

# \*\*\_Die Webseite ist umgezogen\_\*\*

\* [Frostwächter: Sonoff TH16 mit Si7021 \(neues Wiki\)](#)

## Frostwächter: Sonoff TH16 mit Si7021

**Quelle ⇒ hier wurde  
gespickelt**

<https://forum.creationx.de/lexicon/index.php?entry/24-rules/>  
<https://tasmota.github.io/docs/Rules/#long-press-on-a-switch>

Für den Frostschutz einer Pumpe im Außenbereich wurde ein Frostwächter benötigt. Als Heizung wird ein [PTC-Heizelement](#) eingesetzt. Aus Sicherheitsgründen wurde noch ein klassischer [Thermostat](#) eingebaut, damit der Pumpenraum nicht unnötig aufgeheizt wird. Die eigentliche Regelung übernimmt ein [SonOff TH16 mit SI7021](#). Die Regelung kann per Variablen angepasst werden. Ist damit auch für andere Heizungsaufgaben geeignet.

## V1 Heizung (08/15) Sekundentakt

- EIN bei Temp <3 Grad
- AUS bei Temp >5 Grad

Rule1

```
ON SI7021#Temperature<3 DO power1 1 ENDON
```

```
ON SI7021#Temperature>5 DO power1 0 ENDON
```

## V2 Heizung Regelung

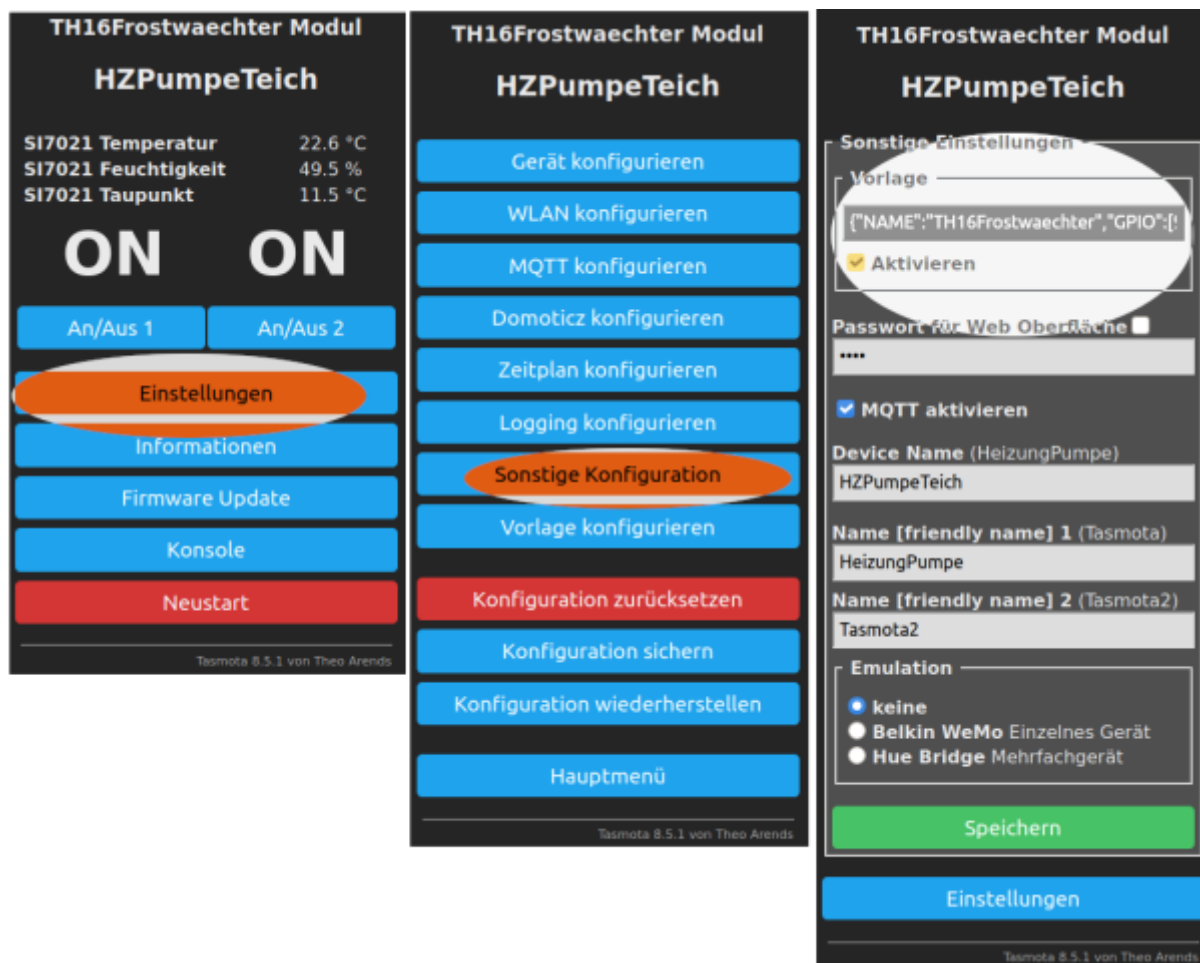
(Vorlage: [9. Einfaches Thermostat Beispiel](#))

