



### OpenValveControl

**Imot:** 25.0 mA    **Temp:** 19.6 °C

### Valve Positions

<input checked="" type="radio"/> VZ1	<div></div>	49 %
<input type="radio"/> VZ2	<div></div>	0 %
<input type="radio"/> VZ3	<div></div>	65 %
<input type="radio"/> VZ4	<div></div>	30 %

Set Position [%]

Imot max. [mA]

Move VZ

Home VZ

Info

WLAN-SSID

WLAN-Password

Save & Reboot

/move?vz=1&set\_pos=70&max\_mA=30.0

OpenValveControl Web-UI v0.3 from Klaus Deutschkämmer

# 1 Inbetriebnahme

## Voraussetzungen

- Die Platine ist mit allen erforderlichen Bauteilen bestückt und
- der PIC Controller ist mit der aktuellen Firmware geflasht (z.B. mit einem PICkit4).

### 1.1 Erstmaliges Flashen des „ESP8266 D1-mini“

Zum erstmaligen Flashen des ESP gibt es viele Möglichkeiten. Allen gemeinsam ist der Anschluss des Boards über die USB-Schnittstelle. Eventuell muss dazu zunächst der geeignete Treiber für den auf dem D1 mini verbauten USB-Interface IC installiert werden.

Unter Linux kann zur Überprüfung das Kommando ‚lsusb‘ verwendet werden. Den Namen des USB Ports kann man im Verzeichnis /dev/ nachsehen:

```
~$ lsusb
...
Bus 002 Device 006: ID 1a86:7523 QinHeng Electronics CH340 serial converter
...

~$ ls -l /dev/ttyUSB*
crw-rw---- 1 root dialout 188, 0 Dez 15 14:00 /dev/ttyUSB0

~$
```

#### 1.1.1 Arduino IDE (2.2.1)

Board: LOLIN(WEMOS) D1 mini

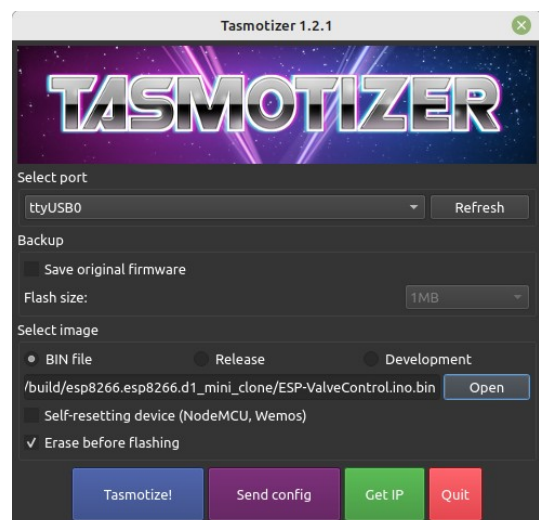
FlashSize: 4 MB (FS: 1MB, OTA: ~1019KB)

#### 1.1.2 tasmotizer

Der „Tasmotizer“ ist eine Python Anwendung und muss deshalb über Python (aktuell python3) gestartet werden. Das Programm tasmotizer.py kann bei github heruntergeladen werden, der Pfad muss dem Speicherort entsprechend angepasst werden:

```
~$ python3 .local/bin/tasmotizer.py
esptool.py v2.8
Serial port /dev/ttyUSB0
Connecting...
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Erasing flash (this may take a while)...
Chip erase completed successfully in 3.1s
Compressed 383456 bytes to 276588...
Wrote 383456 bytes (276588 compressed) at
0x00000000 in 24.4 seconds (effective 126.0
kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```



## Anwender-Handbuch (ENTWURF/DRAFT)

---

In der graphischen Bedienoberfläche einfach USB Port und das binäre Programm (ESP-ValveControl.ino.bin) mit „Open“ auswählen (zum Speicherort navigieren) und „Tasmotize!“ drücken. Der Rest geschieht automatisch.

### 1.1.3 Über Kommandozeile mit „esptool.py“

Das Python Tool „esptool.py“ ist meistens Bestandteil der Arduino IDE, kann aber auch von „Espressiv“ heruntergeladen und installiert werden, falls Python nicht sowieso auf dem Rechner installiert ist. Die Pfadangaben für esptool.py und das ESP-ValveControl Binärfile müssen natürlich wieder angepasst werden:

