Template Sensoren

- Last Day of month
- Pelletsverbrauch
- Wochentag
- Time, Date Formatierung und Templating
- Basiswissen und Links
- Wetterstation über wonderground REST API

Last Day of month

Der Code ist eine Packages Datei

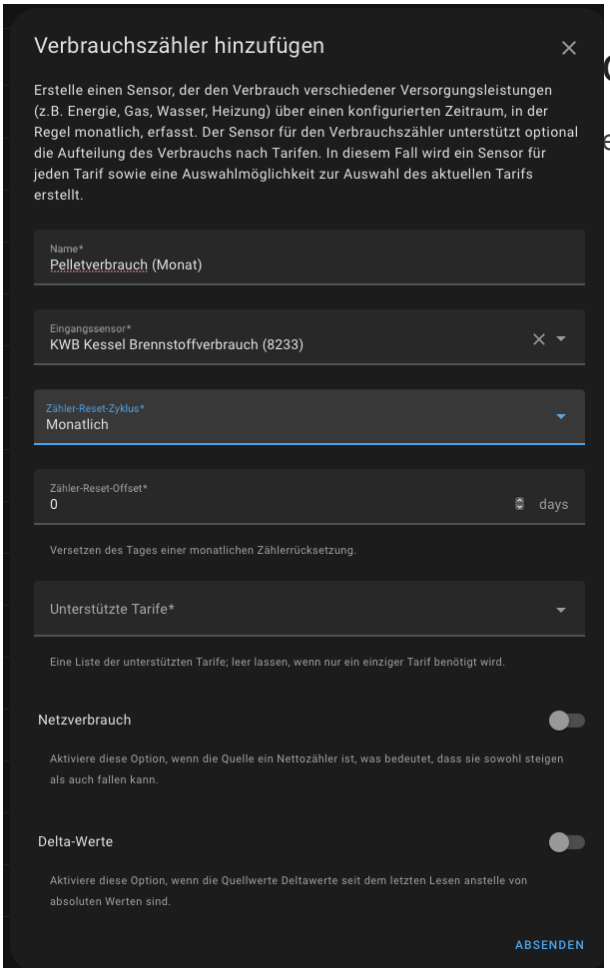
Der Sensor gibt ein Bool zurück.

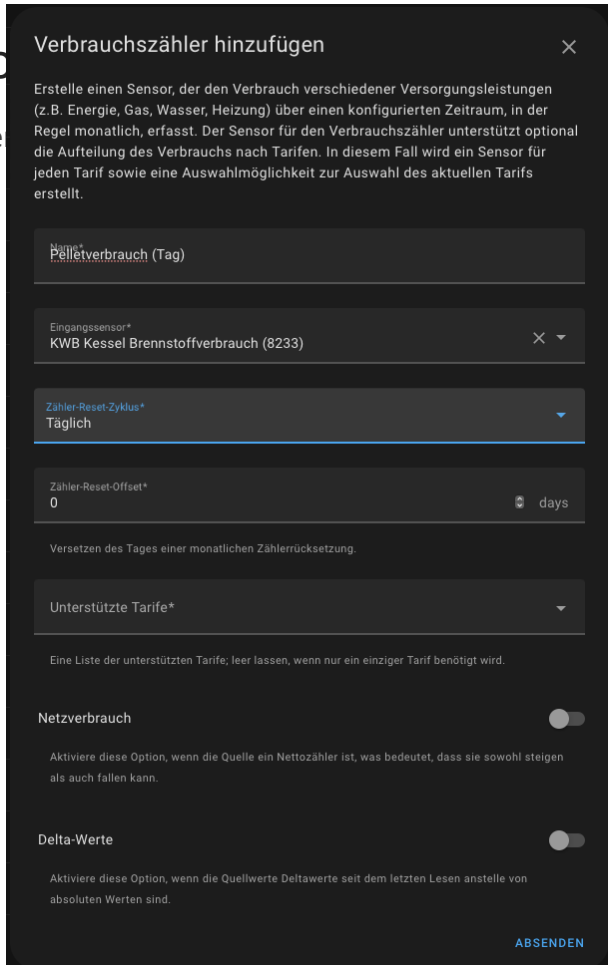
Als Binary Sensor dann auf on / off zu prüfen.