- EIN bei Temp <3 Grad
- AUS bei Temp >4 Grad
- Automatik und Hand Betrieb

# TH16 mit eigenem Profi ausstatten (nur damit funktioniert die RULE1)

## Profil

- Der Taster muss als Switch konfiguriert werden (Switch1 (9))
- Die Blaue LES als Relay2 (dadurch zwei Schalter im WEB) (Relay 2i (30))
- Sensor fest eingebunden. (SI7021 (3))



## Einstellungen -- Vorlage

```
{"NAME": "TH16Frostwaechter", "GPIO": [9,255,255,255,255,0,0,0,21,30,3,0,0], "FLAG": 0, "BASE": 4}
```

## Info: Blaue und Rote LED

Blaue LED wird über GPIO13 gesteuert (Default: LED1i (56) Status Relais ⇒ geändert nach Relais 2i (30))

Blau LED wird als Relais eingerichtet, damit der Automatikmodus angezeigt werden kann  
Rote LED zeigt den Zustand vom Relais an (GPIO12)

# Heizungssteuerung

## Vorbereitung

auf der Konsole sind einige Werte zu Konfigurieren.

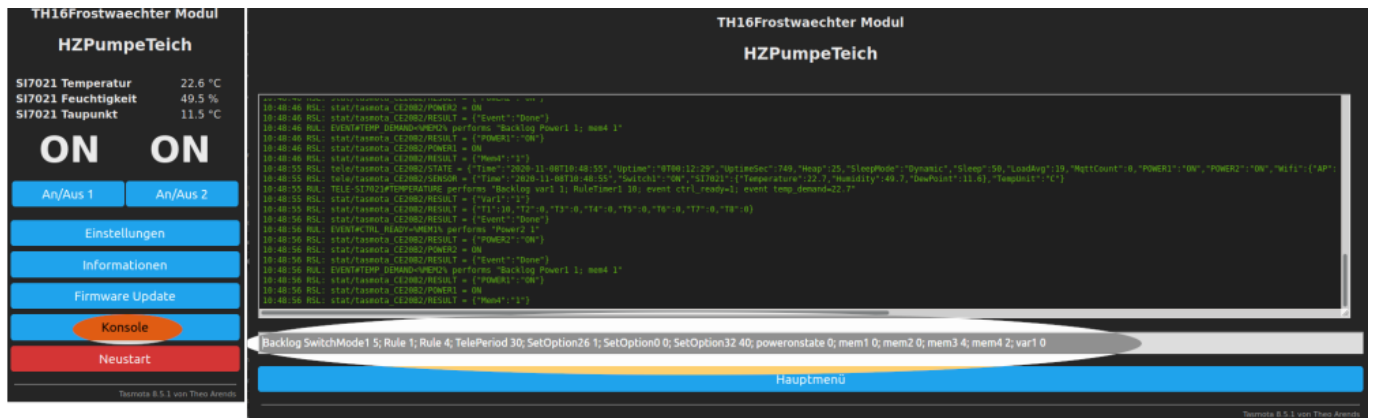
- Alle Werte können an der Konsole gesetzt werden (z.B. Mem3 5)
- oder per MQTT (z.B. cmd/mqttTopic/mem3)

## Basis Parametrierung

Die nächste Zeile **muss** einmalig an der Konsole ausgeführt werden! (Copy & Paste)

an der Konsole

```
Backlog SwitchModel 5; Rule 1; Rule 4; TelePeriod 30; SetOption26 1;
SetOption0 0; SetOption32 40; poweronstate 0; mem1 0; mem2 0; mem3 4;
mem4 2; var1 0
```



## Definition der Variablen

- mem3 5  $\Leftarrow$  maximale Temperatur Power OFF (>5 Grad)
- mem4 3  $\Leftarrow$  minimale Temperatur Power ON (<3 Grad)
- mem1 0/1 Aus/Ein der Regelung
- mem2 0/1 Relais Manuell Aus/Ein
- var1  $\Leftarrow$  aktueller Status vom Regelung 1-OK 0-NOT READY - View by MQTT cmd/mqttTopic/var1

## switchmode1

switchmode1 5 ⇐ damit wird kurzer Tastendruck als TOGGE und langer Tastendruck als HOLD aktiviert

Abfrage in der RULE:

- Switch1#State ⇐ kurzer Tastendruck
- Switch1#State=3 ⇐ langer Tastendruck

!!Langer Tastendruck löst auch kurzen Tastendruck aus!!

