Rules

Inhaltsverzeichnis

- Erstellt von HoerMirAuf

I - Rules (Regeln)

Dies ist eine Übersetzung von Github in der Originalversion von Theo Arends

Seit Beginn der Version 5.12.0I stellt Sonoff-Tasmota, inspiriert durch die ESP-Easy Gestaltung, optional “Rules“ bereit.

(Um Rules nutzbar zu machen muss *define USE_RULES* in der Datei *user_config_override.h* aktiviert sein.)

(Derzeit wird keine „if“ Richtlinie oder Verschachtelungen der Rules unterstützt)

I.I - Inhaltsverzeichniss

I.II - Einführung

Die Rules (Regeln) erweitern die verfügbaren Funktionen auf die individuelle Anpassung an Benutzeranwendungen. Sie erlauben es verschiedenste Aktionen mit Button (Taster)/Switch(Schalter) oder Zustandsrückmeldungen durchzuführen. Die Rules werden im Flash gespeichert und bleiben so auch nach einem Neustart verfügbar. Rules reagieren auf Trigger, ausgelöst durch Systemereignisse wie den Systemstart oder eine Timer Aktivierung.

Mit Hilfe der Rules lassen sich so die unterschiedlichste Problemstellungen lösen.

Die Rules können direkt über die Tasmota Konsole, über MQTT oder über HTTP, wie jeder andere Befehl auch, eingegeben werden.

I.III - Rule Commands (Rule Befehle)

Wenn die Rules verfügbar sind stehen zusätzlich fünf mächtige Befehle zur Verfügung.

- Rule steuert den Speicher der bis zu 511 Zeichen großen, benutzerdefinierten Rules. Erlaubt ebenso die Rules aktiv als auch inaktiv zu schalten. in manchen Fällen, wie bei sich langsam ändernden Sensoren (Temperatur) ist so die Aktivierung/Deaktivierung einer einmalige Rule Ausführung möglich.

- RuleTimer stellt bis zu acht Timer für Countdown Anwendungen.
- Event ermöglicht es einem Benutzer eine Rule auszuführen
- Var ermöglicht das vorübergehende speichern von bis zu fünf Variablen
- Mem ermöglicht das dauerhafte speichern von bis zu fünf Variable

siehe in der Command Beschreibung für den richtigen Syntax und deren Möglichkeiten.

I.IV - Rule Syntax (Befehlsaufbau)

Eine Rule verwendet folgenden Syntax:

Code

```
on <trigger> do <command> endon
```

Mehrere Rule können zu einem Einzeiler aneinandergehängt werden

Code

```
on <trigger1> do <command> endon on <trigger2> do <command> endon …
```

Leerzeichen nach jedem **on,** um ein **do** herum und vor einem **endon** sind zwingend!

Eine Rule unterscheidet nicht zwischen Groß und Kleinschreibung.

Vor 6.1.0 20180706, stand nur ein einziger Rule Speicher für alle Befehle zur Verfügung. Um einen kompletten Rule-Satz im Flash zu speichern kann der rule Befehl verwendet werden:

Code

```
rule on <trigger1> do <command> endon on <trigger2> do <command> endon …
```

Seit 6.1.0 20180706 sind 3 Rule Speicher für Rule Befehle verfügbar. Jeder dieser Speicher kann individuell an bzw. ausgeschaltet werden. Statt rule wird hier rule[123] verwendet:

Code

```
rule1 on <trigger1> do <command> endon on <trigger2> do <command> endon …
```

I.V - Trigger (Auslöser)

Ein Trigger setzt sich folgendermaßen zusammen:

Code

```
<SensorName>#<ValueName>
<SensorName>#<ValueName>=<value>
<SensorName>#<ValueName><<value>
<SensorName>#<ValueName>><value>
Tele-<SensorName>#<ValueName>
```

Mögliche Trigger sind:

- System#Boot einmalig wenn MQTT gestartet wird. Aufgrund der Befehlsausführung kann dieser nicht früher ausgeführt werden
- Mqtt#Connected wenn MQTT verbunden ist
- Mqtt#Disconnected wenn MQTT getrennt ist
- Wifi#Connected wenn WLAN verbunden ist (6.1.1c)
- Wifi#Disconnected wenn WLAN getrennt ist (6.1.1c)
- Time#Initialized einmalig wenn NTP gestartet ist und die Zeit synchronisiert
- Time#Initilaized>120 einmalig wenn NTP gestartet ist und die Synchronisierung über 2 min dauert (6.1.0)
- Time#Set jede Stunde wenn NTP die Zeit synchronisiert
- Time#Minute jede Minute
- Time#Minute=241 jeden Tag um 04:01 (=241 Minuten nach Mitternacht) (6.1.0)
- Time#Minute|5 alle 5 Minuten (6.1.1.6)
- Clock#Timer=3 wenn der allgemeine Timer3 aktiviert ist
- Rules#Timer=1 wenn der RuleTimer1 Countdown abgelaufen ist
- Event#Anyname wenn das Befehlsereignis Anyname ausgeführt wird
- Power1#State wenn sich der Schaltzustand ändert
- Power1#Boot tritt ein wenn Tasmota startet
- Button#2State wenn sich ein Tastereingang ändert
- Switch1#State wenn sich ein Schaltereingang ändert (zu beachten: SwitchTopic=0 muss für diesen Trigger gesetzt werden)
- Switch1#Boot tritt ein nachdem Tasmota gestartet ist
- Dimmer1#State wenn sich der Dimmer Wert ändert
- Dimmer#Boot tritt ein nachdem Tasmota gestartet ist
- Analog#A0div10 wenn sich der A0 Eingang um mehr als 1% verändert. Unterstützt werden Werte zwischen 0 und 100

Alle angeschlossenen Sensoren können als Trigger verwendet werden, in der Form wie sie in der **tele** oder **Status 8** JSON Ausgabe (Terminal/MQTT) dargestellt werden.

