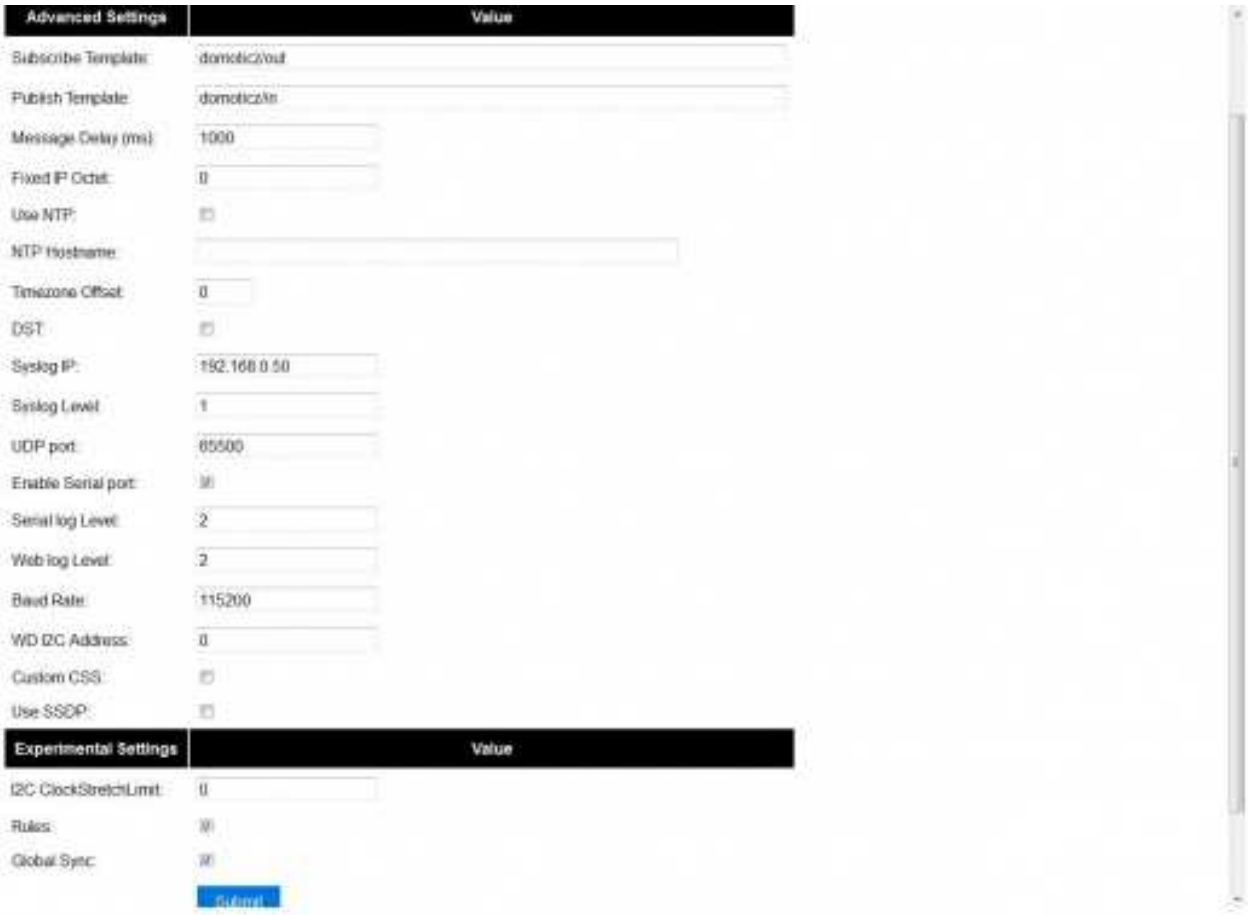
Introduction

Avec ESP Easy, une nouvelle fonctionnalité a été activée, appelée Règles. Les règles peuvent être utilisées pour créer des flux très simples pour contrôler les périphériques de votre ESP.