## TelePeriod

Die Funktion TelePeriod 30 stellt einen 30 Sekunden Timer.

Alle 30 Sekunden werden die Werte die mit einem tele-xxxx beginnen abgefragt.

tele-SI7021#temperature ⇐ die Temperatur alle 30 Sekunden einlesen.

TelePeriod 0 ⇐ damit wird tele-xxxx ausgeschaltet.

## Messergebniss

event temp\_demand=%value% ⇐ in der Variable „temp\_demand“ wird der Wert %value% gespeichert.

%value% wird von tele-SI7021#temperature befüllt.

Falls man mehrere Werte abfragen möchte, müssen die sofort im Anschluss in einer Variable gespeichert werden

## Setoption

- SetOption26 1 ⇒ Status Relais wird um den Index erweitert „power1“ und „power2“ anstatt nur „power“)
- SetOption0 0 ⇒ Status Relais nicht im EPROM abspeichern (schont das EPROM)
- SetOption32 40 ⇒ langer Tastendruck auf 4 Sekunden (Automatik ein/aus)

## RULE(x) ⇒ Regeln

RULE == RULE1 ⇐ kompatibilitäts- Modus zu älteren Softwareständen als es nur eine RULE gab Steuern/Beeinflussen der Regeln (RULE1) z.B an der ersten RULE:

- Rule1 0 = Regel ausschalten (Off)
- Rule1 1 = Regel einschalten (On)
- Rule1 2 = Umschalten (Toggle) Regel off ⇔ on

- Rule1 4 = Befehl solange ausführen wie der Trigger stimmt (Once OFF)
- Rule1 5 = Perform commands once until trigger is not met (Once ON)
- Rule1 6 = Toggle Once state

## Regel

Die Regel kann mit Copy & Paste einfach in die Konsole kopiert werden. Der Mehrzeiler wird automatisch zum Einzeiler 😊

zum Temp Sensor SI7021 V2

```
Rule1
ON system#boot D0 RuleTimer1 70 ENDON
ON Switch1#State D0 event toggling2=%mem2% ENDON
ON event#toggling2=0 D0 Backlog mem2 1; Power1 1 ENDON
ON event#toggling2=1 D0 Backlog mem2 0; Power1 0 ENDON
ON Switch1#State=3 D0 event toggling1=%mem1% ENDON
ON event#toggling1=0 D0 Backlog mem1 1;TelePeriod 30; Power2 1 ENDON
ON event#toggling1=1 D0 Backlog mem1 0;TelePeriod 0; Power2 0 ENDON
ON tele-SI7021#temperature D0 Backlog var1 1; RuleTimer1 10; event
ctrl_ready=1; event temp_demand=%value% ENDON
ON event#ctrl_ready>%mem1% D0 Backlog Power2 0; var1 0 ENDON
ON event#ctrl_ready=%mem1% D0 Power2 1 ENDON
ON event#temp_demand>%mem4% D0 Backlog Power1 0; mem2 0 ENDON
ON event#temp_demand<%mem3% D0 Backlog Power1 %var1%; mem2 %var1%
ENDON
```

## TH16 mit Tasmota Flashen

Die Vorbereitung der Hardware wurde von [bastelgarage.ch](https://bastelgarage.ch) übernommen.

Der Flash Vorgang wird unter Linux (Windos geht auch) durchgeführt. Dazu kommt die Software [esptool.ph](https://esptool.ph) zum Einsatz. In der Praxis hat sich das löschen der alten Software bewährt

## Download vom Tasmota Version 8.5.0 DE

```
cd /tmp
wget
https://github.com/arendst/Tasmota/releases/download/v8.5.1/tasmota-DE.bin
```

## Löschen des Flashspeicher auf dem TH16

```
# ./esptool.py --port /dev/ttyUSB0 erase_flash
```

## Schreiben von Tasmota in den Flashspeicher

```
./esptool.py -p /dev/ttyUSB0 write_flash -fs 1MB -fm dout 0x0 /tmp/tasmota-DE.bin
```

## Tasmota: der Trick mit dem Umschalten

Damit der Zustand des Relay (der Funktion) mit dem Taster umgeschaltet werden kann, kommt das toggelingX und die Variable memX zum Einsatz.