last_day_of_month.yaml

```
binary_sensor:
  - platform: template
    sensors:
      last_day_of_the_month:
        friendly_name: 'Last Day of the Month'
        entity_id: sensor.date
        icon_template: "mdi:calendar"
        value_template: "{{ int(((as_timestamp(now()) + 86400) | timestamp_custom('%d', true))) == 1 | bool }}"
```

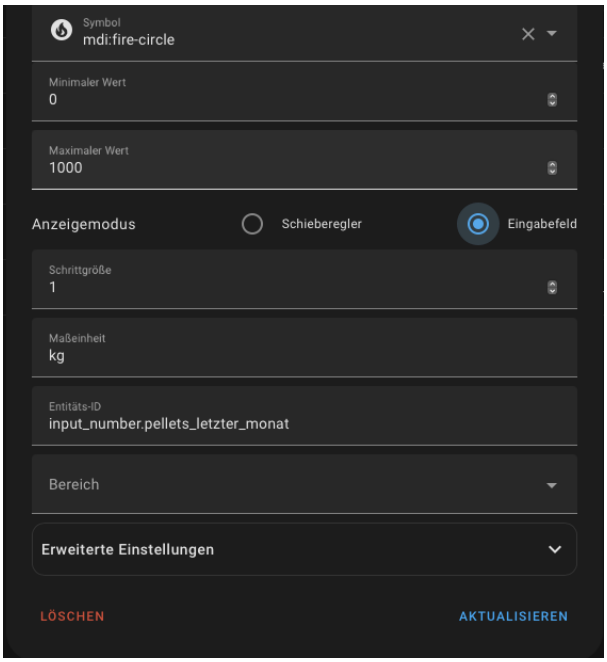
Pelletsverbrauch

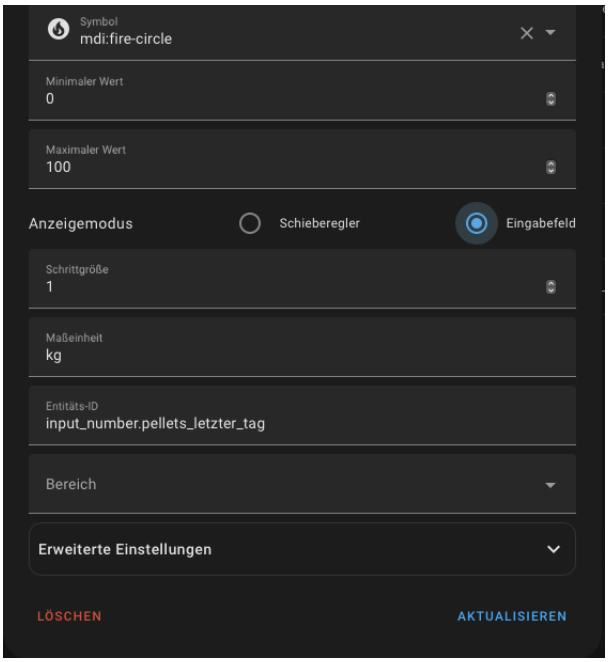




Letzer Tag / Monat Sensor

Um den Verbrauch des letzten Tages oder Monats zu zeigen, nutze ich einen Helfer "Input Number" mit einer Automation, die diesen Helfer immer um 23:59 des Vortages oder Vormonats setzt.





Automatisierungen

Letzer Tag

```
alias: Pellets letzter Tag
description: ""
trigger:
  - platform: time
    at: "23:59:50"
condition: []
action:
  - service: input_number.set_value
    data:
      value: "{{ states('sensor.pellets_tag') }}"
      entity_id: input_number.pellets_letzter_tag
mode: single
```

Letzter Monat

Um den letzten Tag im Monat zu bestimmen wird ein zusätzlicher Template Sensor eingesetzt. Last Day of month

