Automatyzacje i powiadomienia: alarm temperatury w serwerowni bez dublowania Zabbixa

◫◫ Poziom: średnio

◫◫ Dotyczy: Home Assistant 2026.9, ESPHome 2026.9.1, Zabbix na mon01, poczta MailCow na docker01

▯ Czas: ~75 min

Pokazujemy, jak z czujników `esp01` i `esp02` budujemy w Home Assistant automatyzacje i powiadomienia, a jednocześnie nie dublujemy alarmów, które należą do Zabbixa. Przykłady to alarm temperatury w serwerowni (firma) i zalanie w pralni (dom). Wracamy tu przy pisaniu nowej automatyzacji, przy podłączaniu poczty albo telefonu i przy pytaniu, kto w Zielonej odpowiada za który alarm.

◫◫ Skróty użyte na tej stronie (19)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
BMS	Building Management System: system zarządzania budynkiem, który zbiera odczyty z czujników (temperatura, wilgotność) i steruje klimatyzacją oraz wentylacją
DCIM	Data Center Infrastructure Management: dokumentacja tego, co fizycznie stoi w serwerowni, czyli szafy, urządzenia i okablowanie
ESP32	Tani mikrokontroler z Wi-Fi i Bluetooth firmy Espressif: malutki komputer, z którego budujemy własne czujniki sieciowe
ESPHome	Oprogramowanie, które zamienia ESP32 w gotowy czujnik sieciowy bez pisania kodu od zera, a odczyty udostępnia przez sieć
GSM	Global System for Mobile Communications: standard sieci komórkowej, z którego korzysta modem wysyłający SMS

Skrót	Co znaczy
HTTP	Hypertext Transfer Protocol: protokół stron WWW
ICMP	Internet Control Message Protocol: protokół diagnostyczny sieci, którym działa między innymi ping
JSON	JavaScript Object Notation: tekstowy format danych z klamrami i cudzysłowami
JSONPath	Zapis ścieżki, która wskazuje jedną wartość w dokumencie JSON, np. $.value to pole value na samym wierzchu
OS	Operating System: system operacyjny
PoE	Power over Ethernet: zasilanie urządzenia tym samym kablem sieciowym, którym płyną dane
REST	sposób budowania API przez zwykłe żądania HTTP (GET, POST) zamiast osobnego protokołu
SMS	Short Message Service: krótka wiadomość tekstowa na telefon, która dociera także bez internetu
SMTP	Simple Mail Transfer Protocol: protokół wysyłania poczty
STARTTLS	włączenie szyfrowania w trakcie już nawiązanej, nieszyfrowanej sesji pocztowej
UPS	Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS
YAML	YAML Ain't Markup Language: format plików konfiguracyjnych oparty na wcięciach

Fiszka

□□ TYLKO ODCZYT

Włączenie pakietów w configuration.yaml: packages

```
homeassistant:
  packages: !include_dir_named packages
```

Pakiet to plik YAML w katalogu `packages`, który zbiera w jednym miejscu wszystko, co dotyczy jednego tematu: pomocniki, czujniki pochodne, automatyzacje i polecenia [1]. Plik `serwerownia.yaml` trzyma wszystko o serwerowni, a plik `dom.yaml` wszystko o domu. Blok `homeassistant:` jest w `configuration.yaml` jeden, więc dopisujemy do niego tylko klucz `packages`.

□□ **YAML** = YAML Ain't Markup Language: format plików konfiguracyjnych oparty na wcięciach

PLIK KONFIGURACYJNY

Pomocnik i czujnik pochodny: pakiet serwerownia.yaml

```
input_boolean:
  serwerownia_tryb_serwisowy:
    name: "Serwerownia: tryb serwisowy"
    icon: mdi:wrench

template:
  - sensor:
      - name: "Serwerownia wlot max"
        unique_id: serwerownia_wlot_max
        unit_of_measurement: "°C"
        device_class: temperature
        state_class: measurement
        state: >-
          {{ [
               states('sensor.esp01_wlot_r1_dol') | float,
               states('sensor.esp01_wlot_r1_srodek') | float,
               states('sensor.esp01_wlot_r1_gora') | float
             ] | max }}
        availability: >-
          {{ has_value('sensor.esp01_wlot_r1_dol')
             and has_value('sensor.esp01_wlot_r1_srodek')
             and has_value('sensor.esp01_wlot_r1_gora') }}
```

Przełącznik `serwerownia_tryb_serwisowy` wyłącza reakcje lokalne na czas prac w szafach. Czujnik `serwerownia_wlot_max` bierze najwyższy z trzech wlotów szafy R1. Gdy którakolwiek sonda jest niedostępna, blok `availability` robi z czujnika `unavailable`, zamiast liczyć maksimum z dwóch sond i udawać, że wszystko jest w porządku [2].