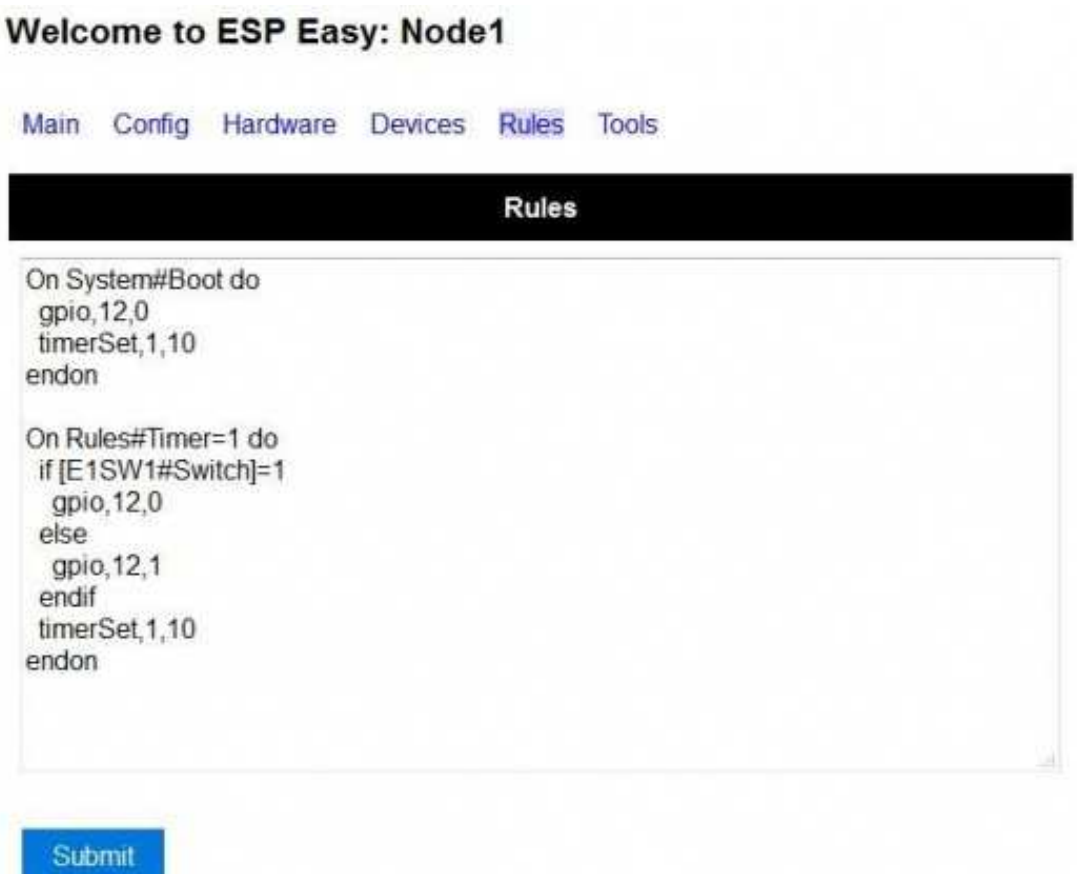
Enable Rules

Pour activer les règles, allez dans Outils / Avancé et cochez la case Règles.



Advanced Settings	Value
Subscribe Template:	domoticz/out
Publish Template:	domoticz/in
Message Delay (ms):	1000
Fixed IP Octet:	0
Use NTP:	☐
NTP Hostname:	
Timezone Offset:	0
DST:	☐
Syslog IP:	192.168.0.50
Syslog Level:	1
UDP port:	65500
Enable Serial port:	☑
Serial log Level:	2
Web log Level:	2
Baud Rate:	115200
WD I2C Address:	0
Custom CSS:	☐
Use SSDP:	☐
Experimental Settings	**Value**
I2C ClockStretchLimit:	0
Rules:	☑
Global Sync:	☑
	Submit

Après avoir cliqué sur Soumettre, vous trouverez une nouvelle page ajoutée. Ici vous pouvez commencer à expérimenter avec des règles:



L'exemple ci-dessus montre une expérience avec une LED, connectée via une résistance de 1k à GPIO12. Pour pouvoir lire l'état de la LED (allumée ou éteinte), une entrée de commutation est créée avec le même port GPIO:

Après le redémarrage de l'ESP, la DEL commencera à clignoter 10 secondes après 10 secondes d'arrêt.

Enjoy.

