Home Assistant

- IKEA RODRET Fernbedienung für Home Assistant
- IKEA Starkvind Dashboard-Badge mit PM 2.5 Werten
- IKEA STYRBAR für Home Assistant
- Blueprint Entwicklungshilfe
- VALLHORN Bewegungssensoren

IKEA RODRET Fernbedienung für Home Assistant

Ganze Automationen als YAML Code... Komplett leere Automation anlegen und direkt in den YAML Code Editor rein

Genutzt wird Zigbee Home Automation, für MQTT scheinen andere Automationen verwendet werden zu müssen

Einfacher Weg

Der einfachere Weg nennt sich Blueprint Exchange. Sprich:

https://community.home-assistant.io/t/ikea-rodret-somrig-tradfri-remotes-zha-z2m-control-light/591182
und ein paar andere...

Da ich den einfachen Weg genommen habe, kommen hier nur Anpassungen die ich nicht in Blueprints wiederfinden konnte

IKEA Starkvind Dashboard-Badge mit PM 2.5 Werten

YAML Code für einen Dashboard-Badge, welcher den PM2,5 Wert anzeigt, beim Klick aber die Einstellung vom Gerät aufruft, nicht die vom Sensor...

Das wichtigste ist der aktuelle PM 2.5 wert. Nicht der Zustand oder die Stufe. Kann die GUI nicht, also brauchen wir Sensoranzeige, wollen beim Klick aber trotzdem die Luftreinigereinstellungen haben

```
type: entity
show_name: true
show_state: true
show_icon: true
entity: sensor.LUFTREINIGER_pm25
icon: mdi:air-filter
state_content: state
tap_action:
  action: more-info
  entity: fan.LUFTREINIGER
name: Luft
```

IKEA STYRBAR für Home Assistant

Ausgangspunkt für Weißspektrum-Lampen

Ausgangspunkt für RGB-Farblampen

alles weitere sind Anpassungen die ich so in den Blueprints machte oder die mir von Funktion her nicht gefielen...

IKEA STYRBAR / TRADFRI Weißspektrum schalten

Das ist kein voller Automation YAML, nur der Teil für die Buttons der RODRET Fernbedienung um die Lichttemperatur anzupassen.
Werden nicht ganz die IKEA DIRIGERA Schritte sein, weil wir hier mit Wertveränderungen statt mit Presets arbeiten.

Die Entitäts-ID habt ihr in den Steuerelementen, dort die Eigenschaften von der Lampe öffnen, die Device ID habt ihr in der URL wenn ihr auf der Seite des Geräts seid.

```
button_left_short:
      - service: light.turn_on
        target:
          device_id: GLÜHBIRNE
        data:
          color_temp: >
            {% set step = 50 %}
            {% set current = state_attr('ENTITÄTS-ID', 'color_temp') | int %}
            {% set new = [current + step, 500] | min %}
            {{ new }}
    button_right_short:
      - service: light.turn_on
        target:
          device_id: GLÜHBIRNE
        data:
```

```
          color_temp: >
            {% set step = 50 %}
            {% set current = state_attr('ENTITÄTS-ID', 'color_temp') | int %}
            {% set new = [current - step, 153] | max %}
            {{ new }}
```

Blueprint Entwicklungshilfe

Ereignisse auslesen

Open your Home Assistant instance and show your event developer tools.

Die Entwicklungstools von Home Assistant sind euer Freund.
Der Punkt "Ereignisse Abonnieren" zeigt euch an, welche Tastendruck-Aktionen welche Ereignisse auslösen auf die euer Blueprint gebunden werden muss.

Weil hier Zigbee läuft müssen wir die ZHA_Events abonnieren

Im Falle von IKEA SYMFONISK sieht das dann in etwa so aus.

```
event_type: zha_event
data:
  device_ieee: xx:xx:xx:xx:xx:xx:xx:00
  device_id: xxxxx0000xx0xx00x0xxx0000000x000
  unique_id: xx:xx:xx:xx:xx:xx:xx:xx:x:0x0000
  endpoint_id: 1
  cluster_id: 6
  command: "on"
  args: []
  params: {}
origin: LOCAL
time_fired: "2025-11-09T10:26:43.663802+00:00"
context:
  id: 00X0X0XXXXXXXX0X0X000X0XXX
  parent_id: null
  user_id: null
```

In der YAML Notation sind die Zeileneinrückungen am Anfang ziemlich wichtig. Nicht so kritisch wie bei FORTRAN, aber die geben statt den {} Klammern aus jeder anderen Sprache an, was zu welchem Element gehört.

