Mapa cieplna na ESP32: ΔT szafy i różnica ciśnień między korytarzami (DIY)

◫◫ Poziom: średnio

◫◫ Dotyczy: ESP32 (Olimex ESP32-POE), ESPHome 2026.6, DS18B20, Sensirion SDP810, Zabbix 7.0 LTS, Home Assistant, Raspberry Pi OS

▯ Czas: ~120 min

Budujemy moduł `esp03` dla szafy R2: sześć czujników temperatury (trzy na wlocie, trzy na wylocie), czujnik różnicy ciśnień między korytarzami i wysyłkę do Zabbixa na `mon01` oraz do Home Assistanta. Z tego liczymy ΔT szafy i widzimy bypass albo recyrkulację, zanim zrobi się gorąco. Wracamy tu przy montażu, rozbudowie na kolejne szafy i szukaniu przyczyny braku danych. Komendy ESPHome wykonujemy na stacji administratora z Pythonem 3.12 albo nowszym, a komendy wariantu z Raspberry Pi na samej malince.

◫◫ Skróty użyte na tej stronie (26)

Skrót	Co znaczy
1-Wire	Magistrala firmy Dallas, w której czujniki rozmawiają z mikrokontrolerem po jednym przewodzie danych, a każdy czujnik ma własny adres
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers: amerykańskie stowarzyszenie inżynierów, którego zalecenia opisują temperaturę i wilgotność w serwerowniach
DCIM	Data Center Infrastructure Management: dokumentacja tego, co fizycznie stoi w serwerowni, czyli szafy, urządzenia i okablowanie
DIY	Do It Yourself: zrób to sam, czyli własnoręcznie zbudowany czujnik zamiast gotowego urządzenia
DS18B20	Cyfrowy czujnik temperatury z magistralą 1-Wire, z fabrycznym 64-bitowym adresem, popularny w układach domowych i małych serwerowniach
ESP32	Tani mikrokontroler z Wi-Fi i Bluetooth firmy Espressif: malutki komputer, z którego budujemy własne czujniki sieciowe

Skrót	Co znaczy
ESPHome	Oprogramowanie, które zamienia ESP32 w gotowy czujnik sieciowy bez pisania kodu od zera, a odczyty udostępnia przez sieć
HTTP	Hypertext Transfer Protocol: protokół stron WWW
I2C	Inter-Integrated Circuit: prosta dwuprzewodowa magistrala, którą mały układ (np. czujnik) rozmawia z mikrokontrolerem
IT	Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy
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
LAN8720	układ scalony (PHY) obsługujący Ethernet 10/100 w płytkach ESP32 z kablem sieciowym, np. Olimex ESP32-POE; w ESPHome zapisujemy go jako type: LAN8720
LTS	Long Term Support: wersja z długim wsparciem poprawkami
OS	Operating System: system operacyjny
PDU	Power Distribution Unit: listwa zasilająca w szafie rack, w droższych wersjach z gniazdami mierzonymi i wejściem na czujnik temperatury
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
SDP810	Czujnik różnicy ciśnień firmy Sensirion z przyłączem na rurki, mierzy dziesiąte części paskala; u nas mierzy różnicę między zimnym i gorącym korytarzem
SMS	Short Message Service: krótka wiadomość tekstowa na telefon, która dociera także bez internetu
SNMP	Simple Network Management Protocol: protokół odczytu stanu urządzeń sieciowych
UEXT	Universal EXTension: małe złącze na płytce Olimex ESP32-POE, które wyprowadza magistralę I2C, UART i zasilanie do podłączania czujników
URL	Uniform Resource Locator: adres strony lub zasobu
USB	Universal Serial Bus: standard portów i urządzeń podłączanych do komputera, tu pendrive'y i inne nośniki wymienne
YAML	YAML Ain't Markup Language: format plików konfiguracyjnych oparty na wcięciach

Fiszka

□□ ZMIENIA SYSTEM

Instalacja ESPHome: pipx install esphome

```
pipx install esphome
```

ESPHome buduje program dla płytki ESP32 z jednego pliku YAML i wgrywa go przez USB albo przez sieć. Konfigurację z tej strony sprawdziliśmy poleceniem `esphome config` na wersji 2026.6.5, a w nowszych wersjach polecenie `esphome config` pokaże ewentualne zmiany składni.

ESPHome = Oprogramowanie, które zamienia ESP32 w gotowy czujnik sieciowy bez pisania kodu od zera, a odczyty udostępnia przez sieć · **ESP32** = Tani mikrokontroler z Wi-Fi i Bluetooth firmy Espressif: malutki komputer, z którego budujemy własne czujniki sieciowe · **YAML** = YAML Ain't Markup Language: format plików konfiguracyjnych oparty na wcięciach · **USB** = Universal Serial Bus: standard portów i urządzeń podłączanych do komputera, tu pendrive'y i inne nośniki wymienne · **pipx** = instaluje narzedzia napisane w Pythonie, kazde we wlasnym, odizolowanym srodowisku, bez mieszania zaleznosci miedzy programami

TYLKO ODCZYT

Wygenerowanie klucza API: openssl rand -base64 32

```
openssl rand -base64 32
```

Klucz szyfrujący natywne API to 32 losowe bajty zapisane w base64. Musi mieć dokładnie taką postać, bo ESPHome odrzuci dowolny inny tekst [1].

API = Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą · **ESPHome** = Oprogramowanie, które zamienia ESP32 w gotowy czujnik sieciowy bez pisania kodu od zera, a odczyty udostępnia przez sieć · **openssl** = zestaw narzędzi do certyfikatów i szyfrowania

TYLKO ODCZYT

Plik z hasłami: secrets.yaml

```
# secrets.yaml, obok pliku esp03-r2.yaml
api_klucz: "WynikPolecenia_openssl_rand_base64_32"
ota_haslo: "HasloOta"
web_uzytkownik: "nazwa_usera"
web_haslo: "HasloWebServera"
```

Hasła trzymamy w osobnym pliku i odwołujemy się do nich przez `!secret`, więc plik konfiguracyjny można pokazywać innym bez wycieku haseł. Wartości powyżej to placeholdery: klucz API musi pochodzić z poprzedniej fiszki.

API = Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą

PLIK KONFIGURACYJNY

Plik konfiguracyjny ESPHome: esp03-r2.yaml

```
esphome:
  name: esp03
  friendly_name: Mapa cieplna szafa R2

esp32:
  board: esp32-poe
  framework:
    type: esp-idf

logger:

# Ethernet z PoE: ustawienia dla Olimex ESP32-POE (wariant ISO ma te same piny)
ethernet:
  type: LAN8720
  mdc_pin: GPIO23
  mdio_pin: GPIO18
  clk:
    pin: GPIO17
    mode: CLK_OUT
  phy_addr: 0
  power_pin: GPIO12

# Bez klienta API (Home Assistant) płytka restartowałaby się co 15 minut,
# a Zabbix czyta dane przez web_server, więc wyłączamy ten restart.
api:
  encryption:
    key: !secret api_klucz
  reboot_timeout: 0s

ota:
  - platform: esphome
    password: !secret ota_haslo
```

```
web_server:
  port: 80
  auth:
    username: !secret web_uzytkownik
    password: !secret web_haslo

# Magistrala 1-Wire z sześcioma czujnikami DS18B20 (rezystor 4,7 kΩ do 3,3 V)
one_wire:
  - platform: gpio
    pin: GPIO32
    id: magistrala_r2

# Magistrala I2C dla SDP810: numery pinów sprawdzamy w arkuszu płytki
i2c:
  sda: GPIO13
  scl: GPIO16
  scan: true

sensor:
  # Adresy są przykładowe: podmieniamy je na adresy z dziennika (esphome logs).
  - platform: dallas_temp
    one_wire_id: magistrala_r2
    name: "r2_wlot_dol"
    id: r2_wlot_dol
    address: 0x8c0000046e9b4f28
    resolution: 12
    update_interval: 30s
  - platform: dallas_temp
    one_wire_id: magistrala_r2
    name: "r2_wlot_srodek"
    id: r2_wlot_srodek
    address: 0x3d0000046e9c1a28
    resolution: 12
    update_interval: 30s
  - platform: dallas_temp
    one_wire_id: magistrala_r2
    name: "r2_wlot_gora"
    id: r2_wlot_gora
    address: 0xe20000046e9d7b28
```

```
    resolution: 12
    update_interval: 30s
  - platform: dallas_temp
    one_wire_id: magistrala_r2
    name: "r2_wylot_dol"
    id: r2_wylot_dol
    address: 0x510000046e9e2c28
    resolution: 12
    update_interval: 30s
  - platform: dallas_temp
    one_wire_id: magistrala_r2
    name: "r2_wylot_srodek"
    id: r2_wylot_srodek
    address: 0x7a0000046ea02d28
    resolution: 12
    update_interval: 30s
  - platform: dallas_temp
    one_wire_id: magistrala_r2
    name: "r2_wylot_gora"
    id: r2_wylot_gora
    address: 0x1b0000046ea13e28
    resolution: 12
    update_interval: 30s

  # Różnica ciśnień między korytarzami. ESPHome podaje hPa, mnożymy przez 100, żeby mieć
paskale.
  - platform: sdp3x
    name: "dp_korytarze"
    id: dp_korytarze
    address: 0x25
    update_interval: 10s
    filters:
      - multiply: 100
    unit_of_measurement: "Pa"
    accuracy_decimals: 1

  # ΔT szafy: średnia z wylotów minus średnia z wlotów
  - platform: template
    name: "r2_delta_t"
```

```
    id: r2_delta_t
    unit_of_measurement: "K"
    accuracy_decimals: 1
    update_interval: 30s
    lambda: |-
      float wlot = (id(r2_wlot_dol).state + id(r2_wlot_srodek).state +
id(r2_wlot_gora).state) / 3.0;
      float wylot = (id(r2_wylot_dol).state + id(r2_wylot_srodek).state +
id(r2_wylot_gora).state) / 3.0;
      if (std::isnan(wlot) || std::isnan(wylot)) return NAN;
      return wylot - wlot;

  # Najwyższy wlot z trzech czujników: jedna liczba do alarmu i do wskaźnika w Home
Assistant
  - platform: template
    name: "r2_wlot_max"
    id: r2_wlot_max
    unit_of_measurement: "°C"
    accuracy_decimals: 1
    update_interval: 30s
    lambda: |-
      float a = id(r2_wlot_dol).state;
      float b = id(r2_wlot_srodek).state;
      float c = id(r2_wlot_gora).state;
      if (std::isnan(a) || std::isnan(b) || std::isnan(c)) return NAN;
      return std::max(a, std::max(b, c));
```

Plik opisuje płytkę Olimex ESP32-POE: Ethernet z zasilaniem PoE (ustawienia układu LAN8720 z dokumentacji ESPHome [2][3]), szyfrowane API, aktualizacje przez sieć, prosty serwer HTTP , magistralę 1-Wire na GPIO32 z sześcioma czujnikami DS18B20 [4], magistralę I2C z czujnikiem SDP810 [5] oraz trzy wyliczane wartości: ΔT szafy, najwyższy wlot i różnicę ciśnień w paskalach. Czujnik SDP810 obsługuje komponent `sdp3x`, który podaje ciśnienie w hPa, więc filtr `multiply: 100` zamienia je na Pa [5]. Domyślny adres komponentu to 0x21, a SDP800 ma 0x25, dlatego jest wpisany jawnie. Numery pinów I2C (GPIO13 i GPIO16) przyjmujemy z gniazda UEXT płytki i sprawdzamy w arkuszu pinów konkretnej wersji płytki, bo nie potwierdziliśmy ich w dokumentacji. Nazwy czujników celowo nie mają spacji ani polskich liter, bo wchodzą do adresu URL.

Sekcja `api` ma `reboot_timeout: 0s`, bo bez klienta API płytka restartowałaby się co 15 minut, a u nas dane pobiera Zabbix przez serwer HTTP. W Home Assistancie dodajemy płytkę jako urządzenie ESPHome i wpisujemy ten sam klucz.

ESPHome = Oprogramowanie, które zamienia ESP32 w gotowy czujnik sieciowy bez pisania kodu od zera, a odczyty udostępnia przez sieć · **PoE** = Power over Ethernet: zasilanie urządzenia tym samym kablem sieciowym, którym płyną dane · **LAN8720** = układ scalony (PHY) obsługujący Ethernet 10/100 w płytkach ESP32 z kablem sieciowym, np. Olimex ESP32-POE; w ESPHome zapisujemy go jako type: LAN8720 · **API** = Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą · **HTTP** = Hypertext Transfer Protocol: protokół stron WWW · **1-Wire** = Magistrala firmy Dallas, w której czujniki rozmawiają z mikrokontrolerem po jednym przewodzie danych, a każdy czujnik ma własny adres · **DS18B20** = Cyfrowy czujnik temperatury z magistralą 1-Wire, z fabrycznym 64-bitowym adresem, popularny w układach domowych i małych serwerowniach · **I2C** = Inter-Integrated Circuit: prosta dwuprzewodowa magistrala, którą mały układ (np. czujnik) rozmawia z mikrokontrolerem

TYLKO ODCZYT

Sprawdzenie pliku: esphome config esp03-r2.yaml

```
esphome config esp03-r2.yaml
```

Polecenie wczytuje plik wraz z `secrets.yaml`, sprawdza składnię i niczego nie wgrywa. Kończy się komunikatem `Configuration is valid!`.

esphome = ESPHome: buduje i wgrywa oprogramowanie na ESP32 z pliku YAML, pokazuje logi urządzenia

TYLKO ODCZYT

Wgranie programu przez USB: esphome run esp03-r2.yaml

```
esphome run esp03-r2.yaml
```

Przy pierwszym wgraniu płytkę podłączamy kablem USB do stacji administratora. Kolejne aktualizacje idą przez sieć, według sekcji `ota`.