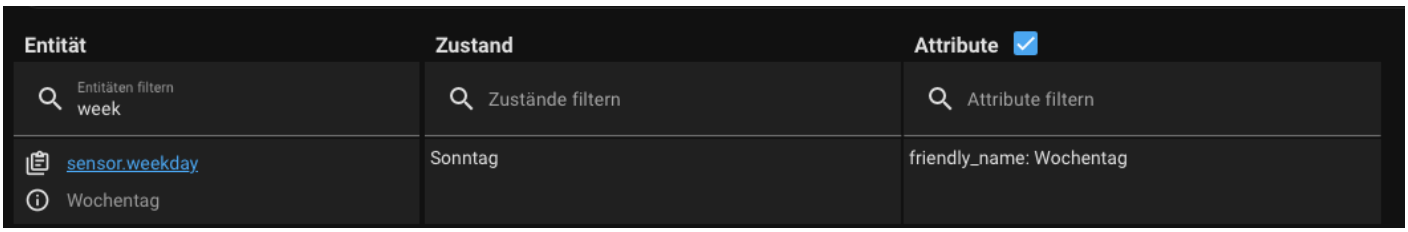
```
alias: Pellets letzter Monat
description: ""
trigger:
  - platform: time
    at: "23:59:50"
condition:
  - condition: state
    entity_id: binary_sensor.last_day_of_the_month
    state: "on"
action:
  - service: input_number.set_value
    data:
      value: "{{ states('sensor.pellets_monat') }}"
      entity_id: input_number.pellets_letzter_monat
mode: single
```

Wochentag

Manchmal möchte man in Automationen auf den Wochentag setzen.

Um das zu vereinfachen hier der Templatesensor "sensor.weekday"

```
- platform: template
  sensors:
    weekday:
      friendly_name: "Wochentag"
      value_template: >
        {% set weekday = int(as_timestamp(states('sensor.date_time_iso')) | timestamp_custom('%u')) %}
        {% if weekday == 7 -%}
          Sonntag
        {%- elif weekday == 6 -%}
          Samstag
        {%- elif weekday == 5 -%}
          Freitag
        {%- elif weekday == 4 -%}
          Donnerstag
        {%- elif weekday == 3 -%}
          Mittwoch
        {%- elif weekday == 2 -%}
          Dienstag
        {%- elif weekday == 1 -%}
          Montag
        {%- else -%}
          error
        {%- endif %}
```



Time, Date Formatierung und Templating

Das Formatieren von Datum und Zeit ist oft nicht so einfach.

https://www.home-assistant.io/integrations/input_datetime

Formatierung Strings für z.b. die "custom_timestamp" Funktion.

%FORMAT String	Description
%%	a literal %
%a	locale’s abbreviated weekday name (e.g., Sun)
%A	locale’s full weekday name (e.g., Sunday)
%b	locale’s abbreviated month name (e.g., Jan)
%B	locale’s full month name (e.g., January)
%c	locale’s date and time (e.g., Thu Mar 3 23:05:25 2005)
%C	century; like %Y, except omit last two digits (e.g., 21)
%d	day of month (e.g, 01)
%D	date; same as %m/%d/%y
%e	day of month, space padded; same as %_d
%F	full date; same as %Y-%m-%d
%g	last two digits of year of ISO week number (see %G)
%G	year of ISO week number (see %V); normally useful only with %V
%h	same as %b
%H	hour (00..23)
%I	hour (01..12)
%j	day of year (001..366)
%k	hour (0..23)
%l	hour (1..12)
%m	month (01..12)
%M	minute (00..59)

%FORMAT String	Description
%n	a newline
%N	nanoseconds (000000000..999999999)
%p	locale’s equivalent of either AM or PM; blank if not known
%P	like %p, but lower case
%r	locale’s 12-hour clock time (e.g., 11:11:04 PM)
%R	24-hour hour and minute; same as %H:%M
%s	seconds since 1970-01-01 00:00:00 UTC
%S	second (00..60)
%t	a tab
%T	time; same as %H:%M:%S
%u	day of week (1..7); 1 is Monday
%U	week number of year, with Sunday as first day of week (00..53)
%V	ISO week number, with Monday as first day of week (01..53)
%w	day of week (0..6); 0 is Sunday
%W	week number of year, with Monday as first day of week (00..53)
%x	locale’s date representation (e.g., 12/31/99)
%X	locale’s time representation (e.g., 23:13:48)
%y	last two digits of year (00..99)
%Y	year
%z	+hhmm numeric timezone (e.g., **-0400**)
%:z	+hh:mm numeric timezone (e.g., **-04**:00)
%::z	+hh:mm:ss numeric time zone (e.g., **-04**:00:00)
%:::z	numeric time zone with : to necessary precision (e.g., **-04**, +05:30)
%Z	alphabetic time zone abbreviation (e.g., EDT)