KLIKAMY W INTERFEJSIE

Sprawdzenie szablonu na żywo: Tools, Template

```
Settings → Tools → Template
```

W polu szablonu wpisujemy na przykład `{{ states('sensor.serwerownia_wlot_max') }}` oraz `{{ has_value('sensor.esp01_wlot_r1_gora') }}` i od razu widzimy wynik z prawdziwych stanów.

Szablon, który pokazuje dobrą liczbę w tym oknie, wkleja się do pliku bez poprawek. W wersjach starszych niż 2026.8 ta sama zakładka nazywała się Developer tools [17].

ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: okno Settings → Tools → Template w `ha01`, w lewym polu szablon z `states('sensor.serwerownia_wlot_max')`, w prawym wynik z temperaturą.

PLIK KONFIGURACYJNY

Polecenie wysyłania SMS przez bramkę: rest_command

```
rest_command:
  sms_bramka:
    url: "https://sms.zielona.example/api/wyslij"
    method: POST
    content_type: "application/json; charset=utf-8"
    headers:
      Authorization: "Bearer TokenBramkiSms"
    payload: '{"numer": "{{ numer }}", "tekst": "{{ tekst }}"}'
```

`rest_command` tworzy akcję `rest_command.sms_bramka`, którą wywołujemy z automatyzacji, podając `numer` i `tekst` [4]. Adres i format zapytania bramki SMS są przykładowe i zależą od dostawcy, więc dopasowujemy je do dokumentacji naszej bramki. Token wpisujemy przez `!secret`, a w repozytorium zostaje `secrets.yaml.przyklad`.

Strona integracji `sms` dla modemów GSM ma w dokumentacji status integracji usuniętej [18], więc SMS wysyłamy przez bramkę z interfejsem HTTP albo przez dostawcę z własną integracją. Listę integracji powiadomień sprawdzamy w dokumentacji naszej wersji [5].

SMS = Short Message Service: krótka wiadomość tekstowa na telefon, która dociera także bez internetu · **GSM** = Global System for Mobile Communications: standard sieci komórkowej, z którego korzysta modem wysyłający SMS · **HTTP** = Hypertext Transfer Protocol: protokół stron WWW

PLIK KONFIGURACYJNY

Automatyzacja: sygnalizator lokalny przy wysokiej temperaturze wlotu

```
automation:
  - id: serwerownia_sygnalizator_wlot
    alias: "Serwerownia: sygnalizator lokalny przy wysokiej temperaturze wlotu"
    description: "Reakcja lokalna. Alarm dla zespołu IT wysyła Zabbix, ta automatyzacja
nie wysyła poczty ani SMS."
    mode: restart
    triggers:
      - trigger: numeric_state
        entity_id: sensor.serwerownia_wlot_max
        above: 27
        for: "00:05:00"
        id: wysoka
      - trigger: numeric_state
        entity_id: sensor.serwerownia_wlot_max
        below: 25
        for: "00:05:00"
        id: norma
    conditions:
      - condition: state
        entity_id: input_boolean.serwerownia_tryb_serwisowy
        state: "off"
    actions:
      - choose:
          - conditions:
              - condition: trigger
                id: wysoka
            sequence:
              - action: switch.turn_on
                target:
                  entity_id: switch.serwerownia_sygnalizator
              - action: persistent_notification.create
                data:
                  notification_id: serwerownia_wlot_wysoki
                  title: "Serwerownia: wysoka temperatura wlotu"
                  message: >-
                    Najwyższy wlot: {{ states('sensor.serwerownia_wlot_max') }} °C
                    od 5 minut. Alarm i eskalacja są w Zabbixie na mon01.
          - conditions:
              - condition: trigger
```

```
      id: norma
    sequence:
      - action: switch.turn_off
        target:
          entity_id: switch.serwerownia_sygnalizator
      - action: persistent_notification.dismiss
        data:
          notification_id: serwerownia_wlot_wysoki
```

Wyzwalacz `numeric_state` uruchamia się w chwili przekroczenia progu, a nie przez cały czas, gdy wartość jest powyżej niego [6]. Opcja `for` wymaga, żeby stan trwał pięć minut, ale po restarcie Home Assistant albo przeładowaniu automatyzacji licznik zaczyna od zera [6]. Wyższy próg 27 °C i niższy 25 °C dają dwustopniową histerezę (zdjęcie alarmu dopiero 2 °C niżej), taką samą jak w Zabbixie.