USB = Universal Serial Bus: standard portów i urządzeń podłączanych do komputera, tu pendrive'y i inne nośniki wymienne · **esphome** = ESPHome: buduje i wgrywa oprogramowanie na ESP32 z pliku YAML, pokazuje logi urządzenia

TYLKO ODCZYT

Odczyt adresów czujników: esphome logs esp03-r2.yaml

```
esphome logs esp03-r2.yaml
```

Po starcie ESPHome skanuje magistralę 1-Wire i I2C i wypisuje adresy znalezionych urządzeń [4]. Adresy czujników DS18B20 wpisujemy do pliku zamiast przykładowych.

Przykład (skrócony):

```
Found devices:
  0x8c0000046e9b4f28 (DS18B20)
  0x3d0000046e9c1a28 (DS18B20)
  0xe20000046e9d7b28 (DS18B20)
  0x510000046e9e2c28 (DS18B20)
  0x7a0000046ea02d28 (DS18B20)
  0x1b0000046ea13e28 (DS18B20)
Found i2c device at address 0x25
```

ESPHome = Oprogramowanie, które zamienia ESP32 w gotowy czujnik sieciowy bez pisania kodu od zera, a odczyty udostępnia przez sieć · **1-Wire** = Magistrala firmy Dallas, w której czujniki rozmawiają z mikrokontrolerem po jednym przewodzie danych, a każdy czujnik ma własny adres · **I2C** = Inter-Integrated Circuit: prosta dwuprzewodowa magistrala, którą mały układ (np. czujnik) rozmawia z mikrokontrolerem · **DS18B20** = Cyfrowy czujnik temperatury z magistralą 1-Wire, z fabrycznym 64-bitowym adresem, popularny w układach domowych i małych serwerowniach · **esphome** = ESPHome: buduje i wgrywa oprogramowanie na ESP32 z pliku YAML, pokazuje logi urządzenia

TYLKO ODCZYT

Odczyt przez REST: curl

```
curl --user nazwa_usera http://esp03/sensor/r2_delta_t
```

Przy takim zapisie `curl` pyta o hasło, więc nie zostaje ono w historii poleceń. Odpowiedź to obiekt JSON z polem `value`, które Zabbix wyciąga w następnym kroku [6].

Przykład:

```
{"id":"sensor/r2_delta_t","state":"12.1 K","value":12.1}
```

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

TYLKO ODCZYT

Test znaku różnicy ciśnień: curl

```
curl --user nazwa_usera http://esp03/sensor/dp_korytarze
```

Dmuchamy delikatnie w rurkę podłączoną do portu, który ma mierzyć zimny korytarz, i powtarzamy polecenie. Wartość ma być dodatnia. Jeśli wychodzi ujemna, zamieniamy rurki albo dopisujemy w filtrach `multiply: -100`.

curl = client URL: pobieranie i wysyłanie przez HTTP z terminala

ZMIENIA SYSTEM

Wariant z Raspberry Pi: dt_do_zabbixa.sh

```
#!/usr/bin/env bash
# Odczyt sześciu DS18B20 z 1-Wire, wyliczenie ΔT szafy i wysyłka do Zabbixa.
set -u

SERWER_ZABBIXA="mon01"
NAZWA_HOSTA="czujnik-rpi02"
KATALOG_W1="${KATALOG_W1:-/sys/bus/w1/devices}"

# numer czujnika z katalogu 28-... : klucz elementu typu trapper
declare -A CZUJNIKI=(
  ["28-3c01d6075a21"]="temp.r2.wlot.dol"
  ["28-3c01d6075b32"]="temp.r2.wlot.srodek"
  ["28-3c01d6075c43"]="temp.r2.wlot.gora"
  ["28-3c01d6075d54"]="temp.r2.wylot.dol"
  ["28-3c01d6075e65"]="temp.r2.wylot.srodek"
  ["28-3c01d6075f76"]="temp.r2.wylot.gora"
)

odczytaj() {
  # wypisuje temperaturę w °C albo nic, gdy odczyt jest niepewny
  local plik="$KATALOG_W1/$1/w1_slave" l1 l2 mc
  [[ -r "$plik" ]] || return 1
  { read -r l1; read -r l2; } < "$plik"
  [[ "$l1" == *YES ]] || return 1
  mc="${l2##*t=}"
  [[ "$mc" =~ ^-?[0-9]+$ ]] || return 1
```

```
  [[ "$mc" -ne 85000 ]] || return 1   # 85,0 °C to wartość startowa układu
  awk -v m="$mc" 'BEGIN { printf "%.3f", m / 1000 }'
}

wejscie=""
sum_wlot=0; n_wlot=0; sum_wylot=0; n_wylot=0
for numer in "${!CZUJNIKI[@]}"; do
  klucz="${CZUJNIKI[$numer]}"
  if temp=$(odczytaj "$numer"); then
    wejscie+="$NAZWA_HOSTA $klucz $temp"$'\n'
    if [[ "$klucz" == *.wlot.* ]]; then
      sum_wlot=$(awk -v a="$sum_wlot" -v b="$temp" 'BEGIN { print a + b }');
n_wlot=$((n_wlot + 1))
    else
      sum_wylot=$(awk -v a="$sum_wylot" -v b="$temp" 'BEGIN { print a + b }');
n_wylot=$((n_wylot + 1))
    fi
  else
    echo "Brak poprawnego odczytu: $numer" >&2
  fi
done

if [[ "$n_wlot" -gt 0 && "$n_wylot" -gt 0 ]]; then
  dt=$(awk -v w="$sum_wylot" -v nw="$n_wylot" -v i="$sum_wlot" -v ni="$n_wlot" 'BEGIN {
printf "%.2f", w / nw - i / ni }')
  wejscie+="$NAZWA_HOSTA temp.r2.delta_t $dt"$'\n'
fi

if [[ -z "$wejscie" ]]; then
  echo "Żaden czujnik nie dał poprawnego odczytu" >&2
  exit 1
fi

printf '%s' "$wejscie" | zabbix_sender -z "$SERWER_ZABBIXA" -i -
```

Skrypt czyta sześć czujników z magistrali 1-Wire, odrzuca odczyty z błędną sumą kontrolną i wartość startową 85 °C, liczy ΔT szafy i wysyła wszystko jednym wywołaniem `zabbix_sender` [7]. Numery z katalogów `28-...` podmieniamy na numery z polecenia `ls /sys/bus/w1/devices/`, a włączenie magistrali 1-Wire opisują fiszki na stronie Termometry w

praktyce: wybór, montaż i własny czujnik (DIY). Składnię sprawdzamy poleceniem `bash -n`.