- Dsb18b20#Temperatur<20 Immer wenn die Temperatur des Sensors DSB18B20 kleiner 20 Grad fällt
- AM2301-12#Humidity=55.5 Immer wenn die Feuchtigkeit des Sensors AM2301-12 gleich 55,5% ist
- IMA219#Current>0.100 Immer wenn der aktuell gemessene Wert größer 0.1A ist
- Energy#Power>100 Immer wenn der Leistungsbedarf über 100W steigt

Geräte die RfReceived (Sonoff Bridge) oder IrReceived verwenden, werden ebenfalls gemäß ihrer JSON Ausgabe unterstützt, z.B:

- IrReceived#Data=801 Immer wenn ein IR Signal einer RC5 Fernbedienung, Taste 1 empfangen wird (6.1.0)
- RfReceived#RfKey=4 Immer wenn die Sonoff Bridge das auf Taste 4 gespeicherte RF Signal empfängt

Um mit einen Sensor nur zu jeder tele Ausgabe zu triggern, muss nur das Wort tele als Präfix vor die Sensorbezeichnung gesetzt werden um von den oben beschriebenen Trigger zu unterscheiden.

Siehe unten, Beispiel 5.

<value> kann jede Zahl, Text, %var1% - %var5%, %mem1% - %mem5%, %time%, %uptime%, %sunrise% oder %sunset% sein um die Regel zu testen.

I.VI - Command (Befehl)

Ein Command ist jeder verfügbare Befehl, einschließlich des Befehls Backlog. Ein Command Parameter kann durch die Variable %value% ersetzt werden die für den Trigger Wert <value> steht.

Siehe Beispiel unten.

Um in einer Rule mehrere Befehle auszuführen wird in dieser der Command backlog verwendet und die Commands mit ; abgegrenzt: (siehe backlog)

Code

```
on <trigger1> do backlog <command1>;<command2>;<command3> … endon
```

Spezielle Commands sind die Ausdrücke: var1 bis var6. Diese bieten die Möglichkeit den Trigger Wert <value> in einer Variablen %var1% bis %var6% **temporär** zu speichern um diese so in jeder anderen Rule mitverwenden zu können. Diese Variablen sind zum Programmstart stets leer.

Siehe unten, Beispiel 5.

Zusätzlich zu %var[x]% sind die Ausdrücke: mem1 bis mem5, ebenfalls spezielle Commands. Diese bieten die Möglichkeit einen Trigger Wert <value> in einer Variablen %mem1% bis %mem6% **dauerhaft** zu speichern um diese so in jeder anderen Rule mitverwenden zu können. Diese Variablen enthalten beim Programmstart die zuvor gespeicherten Werte.

Siehe unten, Beispiel 5.

Zu beachten:

Derzeit wird keine „if“ Richtlinie oder Verschachtelungen der Rules unterstützt

I.VII - Beispiele

Die folgenden Beispiele erklären eine Anwendungsmöglichkeiten

1. verhindern einer Überlastung bei einem Wemos D1 mini
2. absetzen einer MQTT Nachricht wenn eine Taste gedrückt wird
3. ausführen mehrerer Commands wenn ein Timer abgelaufen ist
4. Nutzung der Einmalig-Ausführung (once)
5. Nutzung von Variablen und tele-
6. Nutzung eines Potentiometers
7. Variablen setzen
8. Arithmetische Befehle zur Verwendung mit Variablen
    - 8a Senden eines Sensorwertes an MQTT nur dann, wenn ein Delta erreicht wird
    - 8b Anpassen eines Sensorwertes und versenden über MQTT
9. Einfaches Thermostat Beispiel
10. Einfache Treppenhaus Zeitsteuerung
    - a fortgeschrittene PIR (Bewegungsmelder) Rule für’s Treppenhaus
11. Energiespar Regelung (helligkeitsgeführt)
12. Timer steuern mit einem Schalter
13. Umschalten des Relay’s wenn der Button länger als 2 Sekunden gedrückt wird
14. Rule um sicher zu gehen, dass Licht nachts eingeschaltet ist (Nachtlicht)
15. Rule zur Aktivierung eines PIR-Sensors nur bei Nacht (Dämmerungsschaltung)

16. Verwenden einer externen Taste mit einzel - doppel Tastendruck und gedrückt halten
17. Rule um Aktivieren oder Deaktivieren der Türklingel mit HTTP-Aufruf

1. verhindern einer Überbelastung bei einem Wemos D1 mini

Ein WS2812 24-LED-Ring benötigt ungefähr 24 x 3 x 20 mA = 1,44 A Strom. Da ein Wemos D1 mini, der über den USB-Anschluss eines PCs mit Strom versorgt wird, nur bis zu 0,5 A liefern kann, ist eine Strombegrenzungsfunktion sinnvoll die diesen auf 0,1 A begrenzt. Dieser ist ausreichend um alle 24 LED's bis zur Farbe 202020 zu betreiben.