Zasady dla każdej automatyzacji w pakiecie: stałe pole `id`, opis mówiący, co automatyzacja robi i czego nie robi, oraz identyfikatory wyzwalaczy (`wysoka`, `norma`), do których odwołuje się warunek `trigger` [7]. Przełącznik `switch.serwerownia_sygnalizator` to lampa przy drzwiach, podłączona do płytki ESPHome przekaźnikiem, i nie opisujemy tu jej budowy.

ESPHome = Oprogramowanie, które zamienia ESP32 w gotowy czujnik sieciowy bez pisania kodu od zera, a odczyty udostępnia przez sieć · **IT** = Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy · **SMS** = Short Message Service: krótka wiadomość tekstowa na telefon, która dociera także bez internetu

PLIK KONFIGURACYJNY

Automatyzacja: zapasowy alarm, gdy Zabbix milczy

```
- id: serwerownia_zapasowy_alarm_gdy_zabbix_milczy
  alias: "Serwerownia: zapasowy alarm, gdy Zabbix milczy"
  description: "Działa tylko wtedy, gdy mon01 nie odpowiada na ping od 10 minut."
  mode: single
  triggers:
    - trigger: numeric_state
      entity_id: sensor.serwerownia_wlot_max
      above: 32
      for: "00:03:00"
  conditions:
    - condition: state
      entity_id: binary_sensor.mon01_ping
```

```
        state: "off"
        for: "00:10:00"
    actions:
      - action: notify.dyzur_it
        data:
          title: "Serwerownia: wlot powyżej 32 °C"
          message: "Zabbix na mon01 milczy, a temperatura wlotu to {{
states('sensor.serwerownia_wlot_max') }} °C."
          data:
            tag: serwerownia-wlot-krytyczny
            group: serwerownia
            push:
              interruption-level: time-sensitive
      - action: rest_command.sms_bramka
        data:
          numer: "+48500000000"
          tekst: "Zielona: serwerownia, wlot {{ states('sensor.serwerownia_wlot_max') }}
C, Zabbix milczy."
```

Ten fragment dopisujemy w pliku zaraz pod pierwszą automatyzacją, jako kolejny element listy pod tym samym kluczem `automation:`. Numer telefonu i adres bramki są przykładowe. Warunek `for: "00:10:00"` na stanie ping sprawia, że krótka przerwa na restart `mon01` nie wywołuje zapasowego alarmu.

□□ TYLKO ODCZYT

Czujnik dostępności mon01: integracja Ping

```
Settings → Devices & services → Add integration → Ping (ICMP)
```

W polu Host wpisujemy 10.20.10.51, a liczbę zapytań zostawiamy domyślną. Integracja tworzy czujnik binarny pokazujący, czy host odpowiada, a domyślnie pyta co 30 sekund [8]. Identyfikator encji zmieniamy ręcznie na `binary_sensor.mon01_ping`, żeby pasował do automatyzacji.

□□ PLIK KONFIGURACYJNY

Grupa telefonów dyżurnych: notify, platform group

```
notify:
  - platform: group
    name: "Dyzur IT"
    services:
      - action: mobile_app_telefon_dyzurny
      - action: mobile_app_telefon_drugiego_admina
```

Grupa powiadomień daje jedną akcję, `notify.dyzur_it`, która wysyła wiadomość na kilka telefonów. Zmianę osoby dyżurnej robimy w jednym miejscu, a nie w każdej automatyzacji [9]. Nazwy `mobile_app_...` zależą od nazw telefonów zarejestrowanych w aplikacji i sprawdzamy je w Tools → Actions.

IT = Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy

TYLKO ODCZYT

Podłączenie poczty z MailCow: integracja SMTP

```
Settings → Devices & services → Add integration → SMTP
```

W kreatorze wpisujemy adres nadawcy `homeassistant@zielona.example`, hosta `poczta.zielona.example`, port 587, zabezpieczenie połączenia STARTTLS oraz nazwę użytkownika i hasło skrzynki [10]. Skrzynkę `homeassistant@zielona.example` zakładamy w MailCow, jak opisuje strona MailCow: codzienna praca z domenami, skrzynkami i spamem. W kreatorze dodajemy też pierwszego odbiorcę (`it@zielona.example`), a każdy odbiorca dostaje własną encję powiadomień [10].

ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran integracji SMTP w `ha01` z dodanym odbiorcą `it@zielona.example` i encją powiadomień, której identyfikator wpisujemy potem w automatyzacjach.

