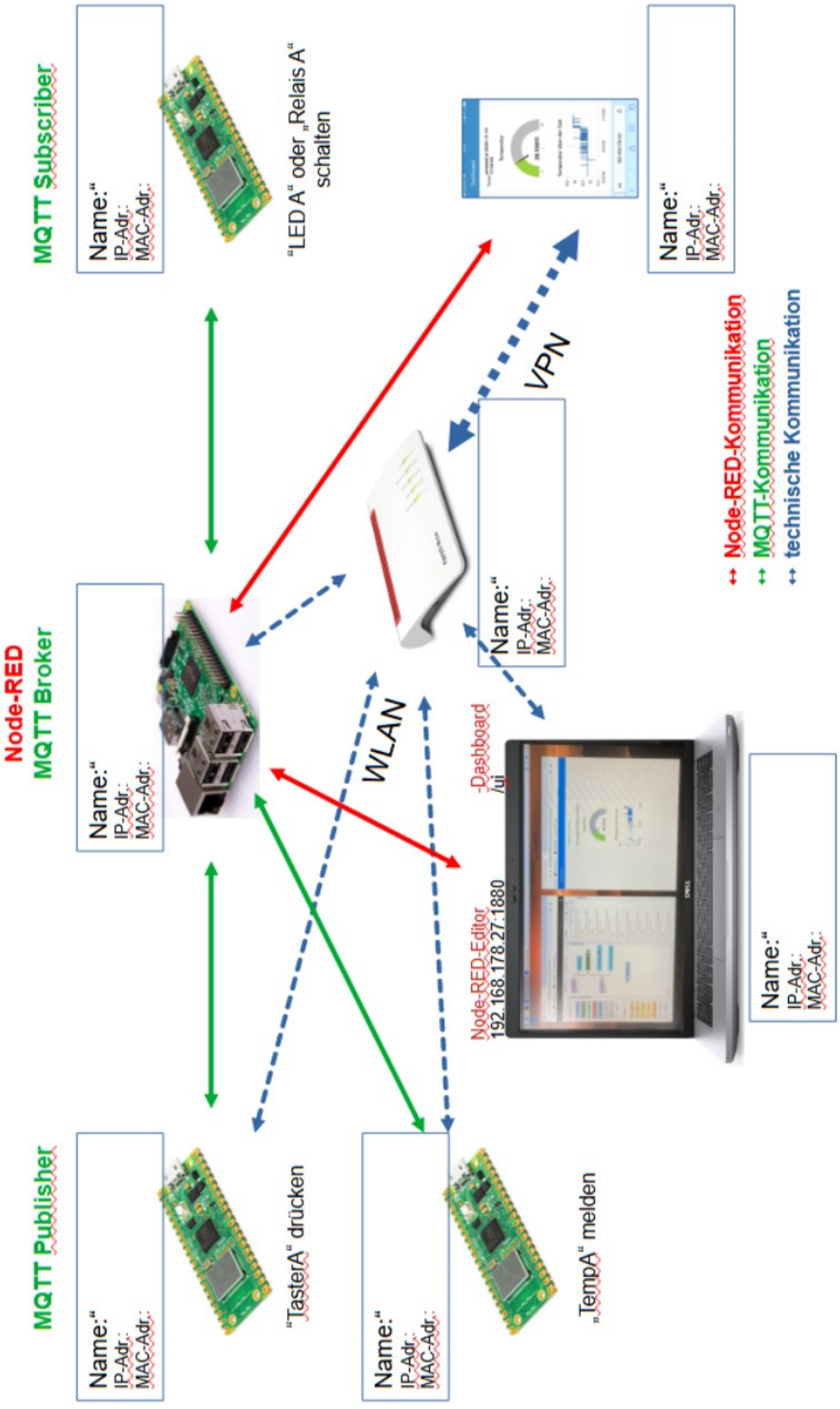
18.1 … entsprechend der Router-Anleitung.