Syntax

La syntaxe d'une règle peut être une seule ligne:

```
on <trigger> do <action> endon
```

ou multi-lignes (doivent être fermés avec un "endon"):

```
on <trigger> do
<action>
<action>
<action>
endon
```

Aussi simple si ... else ... les déclarations sont possibles:

```
on <trigger> do
 if <test>
   <action>
```

```
   <action>
 else
  <action>
 endif
endon
```

Si la partie "else" n'est pas nécessaire, elle peut être supprimée:

```
on <trigger> do
 if <test>
   <action>
   <action>
 endif
endon
```

MAIS, seul if / else est possible - donc l'imbrication et la logique booléenne ne sont pas supportées. Cependant, il existe une solution de contournement pour la limitation de ne pas pouvoir imbriquer. Un "événement" peut être appelé à partir d'un "déclencheur".

```
on <trigger> do
 if <test1>
   event <EventName1>
 endif
endon
on <EventName1> do
 if <test2>
   <action>
 endif
endon
```

Trigger

`<trigger>`
Le déclencheur peut être une valeur de périphérique en cours de modification:

`DeviceName#ValueName`
Ou une fonction d'inégalité:

```
DeviceName#ValueName<inequality function><value>
```

Où la "fonction d'inégalité" est un simple "égal (=), moins (<), plus grand (>) que" vérifier :

```
DeviceName#ValueName<<value>
DeviceName#ValueName=<value>
DeviceName#ValueName><value>
```

Certains cas particuliers sont les déclencheurs du système qui sont déclenchés au démarrage / redémarrage / heure / sommeil etc. de l'unité:

Event	Information	[Collapse] Example
<taskname>#<valuename>	Comme décrit précédemment, chaque tâche peut produire un ou plusieurs événements, un pour chaque valeur mesurée	on DHT11-Outside#Temperature>20 GPIO,2,1 endon
System#Boot	Déclenché au démarrage	on System#Boot

Event	Information	[Collapse] Example
		GPIO,2,1 timerSet,1,30 endon
System#Sleep	Déclenché juste avant que l'ESP ne passe au sommeil profond	on System#Sleep GPIO,2,0 endon

Clock#Time	Déclenché chaque minute avec le jour et l'heure comme: Lun, 12h30 ou Mar, 14h45. Vous pouvez définir des déclencheurs certains jours ou tous les jours en utilisant l'indicateur "Tous" pour les jours. Vous pouvez également utiliser des caractères génériques dans le réglage de l'heure comme Tous, **: 00 pour exécuter toutes les heures	on Clock#Time=All,12:00
		//will run once a day at noon
		GPIO,2,1
		endon
		
		on Clock#Time=All,**:30
		//will run half past every hour
		GPIO,2,1
		endon

Rules#Timer	Comme déjà décrit, déclenché quand une minuterie de règles se termine (mettre un minuteur à "0" désactivera le minuteur)	on Rules#Timer=1 GPIO,2,1 endon

Test

`<test>`
Comme décrit dans la section trigger, le test est une vérification effectuée en vérifiant si le DeviceName # ValueName répond à un critère:

```
[DeviceName#ValueName]<inequality function><value>
```

Où la valeur doit être une valeur flottante avec un point comme signe décimal. Le DeviceName # ValueName est fermé par des crochets (carrés) "[" et "]".

Action

`<action>`

L'action peut être n'importe quelle commande système trouvée dans la liste des commandes. Des commandes spécifiques au plugin sont également disponibles tant que le plugin est utilisé. Dans le cas mentionné plus haut, nous utilisons une action pour déclencher plusieurs tests logiques (la commande "event").