Hardware

- Wemos D1 mini
- INA219 I2C sensor
- WS2812 LED-Ring mit 24 LED's versorgt vom Wemos D1 mini 5V durch den NA219 sensor

Software

- Sonoff-Tasmota v5.12.0l oder höher mit aktivierten define USE_RULES

Rule

Code

```
on INA219#Current>0.100 do Backlog Dimmer 10;Color 10,0,0 endon
```

Ergebnis

- Wenn die Helligkeit soweit erhöht wird das sie den Wert von 0,1A überschreitet, wird die Rule aktiv und löst den Befehl Dimmer 10 aus. Durch das herunterdimmen mit Dimmer 10 und der Farbänderung auf Rot 10,0,0 wird der Strombedarf der LED's wieder reduziert

2. absetzen einer MQTT Nachricht wenn eine Taste gedrückt wird

Der Nutzer hat hier die Möglichkeit durch einen Tastendruck eine MQTT Nachricht, basierend auf FullTopic und ButtonTopic abzusetzen. Diese MQTT Nachricht wird vom MQTT Broker empfangen und ebenso von allen Geräten die diesen Topic abonniert haben. Auf diese Weise können Nachrichten und Befehle an die Hausautomation und/oder jedes beliebige MQTT Gerät gesendet werden.

Die Problemstellung bei einem Sonoff 4CH besteht darin das dieser mit allen Tasten den gleichen Topic, jedoch mit unterschiedlichen Zahlenindex für die Relays sendet.

Beispiel Umschaltfunktion:`cmnd/buttontopic/power3 toggle`

Durch die Verwendung einer Regel kann jetzt eine Taste jede beliebige MQTT-Nachricht senden, wodurch die individuellen Einsatzmöglichkeiten erhöht werden.

Hardware

- Sonoff 4CH

Software

- Sonoff-Tasmota v5.12.0l oder höher mit aktivierten define USE_RULES
- Deaktivieren von ButtonTopic da er sonst die Rule überschreiben würde: ButtonTopic 0

Rule

Code

```
rule on button1#state do publish cmnd/ring2/power %value% endon on button2#state do publish
```

Ergebnis

- Durch drücken der Taste 1 greift die Rule und sendet die MQTT Nachricht, wobei die Variable %value% den Tastenstatus der Taste 1 beinhaltet: `cmnd/ring2/power 2` Bei Betätigung der Taste 2 wird durch die Rule eine MQTT Nachricht: `cmnd/strip1/power 2` gesendet, wobei hier die Variable %value% den Tastenstatus der Taste 2 beinhaltet

3 .ausführen mehrerer Commands wenn ein Timer abgelaufen ist

Die Standardfunktion Timer1..16 ermöglicht die Steuerung eines Ausgangs. Entweder aus, ein, umschalten oder blinken. Bei aktiven Rules, wird die Blinkoption durch eine Regelfunktionalität ersetzt, die mehr Flexibilität bietet.

Hardware

- Sonoff 4CH

Software

- Sonoff-Tasmota v5.12.0 oder höher mit aktivierten define USE_RULES
- Den Timer5 für die Rule Verwendung vorbereiten: timer5 mit Einstellungen: {"Arm":1,"Mode":0,"Time":"16:00","Days":"1111111","Repeat":1,"Action":3}

Rule

Code

```
on clock#timer=5 do backlog power2 on;power1 off;power3 2 endon
```

Ergebnis

- Ist der Timer abgelaufen, greift die Regel und setzt Power1 auf OFF, Power2 auf ON und schaltet Power3 um

Ist eine Blink-Funktion gewünscht, ist die Rule wie folgt zu definieren:

Code

```
on clock#timer=5 do power 3 endon
```

4 .Nutzung der Einmalig-Ausführung (once)

Die Option once bietet die Möglichkeit, bei einer langsamen Zustandsänderung nur einmalig auszulösen, während sich der Zustand noch innerhalb der Auslösegrenzen befindet.

Rule

Code

```
on ENERGY#Current>0.100 do publish tool/tablesaw/power 1 endon on ENERGY#Current<0.100 do pu
```

Durch diese Rule wird ein MQTT-Befehle wiederholt abgesetzt, solange ein Sonoff-POW einen Stromfluß gößer 0,100 misst, während wiederholt ein anderer abgesetzt wird sobald der Wert kleiner 0,100 ist.

```
Rule 5
```

Jetzt wird der MQTT Befehl jeweils nur einmalig abgesetzt wenn die Bedingung erfüllt wurde. Dies ist ideal für das Ein / Ausschalten eines Thermostats in Abhängigkeit von der Temperatur, eines Badezimmersventilators in Abhängigkeit von der Luftfeuchtigkeit oder das Ein / Ausschalten eines Werkstattstaubsaugers, je nachdem, ob eine stauberzeugende Maschine läuft.