Hier ist der Anschalter gedrückt worden, der meldet ein "on" command, auf das wir im Blueprint Aktionen eintragen können.

```
event_type: zha_event
data:
```

```
  device_ieee: xx:xx:xx:xx:xx:xx:xx:00
  device_id: xxxxx0000xx0xx00x0xxx0000000x000
  unique_id: xx:xx:xx:xx:xx:xx:xx:x:0x0000
  endpoint_id: 1
  cluster_id: 5
  command: press
  args:
    - 256
    - 13
    - 0
  params:
    param1: 256
    param2: 13
    param3: 0
origin: LOCAL
time_fired: "2025-11-09T10:27:18.174521+00:00"
context:
  id: 00X0X0XXXXXXXX0X0X000X0XXX
  parent_id: null
  user_id: null
```

Das hier ist jetzt keine Standardtaste mehr, sondern eine Funktionstaste der STYRBAR Fernbedienung. command:press bedeutet schlicht dass sie gedrückt wurde, und der args: Abschnitt sagt uns welche Taste das eigentlich war.

Jetzt kann die Taste aber auch lange gedrückt werden, was zwei Events erzeugt:

```
  endpoint_id: 1
  cluster_id: 6
  command: "on"
  args: []
  params: {}
origin: LOCAL
```

```
  endpoint_id: 1
  cluster_id: 5
  command: release
  args:
    - 0
  params:
    param1: 0
```

```
origin: LOCAL
```

Das ist mit allen Zigbee Geräten so, die irgendwelche Tasten haben, die irgendwelche Zustände an den Coordinator melden müssen

Zustände

Open your Home Assistant instance and show your state developer tools.

Zustände sind aktuelle Eigenschaften von Zigbee-Geräten. Dinge wie ob eine Glühbirne gerade leuchtet, welche Helligkeit sie hat, oder Lichttemperatur, Lichtfarbe und so weiter.

Hier braucht ihr eure Entitäts-ID, die findet ihr wenn ihr im Home Assistant auf euer Gerät geht und im Abschnitt "Steuerelemente" aufs Gerät und dann das Zahnrad klickt.

Nehmen wir jetzt mal Beispielhaft eine IKEA TRADFRI E14 Glühbirne:

```
min_color_temp_kelvin: 2202
max_color_temp_kelvin: 4000
min_mireds: 250
max_mireds: 454
effect_list: off, colorloop
supported_color_modes: color_temp, xy
effect: null
color_mode: null
brightness: null
color_temp_kelvin: null
color_temp: null
hs_color: null
rgb_color: null
xy_color: null
off_with_transition: false
off_brightness: 254
friendly_name: ******
supported_features: 44
```

Das ist eine ganze Menge: Sie ist gerade ausgeschaltet, weil wir keine Brightness haben und auch keine anderen Werte die ein Licht beschreiben würden.

Sie kann Lichttemperatur (color_temp), und Farblicht (sieht man am xy, das ist ein anderes Farbsystem, scheinbar passender für LED als RGB)...

WEITERER ARTIKEL IN ARBEIT

VALLHORN Bewegungssensoren

Die VALLHORN Sensoren haben einen sehr engen Bereich in dem Sie vernünftig funktionieren: Akkuspannung von 1,2-1,3 V ist Pflicht, Akkus müssen schon bei 40% Kapazität nachgeladen werden.

Nach Berichten im Netz laufen die Sensoren mit einem Paar Akkus aber in der Spanne zwischen 100 und 40% Kapazität auch gut 10-12 Monate lang.

Außerhalb von 2,4-2,6 V gibt es mehrfach falsche Auslösungen.

Blueprint

Setzt zwei Sensoren mit einer besonderen Logik zusammen, einer von beiden darf einschalten, erst wenn beide Sensoren Freiheit melden, wird abgeschaltet.

DEV-Version: https://github.com/paradonym/ha-blueprints/blob/main/vallhorn_dev.yaml