```
~$ python3 ~/.arduino15/packages/esp8266/hardware/esp8266/3.1.2/tools/esptool/esptool.py -p /dev/ttyUSB0  
write_flash 0x000000 ~/.git/OpenValveControl/ESP-ValveControl/build/esp8266.esp8266.d1_mini_clone/  
ESP-ValveControl.ino.bin  
esptool.py v3.0  
Serial port /dev/ttyUSB0  
Connecting....  
Detecting chip type... ESP8266  
Chip is ESP8266EX  
Features: WiFi  
Crystal is 26MHz  
MAC: 24:a1:60:3a:bb:83  
Uploading stub...  
Running stub...  
Stub running...  
Configuring flash size...  
Compressed 383456 bytes to 276588...  
Wrote 383456 bytes (276588 compressed) at 0x00000000 in 24.4 seconds (effective 125.9  
kbit/s)...  
Hash of data verified.  
  
Leaving...  
Hard resetting via RTS pin...  
~$
```

## 1.2 Konfiguration des WiFi (WLAN)

Hat das Flashen geklappt, sollte in den ersten beiden Zeilen des **OLED Display** folgende Meldung angezeigt werden<sup>1</sup> (als Hinweis, dass der ESP als Accesspoint arbeitet):

```
OVC-Access-Point  
OVC-password
```

Kurz danach wechselt die Nachricht und es wird die eingestellte IP-Adresse angezeigt:

```
IP: 192.168.4.1  
21.1 C 0.0 mA
```

---

1 Falls die OLED Anzeige nicht funktioniert, muss die Hardware und speziell die Verbindung (und Polung) des OLED überprüft werden.

## Anwender-Handbuch (ENTWURF/DRAFT)

---

Jetzt kann man sich per WiFi mit diesem Accesspoint „OVC-Access-Point“ und dem Passwort „OVC-password“ (Groß/Kleinschreibung beachten!) verbinden und in einem Browser die angezeigte IP Adresse eingeben.

Im Browser sollte folgender Hinweis erscheinen:

Keine Dateien ausgewählt.




Lade die fs.html hoch.

<http://192.168.4.1>

Mit „Durchsuchen“ muss die Datei „fs.html“ aus dem ESP-Verzeichnis ausgewählt und anschließend mit „Upload“ in das ESP Filesystem kopiert werden.

Die Seite aktualisiert sich anschließend von selbst und zeigt den ESP8266 Filemanager mit dem Inhalt „fs.html“, der gerade hochgeladenen Datei:

### ESP8266 Filesystem Manager

Keine Dateien ausgewählt.   
   
    
[fs.html](#) 3.51 KB [Download](#) or [Delete](#)  
▲ **LittleFS** belegt 24 KB von 1000 KB

<http://192.168.4.1/fs.html>

Im Anschluss laden wir auf dieselbe Weise die Dateien

„**style.css**“ (css-Formatierungen für fs.html) und

„**index.html**“ (die ESP-Valve-Control Bedienoberfläche)

in das ESP Filesystem hoch:



<http://192.168.4.1/fs.html>

(mit style.css und index.html)

Nun können wir die graphische Bedienoberfläche von ESP-ValveControl anzeigen lassen (sie wird aus dem ESP Filesystem geladen):

**OpenValveControl**

**Imot:** 0 mA    **Temp:** 20.2 °C

**Valve Positions**

VZ1	0 %
VZ2	0 %
VZ3	0 %
VZ4	0 %

Set Position [%]

Imot max. [mA]

**Move VZ**

**Home VZ**

**Info**

WLAN-SSID

WLAN-Password

**Save & Reboot**

`{"ESP": "v0.7", "PIC": "v0.6.1", "dTemp": "0.00", "SSI": "access-point"}`

OpenValveControl Web-UI v0.3.2 from Klaus Deutschkämmer

- ← Hier können wir jetzt die WiFi SSID eintragen,
- ← sowie das WiFi Passwort.
- ← Mit dieser Taste werden die Einstellungen gesichert.

### 1.3 Reboot und Anmeldung im Heimnetzwerk

Nach dem Reboot versucht sich der ESP mit den eingegeben Daten im Heimnetzwerk anzumelden. Gelingt das nicht innerhalb etwa 2 Minuten, wird erneut der Accesspoint aktiviert.

**i** Bei Problemen überprüfen Sie bitte, ob ihr Router so konfiguriert ist, dass sich nur bekannte Geräte verbinden dürfen. Ändern Sie ggfs. vorübergehend diese Einstellung.

Beim Neustart wird in der obersten Zeile des OLED die von Ihnen eingegebene SSID Ihres Heimnetzwerks angezeigt. Während des Verbindungsaufbaus blinkt diese Zeile. In der zweiten Zeile wird die Version der ESP8266 Software ausgegeben:

Beispiel:

```
FRITZ!Box 6390 Cable  
ESP v0.7
```

Nach erfolgreicher Anmeldung wechselt die Anzeige und es wird die eingestellte IP-Adresse angezeigt. In der zweiten Zeile werden Motorstrom und Temperatur<sup>2</sup> ausgegeben.

Beispiel:

```
IP: 192.168.2.131  
19.5 C          0.0 mA
```

Jetzt kann man sich mit dem Heimnetzwerk verbinden und die angezeigte IP-Adresse eingeben. Danach sollte wieder die graphische Bedienoberfläche von ESP-ValveControl angezeigt werden, jetzt allerdings mit der SSID Ihres Heimnetzwerks.

## 1.4 Weitere Einstellungen (ovc.ini)

Nach der Eingabe und Speichern der WiFi-Parameter wird eine Datei „ovc.ini“ angelegt, in der die Einstellungen dauerhaft gespeichert werden. Auch nach einem Firmware-Update sind die Daten normalerweise noch vorhanden, es sei denn, Sie geben an, dass das Flash komplett gelöscht wird (z.B. beim Tasmotizer: „Erase before flashing“).

Ein Verlust ist jedoch kein Problem, Die beschriebene Installation kann jederzeit wiederholt werden.

Als weitere Alternative können Sie die Datei ovc.ini auch auf Ihren Rechner herunterladen und sichern oder auch mit einem Text-Editor ändern und mit weiteren Angaben (z.B. für MQTT) erweitern.

So kann man die Einstellungen auch auf mehrere Geräte duplizieren oder die Eingabe der WiFi Einstellungen im Accesspoint übergehen und gleich mit dem Filesystem Manager eine vorbereitete Datei ovc.ini hochladen. Nach dem Reboot, sind dann gleich alle Einstellungen aktiv.

---

2 Falls kein DS18B20 angeschlossen ist, werden -128,2 C angezeigt.

## Anwender-Handbuch (ENTWURF/DRAFT)

Eine Beispieldatei **ovc.ini** ist in github im Ordner „/ESP-ValveControl/ovc.ini“ enthalten:

```
## ovc.ini
# As default (or when it cannot connect within 2 minutes to the WiFi using the current
# settings), the ESP creates an access point with the credentials shown below.
# Please change the SSID and PSK to meet the settings of your WiFi, then upload the file into
# the ESP's file system, which can be accessed in your browser with the URI <local IP>/fs.html

# WLAN credentials
#SSID = Your SSID
#PSK = Your WiFi password
SSID = OVC-access-point
PSK = OVC-password

# MQTT: server (IP) and token to publish data every period (seconds) as "<mqtt_prefix>/status"
MQTT_HOST   = 192.168.2.72
MQTT_PREFIX = OVC-1
MQTT_PERIOD = 900