Comment

Si vous le souhaitez, vous pouvez ajouter des commentaires à n'importe quelle ligne de votre code de règles. N'oubliez pas de les ajouter après le code et commencez toujours par "//":

```
on <trigger> do //If this happens then do that...
 if <test>
   <action>
   <action>
 else
  <action>
 endif //this is another comment
endon
```

Best practice

Il est possible d'utiliser des majuscules et des lettres minuscules à votre guise, mais la meilleure pratique consiste à utiliser les mêmes types de lettres que celles qui se trouvent dans la liste des références de commande, ainsi que les commandes spécifiques au plugin. Pour les logiques (on, if, else ...) l'idée générale est d'utiliser des minuscules.
En ce qui concerne les espaces dans les noms, il est recommandé de ne pas les utiliser car cela rend les règles de test de bogue beaucoup plus difficiles. Les espaces entre les morceaux de code sont possibles pour rendre le code plus lisible:

```
[DeviceName#ValueName]<<value> //These work the same...
[DeviceName#ValueName] < <value>
```

Some working examples:

Event value (%eventvalue%)

Rules engine specific:

% eventvalue% - remplace la valeur de l'événement (tout ce qui vient après le signe '=', une seule valeur est possible)

Sample rules section:

```
 on remoteTimerControl do
   timerSet,1,%eventvalue%
 endon
```

Maintenant, envoyez cette commande à l'ESP:

```
http://<espeasyip>/control?cmd=event,remoteTimerControl=5
```

et il mettra les règles timer nr 1 à 5 secondes. En utilisant cette technique, vous pouvez analyser une valeur d'un événement dans le moteur de règles.
Notez que 'timerSet' est une commande de règle qui ne peut pas être exécutée directement à partir d'une commande distante.

Si vous voulez vérifier la valeur transférée dans les règles de l'ESP récepteur (condition dans if-statement), vous devrez écrire la valeur transférée dans un appareil Dummy en utilisant la

commande TaskValueSet. Il est alors possible de vérifier la valeur du périphérique Dummy comme condition dans if-statement dans les règles.

PIR and LDR

```
On PIR#Switch do
  if [LDR#Light]<500
    gpio,16,[PIR#Switch]
  endif
endon
```

En d'autres termes: Si le commutateur PIR est réglé (sur 1 ou 0) et si la valeur de la lumière est inférieure à 500, réglez le port GPIO 16 de l'ESP.

```
on PIR#Switch=1 do
  if [LDR#Light]<500
    gpio,16,[PIR#Switch]
  endif
endon
```

Maintenant, l'événement n'est déclenché que lorsque le pir s'allume.

SR04 and LDR

```
on SR04#range<100 do
  if [ldr#lux]<500
    gpio,2,0
    gpio,16,1
  else
    gpio,2,1
    gpio,16,0
  endif
endon
```

Timer

Il y a 8 minuteurs (1-8) que vous pouvez utiliser:

```
On System#Boot do     //When the ESP boots, do
  servo,1,12,0
  timerSet,1,10       //Set Timer 1 for the next event in 10 seconds
endon
On Rules#Timer=1 do   //When Timer1 expires, do
  servo,1,12,30
  timerSet,2,1        //Set Timer 2 for the next event in 1 second
endon
On Rules#Timer=2 do   //When Timer2 expires, do
  servo,1,12,0
  timerSet,1,30       //Set Timer1 for the next event in 30 seconds
endon
```

Démarrer / arrêter des minuteurs répétitifs avec des événements

Pour désactiver une minuterie existante, réglez-la sur 0. Ceci est utile pour faire des minuteries répétitives pour des choses comme des alarmes ou des avertissements:

```
//start the warning signal when we receive a start_warning event:
On start_warning do
  timerSet,1,2
endon

//stop the warning signal when we receive a stop_warning event:
On stop_warning do
  timerSet,1,0
endon

//create an actual warning signal, every time timer 1 expires:
On Rules#Timer=1 do
  //repeat after 2 seconds
  timerSet,1,2

  //pulse some led on pin 4 shortly
  Pulse,4,1,100

  //produce a short 1000hz beep via a piezo element on pin 14
  tone,14,1000,100

endon
```

Pour démarrer ou arrêter le signal d'avertissement, utilisez http:

```
http://<espeasyip>/control?cmd=event,start_warning
http://<espeasyip>/control?cmd=event,stop_warning
```

HTTP call

Lorsque vous entrez cette première commande avec l'adresse IP correcte dans l'URL de votre navigateur:

```
http://<espeasyip>/control?cmd=event,startwatering
http://<espeasyip>/control?cmd=event,stopwatering
```

Et avoir cette règle dans l'ESP adressé:

```
On startwatering do
  gpio,12,1 //start watering (open valve)
  timerSet,1,600 //timer 1 set for 10 minutes
endon

On stopwatering do
  timerSet,1,0 //timer 1 set to halt, used to stop watering before the
timer ends!
endon

On Rules#Timer=1 do
   gpio,12,0 //stop watering (close valve)
endOn
```

Provided that you also have the valve etc, the plants will be happy.