5. Nutzung von Variablen und tele-

Die Verwendung von Variablen ermöglicht das Speichern von Sensorwerten um z.B. eine komplette HA-Nachricht, wie sie bei Domoticz verwendet wird, zu erstellen. Um den Datenverkehr zu Domoticz gering zu halten, wird so nur eine Nachricht zur tele Übertragungszeit gesendet. Dies wird erreicht, indem dem Sensor Name das Label tele- vorangestellt wird. In diesem Beispiel wird eine Variable für die Temperatur gesetzt, die zusammen mit der Luftfeuchtigkeit in einer Domoticz MQTT-Nachricht verwendet werden soll.

Hardware

- Sonoff TH oder Wemos D1 mini
- AM2301 Temperature and Humidity sensor

Rule

Code

```
on tele-am2301-12#temperature do var1 %value% endon on tele-am2301-12#humidity do publish do
```

Ergebnis

- Aufgrund des Präfix tele- wird durch die Rule zur tele Übertragungszeit für den Sensor AM2301-12, die Temperatur und Luftfeuchtigkeit ausgelesen.

Die erste Rule verwendet den in %value% enthaltenen Temperaturwert und schreibt diesen zur späteren Verwendung in %var1%. Die zweite Rule verwendet den in %value% enthaltenen Luftfeuchtigkeitswert und die in %var1% gespeicherte Temperatur, um eine einzelne, für Domoticz geeignete MQTT-Nachricht zu erstellen.

Schlaue Köpfe haben somit endlich die Möglichkeit, Temperaturen von mehreren DS18B20 Sensoren an Domoticz zu senden.

6. Nutzung eines Potentiometers

Mit dem anschließen eines Potentiometers an den Eingang Analog A0 und einer Rule kann hier der Dimmerstatus eines Geräts gesteuert werden.

Hardware

- Wemos D1 mini
- Potentiometer of 2k2 connected to Gnd, A0 and 3V3
- WS2812 led

Software

- Sonoff-Tasmota v5.12.0n or up with define USE_RULES enabled

Rule

Code

```
on analog#a0div10 do dimmer %value% endo
```

Ergebnis

- Durch Drehen am Potentiometers wird eine Spannungsänderung am Analogeingang, zwischen 0(Off) und 100(On) bewirkt die somit den Trigger auslöst. Der erfasste Analogwert entspricht dabei dem LED-Dimmwert für den WS2812.

Rule

Code

```
on analog#a0div10 do publish cmnd/grouplight/dimmer %value% endon
```

Ergebnis

- Hier ändern alle mit GroupTopic grouplight konfigurierten Leuchten ihre Helligkeit entsprechend der Position des Potentiometers.

Hinweis:

- um Flash-Schreibvorgänge zu reduzieren kann der Command SaveData 2 verwendet werden

7. Variablen setzen

In diesem Beispiel wird die Verwendung von Variablen gezeigt. Dazu muss zuerst der Command Rule 4 (einmalig Erkennung deaktivieren) ausgeführen werden.

- Setzen einer Variablen

benötigte Rule: `on event#setvar1 do var1 %value% endon`
Command: `event setvar1=1`

- Variable ausgeben

benötigte Rule: `on event#getvar1 do var1 endon`
Command: `event getvar1`

- Variablen Toogeln (Umschalten)

benötigte Rules:
```
on event#togglevar1 do event toggling1=%var1% endon
on event#toggling1<1 do event setvar1=1 endon
on event#toggling1>0 do event setvar1=0 endon
on event#setvar1 do var1 %value% endon
```
Command: `event togglevar1`

- Nachrichten anzeigen

benötigte Rule: `on event#message do publish stat/[topic]/log %value% endon`

Command: `event message=INIT`

Alle Event Befehle können ausgeführt werden von:

Konsole: `event anyname=number`
MQTT: `cmnd/[topic]/event anyname=number`

- alles zusammengefasst

Code

```
rule on event#togglevar1 do event toggling1=%var1% endon on event#toggling1<1 do event setva
```

Hinweis

Folgendes funktioniert nicht:

Code

```
rule on event#setvar1 do backlog var1 %value%; power1 %var1% endon
```

Zumindest nicht so, wie man es erwarten könnte. Der vom Befehl power1 verwendete Wert var1 ist der Wert, der vorhanden ist, bevor der Befehl backlog ausgeführt wird. Dies ist der Fall, weil die Rule den Wert in %var1% einsetzt, BEVOR die Backlog-Befehle in das Backlog-Protokoll eingestellt werden.

8. Arithmetische Befehle zur Verwendung mit Variablen

- ADD

ADD1 bis ADD5: addiert einen Wert zur VARx

Syntax: ADDx value
Anwendung: ADD1 15
Ergebins: VAR1 = VAR1 + 15

- SUBSTRACT

SUB1 bis SUB5: substrahiert einen Wert von VARx

Syntax: SUBx value
Anwendung: SUB1 15
Ergebnis: VAR1 = VAR1 - 15

- MULTIPLY

MULT1 bis MULT5: mutipliziert einen Wert mit VARx

Syntax: MULTx value
Anwendung: MULT1 15
Ergebnis: VAR1 = VAR1 * 15

- SCALE A VALUE

SCALE1 bis SCALE5:

skaliert einen Wert von einem unteren/oberen Grenzwert zu einem anderen unteren/oberen Grenzwert und schreibt diesen in VARx (entspricht dem Ardunino MAP Befehl)