# ValveZone aliases (not used yet)
VZ1 = ESS
VZ2 = KÜ
VZ3 = BAD
VZ4 = WZ1

# Adjust temperature sensor
dTemp = -0.5
```

Es bedeuten:

SSID	Der Name Ihres Heimnetzwerks
PSK	Passwort Ihres Heimnetzwerks
MQTT_HOST	IP-Adresse des MQTT Servers.
MQTT_PREFIX	Prefix für die gesendeten Daten (zur Unterscheidung mehrerer identischer Geräte (alle Daten werden mit dem MQTT token "<mqtt_prefix>/status" gesendet).
MQTT_PERIOD	Zeitintervall in Sekunden für das Senden der MQTT Daten („publishing“).
VZ1	Alternative Bezeichner für die Ventil-Zonen (noch nicht verwendet)
VZ2	
VZ3	
VZ4	
dTemp	Korrekturwert für den DS18B20 Temperatur-Sensor

## 1.5 Erste Fahrversuche

**i** Ein angeschlossener Motor ohne Ventil kann über seine Endlage fahren, wenn er beim Fahren keinen Widerstand erfährt. Das kann möglicherweise zu Schäden an der Mechanik führen. Der Betrieb der Steuerung erfolgt auf eigenes Risiko.

### 1.5.1 Auslieferungszustand der Antriebe

Nachfolgendes Foto zeigt einen Antrieb im Auslieferungszustand. Man erkennt, dass der runde Stempel (im Zentrum) einige Millimeter nach innen versetzt ist. Dies entspricht bei meinem Rotex Heizkreisverteiler der Ventilstellung AUF und erleichtert die Montage des Motors.

Beim Referenzieren (Home) fährt der Antrieb den Stempel heraus, im eingebauten Zustand drückt er also den Zapfen des Ventils hinein, was (bei meinem Rotex Heizkreisverteiler) der Ventilstellung ZU entspricht. Bei geschlossenem Ventil drückt der Stempel also gegen die Federkraft des Ventil-Zapfens, weshalb die Montage des Motors in Motorstellung ZU nicht empfehlenswert ist.

**i** Möglicherweise ist das bei anderen Verteilern anders!  
OpenValveControl ermöglicht derzeit keine Änderung des Drehsinns!



### 1.5.2 Überprüfen der Referenzfahrt

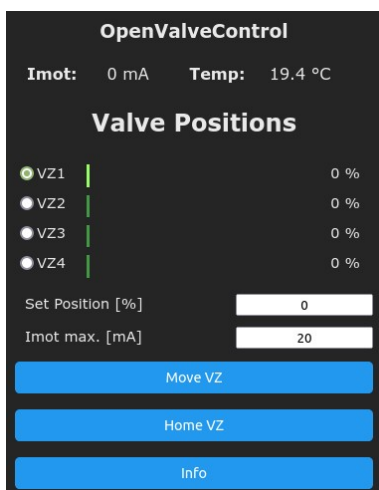
Als erstes sollten Sie eine Referenzfahrt durchführen - OHNE den Motor in den Heizkreisverteiler einzubauen. Am einfachsten geht das auf dem Schreibtisch. Schließen Sie den Motor z.B. an die linke Buchse an, wie im nächsten Bild gezeigt. Dies entspricht der Ventilzone VZ1 in der Bedienoberfläche.





Anschluss eines Motors an VZ1  
zum Test der Referenzfahrt.

- In der Bedienoberfläche wählen Sie nun VZ1 an (durch Anklicken mit der Maus)
- Danach geben Sie bei Imot max. [mA] den (kleinen) Wert 20 ein
- klicken nun auf den Button „HOME VZ“.



Danach sollte der Motor den Stempel „heraus“ fahren und während dessen den Motorstrom anzeigen. Als Indikator leuchtet auch die LED.

Die Referenzfahrt wird beendet, wenn der Motorstrom den eingegebenen Grenzwert überschreitet. Dann wird Ventilstellung „ZU“ angenommen und die Ist-Position auf den Wert 0 % gesetzt.

Dies können Sie bei so niedrigem Grenzwert leicht simulieren, indem Sie mit einem geeigneten Stift oder mit dem Daumen gegen den Stempel drücken, während er herausgefahren wird.

**Achtung!** Der Stempel darf nicht so weit herausfahren, dass die Führungsnase ganz aus der Nut heraus kommt, sonst kann der Stempel nicht mehr ohne manuelle Unterstützung hinein gefahren werden.

Die Referenzfahrt wird deshalb nach ca. 2 Minuten abgebrochen, auch wenn der Stromgrenzwert nicht überschritten wird. Die LED erlischt.

Bei Problemen ziehen Sie einfach rechtzeitig den Stecker!

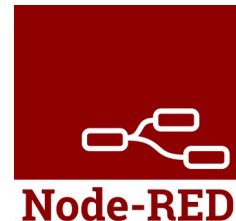
Wenn diese „Referenzfahrt“ funktioniert hat, können Sie dem Motor eine Sollposition von z.B. 20 % angeben und diese Position mit dem Button „Move VZ“ anfahren.

Bleibt der Motor vorzeitig stehen, können Sie den Strom-Grenzwert in kleinen Schritten erhöhen. Ähnlich gehen Sie vor, wenn Sie den Motor eingebaut haben. Zum Vergleich: Bei meinem Rotex Heizkreisverteiler habe ich den Strom-Grenzwert 50 mA eingegeben.

## 2 Ansteuerung mit node-RED<sup>3</sup>

node-RED ist das „Schweizer Taschenmesser“ des Internet-der-Dinge und das ideale Werkzeug, wenn es um Datenaustausch zwischen Sensoren und Aktoren geht. In diesem Kapitel wollen wir OpenValveControl mit einem node-RED flow ansteuern.

„flows“ bezeichnen in node-RED die graphische Darstellungen und Verknüpfungen von verbundenen „nodes“. Ein node kann z.B. eine Schnittstelle zu MQTT darstellen und wird als Block (node) mit Ein- und Ausgängen dargestellt. Über Parameter werden die Eigenschaften dieser nodes definiert (z.B. die IP-Adresse des MQTT servers).



Man kann node-RED z.B. auf einem RaspberryPi installieren, so dass er permanent (24/7) zur Verfügung steht. Auf demselben RaspberryPi kann parallel ein mosquitto-Server als MQTT Host laufen und noch weitere Anwendungen (z.B. grafana).

Bei mir läuft diese Konfiguration seit Jahren problemlos auf einem Pi der 3. Generation.

### 2.1 Ventil verstellen mit node-RED

#### 2.1.1 Aufgabenstellung

Wie wir gesehen haben, können wir unsere Ventile über die Web-Schnittstelle verstellen. Dazu müssen wir die IP-Adresse des Web-UI kennen.

Dann können wir die VentilZone (den Antrieb bzw. das Ventil) auswählen, die Soll-Position eingeben (0 - 100%), den maximalen Motorstrom in Milliampere eingeben und den Button „Move“ betätigen.

Dadurch wird ein „http GET“-Befehl an den Server gesendet, in dem die erwähnten Parameter enthalten sind. Der Befehl wird im Klartext am unteren Bildschirmrand des Web-UI angezeigt.

#### Beispiel

Ventil 1 (vz=1) soll auf 26% gefahren werden. Falls der Motorstrom den Wert 26 mA überschreitet, soll der Vorgang abgebrochen werden (Blockierschutz):