SMTP = Simple Mail Transfer Protocol: protokół wysyłania poczty · **STARTTLS** = włączenie szyfrowania w trakcie już nawiązanej, nieszyfrowanej sesji pocztowej

PLIK KONFIGURACYJNY

Automatyzacja domowa: zalanie w pralni (pakiet dom.yaml)

```
automation:
  - id: dom_zalanie_pralnia
    alias: "Dom: zalanie w pralni"
    description: "Czujnik wody pod pralką: powiadomienie w aplikacji i poczta."
    mode: single
    triggers:
      - trigger: state
        entity_id: binary_sensor.esp_pralnia_zalanie
        to: "on"
        for: "00:00:10"
    actions:
      - action: notify.mobile_app_telefon_tomka
        data:
          title: "Dom: woda w pralni"
          message: "Czujnik pod pralką wykrył wodę o {{ now().strftime('%H:%M') }}."
          data:
            tag: dom-zalanie-pralnia
            push:
              interruption-level: time-sensitive
      - action: notify.send_message
        target:
          entity_id: notify.poczta_dom
        data:
          title: "Dom: woda w pralni"
          message: "Czujnik pod pralką wykrył wodę o {{ now().strftime('%H:%M') }}."
```

W domu Home Assistant jest jedynym systemem, więc sam wysyła alarm i nie ma tu drugiego właściciela. Opcja `for: "00:00:10"` odsiewa chwilowe zwarcia styku, a `tag` powoduje, że kolejne powiadomienie o tym samym zdarzeniu zastępuje poprzednie, zamiast się dokładać [11]. Poziom `time-sensitive` na iPhonie przebija tryb skupienia, a poziom `critical` ma osobną dokumentację, którą sprawdzamy przed użyciem [11].

TYLKO ODCZYT

Odczyt stanu encji przez REST API: curl

```
curl --silent --header "Authorization: Bearer $(cat PlikZTokenem)"
http://AdresHomeAssistant/api/states/NazwaEncji
```

Odpowiedź to dokument JSON z polami `entity_id`, `state`, `last_changed`, `last_updated` i `attributes` [12]. Token długoterminowy tworzymy na stronie profilu użytkownika, w sekcji Security, i widzimy go tylko raz. W Zielonej zakładamy do tego osobne konto `zabbix` bez uprawnień administratora i trzymamy token w Vaultwardenie oraz w makrze sekretnym Zabbixa. Konto bez uprawnień administratora nie dociera do ustawień, ale nie zakładamy, że token jest tylko do odczytu, więc traktujemy go jak hasło.

REST = sposób budowania API przez zwykłe żądania HTTP (GET, POST) zamiast osobnego protokołu · **API** = Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą · **JSON** = JavaScript Object Notation: tekstowy format danych z klamrami i cudzysłowami · **curl** = client URL: pobieranie i wysyłanie przez HTTP z terminala

KLIKAMY W INTERFEJSIE

Element HTTP agent w Zabbixie czytający Home Assistant: Items

```
Data collection → Hosts → ha01 → Items → Create item
```

Pola elementu zapisujemy w osobnym bloku:

```
Name: Wlot max z Home Assistant
Type: HTTP agent
Key: ha.serwerownia.wlot_max
URL: http://10.20.10.54/api/states/sensor.serwerownia_wlot_max
Request type: GET
Headers: Authorization: Bearer {$HA.TOKEN}
Type of information: Numeric (float)
Update interval: 1m
Preprocessing: JSONPath, parametr $.state
```

Makro `{$HA.TOKEN}` ma typ Secret text, a makra, także sekretne, można wpisywać w nagłówkach elementu [13]. Gdy encja jest `unavailable`, wartość nie przechodzi na liczbę i element staje się nieobsługiwany, co zgłasza wyzwalacz oparty na `nodata`.

HTTP = Hypertext Transfer Protocol: protokół stron WWW · **HA** = High Availability: wysoka dostępność, czyli działanie mimo awarii jednego elementu

TYLKO ODCZYT

Pilnowanie ha01 przez Zabbixa: ping i port

```
Key: icmpping[10.20.10.54]
Key: net.tcp.service[http,10.20.10.54,80]
```

Oba elementy dodajemy na hoście `ha01` jako proste sprawdzenia (Simple check). Pierwszy mówi, czy maszyna odpowiada, drugi, czy interfejs WWW nasłuchuje. Port 80 dotyczy nowej instalacji Home Assistant OS, a dla kontenera na `docker01` wpisujemy 8123 i adres 10.20.10.42.