Envoyer à et publier

Avec SendTo, vous pouvez ajouter une règle à votre ESPEasy, capable d'envoyer un événement à une autre unité. Cela peut être utile dans les cas où vous voulez prendre des

mesures immédiates. Il y a deux variantes: - SendTo pour envoyer des commandes de contrôle d'unité distante en utilisant la messagerie UDP peer to peer interne - Publier pour envoyer des commandes distantes à d'autres ESP en utilisant le courtier MQTT

SendTo: SendTo <unit>,<command>

Imaginez que vous avez deux modules ESPEasy, ESP # 1 et ESP # 2 Dans la section Règles de l'ESP # 1, vous avez ceci:

```
on demoEvent do
  sendTo 2,event,startwatering //(to use the previous example.)
endon
```

Et ESP # 2 a les règles selon l'exemple précédent (givemesomewater)

Si vous l'entrez ensuite avec l'adresse IP correcte dans l'URL de votre navigateur:

```
http://<ESP#1-ip >/control?cmd=event,demoEvent
```

Ensuite, ESP # 1 enverra l'événement 'startwatering' à ESP # 2.
Il est également possible de commander directement les modifications gpio, comme:

```
on demoEvent do
   sendTo 2,GPIO,2,1
 endon
```

Publish

```
Publish <topic>,<value>
```

Pour être créé.

Time

Avec les règles, vous pouvez également démarrer ou arrêter des actions sur un jour et une heure donnés, ou même tous les jours.

```
On Clock#Time=All,18:25 do // every day at 18:25 hours do ...
 gpio,14,0
endon
```

Ou pour un jour spécifique:

```
On Clock#Time=Sun,18:25 do  // for Sunday, but All, Sun, Mon, Tue, Wed,
Thu, Fri, Sat will do.
  gpio,14,0
 endon
```

Il est également possible d'utiliser la valeur système% systime% dans les conditions de règles pour que les choses se passent à certaines heures de la journée:

```
  On Pir#Switch=1 do
   If %systime% < 07:00:00
    Gpio,16,1
```

```
   Endif
   If %systime% > 19:00:00
    Gpio,16,1
   Endif
  Endon
```

Cela met gpio 16 à 1 lorsque le pir est déclenché, si l'heure est avant 7 heures du matin ou après 19h00 le soir. (utile si vous n'avez pas de capteur de lumière)

SendToHTTP

Pour envoyer un message à un autre appareil, comme une commande pour allumer une lumière sur Domoticz

```
 On System#Boot do    //When the ESP boots, do
   timerSet,1,10      //Set Timer 1 for the next event in 10 seconds
 endon

 On Rules#Timer=1 do  //When Timer1 expires, do
   SendToHTTP
192.168.0.243,8080,/json.htm?type=command&param=switchlight&idx=174&switchc
md=On
 endon
```

De nombreux utilisateurs ont signalé des problèmes avec les commandes tronquées, en particulier lorsque vous essayez d'envoyer des commandes à domoticz. Cela semble être une erreur d'analyse. Il existe la solution de contournement suivante :

```
   SendToHTTP
192.168.0.243,8080,/json.htm?type=param=switchlight&command&idx=174&switchc
md=On
```