Syntax: SCALEx value, fromLow, fromHigh, toLow, toHigh

Begriffserklärung:

value:	der zu skalierende Wert
fromLow:	Die Untergrenze des aktuellen Wertebereichs
fromHigh:	Die Obergrenze des aktuellen Wertebereichs
toLow:	Die Untergrenze des Zielbereichs des Werts
toHigh:	Die Obergrenze des Zielbereichs des Werts

(ausgelassene Werte werden als Null angenommen)

Anwendung:	SCALE1 15, 0, 100, 0, 1000
Ergebnis:	VAR1 = 150

8a. Senden eines Sensorwertes an MQTT nur dann, wenn ein Delta erreicht wird

Sollte es erforderlich sein einen Sensorwert nur dann zu senden, wenn er ein Delta überschreitet, kann das mit folgendem Regelbeispiel gelöst werden:

Code

```
rule
on SI7021#temperature>%var1% do backlog var1 %value%; publish stat/sonoff/temp %value%; var2 %value% add1 0.2 endon
on SI7021#temperature<%var2% do backlog var2 %value%; publish stat/sonoff/temp %value%; var
```

8b. Anpassen eines Sensorwertes und versenden über MQTT

In diesem Beispiel werden der gemessenen Temperatur 2 Grad hinzugefügt und dieser Wert anschließend als MQTT-Topic gesendet.

Code

```
rule
on tele-SI7021#temperature do backlog var1 %value%; add1 2; event sendtemp endon
on event#sendtemp do publish stat/sonoff/temp %var1% endon
```

9. Einfaches Thermostat Beispiel

Diese Beispiel ist zur Verwendung auf einem Sonoff TH10 mit Sensor Si7021

In diesem Beispiel wird ein Ausgang basierend auf dem Temperaturwert, dem oberen Sollwert und dem unteren Sollwert ein- und ausgeschaltet. Es wird gewartet, bis es durch Drücken der Taste oder durch den Mqtt Befehl 1 an Mem1 zur Aktivierung kommt.

Dieser Wert wird gespeichert. Nach Stromwiederkehr wird der Betrieb wieder aufgenommen. Die Sollwerte können im laufenden Betrieb durch die Befehle mqtt oder Konsole geändert werden. Wenn der

Temperatursensor getrennt wird, werden die Ausgänge ausgeschaltet, bis der Sensor wieder verfügbar ist und den Betrieb fortsetzt. Wenn das Gerät eingeschaltet ist, wartet der Thermostat, bis der Sensorwert den Betrieb wieder aufnimmt.

Ursprungs Config

- Verfügbare physische Taste als Schalter1
- Relay1 wird als Controller verwendet
- Es werden Rule zur Steuerung des Relais benötigt, sodass der Taster nur switch1 und nicht direkt das Relais schaltet. Dazu verwenden wir den switchmode1 3 wie unten beschrieben und erstellen die notwendigen Rule, da die Tastensteuerung des Relais nur deaktiviert wird, wenn die Rules aktiv sind

Konsolenbefehle:

`switchmode1 3`	verwendet switch1 as Taste (Dies ermöglicht es uns, die Verbindung zwischen der Taste und dem Relais zu deaktivieren, indem eine Rule eingefügt wird, die vorschreibt, was die Taste tun soll - **HINWEIS: Bis zum erstellen der Rule steuert der Taster das Relais!)**
`rule 1`	aktiviert die Rule
`rule 4`	deaktiviert die einmalig Funktion
`teleperiod 60`	Temperatur jede Minute abfragen
`setoption26 1`	verwendet power1 bei mqtt Nachricht
`setoption0 0`	den Relais status nicht im eprom speichern
`poweronstate 0`	bei Systemstart alle Relais aus
`mem1 0`	Thermostat Status: 0-off 1-enabled - Wird angezeigt oder gesetzt mit MQTT Befehl `cmnd/sonoff/mem1`
`mem2 25`	Sollwert Temp oberer Grenzwert - Wird angezeigt oder gesetzt mit MQTT Befehl `cmnd/sonoff/mem2`
`mem3 23`	Sollwert Temp unterer Grenzwert - Wird angezeigt oder gesetzt mit MQTT Befehl `cmnd/sonoff/mem3`
`var1 0`	Thermostat aktueller Status: 1-OK 0-NOT READY - Wird angezeigt oder gesetzt mit MQTT Befehl `cmnd/sonoff/var1`

Rule:

Beim Booten wird ein Watch-Dog-Timer gestartet, um die Verbindung des Temperatur Sensors zu überprüfen

Code

```
rule on system#boot do ruletimer1 70 endon
```

Eine vorhandene Taste ist als Schalter konfiguriert, um den Thermostat ein- oder auszuschalten

Code

```
on switch1#state do backlog event toggling1=%mem1% endon on event#toggling1=0 do mem 1 endo
```

Überprüft die Verbindung des Temperatursensors. Wenn diese nicht besteht wird ausgeschaltet und der Temperaturabfragewert auf 0 gesetzt, und weiterhin abgefragt.