WWW = World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS · **OS** = Operating System: system operacyjny

TYLKO ODCZYT

Sprawdzenie i przeładowanie: ha core check

```
ha core check
```

W kontenerze używamy `docker exec homeassistant python -m homeassistant --script check_config --config /config`. Po poprawnym wyniku przeładowujemy zmiany w Settings → Tools → YAML. Zmiany w sekcjach `http` i `recorder` wymagają restartu, a dla pozostałych sprawdzamy w tym oknie, które mają przycisk przeładowania.

YAML = YAML Ain't Markup Language: format plików konfiguracyjnych oparty na wcięciach

Z życia wzięte

Z ŻYCIA WZIĘTE

Środa, 2:40 w nocy. Tomek dostaje dwa powiadomienia o tym samym: SMS z bramki, że wlot w szafie R1 ma 32,4 °C, i sygnał z aplikacji Home Assistant o tej samej temperaturze. Dzwoni do Oli, a ona odbiera tak samo zdezorientowana, bo jej telefon pokazał trzecie, w innym brzmieniu. Rano ustalają zasadę: alarm ma jednego właściciela, a drugi system tylko sygnalizuje na miejscu i wchodzi do akcji, gdy ten pierwszy milczy.

Jak to działa

Rozwiń: jak to działa pod spodem

Anatomia automatyzacji

> **Pojęcie: automatyzacja.** Zapisana reguła, która na zdarzenie (wyzwalacz) sprawdza, czy zachodzi dodatkowy warunek, i jeśli tak, wykonuje akcje. Home Assistant wykonuje ją u siebie, bez chmury i bez udziału człowieka.

Część	Klucz w YAML	Pytanie, na które odpowiada	Przykład z Zielonej
Wyzwalacz	`triggers`	co uruchamia automatyzację	wlot powyżej 27 °C przez 5 minut
Warunek	`conditions`	czy wolno jej zadziałać teraz	tryb serwisowy wyłączony
Akcja	`actions`	co robimy	włączamy lampę, tworzymy powiadomienie
Tryb	`mode`	co się dzieje, gdy wyzwalacz wróci w trakcie	`restart` dla lampy, `single` dla alarmu

Od wersji 2026.7 edytor oferuje wyzwalacze i warunki opisane celem, na przykład „gdy w sypialni spadnie poniżej 18 °C”, a istniejące automatyzacje w YAML działają bez zmian [3]. Dokumentacja zaznacza jednak, że część kluczy nowych wyzwalaczy zmieniła nazwy, więc automatyzacje używające starych kluczy poprawiamy ręcznie po aktualizacji [3].

Edytor czy YAML

W edytorze klikamy automatyzację i zapisujemy ją do pliku `automations.yaml`. W pakiecie piszemy ją w YAML i trzymamy w repozytorium Git razem z resztą konfiguracji. W Zielonej przyjmujemy zasadę: automatyzacje związane z serwerownią żyją w pakiecie `serwerownia.yaml`, a domowe i próbne w edytorze, żeby każda miała jedno miejsce życia.

Pakiety mają swoje reguły łączenia. Pomocniki wymagają unikalnych kluczy w całej konfiguracji, a automatyzacje w pakiecie zapisujemy pod jednym kluczem `automation:` i łączymy listy [1]. Ustawienie `auth_providers` musi zostać w głównym `configuration.yaml` [1].

Szablony

> **Pojęcie: szablon.** Fragment tekstu w podwójnych nawiasach klamrowych, który Home Assistant wylicza z bieżących stanów, na przykład `{{ states('sensor.esp01_wlot_r1_gora') | float }}`. Szablony budują czujniki pochodne i treść wiadomości.

Stan encji jest tekstem, więc liczbę zawsze zamieniamy filtrem `float`. Gdy czujnik jest niedostępny, `states()` zwraca tekst `unavailable`, który `float` bez wartości domyślnej zamienia na błąd, a z wartością domyślną na liczbę udającą pomiar [2]. Dlatego w czujniku `serwerownia_wlot_max` stan wylicza szablon, a o dostępności decyduje osobny szablon z `has_value`, który zwraca fałsz dla `unavailable` i `unknown` [2].