Basiswissen und Links

Sensor

https://www.home-assistant.io/integrations/sensor/

Template

https://www.home-assistant.io/integrations/template/

Template Editor

https://www.home-assistant.io/docs/tools/dev-tools/#template-editor

Jinja2 Dokumentation

https://jinja.palletsprojects.com/en/latest/templates/

Wetterstation über wonderground REST API

Wetterstation Besser WI-FI Colour Weather Station + 5in1

Supplement WLAN DE.pdf Manual EN DE.pdf

Die Daten hole ich mir über die REST API

secret.yaml

```
wunderground_url:
https://api.weather.com/v2/pws/observations/current?stationId=IKLEIN82&format=json&units=e&apiKey=XXXX
```

config/include_sensors/wonderground_rest.yml

```
- platform: rest
  name: pws_report
  #friendly_name: PWS Report
  json_attributes:
    - observations
  value_template: '{{ value_json["observations"][0]["obsTimeLocal"].title() }}'
  resource: !secret wunderground_url
  scan_interval: 300

### 29.12.2022
# {"observations":[{
# "stationID":"IKLEIN82",
# "obsTimeUtc":"2022-12-29T12:15:30Z",
# "obsTimeLocal":"2022-12-29 13:15:30",
# "neighborhood":"Kleinwallstadt",
# "softwareType":null,
# "country":"DE",
# "solarRadiation":null,
# "lon":9.164517,
# "realtimeFrequency":null,
```

```
# "epoch":1672316130,
# "lat":49.866959,
# "uv":null,
# "winddir":192,
# "humidity":90,
# "qcStatus":1,
# "imperial":{"temp":51,"heatIndex":51,
# "dewpt":48,
# "windChill":51,
# "windSpeed":1,
# "windGust":1,
# "pressure":29.62,
# "precipRate":0.00,
# "precipTotal":0.07,
# "elev":115
# }}]
###

- platform: template
  sensors:
    pws_location:
      value_template: "{{ states.sensor.pws_report.attributes.observations[0].neighborhood }}"
      friendly_name: Location
    pws_station_id:
      value_template: "{{ states.sensor.pws_report.attributes.observations[0].stationID }}"
      friendly_name: Staion ID
    pws_type:
      value_template: "{{ states.sensor.pws_report.attributes.observations[0].softwareType }}"
      friendly_name: Station Type
    pws_wind_dir:
      value_template: "{{ states.sensor.pws_report.attributes.observations[0].winddir }}"
      friendly_name: Wind Direction
      unit_of_measurement: "°"
    pws_wind_gust:
      value_template: "{{ states.sensor.pws_report.attributes.observations[0].windGust }}"
      friendly_name: Wind Böe
    pws_wind_speed:
      value_template: "{{ states.sensor.pws_report.attributes.observations[0].imperial.windSpeed }}"
      friendly_name: Wind Speed
      unit_of_measurement: "km/h"
```

```
pws_humidity:
  value_template: "{{ states.sensor.pws_report.attributes.observations[0].humidity }}"
  friendly_name: Humidity
  unit_of_measurement: "%"
pws_precip_total:
  value_template: "{{ states.sensor.pws_report.attributes.observations[0].imperial.precipTotal }}"
  friendly_name: Rain Total
  unit_of_measurement: "in"
pws_precip_rate:
  value_template: "{{ states.sensor.pws_report.attributes.observations[0].imperial.precipRate }}"
  friendly_name: Rain Rate
  unit_of_measurement: "in"
pws_temp:
  value_template: "{{ states.sensor.pws_report.attributes.observations[0].imperial.temp }}"
  friendly_name: Temp (F)
  unit_of_measurement: "°F"
pws_solar:
  friendly_name: Sonnenstrahlung
  value_template: "{{ states.sensor.pws_report.attributes.observations[0].solarRadiation }}"

pws_windstaerke_ms:
  friendly_name: "Windstärke"
  unit_of_measurement: "m/s"
  value_template: "{{ states('sensor.pws_wind_speed') | float(0) / 3.6 | round() }}"

pws_temp_c:
  friendly_name: "Temperatur (°C)"
  unit_of_measurement: "°C"
  value_template: "{{ (( states('sensor.pws_temp') | float(0) - 32 ) * 5 / 9 ) | round(2, 'floor') }}"
```