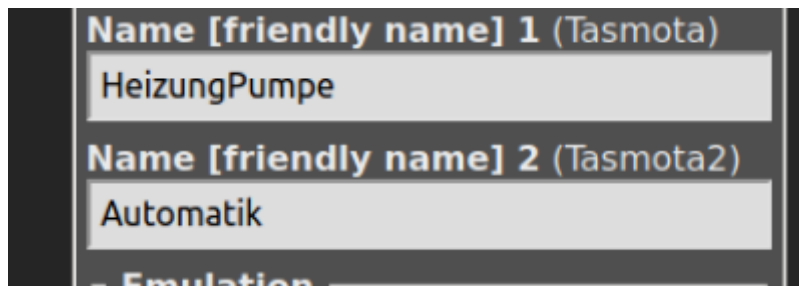
1. ON Switch1#State DO event toggling2=%mem2% ENDON ← Tastendruck erkennen und den Wert von mem2 in toggeling2 speichern
  2. ON event#toggling2=0 DO Backlog mem2 1; Power1 1 ENDON ← wenn sich der Wert von toggeling ändert, wird ein event ausgelöst. In diesem Beispiel wird geprüft, ob der Wert 0 in toggeling2 steht. Wenn ja, dann wird mem2 den Wert 1 zugewiesen und Power1 auf ON geschaltet.
  3. ON event#toggling2=1 DO Backlog mem2 0; Power1 0 ENDON ← enthält toggeling2 den Wert 1, wird mem2 auf 0 und Power1 auf OFF geschaltet.
- TRICK: wenn bei toggelingX=0 erkannt wird, wird memX auf 1 gesetzt

Backlog hilft, dass mehrere Befehle nacheinander ausgeführt werden können (wird Quasi in den Tastaturpuffer geschrieben)

## Tasmota: Sprechende Namen der Relais

Per Default haben die Relais einheitliche Namen. Dabei verliert man z.B. TasmAdmin oder MQTT schnell die Übersicht. Daher kann man „Freundliche“ Namen für jedes Relais vergeben.

- Name1 ⇒ HeizungPumpe
- Name2 ⇒ Automatik



Ansicht im TasmotAdmin:

Pos.	Name	IP	Status	Version	Laufzeit	Energie	Temp.	L.-Feucht.	
1	1 - WZSteckdose3x	10.		8.5.1(tasmota)	10T 0h 59m 31s	0W 0.005/0.019kWh 0A	-	-	
1	2 - WZSteckdose3x	10.		8.5.1(tasmota)	10T 0h 59m 31s	0W 0.005/0.019kWh 0A	-	-	
1	3 - WZSteckdose3x	10.		8.5.1(tasmota)	10T 0h 59m 31s	0W 0.005/0.019kWh 0A	-	-	
1	USB - WZSteckdose3x	10.		8.5.1(tasmota)	10T 0h 59m 31s	0W 0.005/0.019kWh 0A	-	-	
2	Subwoofer	10.		8.5.1(tasmota)	10T 0h 59m 34s	-	-	-	
3	blitzwolfshp6	10.		6.5.0(release-sonoff)	71T 0h 12m 40s	32W 0.076/0.175kWh 0.163A	-	-	
4	3DDruckerEnder3LSD	10.		8.5.1(tasmota)	17T 1h 46m 39s	0W 0.00kWh 0A	-	-	
4	BW-SHP7	10.		8.5.1(tasmota)	17T 1h 46m 39s	0W 0.00kWh 0A	-	-	
5	co2ampel	10.		9.0.0.2(sensors)	2h 49m 15s	-	-	-	
6	Stern_Flur	10.		-	-	-	-	-	
7	HeizungPumpe	10.		8.5.1(tasmota)	7m 52s	-	9.4°C	90%	
7	Automatik	10.		8.5.1(tasmota)	7m 52s	-	9.4°C	90%	
Pos.	Name	IP	Status	Version	Laufzeit	Energie	Temp.	L.-Feucht.	

From:

<https://quad4.logout.de/> - **quad.logout.de**

Permanent link:

<https://quad4.logout.de/tasmota:pumpenheizung?rev=1605006271>

Last update: **2020/11/10 11:04**