Code

```
rules#timer=1 do backlog var1 0; ruletimer1 70; power1 0 endon
```

Setzt die Timerüberprüfung zurück, wenn der Temperatursensor vorhanden ist

Code

```
on tele-SI7021#temperature do backlog var1 1; ruletimer1 30; event ctrl_ready=1; event temp_
```

Thermostatsteuerung - oberer und unterer Grenzwert setzen und aktivieren

Code

```
on event#ctrl_ready>%mem1% do var1 0 endon on event#temp_demand>%mem2% do power1 0 endon on
```

Der Thermostat kann eingeschaltet werden durch:

- Tastendruck
- über einen lokalen Konsolenbefehl: mem1 1
- durch Befehl auf einer anderen Konsole: publish cmnd/sonoff/mem1 1
- über MQTT: cmnd/sonoff/mem1 1

Der Thermostat kann ausgeschaltet werden durch:

- Tastendruck
- über einen lokalen Konsolenbefehl: mem1 0
- durch Befehl auf einer anderen Konsole: publish cmnd/sonoff/mem1 0
- über MQTT: cmnd/sonoff/mem1 0

Statusabfrage durch:

- `mem1` <- Thermostat status: 0-off 1-enabled -Wird angezeigt oder gesetzt mit MQTT Befehl `cmnd/sonoff/mem1`
- `mem2` <- setpoint Temp upper limit - Wird angezeigt oder gesetzt mit MQTT Befehl MQTT `cmnd/sonoff/mem2`
- `mem3` <- setpoint Temp lower limit - Wird angezeigt oder gesetzt mit MQTT Befehl `cmnd/sonoff/mem3`
- `var1` <- thermostat actual status: 1-OK 0-NOT READY - Wird angezeigt oder gesetzt mit MQTT Befehl `cmnd/sonoff/var1`

Zusammengefasst:

Konsolenbefehle:

(Bitte beachten, dass ruletimer1 größer als teleperiod sein muss um eine korrekte Funktion zu erzielen)

```
backlog switchmode1 3
```

```
rule 1
```

```
rule 4
```

```
teleperiod 60
```

```
setoption26 1
```

```
setoption0 0

poweronstate 0

mem1 0

mem2 25

mem3 23

var1 0
```

RULE:

Code

```
rule1 on system#boot do ruletimer1 70 endon on Switch1#State do event toggling1=%mem1% endo
```

BEISPIELREGELN OHNE TEMPERATURSENSOR, ZUM TEST DER THERMOSTATREGELN:

Code

```
rule on system#boot do ruletimer1 70 endon on Switch1#State do event toggling1=%mem1% endon
```

TESTS

- Drücken der Taste1. Der Thermostat wechselt zu ENABLED (mem1 = 1)
- in der Konsole: event temp=20 (wird vom System vom Temperatursensor wie eine Tele-Nachricht empfangen) und schaltet das Relais1 ein (heizen)
- in der Konsole: event temp=26 (der Thermostat schaltet die Heizung aus)
- in der Konsole: event temp=22 (der Thermostat schaltet die Heizung ein)
- Wird länger als eine Minute gewartet, ohne die event temp zu verändern, schaltet sich der Thermostat aus, da es keinen gemeldeten Temperaturwert gibt (z. B. bei einem Sensorfehler oder einer Unterbrechung).
- Wird wieder aufgenommen, wenn event temp erneut verwendet wird
- in der Konsole: mem1 0, DISABLED, bzw. mem1 1, ENABLED

TIMERS

Mit den oben genannten Timern kann mem1 gesteuert und einen Zeitplan hinzufügen werden, mit dem der Thermostat aktiviert wird.

Code

```
rule2 on Clock#Timer=1 do mem 1 endon on Clock#Timer=2 do mem 0
```

10. Einfache Treppenhaus Zeitsteuerung

Rule

Code

```
rule1 on button1#state do backlog power1 %value%; ruletimer1 600 endon on rules#timer=1 do p
```

Ergebnis

```
on button1#state do backlog power1 %value%;
```

- Beim Drücken der Taste wird das Licht im Treppenhaus ein- / ausgeschaltet

```
ruletimer1 600 endon
```

- Zusätzlich beginnt der ruletimer 1 mit einem 10 Minuten Countdown

```
on rules#timer=1 do power1 off endon
```

- Nachdem ruletimer1 abgelaufen ist, wird das Licht ausgeschaltet (sollte es noch an sein)

10a fortgeschrittene PIR (Bewegungsmelder) Rule für's Treppenhaus

Diese Beispiel funktioniert gut auf einem Wemos D1 Mini mit Tasmota 6.2.1. Wird als Nachtlicht mit Bewegungssensor oder als Umgebungslicht für Boden oder Küche verwendet. Es ist ein LED Strip WS2812 an D1, ein PIR an D2 und einen LDR an A0 (Spannungsteiler mit 10k Ohm Widerstand) anzuschließen

Besipiel PIR: HR-SC501

Die Einstellungen

18 Generic

D1 WS2812

D2 Switch1

LDR an Wemos A0 (aktiviert in user_config.h)

Rules

Code

```
rule1 on analog#a0<400 do backlog rule3 0; rule2 1 endon on analog#a0>500 do backlog rule2 0
```

Code

```
rule2 on switch1#state do backlog power1 1; ruletimer1 30 endon on rules#timer=1 do power1 0
```

Code

```
rule3 on switch1#state do power1 off endon
```

Konsolenbefehle:

```
switchmode1 1
```

`rule1 1` Rules aktivieren

`rule1 6` einmalig Erkennung

Optional

```
rule2 4
rule3 4
```

Ergebnis

`on analog#a0>400` Deaktiviert rule3 und aktiviert rule2

`on analog#a0>500` Deaktiviert rule2 und aktiviert rule3

- rule2 aktiviert die LEDs und startet den ruletimer1 30 Sekunden. Bei jedem Triggerimpuls vom PIR wird der rultimer neu gestartet.

```
on rules#timer=1 do power1 off
```

- rule3 ist bei Tageslicht aktiv und bewirkt mit dem PIR-Signal ein Power1 Off Signal. Die LEDs bleiben aus.

