

Programmation Chauffage avec le noeud "Ramp-Thermostat"

node-red-contrib-ramp-thermostat 0.8.3

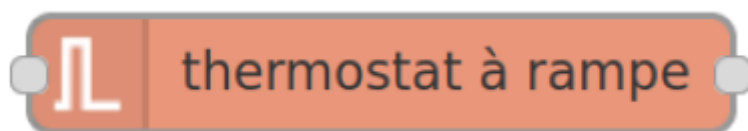
Un noeud Node-RED qui émule un thermostat programmable.

```
npm install node-red-contrib-ramp-thermostat
```

thermostat à rampe

version npm : v0.8.3

Un noeud de contribution Node-RED qui émule un thermostat programmable.



Source Wikipédia : Un thermostat programmable est un thermostat conçu pour ajuster la température selon une série de réglages programmés qui s'appliquent à différents moments de la journée. Les thermostats programmables sont également appelés thermostats à horloge .

Le thermostat à rampe commande un actionneur en fonction de la température d'entrée actuelle et de la température cible (consigne). La température cible est définie pour une période de 24 heures (00:00-23:59). Une programmation hebdomadaire ou pour les jours fériés est possible grâce à différents profils.

Configuration

La température cible est définie par un profil qui fournit sa valeur en fonction de l'instant présent . Ce profil est constitué de plusieurs points dont les connexions forment une séquence de lignes. Le moment de commutation peut être optimisé en définissant une ligne de gradient , 00:00-23:59 par exemple une courbe en U.ramp



Un profil doit comporter au moins 2 points et doit commencer à 00:00 et se terminer à 23:59.

L'hystérésis sert à éviter les oscillations. La [+] valeur est ajoutée à la cible et la [-] valeur (absolue) est soustraite de la cible. Dans cette zone neutre, aucune action n'est effectuée.

Usage

Ce nœud attend un `numericmessage` (`msg.payload`) contenant la température actuelle (valeur numérique). Le sujet (`msg.topic`) doit être défini `setCurrent`. Il calculera la température cible en fonction de la valeur de `msg.payload` à l'instant présent et affichera trois valeurs :

- état (booléen)
- température actuelle (nombre)
- température cible (nombre)

L'état (vrai/faux) sert à commander un actionneur. Les sorties de température actuelle et cible peuvent être connectées, par exemple, à un nœud `ui_chart`.

Paramètres d'exécution

définir la cible

```
msg.topic: setCurrent
msg.payload: nn.n (number)
```

La cible restera valide jusqu'à ce qu'une nouvelle cible ou un nouveau profil soit défini, ou jusqu'à ce que Node-RED soit redémarré.

setHysteresisPlus

```
msg.topic: setHysteresisPlus
msg.payload: nn.n (number)
```

L'hystérésis restera valable jusqu'à ce qu'une nouvelle hystérésis soit définie ou jusqu'à ce que Node-RED soit réinitialisé.

définirHysteresisMinus

- `msg.topic: setHysteresisMinus`
- `msg.payload: nn.n (number)`

L'hystérésis restera valable jusqu'à ce qu'une nouvelle hystérésis soit définie ou jusqu'à ce que Node-RED soit réinitialisé.

obtenirProfil

- `msg.topic: getProfile`
- `msg.payload: profile-name`

L'objet profil est envoyé à la sortie 3 :

[profil.json](#)

```
msg.topic: getProfile
msg.payload: {
  "name": "profile-name",
  "points": [{
    "00:00": 18
  }, {
    "04:00": 18
  }, {
    "08:00": 20.5
  }, {
    "12:00": 20.5
  }, {
    "12:00": 19
  }, {
    "12:30": 19
  }, {
    "13:30": 20.5
  }, {
    "19:00": 20.5
  }, {
    "19:00": 18
  }, {
    "23:59": 18
  }]
}
```

From:

<https://chanterie37.fr/fablab37110/> - Castel'Lab le Fablab MJC de Château-Renault

Permanent link:

<https://chanterie37.fr/fablab37110/doku.php?id=start:nodered:thermostat&rev=1766094787>

Last update: 2025/12/18 22:53