1-Wire = Magistrala firmy Dallas, w której czujniki rozmawiają z mikrokontrolerem po jednym przewodzie danych, a każdy czujnik ma własny adres · **DIY** = Do It Yourself: zrób to sam, czyli własnoręcznie zbudowany czujnik zamiast gotowego urządzenia · **declare** = zadeklaruj zmienną powłoki z określonym typem, na przykład jako liczbę całkowitą albo tablicę · **awk** = od nazwisk autorów Aho, Weinberger, Kernighan: język do danych w kolumnach · **echo** = od angielskiego echo: wypisuje podany tekst na standardowe wyjście, często łączone z przekierowaniem do pliku · **exit** = zakończ bieżącą powłokę albo skrypt z podanym kodem wyjścia · **printf** = wypisz sformatowany tekst, ściślejsza odmiana echo

ZMIENIA SYSTEM

Instalacja skryptu: sudo install

```
sudo install -m 0755 dt_do_zabbixa.sh /usr/local/bin/dt_do_zabbixa.sh
```

install = install: kopiuje pliki i tworzy katalogi z jednoczesnym ustawieniem uprawnień

ZMIENIA SYSTEM

Zadanie cykliczne: /etc/cron.d/dt-zabbix

```
* * * * * nazwa_usera /usr/local/bin/dt_do_zabbixa.sh 2>&1 | logger -t dt-zabbix
```

Wpis w katalogu `/etc/cron.d/` uruchamia skrypt co minutę, a wynik trafia do dziennika systemowego pod znacznikiem `dt-zabbix`.

TYLKO ODCZYT

Element wyliczany w Zabbixie: ΔT szafy

```
(last(/esp03/temp.r2.wylot.dol)+last(/esp03/temp.r2.wylot.srodek)+last(/esp03/temp.r2.wylot.gora))/3
-
(last(/esp03/temp.r2.wlot.dol)+last(/esp03/temp.r2.wlot.srodek)+last(/esp03/temp.r2.wlot.gora))/3
```

Element typu Calculated liczy ΔT w samym Zabbixie, z odczytów pojedynczych czujników [8]. Używamy go, jeśli płytka nie liczy ΔT sama albo gdy chcemy porównać wynik z liczonym na płytce. Formułę wpisujemy w jednej linii.

PLIK KONFIGURACYJNY

Wskaźnik w Home Assistant: karta gauge

```
type: gauge
entity: sensor.mapa_cieplna_szafa_r2_r2_wlot_max
name: Wlot R2, najwyższy
unit: °C
min: 10
max: 40
severity:
  green: 10
  yellow: 27
  red: 32
```

Karta pokazuje najwyższy wlot szafy R2 z progami zielonym, żółtym i czerwonym [9]. Identyfikator encji sprawdzamy w Home Assistancie w ustawieniach urządzenia, bo wynika z nazw w pliku ESPHome.

ESPHome = Oprogramowanie, które zamienia ESP32 w gotowy czujnik sieciowy bez pisania kodu od zera, a odczyty udostępnia przez sieć

Z życia wzięte

Z ŻYCIA WZIĘTE

Piątek, 16:20. Wlot u góry szafy R5 pokazuje 29 °C, choć obie klimatyzacje pracują, a Zabbix nie ma żadnego alarmu, bo próg ustawiono na 32 °C. Tomek chce zobaczyć, co się dzieje przed alarmem. Tydzień później wisi na R2 moduł `esp03` i na wykresie widać, że ΔT szafy w nocy spada z 12 do 7 K, a różnica ciśnień między korytarzami jest ujemna przez cały dzień. Okazuje się, że drzwi gorącego korytarza nie domykają się po ostatnich pracach. Poprawka zamka kosztuje mniej niż dobry obiad.

Projekt DIY krok po kroku

Moduł mierzy w jednej szafie sześć temperatur i różnicę ciśnień. Dlaczego akurat tak i co oznaczają wyniki, wyjaśnia strona Przepływ powietrza w serwerowni: szafa, korytarze, bypass i recyrkulacja.