```
/move?vz=1&set_pos=26&max_mA=30.0
```

Genau dasselbe erreichen wir, wenn wir in unserem LAN den GET-Befehl über einen Web-Browser senden. Wir müssen lediglich die IP-Adresse unserer Steuerung voranstellen (die IP-Adresse wird nach erfolgreicher Anmeldung im Heimnetzwerk im OLED angezeigt):

```
http://192.168.2.131/move?vz=1&set_pos=26&max_mA=30.0
```

---

3 **node-RED** is a flow-based programming tool, originally developed by IBM's Emerging Technology Services team (<https://emerging-technology.co.uk/>) and now a part of the OpenJS Foundation (<https://openjsf.org/>).

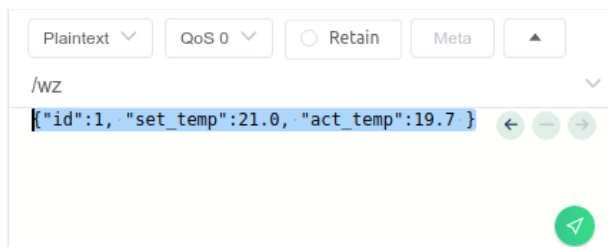
### 2.1.2 Sollwert vorgeben

Um die Aufgabe praxistauglich zu lösen, wollen wir zunächst die Vorgabe des Sollwerts lösen, wir wollen ja nicht alle Einstellungen ständig mit dem Smartphone vornehmen.

Wir nehmen an, dass wir in unserem Wohnzimmer für eine brauchbare Einzelraumregelung einen Temperatursensor mit Anzeige sowie eine Möglichkeit für eine Sollwertvorgabe haben. Dies könnte durch einen kleinen ESP realisiert werden oder mit einem käuflichen Industrieprodukt (Anzeige Ist- und Solltemperatur, 2 Tasten  $\updownarrow$  für den Sollwert).

Dieser ESP könnte nun über WiFi direkt an unsere Steuerung den beschriebenen GET Befehl senden und z.B. das Ventil auf 100% steuern wenn es im Raum zu kalt ist oder auf 0%, wenn es zu warm ist (einfache „Zweipunkt“-Regelung). Zusätzlich muss natürlich jedem „Thermostaten“ die Ventilzuordnung, der maximale Strom und evtl. der Regelalgorithmus mitgeteilt werden. Es gibt aber noch weitere Informationen, die für die Regelung nützlich sind, z.B. Außentemperatur, Sonneneinstrahlung, Nachtab senkung, längere Abwesenheit (Urlaub) und andere. Deshalb macht es Sinn, nur den Istwert und die Solltemperatur des jeweiligen Raumes zu melden und alle Räume zentral zu verwalten, z.B. mit node-RED.

Dazu senden wir die Sensor-ID, den Temperatur Ist- und Sollwert an einen MQTT Server (auch als Broker bezeichnet, z.B. unseren mosquitto). Bis wir den Sensor programmiert haben, können wir das mit dem Tool MQTTX<sup>4</sup> simulieren:



Wenn wir mit dem MQTT Broker verbunden sind (z.B. als test@raspi3:1880), können wir einen **Topic** (z.B. „/wz“) definieren und die eingegebene **Payload** durch Anklicken des grünen Pfeils an den MQTT Broker senden. Die Daten sind beliebig änderbar.

### 2.1.3 flow: Sollwert vom MQTT Broker „subscriben“

In node-RED platzieren wir nun einen „mqtt in“ node in unseren „Flow“. Ein Doppelklick auf den node öffnet das „Properties“-Fenster, in das wir folgende Daten eintragen:

Als **Server** habe ich „localhost:1883“ eingetragen, weil sowohl node-RED als auch mosquitto (der MQTT broker) auf demselben RaspberryPi laufen. Andernfalls muss statt „localhost“ die entsprechende IP-Adresse des MQTT Servers angegeben werden.

---

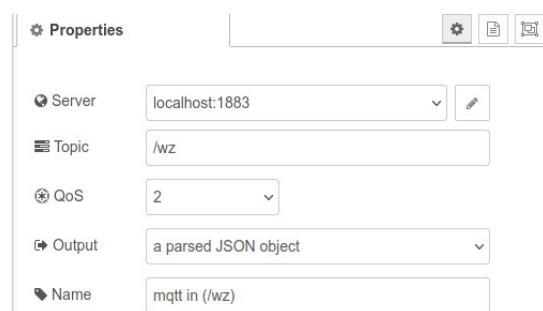
4 „Your All-in-One MQTT Client Toolbox“ (<https://mqttx.app/>). MQTTX is a free open-source project available under the [Apache-2.0 LICENSE](#).

Den **Topic** „/wz“ können wir im Thermostaten (bzw. Temperatursensor) vergeben. Wir haben dies testweise in MQTTX so eingetragen und dient zur Unterscheidung der verschiedenen Räume bzw. Temperatur-Sensoren.

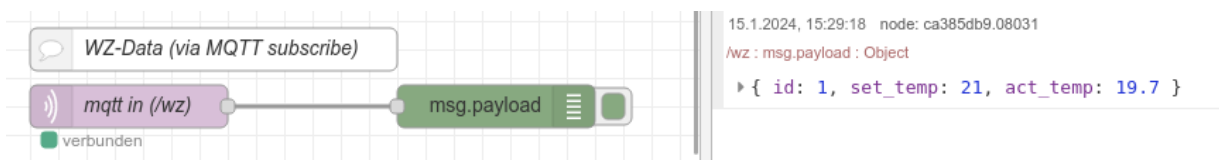
Quality of Service „**QoS**“ bezeichnet den zunehmenden Grad der Zuverlässigkeit bei der Zustellung von Nachrichten, der höchste Wert ist 2.

Als **output** wollen wir ein „parsed JSON object“, somit dieselbe Nachricht, wie sie vom Sensor zum MQTT Broker gesendet wurde.

Der **Name** ist frei wählbar und dient vor allem der Übersichtlichkeit des Flows.



Wir fügen nun einen „debug“ node hinzu, so dass wir die Ausgabe des „mqtt in“ nodes kontrollieren können. Sobald wir in MQTTX einen Datensatz an den MQTT Broker senden, werden die abonnierten Daten an node-RED gesendet und stehen am Ausgang des „mqtt in“ nodes zur Verfügung:



#### 2.1.4 flow: Temperatur-Differenz in Ventil-Position wandeln

Wir kennen nun den Raum und die Soll- und Ist-Temperatur und können nun eine Entscheidung für die Ventilposition treffen.

Zur Vereinfachung nehmen wir die Raum-ID als VentilZone (vz) und programmieren den beschriebenen Zweipunkt-Regler, den wir in JavaScript so formulieren wollen:

Bei mehr als 0,5 Grad Abweichung schalten wir das Ventil zu (20%) bzw. auf (80%), andernfalls lassen wir es wie es ist.

Anmerkung: Die vereinfachte Annahme ist, dass die Vorlauftemperatur witterungsgeführt ist und wir mit einem Ventil lediglich eine Feinjustierung vornehmen. Natürlich können wir hier auch einen beliebig komplizierten Regelalgorithmus programmieren!

## Anwender-Handbuch (ENTWURF/DRAFT)

Dafür gibt es in node-RED den „function“ node, in dem fast beliebig JavaScript ausgeführt werden kann. Als Eingang wird eine „message“ angenommen, die durch das Programm (fast) beliebig modifiziert und an einem oder mehreren Ausgängen ausgegeben werden kann. Dazu dient die Anweisung `return (x)`, wobei x auch `null` sein kann, dann wird nichts ausgegeben. Dies nützen wir aus, um keinen Verstellvorgang auszuführen, wenn die Temperatur einigermaßen stimmt.

Zu Beginn des JavaScript Programms hat unsere message z.B. folgende Werte:

```
msg.payload = {"id":1,"set_temp":21,"act_temp":19.7}
```

Als Ausgang wollen wir die passenden Parameter für den Fahrbefehl via „http GET“ haben:

```
msg2.payload = {"id":1, "set_pos":80, "max_mA":50.0}
```

Dazu müssen wir lediglich eine neue message msg2 generieren. Den fehlenden Wert für „max\_mA“ nehmen wir vorläufig aus einer Variablen, die z.B. irgendwie im Flow mit der Anweisung `flow.set(„max_mA“, <wert>)` gespeichert wurde.

Unser JavaScript Programm könnte dann etwa so aussehen:

The screenshot displays the Node-RED web interface. At the top, a flow is visible with three nodes: 'WZ-Data (via MQTT subscribe)', 'mqtt in (/wz)', and 'control script'. The 'control script' node is selected, and its configuration panel is open on the left. The 'Name' field is set to 'control script'. The 'Funktion' field contains the following JavaScript code:

```
1 msg2 = {};  
2  
3 max_mA = flow.get('max_mA') || 49.5;  
4 vz = msg.payload.id;  
5 set_temp = msg.payload.set_temp;  
6 act_temp = msg.payload.act_temp;  
7  
8 if ((set_temp - act_temp) > 0.5)  
9 {  
10   // too cold:  
11   set_pos = 80;  
12 }  
13 else if ((act_temp - set_temp) > 0.5)  
14 {  
15   // too warm  
16   set_pos = 20;  
17 }  
18 else  
19 {  
20   // acceptable  
21   return(null);  
22 }  
23 msg2.payload = { "vz":vz,  
24                 "set_pos":set_pos,  
25                 "max_mA":max_mA  
26 };  
27 return msg2;
```

On the right side, the 'Debugging' console shows two log entries. The first entry, at 15.1.2024, 17:25:52, shows the initial message payload: `{ vz: 2, set_pos: 20, max_mA: 49.5 }`. The second entry, at 15.1.2024, 17:26:03, shows the modified message payload: `{ vz: 2, set_pos: 80, max_mA: 49.5 }`.

## Anwender-Handbuch (ENTWURF/DRAFT)

Die 2 Ausgaben im Debugger waren die Antworten auf folgende 3 Nachrichten:

```
{"id":2, "set_temp":18.0, "act_temp":19.7 } // too warm
{"id":2, "set_temp":22.0, "act_temp":19.7 } // too cold
{"id":2, "set_temp":20.0, "act_temp":19.7 } // acceptable
```

Man kann auf diese Weise sehr schön testen, dass bei der dritten Eingabe wie beabsichtigt keine Ausgabe erfolgt.

## Anmerkung

Um zu verhindern, dass beim Lesen der Variablen max\_mA eine Fehlermeldung auftritt oder ein ungültiger Wert verwendet wird, wenn (noch) kein Wert gespeichert ist, wurde ein logisches UND angehängt, was in diesem Fall einen Default-Wert zuweist:

```
max_mA = flow.get('max_mA') || 49.5;
```

### 2.1.5 flow: Query Parameter als http GET ausgeben

Zum Senden des „http request“ gibt es einen weiteren („network“) node. Wir setzen ihn an den Ausgang des JavaScript „function“ node und verbinden die beiden. Den Debug node verschieben wir an den Ausgang des http requests. Die Properties füllen wir wie folgt aus:

The screenshot displays the 'http request Node bearbeiten' dialog on the left and the 'Debugging' panel on the right.

**http request Node bearbeiten:**

- Buttons: Löschen, Abbrechen, Fertig
- Properties section:
  - Methode: GET
  - URL: http://192.168.2.131/move?
  - ☒ Append msg.payload as query string parameters
  - ☐ Sichere Verbindung (SSL/TLS) aktivieren
  - ☐ Basisauthentifizierung verwenden
  - ☐ Enable connection keep-alive
  - ☐ Use proxy
  - Rückgabe: eine UTF-8-Zeichenfolge
  - Name: http GET

**Debugging:**

- Buttons: Alle Nodes, [icon]
- Log entries:
  - 15.1.2024, 17:52:35 node: ca385db9.08031  
msg.payload : string[34]  
▶ "/move?vz=1&set\_pos=80&max\_mA=49.5"
  - 15.1.2024, 17:53:54 node: ca385db9.08031  
msg.payload : string[34]  
▶ "/move?vz=1&set\_pos=20&max\_mA=49.5"

## Anwender-Handbuch (ENTWURF/DRAFT)

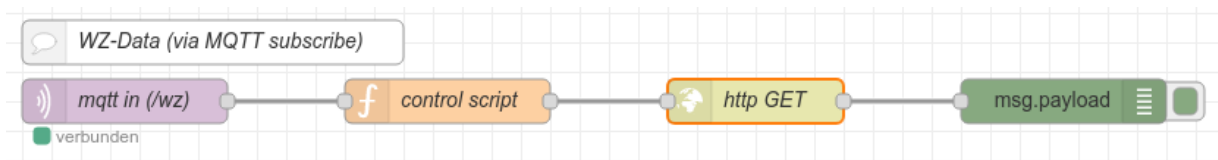
---

Die Debug Ausgaben beim Publishen unserer Messwerte zeigen, dass zur URL mit der Ergänzung „/move?“ für den Fahrbefehl alle erforderlichen Parameter aus der message angehängt wurden.

Entsprechend kann man auch die Befehle „home“ oder „info“ oder „status“ realisieren und in node-RED weiterverarbeiten.

Den Debug node darf man natürlich löschen oder deaktivieren, wenn man ihn nicht mehr braucht.

Damit ist die Aufgabe gelöst und wir haben unseren fertigen Flow:



@todo Translation.

to be continued