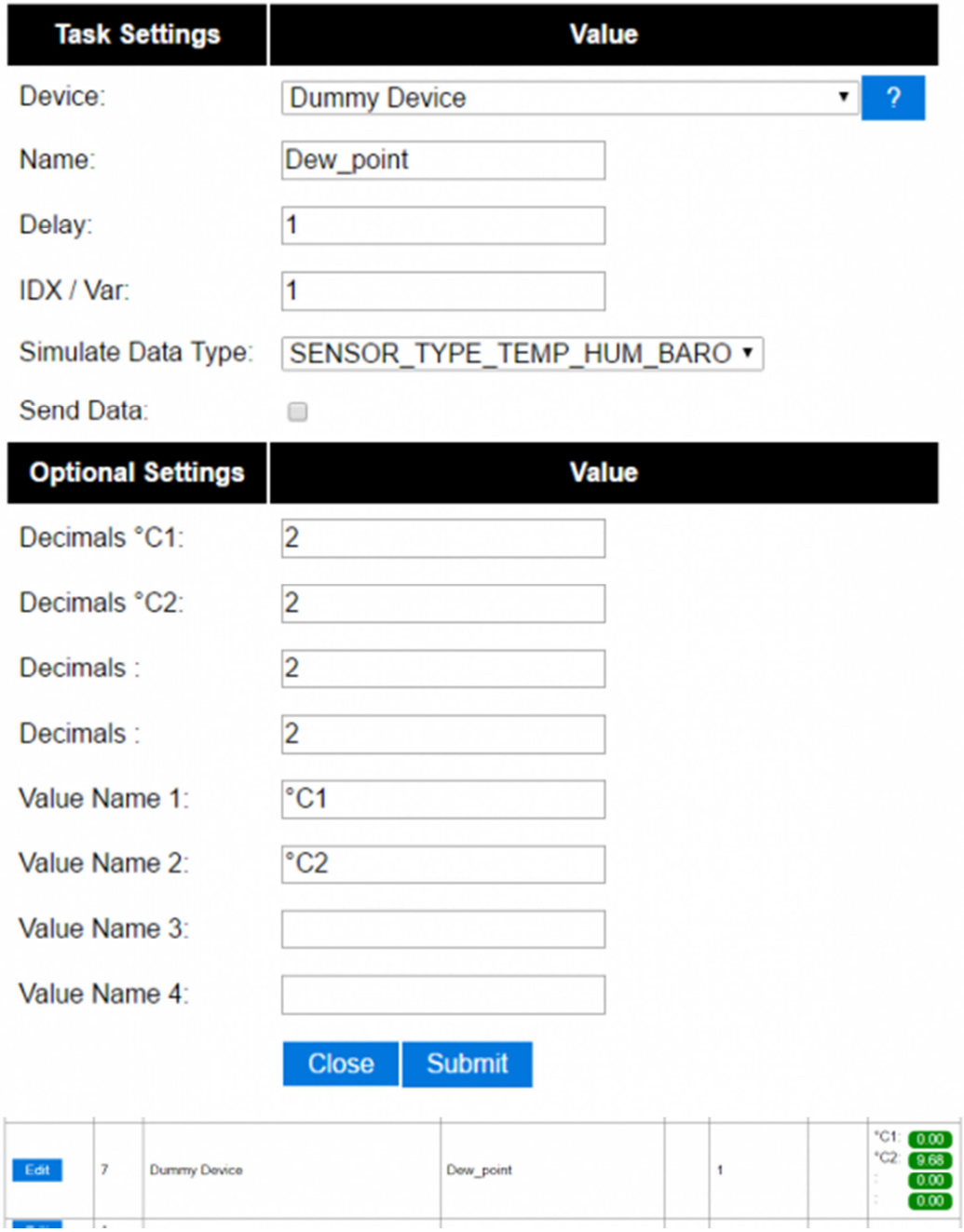
Point de rosée pour capteurs de température / humidité (BME280 par exemple)

Si vous avez un capteur qui surveille la température de l'air et l'humidité relative, vous pouvez calculer le point de rosée avec des règles. Cet exemple utilise MQTT pour publier les valeurs, mais vous pouvez changer cela pour ce que vous voulez. Nous utilisons également un "périphérique fictif" pour vider les valeurs, cet exemple utilise deux BME280 avec différentes adresses i2c.