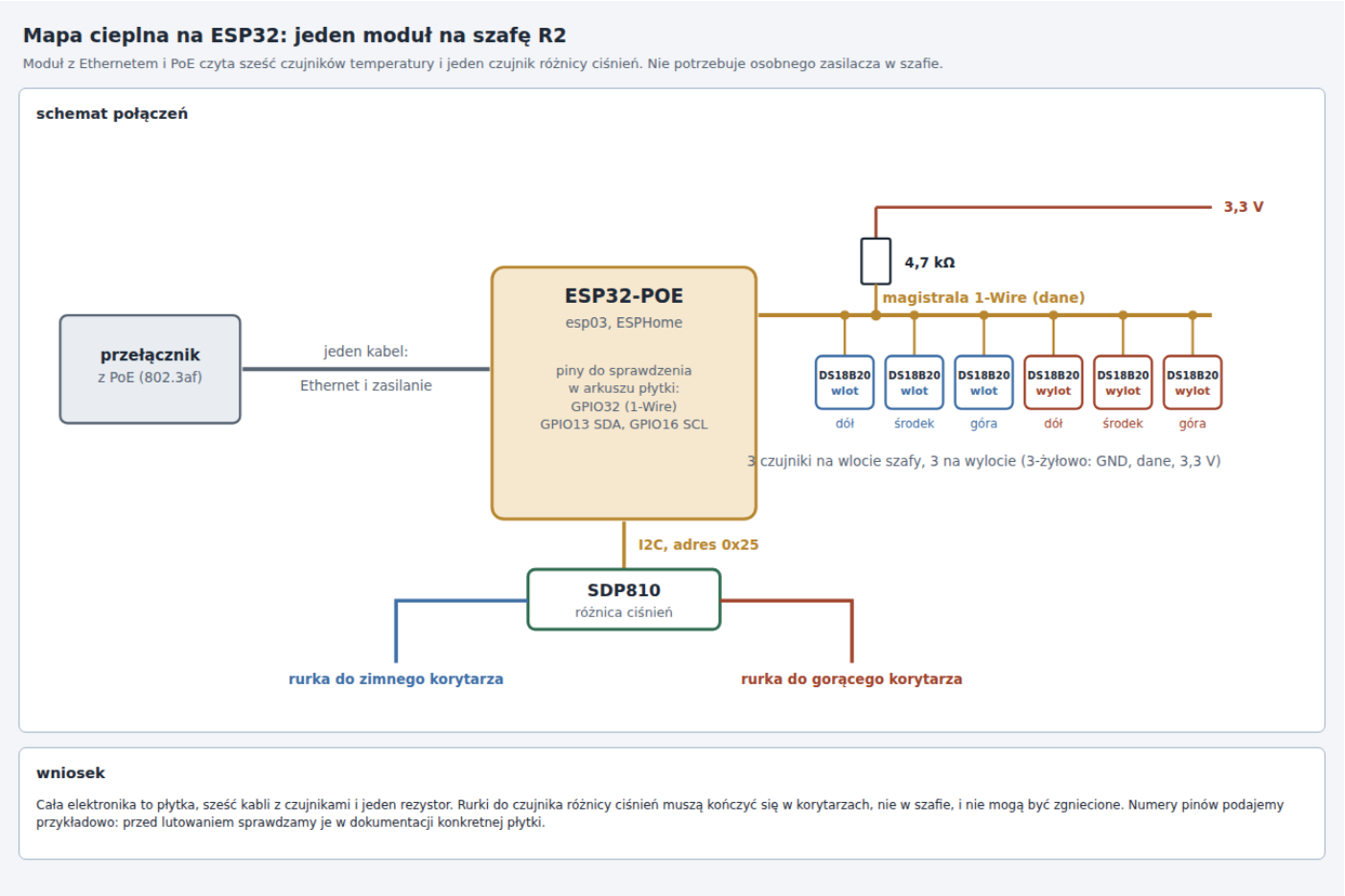
Lista części

Część	Po co	Uwagi
Olimex ESP32-POE (albo ESP32-POE-ISO)	procesor z Ethernetem i zasilaniem z przełącznika	wariant ISO ma izolację galwaniczną
Sześć sond DS18B20 w stalowej gilzie na kablu	temperatury wlotu i wylotu	wodoodporne sondy mają trzy przewody, kolory różnią się między producentami
Sensirion SDP810 w wersji 125 Pa albo 500 Pa, najlepiej na module lub w zestawie ewaluacyjnym	różnica ciśnień między korytarzami	wersja 125 Pa lepiej rozróżnia małe różnice, a 500 Pa ma szerszy zakres
Dwie rurki silikonowe do końcówek czujnika	doprowadzenie ciśnienia z korytarzy	krótkie, tej samej długości, bez załamań
Rezystor 4,7 kΩ	podciągnięcie linii danych 1-Wire do 3,3 V	jeden na całą magistralę
Kabel trzyżyłowy lub skrętka	przedłużenie sond	trzy żyły: zasilanie, masa, dane
Listwa zaciskowa lub lutowane połączenia z koszulkami	równoległe połączenie sond	pewne styki
Obudowa na płytkę	ochrona przed kurzem i zwarciem	otwory na kable i rurki
Port PoE w przełączniku (802.3af)	zasilanie i sieć jednym kablem	ten sam typ co przy modułach `esp01` i `esp02`
Opaski, uchwyty i taśma dwustronna	mocowanie sond i rurek	z tworzywa, żeby sonda nie dotykała metalowej szafy
Termometr wzorcowy	sprawdzenie sond	albo wzorcowanie w laboratorium

Płytka rozwojowa ESP32, poglądowa. Używamy Olimex ESP32-POE, który ma dodatkowo złącze Ethe

Płytka rozwojowa ESP32, poglądowa. Używamy Olimex ESP32-POE, który ma dodatkowo złącze Ethernet z zasilaniem PoE. Źródło: Wikimedia Commons (autor i wolna licencja podane na stronie pliku).

Schemat połączeń



Rysunek 1. Schemat połączeń: płytka ESP32-POE z kablem Ethernet i PoE z przełącznika, magistrala 1-Wire z sześcioma czujnikami DS18B20 i rezystorem 4,7 kΩ oraz czujnik SDP810 na I2C z dwiema rurkami do korytarzy. Numery pinów podajemy przykładowo. (kliknięcie otwiera pełny rozmiar)

Wszystkie sześć sond łączymy równolegle, bo każda ma własny 64-bitowy adres [10]. Zasilamy je z pinu 3V3 płytki, a rezystor 4,7 kΩ łączymy między 3V3 i linią danych [4]. Odwrotnie podłączone zasilanie nagrzewa i niszczy układ, więc kolory przewodów sprawdzamy w opisie sondy. Czujnik SDP810 pracuje przy 3,3 V i łączy się z płytką dwoma przewodami I2C, a dwie rurki z jego końcówek prowadzimy do korytarzy [11]. Zanim zaczniemy lutować, sprawdzamy w dokumentacji modułu, czy ma rezystory podciągające I2C.

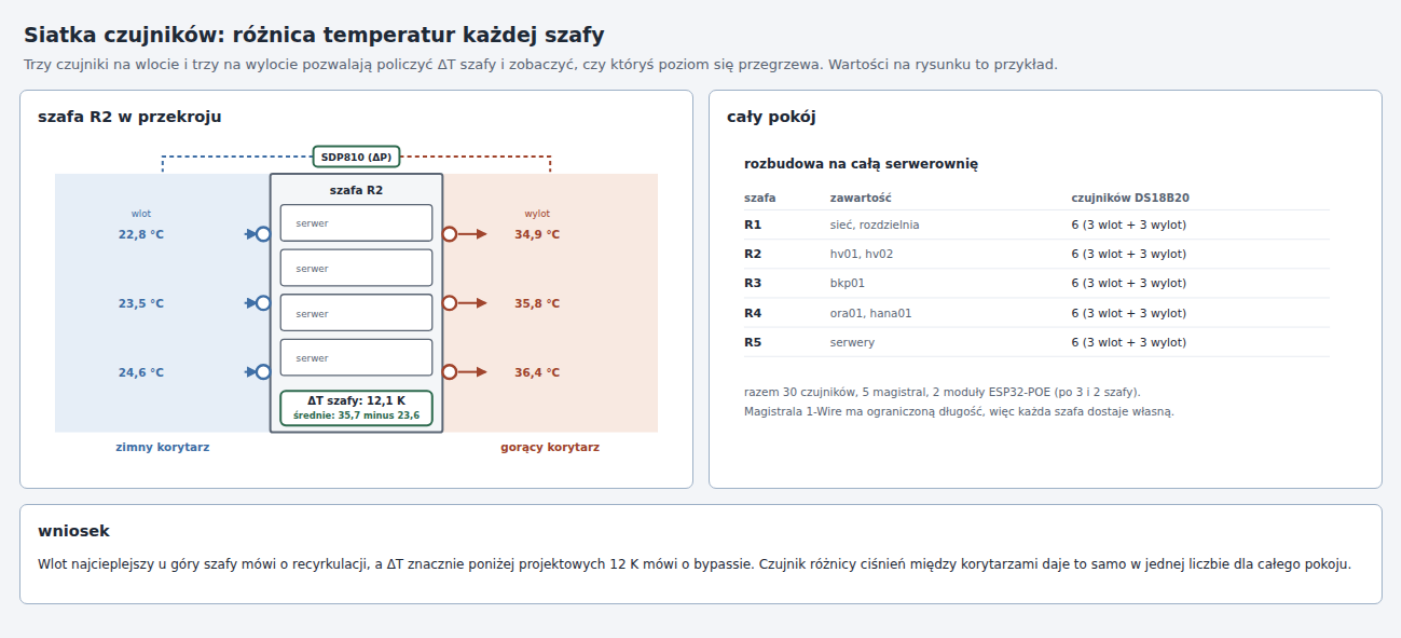
Sonda DS18B20 w stalowej gilzie. Sześć takich sond mocujemy na szafie: trzy na wlocie i trzy na wy

Sonda DS18B20 w stalowej gilzie. Sześć takich sond mocujemy na szafie: trzy na wlocie i trzy na wylocie, po jednej na górze, w środku i na dole. Źródło: Wikimedia Commons (autor i wolna licencja podane na stronie pliku).