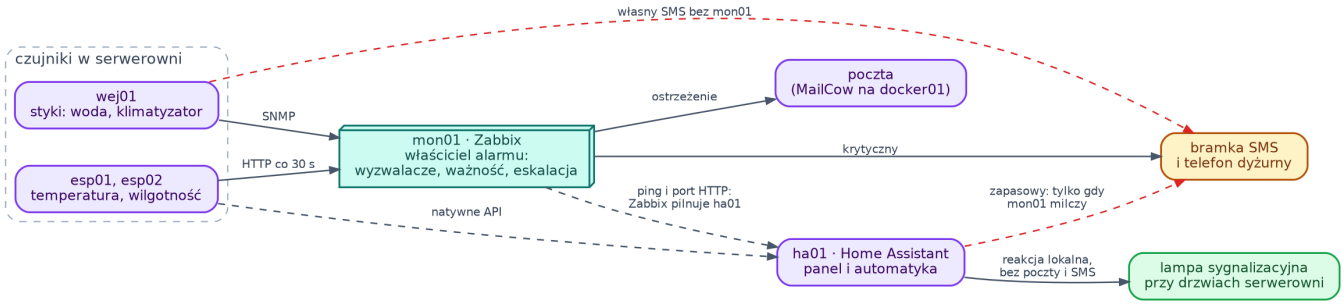
Kanały powiadomień

Kanał	Jak działa	Zalety	Ograniczenia
Panel w interfejsie (`persistent_notification`)	wpis w panelu powiadomień Home Assistant	działa bez internetu	widać go tylko po otwarciu interfejsu
Aplikacja mobilna (`notify.mobile_app_...`)	wiadomość przez usługę push Google albo Apple	dociera poza biurem, ma priorytety	do 500 powiadomień dziennie na urządzenie; treść nie jest szyfrowana w usłudze Firebase [14]
Push lokalny	wiadomość przez połączenie WebSocket z Home Assistant	nie przechodzi przez chmurę, nie liczy się do limitu [15]	działa, gdy telefon ma bezpośrednie połączenie z serwerem
Poczta (integracja SMTP, MailCow)	wiadomość z naszej poczty	ślad w skrzynce, dobre na ostrzeżenia	wolniejsza od push; zależy od MailCow na `docker01`
SMS przez bramkę (`rest_command`)	zapytanie HTTP do bramki	dociera bez internetu w telefonie	wymaga bramki, koszt wiadomości, własny format zapytania

W Zielonej poczta i SMS to kanały Zabbixa. Home Assistant używa ich tylko w zapasowym alarmie, a codzienny sposób dawania znać to panel, lampa i aplikacja na telefonie dyżurnym. Wiadomości z treścią wrażliwą, na przykład hasłami, nie wysyłamy w ogóle przez usługi push.

Kto alarmuje: Zabbix czy Home Assistant

Temperaturę wlotu widzą jednocześnie dwa systemy, więc pierwsza zasada brzmi: jeden alarm ma jednego właściciela. W Zielonej tym właścicielem jest Zabbix na `mon01`, bo ma wyzwalacze z ważnością, eskalację do drugiego administratora i prezesa oraz historię potrzebną do przeglądów [16].



Rysunek 1. Kto alarmuje: czujniki z serwerowni idą do Zabbixa na mon01, który wysyła pocztę (ostrzeżenie) i SMS (krytyczny), i równolegle do Home Assistant na ha01, który zapala lampę przy drzwiach. Moduł wejść wej01 ma własny SMS bez mon01, a Home Assistant ma zapasowy SMS tylko wtedy, gdy mon01 milczy (przerywane czerwone linie). (kliknięcie otwiera pełny rozmiar)

Zdarzenie	Kto alarmuje	Co robi Home Assistant
Wlot powyżej 27 °C (ostrzeżenie)	Zabbix, poczta do zespołu IT	zapala lampę przy drzwiach i zostawia wpis w panelu
Wlot powyżej 32 °C (krytyczny)	Zabbix, SMS i telefon dyżurny	to samo, a przy milczącym `mon01` powiadomienie i SMS jako zapasowy alarm
Woda pod klimatyzacją	`wej01` własnym SMS i Zabbix	tylko pokazuje stan na panelu
Pożar	centrala pożarowa	nic: pożar nigdy nie zależy od Home Assistant
Awaria `ha01`	Zabbix, sprawdzenie ping i portu	nic, bo nie działa
Dom	Home Assistant	wszystko, bo nie ma innego systemu

Zapasowy alarm to zgodna z mapą serwerowni druga droga: gdy poczta i `mon01` stoją na klastrze, który właśnie padł od przegrzania, ktoś musi o tym powiedzieć [16]. Home Assistant jest w tej roli dobrym kandydatem, bo działa na osobnej maszynie `ha01` albo na mini PC i ma własny kontakt z telefonem. Warunek ping `mon01` w stanie `off` od 10 minut sprawia, że w zwykłych dniach zapasowy alarm milczy, więc nie dubluje Zabbixa.

Dane z Home Assistant do Zabbixa

Zabbix czyta czujniki bezpośrednio z płytek, więc dla `esp01` i `esp02` nie potrzebuje Home Assistant. Zostają trzy drogi dla danych, które istnieją tylko w Home Assistant.