Set up BME280 devices as follows:

Edit	5	Temperature & Humidity & Pressure - BME280	TempHumidityPressure_OUTSIDE		1	GPIO-4 GPIO-5	°C: 22.76 %RH: 39 hPa: 1020
Edit	6	Temperature & Humidity & Pressure - BME280	TempHumidityPressure_INSIDE		1	GPIO-4 GPIO-5	°C: 22.27 %RH: 38 hPa: 1021

Configurez le périphérique factice comme suit:



Pour le point de rosée à l'extérieur:

```
 on TempHumidityPressure_OUTSIDE#%RH do
  TaskValueSet,7,1,[TempHumidityPressure_OUTSIDE#°C]-(100-
[TempHumidityPressure_OUTSIDE#%RH])/5  // "7" is the number of the task
that the dummy device is on, "1" is its first value where we dump our
result
  if [TempHumidityPressure_OUTSIDE#%RH]>49
   Publish %sysname%/DewPoint_OUTSIDE/°C,[Dew_point#°C1]
  else
   Publish %sysname%/DewPoint_OUTSIDE/°C,[Dew_point#°C1]*  //This asterix
shows that the calculation is not correct due to the humidity being below
50%!
  endif
 endon
```

Pour le point de rosée à l'intérieur:

```
 on TempHumidityPressure_INSIDE#%RH do
  TaskValueSet,7,2,[TempHumidityPressure_INSIDE#°C]-(100-
[TempHumidityPressure_INSIDE#%RH])/5  // "7" is the number of the task that
the dummy device is on, "2" is its second value where we dump our result
  if [TempHumidityPressure_INSIDE#%RH]>49
   Publish %sysname%/DewPoint_INSIDE/°C,[Dew_point#°C2]
  else
   Publish %sysname%/DewPoint_INSIDE/°C,[Dew_point#°C2]*  //This asterix
shows that the calculation is not correct due to the humidity being below
50%!
  endif
 endon
```

Signaler IP toutes les 30 secondes en utilisant MQTT

Cette règle fonctionne également comme un ping ou un battement de coeur de l'unité. S'il n'a pas publié de numéro IP pendant plus de 30 secondes, l'unité rencontre des problèmes.

```
On System#Boot do    //When the ESP boots, do
 Publish %sysname%/IP,%ip%
  timerSet,1,30      //Set Timer 1 for the next event in 30 seconds
endon

On Rules#Timer=1 do  //When Timer1 expires, do
 Publish %sysname%/IP,%ip%
  timerSet,1,30       //Resets the Timer 1 for another 30 seconds
endon
```

Rapports personnalisés à Domoticz avec IDX propre

Cette règle a été présentée comme une solution de contournement pour un problème où un capteur avait trois valeurs différentes mais une seule valeur IDX. Vous pouvez publier vos propres messages Domoticz (MQTT ou HTTP) en utilisant cette méthode. Ci-dessous nous utilisons le plugin INA219 qui a 3 valeurs dont les deux secondes sont Amps et Watts, à titre d'exemple, nous voulons les publier en tant que messages personnalisés avec une valeur IDX unique.

MQTT

```
on INA219#Amps do
  Publish domoticz/in,{"idx":123456,"nvalue":0,"svalue":"[INA219#Amps]"}
//Own made up IDX 123456
endon

on INA219#Watts do
  Publish domoticz/in,{"idx":654321,"nvalue":0,"svalue":"[INA219#Watts]"}
//Own made up IDX 654321
endon
```

HTTP

```
on INA219#Amps do
  SendToHTTP
192.168.1.2,8080,/json.htm?type=command&param=udevice&idx=123456&nvalue=0&s
value=[INA219#Amps] //Own made up IDX 123456
```

```
endon

on INA219#Watts do
  SendToHTTP
192.168.1.2,8080,/json.htm?type=command&param=udevice&idx=654321&nvalue=0&s
value=[INA219#Watts] //Own made up IDX 654321
endon
```

(Étant donné que votre serveur Domoticz est sur "192.168.1.2:8080", vous devez changer le numéro IP et PORT de votre serveur.Si la publication HTTP ne fonctionne pas, reportez-vous à cette rubrique pour obtenir une solution de contournement.)

Un bouton, plusieurs actions en utilisant une pression longue

À l'aide d'un appareil «interrupteur normal» qui est normalement réglé sur faible (0), vous pouvez effectuer l'une des deux actions suivantes. Si vous relâchez le bouton en moins d'une seconde ou appuyez dessus pendant plus d'une seconde:

```
on Button#State=1 do

  timerSet,1,1
endon

on rules#timer=1 do
 if [Button#State]=0
  //Action if button is short pressed
 else
  //Action if button is still pressed
 endif
endon
```