Pierwsze uruchomienie i adresy czujników

Wgrywamy plik z sekcjami `ethernet`, `api`, `ota`, `one_wire` i `i2c`, bez sekcji `sensor`. Po starcie dziennik (fiszka Odczyt adresów czujników) pokazuje sześć adresów 1-Wire i adres I2C 0x25. Żeby poznać, który adres należy do której sondy, ogrzewamy ją w dłoni i obserwujemy, która wartość rośnie. Potem uzupełniamy sekcję `sensor` właściwymi adresami i nazwami. Adresów nie zastępujemy numerami kolejnymi, bo kolejność zmienia się po wymianie sondy.

Montaż w szafie i korytarzach



Rysunek 2. Siatka czujników na szafie R2: trzy sondy na wlocie i trzy na wylocie, czujnik różnicy ciśnień nad szafą z rurkami do obu korytarzy oraz przykład obliczenia ΔT szafy. Po prawej plan rozbudowy na całą serwerownię. (kliknięcie otwiera pełny rozmiar)

Trzy sondy wlotu mocujemy na froncie szafy, na wysokości najwyższego, środkowego i najniższego serwera, około 5 cm przed obudową [12]. Trzy sondy wylotu mocujemy symetrycznie z tyłu, w gorącym korytarzu. Kable prowadzimy wzdłuż prowadnic, a nie przez wlot powietrza.

Rurkę czujnika ΔP kierujemy końcówką do zimnego korytarza, drugą do gorącego, w miejscu, które nie jest w strumieniu z wentylatorów ani przy drzwiach. Obie rurki mają tę samą długość i nie mogą być zgniecione.

ZRZUT EKRANU DO UZUPEŁNIENIA

Zdjęcie do zrobienia: szafa R2 z tyłu z trzema sondami wylotu i końcówką rurki ΔP w gorącym korytarzu, każda sonda opisana na kablu.

Płytkę montujemy z boku szafy od strony zimnego korytarza, poza strumieniem gorącego powietrza. Sondy opisujemy na obu końcach kabla, a adresy zapisujemy w tabeli w NetBoxie przy szafie.

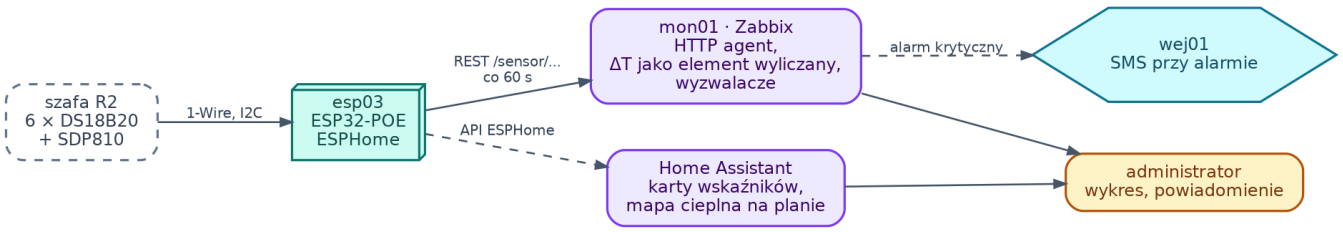
Kalibracja

> **Pojęcie: offset.** Stała poprawka dopisana do odczytu czujnika, która koryguje stały błąd. W ESPHome wpisujemy ją w sekcji `filters` jako `offset`.

Kąpiel lodowa dla punktu 0 °C, wykonanie jej i dobór drugiego punktu opisuje strona o termometrach DIY. Tu dodajemy sprawdzenie, które ma znaczenie dla ΔT: wszystkie sześć sond kładziemy obok siebie na kilkanaście minut w tym samym miejscu i porównujemy odczyty. Różnica między dowolnymi dwiema sondami nie powinna przekraczać 0,5 °C. Jeśli przekracza, korygujemy offsetem, bo błąd rzędu pół stopnia przy ΔT 12 K to 4 procent.

Czujnik SDP810 przed montażem zerujemy: obie końcówki bez rurek wystawiamy na to samo powietrze i odczytujemy wartość. Powinna być bliska zera (dokumentacja podaje błąd zera rzędu dziesiątych części paskala), a stały odczyt kilku paskali korygujemy offsetem [11].

Integracja z monitoringiem



REST oraz Home Assistant przez API. Alarmy krytyczne idą SMS-em przez `wej01`.">

Rysunek 3. Droga danych: sześć czujników i SDP810 na szafie R2, moduł `esp03` z ESPHome, Zabbix na `mon01` odpytujący przez REST oraz Home Assistant przez API. Alarmy krytyczne idą SMS-em przez `wej01`. (kliknięcie otwiera pełny rozmiar)

Zabbix co minutę pyta płytkę o wartości przez serwer HTTP ESPHome, a odpowiedź JSON ma pole `value`, które element wyciąga ścieżką JSONPath [6]. Dzięki temu na płytce nie trzeba dodatków, a gdy milknie, Zabbix to widzi. Home Assistant dostaje te same dane przez natywne API i służy do wskaźników i planu.