11 .Energiespar Regelung (helligkeitsgeführt)

Beispiel eines Schalters, der ein Licht helligkeitsabhängig (LUX) steuert.

Ist der Schalter eingeschaltet, wird das Licht nur dann eingeschaltet, wenn die Raumhelligkeit kleiner 100 Lux ist.

Ist der Schalter ausgeschaltet ist, geht das Licht aus.

```
on switch1#state=1 do var1 100 endon
```

```
on switch1#state=0 do backlog var1 0; power1 off endon
```

```
on APDS9960#Ambient<%var1% do power1 on endon
```

Alles zusammen funktioniert als Rule

Code

```
rule                                                                    1
rule on switch1#state=1 do var1 100 endon on switch1#state=0 do backlog var1 0; power1 off e
```

12 .Timer steuern mit einem Schalter

Annahme: Ein Schalter ist an GPIO00 angeschlossen und als Switch1 konfiguriert:

Konsolenbefehle:

```
switchmode1 1
```

Rule:

Code

```
rule 1 rule on Switch1#state=1 do Timers 0 endon on Switch1#state=0 do Timers 1 endon
```

Switchmode1 1 bewirkt, dass der Switch1#state 1 ist, wenn eingeschaltet ist, und 0, wenn ausgeschaltet ist

Wenn Switchmode1 nicht gesetzt oder 0 ist, ist Switch1#state = 2 (Toggle) aktiv und die Rule funktioniert nicht.

13 .Umschalten des Relay’s wenn der Button länger als 2 Sekunden gedrückt wird

Im folgenden Beispiel wird erläutert, wie die HOLD-Funktion für Tasten konfiguriert und verwendet wird.

Verhalten: Die "kurzer Tastendruck" Funktion des Button1 deaktivieren und stattdessen umschalten des Relais nur dann wenn der Button1 für 2 Sekunden gedrückt wird.

Konsolenbefehle:

```
buttontopic 0

setoption1

setoption32 20

rule on button1#state=3 do power1 2 endon on button1#state=2 do delay endon

rule 1
```

Befehlserklärung

buttontopic 0	(Standard) Um Topics nicht für Buttons verwenden zu machen
setoption1 1	Erlaubt nur Button einzeldruck, doppeldruck, und gedrückt halten Aktionen
setoption32 20	Setzen der Tastendruckdauer von 0.1 auf 10 Sekunden (20 = 2 Sekunden)
rule on button1#state=3 do power1 2 endon	Wenn der Button1 für 2 Sekunden gedrückt wird, wird das Relais1 umgeschaltet (state = 3 bedeudet HOLD)
on button1#state=2 do delay endon	Mache nichts wenn die Taste1 kurz gedrückt wird (state = 2 bedeuted TOGGLE)
rule 1	Aktiviert die Rule

HINWEIS: Es gibt keine Statusausgabe für "Doppeldruck" der Tasten. Dies wurde erstellt damit das Relais durch einen Doppeldruck umgeschaltet wird. Weitere Informationen unter: https://github.com/arendst/Sonoff…l-other-devices

Soll die Funktion "Doppeldruck" nicht aktiv sein, muss der Button als Schalter konfiguriert werden und ebenso der switchmode auf den gewünschten Modus eingestellt werden (z.B. switchmode 5, damit er sich wie eine Button verhält) [SWITCH hat keinen Doppeldruck]

```
switchtopic1 0

switchmode1 5

setoption32 20

rule on switch1#state=3 do power1 2 endon on switch1#state=2 do delay endon

rule 1
```

14 .Regel, um sicher zu gehen, dass Licht nachts eingeschaltet ist

Zeitschaltuhren können unter anderem zur Wege oder Terrassenbeleuchtung bei Nacht eingesetzt werden. Ist der Sonoff zum Auslösezeitpunkt nicht mit Strom versorgt, bleibt das Licht die Nacht aus. Eine Rule die zum Start(Boot)-Zeitpunkt die Einschaltbedingungen überprüft bringt hier Ausfallsicherheit.