19)Sicherer Zugriff für iPhone auf Node-RED Dashboard

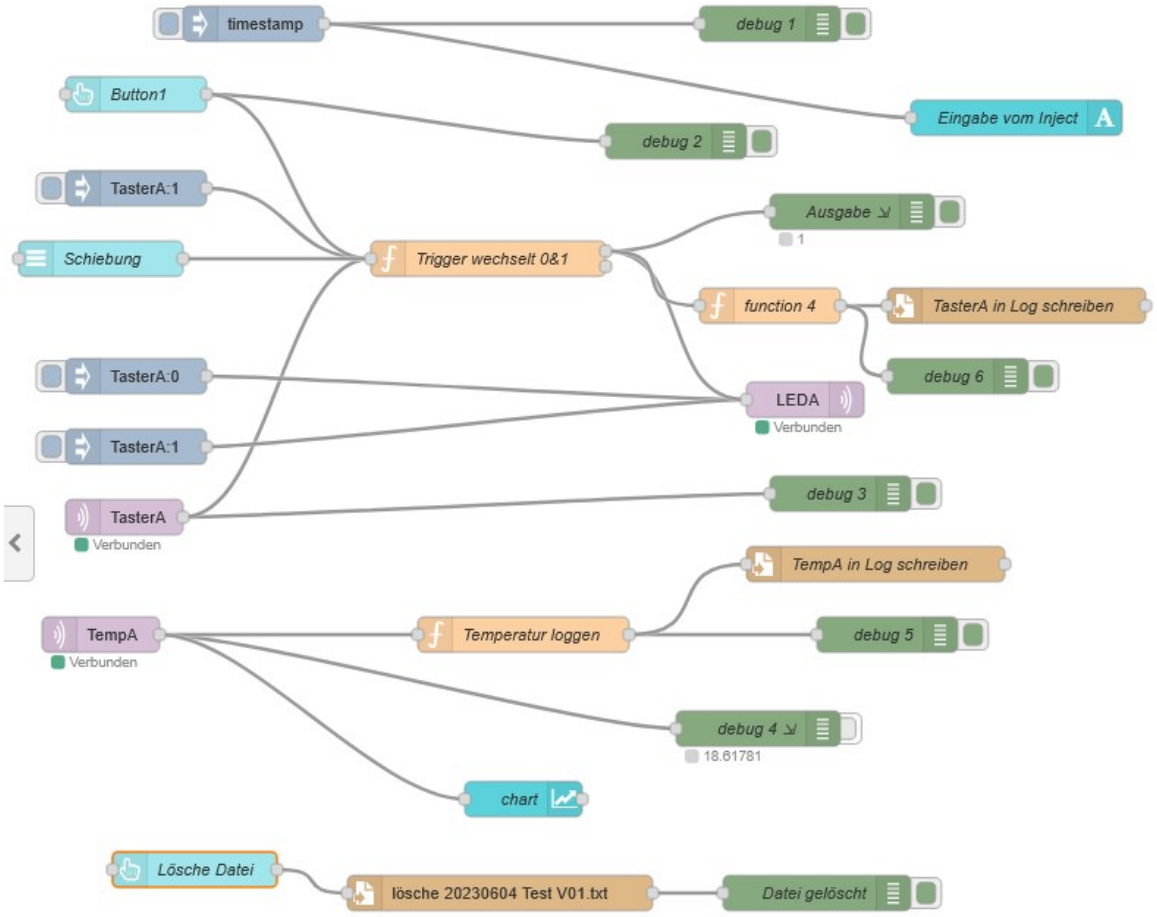
19.1 Aufrufen der VPN-Verdindung im VPN-Client auf dem iPhone.

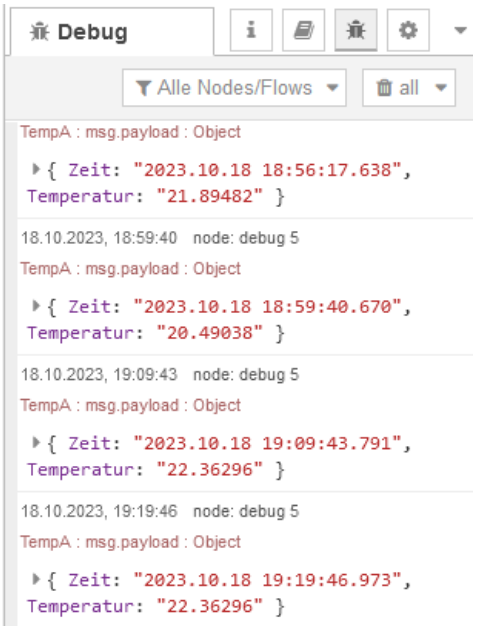
19.2 Aufruf auf Raspi-IP-Adresse, im Beispiel: http://192.168.0.233:1880/ui

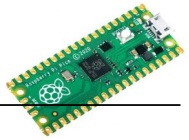
20)Leer-Struktur



21)Node-Red Flows







22)benötigte Dateien

22.1	PB-07-3-10-PicoW07_Pub_Top_TasterA.pyw	Siehe 13.1, zum Betreiben eines Pico W als MQTT-Publishers, welcher im Beispiel das Drücken eines Tasters übermittelt.
22.2	PB-07-3-11-PicoW09_Pub_Top_TempA.pyw	Siehe 13.3, zum Betreiben eines Pico W als MQTT-Publishers, welcher im Beispiel Temperatur misst und übermittelt.
22.3	PB-07-3-20-PicoW08_Sub_Top_TasterA_RelaisA_Kom.pyw	Siehe 14.1, zum Betreiben eines Pico W als MQTT-Subscriber, welcher im Beispiel eine LED ein-/aus-schaltet.
22.4	20251126 PB-07-Test-Flows V02.json	Siehe 15.1 und 21, zum Importieren der unter 21 gezeigten Test-Flows in Node-RED, um Temperaturmessung und Taster zu erfassen, LED zu steuern und Anzeigen und Steuerung im Node-RED Dashboard zu ermöglichen.