Host i makra w Zabbixie

Ścieżka w interfejsie, w której tworzymy host:

```
Data collection → Hosts → Create host
```

Host nazywamy `esp03`, dodajemy go do grupy `Czujniki środowiska` i nie dodajemy interfejsu agenta. Na hoście ustawiamy makra:

Makro	Wartość	Po co
`{$ESP.ADRES}`	adres płytki w VLAN serwerowni	adres płytki
`{$ESP.UZYTKOWNIK}`	nazwa użytkownika z sekcji `web_server`	uwierzytelnianie
`{$ESP.HASLO}`	hasło z sekcji `web_server`, typ Secret text	niewidoczne w interfejsie
`{$TEMP.WLOT.OSTRZEZENIE}`	`27`	górna granica zalecana wlotu (°C)
`{$TEMP.WLOT.KRYTYCZNY}`	`32`	górna granica dopuszczalna klasy A1 (°C)
`{$DT.MIN}`	`6`	ΔT szafy poniżej: podejrzenie bypassu (K), do kalibracji
`{$DT.MAX}`	`20`	ΔT szafy powyżej: za mało powietrza (K), do kalibracji
`{$DP.MIN}`	`0`	różnica ciśnień poniżej: recyrkulacja (Pa), do kalibracji
`{$DP.MAX}`	`15`	różnica ciśnień powyżej: wyciek przez serwery (Pa), do kalibracji

Progi temperatury wynikają z ASHRAE: 27 °C to górna granica zalecanego zakresu 18 do 27 °C, a 32 °C to górna granica zakresu dopuszczalnego klasy A1 [12][13]. Progi ΔT i ΔP są naszymi propozycjami roboczymi, które dostrajamy po kilku tygodniach obserwacji.

Elementy danych

Wszystkie elementy mają typ HTTP agent, uwierzytelnianie podstawowe z makrami, przetwarzanie wstępne JSONPath `$.value`, typ informacji liczba zmiennoprzecinkowa i interwał 1m [14].

Nazwa	Klucz	URL
Wlot, dół	`temp.r2.wlot.dol`	`http://{$ESP.ADRES}/sensor/r2_wlot_dol`
Wlot, środek	`temp.r2.wlot.srodek`	`http://{$ESP.ADRES}/sensor/r2_wlot_srodek`

Nazwa	Klucz	URL
Wlot, góra	`temp.r2.wlot.gora`	`http://{$ESP.ADRES}/sensor/r2_wlot_gora`
Wylot, dół	`temp.r2.wylot.dol`	`http://{$ESP.ADRES}/sensor/r2_wylot_dol`
Wylot, środek	`temp.r2.wylot.srodek`	`http://{$ESP.ADRES}/sensor/r2_wylot_srodek`
Wylot, góra	`temp.r2.wylot.gora`	`http://{$ESP.ADRES}/sensor/r2_wylot_gora`
Wlot, najwyższy	`temp.r2.wlot.max`	`http://{$ESP.ADRES}/sensor/r2_wlot_max`
ΔT szafy (K)	`temp.r2.delta_t`	`http://{$ESP.ADRES}/sensor/r2_delta_t`
Różnica ciśnień (Pa)	`dp.korytarze`	`http://{$ESP.ADRES}/sensor/dp_korytarze`

Wyzwalacze

Wyrażenia zapisujemy w składni Zabbixa 7.0, w której nazwę hosta i klucz podajemy w ukośnikach [15]. Ostrzeżenie o wysokim wlocie (Warning):

```
min(/esp03/temp.r2.wlot.max,5m)>{$TEMP.WLOT.OSTRZEZENIE}
```

Wyrażenie powrotu, z histerezą jednego stopnia:

```
max(/esp03/temp.r2.wlot.max,5m)<{$TEMP.WLOT.OSTRZEZENIE}-1
```

Alarm krytyczny (Disaster):

```
min(/esp03/temp.r2.wlot.max,3m)>{$TEMP.WLOT.KRYTYCZNY}
```

ΔT szafy za niskie, podejrzenie bypassu (Information), i za wysokie, za mało powietrza (Warning):

```
avg(/esp03/temp.r2.delta_t,30m)<{$DT.MIN}
avg(/esp03/temp.r2.delta_t,30m)>{$DT.MAX}
```

Różnica ciśnień ujemna, recyrkulacja (Warning), i zbyt wysoka, wyciek przez serwery (Information):

```
avg(/esp03/dp.korytarze,5m)<{$DP.MIN}
avg(/esp03/dp.korytarze,5m)>{$DP.MAX}
```

Brak danych z któregokolwiek czujnika (High):

```
nodata(/esp03/temp.r2.wlot.max,5m)=1
or nodata(/esp03/dp.korytarze,5m)=1
```

Wyzwalacz ΔT nie jest alarmem awarii, tylko wskazówką do działania: poniżej progu patrzymy na uszczelnienie szafy i korytarza, powyżej na nawiew. Przy małym obciążeniu ΔT naturalnie się zmienia, więc progi ustawiamy po zebraniu historii.

Mapa cieplna w Zabbixie i w Home Assistant

W Zabbixie 7.0 mamy widżet Honeycomb, który pokazuje wartości wielu elementów jako kolorowe komórki z progami [16]. Dla R2 tworzymy komórki z sześciu czujników, a dla całej serwerowni, po rozbudowie, jedną komórkę na szafę z elementu `wlot.max`. Ścieżka w interfejsie: dodajemy widżet Honeycomb na pulpicie i wybieramy elementy z hosta `esp03`.

W Home Assistancie używamy karty gauge z fiszki dla najwyższego wlotu oraz zwykłej karty z wartością ΔT i ΔP. Plan pokoju z kolorowymi punktami na rzucie wykonujemy kartą picture-elements, ale to dodatek i nie opisujemy go szczegółowo.

Akcja: poczta i SMS

Ścieżka w interfejsie:

```
Alerts → Actions → Trigger actions → Create action
```

Dla poziomów od Warning w górę wysyłamy pocztę do grupy IT. Dla Disaster dodajemy SMS przez `wej01` albo bramkę SMS. Test progów robimy bez grzania szafy: tymczasowo obniżamy makra `{$TEMP.WLOT.OSTRZEZENIE}` i `{$TEMP.WLOT.KRYTYCZNY}` o kilka stopni poniżej bieżącej temperatury, sprawdzamy, że alarmy dochodzą, i przywracamy 27 i 32.

Rozbudowa na całą serwerownię

ESP32 obsłuży kilka magistral 1-Wire, ale każda szafa dostaje własną, bo magistrala ma ograniczoną długość i łatwiej szukać usterki.

Szafa	Moduł	Magistrala 1-Wire	Czujniki
R1, R2, R3	`esp03`	trzy, po jednej na szafę	18 DS18B20 i SDP810
R4, R5	`esp04`	dwie, po jednej na szafę	12 DS18B20

Szafa	Moduł	Magistrala 1-Wire	Czujniki
R6	po zajęciu szafy	jedna	6 DS18B20

Razem 30 czujników w pięciu szafach, pięć magistrali i dwa moduły ESP32-POE. Numery pinów dla kolejnych magistral wybieramy z wolnych pinów płytki, według jej arkusza. Czujnik różnicy ciśnień wystarczy jeden na pokój, a drugi daje redundancję.

Bezpieczeństwo i ograniczenia

Własny czujnik nie ma świadectwa wzorcowania ani deklaracji zgodności, a jego dokładność wynika z karty katalogowej i naszej kalibracji. Traktujemy go jako uzupełnienie monitoringu, nie jego zastąpienie.