Droga	Jak	Kiedy
A: bez pośrednika	Zabbix czyta płytkę przez HTTP, jak opisują strony DIY	czujniki ESPHome, nasz wybór
B: REST API Home Assistant	element HTTP agent z tokenem konta `zabbix`	encje, które są tylko w Home Assistant: czujnik pochodny, urządzenie Zigbee, integracja chmurowa
C: Home Assistant wysyła do Zabbixa	skrypt wysyłający wartości do trappera Zabbixa	nie opisujemy, bo wymaga dodatkowego skryptu i jego utrzymania

Droga B ma jedną wadę: Zabbix zależy wtedy od Home Assistant, więc awaria `ha01` wygląda jak awaria czujnika. Dlatego dla krytycznych wartości zostajemy przy drodze A, a drogą B czytamy tylko to, co jest wygodne i niekrytyczne.

Pilnowanie samego Home Assistant

Panel i automatyka, którym nikt nie patrzy na ręce, psują się po cichu, więc `ha01` jest hostem w Zabbixie z dwoma elementami: ping i port WWW. Wyzwalacz na brak odpowiedzi ma ważność ostrzeżenia, bo `ha01` nie jest drogą alarmu. Zabbix pilnuje też płytek: wyzwalacz `nodata` na każdym czujniku, jak w mapie serwerowni.

W domu

W domu nie mamy Zabbixa, więc kanał jest jeden: Home Assistant. Zalanie w pralni, otwarte okno przy włączonym ogrzewaniu albo lodówka, której drzwi stoją otwarte, powiadamiają telefon w aplikacji, a ważniejsze dodatkowo pocztą. Gdy czujnik wody wysyła alarm, a telefon jest wyłączony, wiadomość pocztowa i tak czeka w skrzynce.

Ta sama płytka ESP32 z innym plikiem ESPHome obsługuje więc dwa światy. W firmie jej odczyt czyta Zabbix i Home Assistant, w domu tylko Home Assistant. Pakiety `serwerownia.yaml` i `dom.yaml` trzymamy w osobnych instalacjach, a nie w jednej, żeby awaria domowego eksperymentu nie wyłączyła panelu serwerowni.

> □□ Więcej możliwości albo gotowe rozwiązanie od ręki (out of the box): gotowa lampa lub syrena sygnalizacyjna z wejściem na styk i zasilaniem z sieci PoE, albo sterownik GSM z wejściami alarmowymi i własnym modemem, który wysyła SMS bez żadnego serwera. W dużych centrach danych stosuje się systemy BMS i DCIM, które zbierają alarmy z klimatyzacji, UPS i czujników

> w jednym miejscu, mają zdefiniowaną eskalację i dwa niezależne tory powiadamiania.

Częste błędy

Objaw	Przyczyna	Co robimy
Ten sam alarm dostajemy dwa razy	Zabbix i Home Assistant mają to samo powiadomienie	zostawiamy właściciela alarmu (Zabbix), a Home Assistant sygnalizuje lokalnie i alarmuje tylko przy milczącym `mon01`
Automatyzacja nie reaguje na wysoką temperaturę, choć jest powyżej progu	wyzwalacz `numeric_state` działa przy przekroczeniu progu, a temperatura już była powyżej po restarcie	sprawdzamy w historii, kiedy wartość przekroczyła próg, i pamiętamy, że `for` po restarcie liczy od zera [6]
Czujnik pochodny pokazuje 0 albo nieprawdopodobną liczbę	`float` bez wartości domyślnej albo z zerem przy niedostępnej sondzie	używamy `availability` z `has_value`, żeby stan był `unavailable`, a nie zero [2]
Powiadomienie push nie przychodzi	telefon bez internetu, wyłączone powiadomienia aplikacji albo przekroczone 500 dziennie	sprawdzamy ustawienia aplikacji, a dla ważnych zdarzeń dodajemy poczty i SMS [14]
`check_config` zgłasza błąd na `automation:`	w pakiecie jest drugi klucz `automation:` w tym samym pliku	łączymy automatyzacje w jedną listę pod jednym kluczem [1]
Poczta z Home Assistant nie dochodzi	brak STARTTLS, zły port albo hasło skrzynki	sprawdzamy dziennik Home Assistant i dziennik MailCow na `docker01` [10]
Zabbix nie czyta encji z Home Assistant	zły token, konto bez uprawnień albo encja `unavailable`	testujemy ten sam adres poleceniem `curl` z `mon01` i sprawdzamy kod odpowiedzi [12]

□□ Ćwiczenia

1. Dlaczego automatyzacja „sygnalizator lokalny” nie wysyła poczty ani SMS, choć technicznie mogłaby?

Odpowiedź

Bo alarm w serwerowni ma jednego właściciela, Zabbixa. Gdyby Home Assistant też wysyłał pocztę, zespół dostawałby ten sam alarm dwa razy w różnym brzmieniu, a przy awarii trudno

byłoby ustalić, który system działa [16].