Dazu ist ein Timer zu konfigurieren, der das Licht bei Sonnenuntergang ein- und bei Sonnenaufgang ausschaltet. Mit der Verwendung von poweronstate 0 wird festgelegt das das Licht zum Startzeitpunkt aus ist. Es wird folgende Rule eingesetzet:

Code

```
onTime#Initializeddobacklogeventchecksunrise=%time%;eventchecksunset=%time%endon
on        event#checksunset>%sunset%        do        power1        1        endon
on event#checksunrise<%sunrise% do power1 1 endon
```

Die Rule setzt folgende Logik um:

WENN %time%>%sunset DANN power1 1 / WENN %time%<%sunrise DANN power1 1

15 .Rule zur Aktivierung eines PIR-Sensors nur bei Nacht (Dämmerungsschaltung)

Vorhanden:

- PIR HC-SR501
- GPIO14 09 Switch1 (Sonoff Basic)
- Jumper (Brücke) außen (like this)
- Lat und Lng (Breiten und Längengrad) in der config gesetzt

Konsolenbefehle:

```
switchmode1 1
```

```
rule1 1
```

Rule:

Code

```
rule1
onSwitch1#state=1 do backlog event checksunrise=%time%; event checksunset=%time% endon
on        event#checksunrise<%sunrise%        do        power1        1        endon
on event#checksunset>%sunset% do power1 1 endon
```

16 .Verwenden einer externen Taste mit einzel - doppel Tastendruck und gedrückt halten

Es stehen nur dann alle 3 Funktionen zur Verfügung wenn der verwendete GPIO als Button definiert ist. Dadurch wird das Relais durch zweimaliges Drücken umgeschaltet (toggle).

Es besteht die Option, die Einfachdruck und Doppeldruck Funktion auszutauschen

BUTTON MIT 3 VERSCHIEDENEN FUNKTIONEN

- Anwendungsbeispiel

(Voraussetzung: GPIO0 ist als Button1 konfiguriert)

Dies soll erreicht werden:

Einfachdruck: Relais umschalten; **Doppeldruck:** MQTT Nachricht senden; **2 Sekunden gedrückt halten**: Weiter MQTT Nachricht senden

Konsolenbefehle:

```
buttontopic 0
```

```
setoption1 1
```

```
setoption11 1
```

```
setoption32 20
```

```
rule 1
```

Rule:

Code

```
rule on button1#state=3 do publish cmnd/topicHOLD/power 2 endon on button1#state=2 do publis
```

- weiteres Beispiel

(Voraussetzung: GPIO0 ist als Button1 konfiguriert)

Dies soll erreicht werden:

Einfachdruck: MQTT Nachricht senden; **Doppeldruck:** Relais umschalten; **2 Sekunden gedrückt halten**: Weiter MQTT Nachricht senden

Konsolenbefehle:

```
buttontopic 0
setoption1 1
setoption11 0
setoption32 20
rule 1
```

Rule:

Code

```
rule on button1#state=3 do publish cmnd/topicHOLD/power 2 endon on button1#state=2 do publis
```

(Bitte beachten: **setoption11 0**)

SCHALTER MIT ZWEI UNTERSCHIEDLICHEN FUNKTIONEN

Schalter können keine Doppeldruck Funktion

- Beispiel

(Voraussetzung: GPIO0 ist als Switch1 konfiguriert)

Dies soll erreicht werden:

Einfachdruck: nichts; **2 Sekunden gedrückt halten**: Relais 1 umschalten

Konsolenbefehle:

```
switchtopic1 0
switchmode1 5
setoption32 20
rule 1
```

Rule:

Code

```
rule on switch1#state=3 do power1 2 endon on switch1#state=2 do delay endon
```

17 .Rule zum Aktivieren oder Deaktivieren der Türklingel mit HTTP-Aufruf

Es soll das Relais an GPIO12 über einen HTTP Befehl ein- und ausgeschaltet werden und der aktuelle Status über eine MQTT Nachricht (in diesem Fall wird domoticz verwendet) gesendet werden.

Dazu werden in diesem Beispiel Rules verwendet

Ursprungs Config

- GPIO14 Türklingeltaster (Doorbell)
- GPIO12 Switch4 (Sonoff Basic)

Den Türklingeltaster GND und an den GPIO14 des Sonoff Basic anschließen.

Zur Sicherheit einen 4,7K Pullup-Widerstand zwischen VCC(3,3V) und GPIO14 einsetzten um Scheinauslösungen zu vermeiden. (Kondensator ist optional) siehe: YouTube
Nicht vergessen den IDX value zu ändern

Konsolenbefehle:

```
switchtopic 0

switchmode4 2

setoption0 0

poweronstate 0

var1 1

rule1 1
```

Rule:

Code

```
rule1
on          event#doorbell          do          var1          %value%          endon
on    switch4#state=1    do    publish    domoticz/in    {"idx":11,"nvalue":1}    endon
on          switch4#state=1          do          power1          %var1%          endon
on    switch4#state=0    do    publish    domoticz/in    {"idx":11,"nvalue":0}    endon
on switch4#state=0 do power1 0 endon
```

Verwendung:

Nun kann über einen HTTP Befehl das Relais ausgeschaltet werden:

Code

```
http://<tasmotaIP>/cm?cmnd=event%20doorbell=0
```

oder zum einschalten:

Code

```
http://<tasmotaIP>/cm?cmnd=event%20doorbell=1
```

Ist der Sonoff mit Passwort versehen wird der HTTP Aufruf wie folgt gestaltet:

Aus:

Code

```
http://<tasmotaIP>/cm?&user=<tasmotaUsername>&password=<tasmotaPassword>&cmnd=event%20doorbe
```

An:

Code

```
http://<tasmotaIP>/cm?&user=<tasmotaUsername>&password=<tasmotaPassword>&cmnd=event%20doorbe
```

I.VIII - Änderungsprotokoll der Rule

6.1.1.11:

- %sunset%, %sunrise% hinzugefügt