- Gwarancja i ubezpieczenie: urządzenie własnej roboty w szafie może być sprzeczne z warunkami gwarancji sprzętu albo polisy. Uzgadniamy je z zarządcą serwerowni i ubezpieczycielem, a prace wykonujemy tylko na niskim napięciu i z gotowym zasilaniem PoE.
- Czujnik różnicy ciśnień ma dokładność rzędu 3 procent odczytu i zero rzędu dziesiątych części paskala [11]. Wyniki kilku paskali traktujemy jako trend, nie pomiar bezwzględny.
- Sonda przy wylocie zasilacza serwera pokazuje wylot, nie wlot. Sondy wlotu montujemy tylko na froncie.
- Serwer HTTP ESPHome domyślnie używa uwierzytelniania podstawowego [6]. Na `fw01` ograniczamy dostęp do portu 80 płytki do adresu `mon01` i nie wystawiamy go poza serwery.
- PoE: port przełącznika ma budżet mocy. Płytka z kilkoma czujnikami pobiera niewiele, ale sprawdzamy ten budżet przy podłączeniu.
- Aktualizacje: płytkę aktualizujemy przez sieć według sekcji `ota`, poza oknem zmian w listopadzie i styczniu.
- Przy braku danych system nie ma pewności, czy awaria jest w czujniku, czy w płytce. Dlatego wyzwalacz braku danych jest osobny i ma wysoki priorytet.

> Więcej możliwości albo gotowe rozwiązanie od ręki (out of the box): bezprzewodowe czujniki temperatury i wilgotności z bramką i integracją z Zabbixem przez SNMP, listwa PDU z wbudowanym czujnikiem temperatury i wilgotności (jedna czujka na szafę, podłączana do listwy), albo gotowy zestaw czujników do szafy z wyjściem SNMP. W dużych centrach danych stosuje się czujniki co kilka szaf na trzech wysokościach, mapy cieplne w systemie DCIM, czujniki różnicy ciśnień między korytarzami i sterowanie obrotami wentylatorów klimatyzacji na ich podstawie.

Częste błędy

Objaw	Przyczyna	Co robimy
`esphome config` zgłasza błąd klucza API	klucz nie jest 32 bajtami w base64	generujemy klucz poleceniem `openssl rand -base64 32`
W dzienniku `Found no devices!`	brak rezystora 4,7 kΩ, zły pin albo odwrócone zasilanie	sprawdzamy rezystor, pin i kolory przewodów [4]
Odczyt zawsze 85,0 °C	wartość startowa układu albo spadek napięcia	sprawdzamy zasilanie, dodajemy odrzucanie 85,0 w skrypcie
Różnica ciśnień w szumie, skacze o kilka Pa	rurka w strumieniu powietrza albo zgnieciona	przesuwamy końcówkę w spokojne miejsce, prostujemy rurkę
Znak ΔP odwrotny	rurki zamienione	zamieniamy rurki albo dopisujemy `multiply: -100`
Płytka restartuje się co 15 minut	`api` bez klienta i bez `reboot_timeout: 0s`	dopisujemy `reboot_timeout: 0s`
Element w Zabbixie niewspierany	zły URL, hasło albo brak JSONPath	sprawdzamy `curl` z tego samego adresu
Odczyty skaczą, część brakuje	zbyt długi kabel, słabe styki	skracamy kabel, lutujemy, sprawdzamy rezystor

□□ Ćwiczenia

1. Dlaczego sondy wlotu mocujemy 5 cm przed serwerem, a nie przy klimatyzatorze?

Odpowiedź

Serwer chłodzi się powietrzem z własnego wlotu, a jego temperatura zależy od mieszania w szafie. Czujnik przy klimatyzatorze mierzy nawiew, a nie to, co serwer zasysa [12].

1. Wloty wynoszą 22,8, 23,5 i 24,6 °C, a wyloty 34,9, 35,8 i 36,4 °C. Jakie jest ΔT szafy?

Odpowiedź

Średni wylot to 35,7 °C, średni wlot 23,6 °C, więc ΔT to około 12,1 K. To zgodne z oczekiwanymi 12 K.

1. Czujnik ΔP pokazuje stale minus 4 Pa. Co to znaczy dla przepływu i co sprawdzamy?

Odpowiedź

Jeśli przyjęliśmy ΔP jako zimny minus gorący, ujemna wartość oznacza, że gorący korytarz ma wyższe ciśnienie, więc gorące powietrze idzie na front, czyli recyrkulacja. Najpierw sprawdzamy, czy rurki nie są zamienione (test dmuchnięciem), potem drzwi korytarza i zaślepki.

1. Ile czujników DS18B20 i ile magistral potrzeba dla pięciu szaf R1 do R5?

Odpowiedź

Po sześć na szafę, razem 30 czujników, i pięć magistral, po jednej na szafę, w dwóch modułach ESP32-POE.

1. Dlaczego do hasła do web_server używamy `!secret`, a nie wpisujemy go w pliku?

Odpowiedź

Plik konfiguracyjny pokazujemy w dokumentacji i wrzucamy do repozytorium. Hasła w `secrets.yaml` zostają poza nim.

Powiązane: Przepływ powietrza w serwerowni: szafa, korytarze, bypass i recyrkulacja, Planowanie chłodzenia: bilans, zapas N+1, awaria klimatyzacji, Termometry w praktyce: wybór, montaż i własny czujnik (DIY), Serwerownia Zielonej: gdzie stoją czujniki i dokąd płyną alarmy, Monitoring zasilania i łagodne wyłączanie: UPS, NUT, Zabbix

Źródła:

1. ESPHome, Native API Component
2. ESPHome, Ethernet Component
3. Olimex, ESP32-POE na GitHub
4. ESPHome, 1-Wire Bus oraz 1-Wire przez GPIO
5. ESPHome, SDP3x Sensor oraz I2C Bus
6. ESPHome, Web Server Component oraz Web Server API
7. Zabbix 7.0, zabbix_sender
8. Zabbix 7.0, elementy wyliczane
9. Home Assistant, karta gauge
10. Analog Devices, karta katalogowa DS18B20
11. Sensirion, SDP810-500Pa: cyfrowy czujnik różnicy ciśnień z przyłączem rurki, zasilanie 2,7 do 5,5 V, dokładność 3 procent odczytu, zero 0,1 Pa

12. Upsite, Data Center Temperature Sensor Location
13. ASHRAE, Thermal Guidelines, 5th edition, karta referencyjna
14. Zabbix 7.0, element HTTP agent
15. Zabbix 7.0, wyrażenia wyzwalaczy
16. Zabbix 7.0, widżet Honeycomb

Revision #14
Created 2026-10-03 12:08:57 UTC by Tomasz Konieczny
Updated 2026-10-26 14:20:05 UTC by Tomasz Konieczny