1. Do czego służy warunek `binary_sensor.mon01_ping` w stanie `off` przez 10 minut w zapasowym alarmie?

Odpowiedź

Zapasowy alarm ma działać tylko wtedy, gdy Zabbix nie może powiedzieć o przegrzaniu. Dziesięć minut ciszy odróżnia awarię od chwilowego restartu `mon01`, więc w zwykłych dniach alarm zapasowy milczy.

1. Sonda `esp01_wlot_r1_gora` przestała odpowiadać, a czujnik `serwerownia_wlot_max` pokazuje 0. Co poprawiamy w szablonie?

Odpowiedź

Dodajemy blok `availability` z warunkiem `has_value` dla wszystkich trzech sond. Wtedy czujnik ma stan `unavailable`, automatyzacje na `numeric_state` nie reagują na zero, a Zabbix i panel widzą, że dane są niedostępne [2].

1. Czemu w domowej automatyzacji o zalaniu nie ma Zabbixa ani zapasowego alarmu?

Odpowiedź

W domu nie ma drugiego systemu, który mógłby być właścicielem alarmu, więc Home Assistant jest jedynym. Dla ważnych zdarzeń dokładamy tylko drugi kanał, na przykład pocztę obok powiadomienia w aplikacji.

Powiązane: Instalacja Home Assistant: system na maszynie wirtualnej, mini PC albo w kontenerze · ESPHome i urządzenia: adopcja esp01 i esp02, API albo MQTT, nazwy encji · Serwerownia Zielonej: gdzie stoją czujniki i dokąd płyną alarmy · Monitorowanie typowych usług · MailCow: codzienna praca z domenami, skrzynkami i spamem

Źródła:

1. Home Assistant, pakiety (`packages`, reguły łączenia, `auth_providers`): home-assistant.io/docs/configuration/packages
2. Home Assistant, funkcje szablonów: `has_value`, `states`, `float`: home-assistant.io/template-functions

3. Home Assistant 2026.7, notatki wydania (wyzwalacze i warunki opisane celem, YAML bez zmian): home-assistant.io/blog/2026/07/01/release-20267
4. Home Assistant, integracja `rest_command`: home-assistant.io/integrations/rest_command
5. Home Assistant, integracja `notify`: home-assistant.io/integrations/notify
6. Home Assistant, wyzwalacz `numeric_state`: home-assistant.io/triggers/numeric_state
7. Home Assistant, wyzwalacze i identyfikatory wyzwalaczy: home-assistant.io/docs/automation/trigger
8. Home Assistant, integracja Ping (ICMP): home-assistant.io/integrations/ping
9. Home Assistant, grupy powiadomień (`platform: group`): home-assistant.io/integrations/group
10. Home Assistant, integracja SMTP (kreator w interfejsie, odbiorcy jako encje): home-assistant.io/integrations/smtp
11. Home Assistant Companion, podstawy powiadomień (`tag`, `group`, `interruption-level`): companion.home-assistant.io/docs/notifications/notifications-basic
12. Home Assistant, REST API (token, `/api/states/<entity_id>`): developers.home-assistant.io/docs/api/rest
13. Zabbix, element typu HTTP agent (nagłówki, makra, JSONPath): zabbix.com/documentation/current/en/manual/config/items/itemtypes/http
14. Home Assistant Companion, prywatność i limity powiadomień (500 dziennie na urządzenie): companion.home-assistant.io/docs/notifications/notification-details
15. Home Assistant Companion, push lokalny: companion.home-assistant.io/docs/notifications/notification-local
16. Serwerownia Zielonej: gdzie stoją czujniki i dokąd płyną alarmy, w tym rozdział „Gdy mon01 milczy” (w tej bazie): f0-02-serwerownia-zielonej-mapa-czujnikow.md
17. Home Assistant 2026.8, notatki wydania (zakładka Tools zamiast Developer tools): home-assistant.io/blog/2026/08/05/release-20268
18. Home Assistant, integracja SMS (strona usuniętej integracji): home-assistant.io/integrations/sms

---

Revision #14
Created 2026-10-03 12:08:34 UTC by Tomasz Konieczny
Updated 2026-10-26 14:20:16 UTC by Tomasz Konieczny