DIY: pomiar napięcia, prądu i energii na ESP32

Poziom: zaawansowany

Dotyczy: ESP32, ESPHome 2026.x, moduł PZEM-004T v3, cęgi SCT-013, Home Assistant, Zabbix 7.0 LTS

Czas: ~120 min

Budujemy własny miernik napięcia, prądu, mocy i energii jednego obwodu szafy na ESP32, który wysyła odczyty do Home Assistant i do Zabbixa na `mon01`. Pokazujemy dwa warianty: moduł PZEM-004T v3 i bezpieczniejsze cęgi na przewód, które mierzą tylko prąd. Montaż po stronie 230 V wykonuje wyłącznie osoba z uprawnieniami, a my robimy resztę: program, kalibrację i alarmy. Komendy ESPHome wykonujemy na stacji administratora z Pythonem 3.12 albo nowszym i dostępem do sieci serwerowni.

> Napięcie 230 V zagraża życiu. Wszystko, co dotyka przewodów sieciowych, czyli zaciski modułu PZEM, bezpiecznik i założenie cęgów na przewód w rozdzielnicy, wykonuje osoba z uprawnieniami SEP w zamkniętej obudowie i z zabezpieczeniem nadprądowym. Zasady opisują strony Czym są uprawnienia SEP i kto ich potrzebuje, Organizacja pracy przy urządzeniach elektroenergetycznych i Ochrona przeciwporażeniowa.

Skróty użyte na tej stronie (31)

Skrót	Co znaczy
ADC	Analog to Digital Converter: przetwornik zamieniający napięcie na liczbę, dzięki któremu mikrokontroler czyta sygnały analogowe
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
CE	Community Edition: darmowa, społecznościowa wersja Dockera, w odróżnieniu od płatnej wersji Enterprise
DCIM	Data Center Infrastructure Management: dokumentacja tego, co fizycznie stoi w serwerowni, czyli szafy, urządzenia i okablowanie

Skrót	Co znaczy
DIN	Deutsches Institut für Normung: niemiecki instytut normalizacyjny; szyna DIN to standardowa szyna montażowa aparatów w rozdzielnicy
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
GND	Ground: masa, wspólny punkt odniesienia obwodu elektrycznego
GPU	Graphics Processing Unit: karta graficzna, która liczy modele językowe wielokrotnie szybciej niż procesor
HTTP	Hypertext Transfer Protocol: protokół stron WWW
I2C	Inter-Integrated Circuit: prosta dwuprzewodowa magistrala, którą mały układ (np. czujnik) rozmawia z mikrokontrolerem
JSON	JavaScript Object Notation: tekstowy format danych z klamrami i cudzysłowami
kW	Kilowat: tysiąc watów, miara mocy; prawie cała moc pobrana przez serwery zamienia się w ciepło
kWh	Kilowatogodzina: jednostka energii, 1000 watów pracujących przez godzinę
LTS	Long Term Support: wersja z długim wsparciem poprawkami
Modbus	Prosty protokół komunikacji przemysłowej, którym liczniki energii i moduły pomiarowe wysyłają odczyty
MWh	Megawatogodzina: 1000 kilowatogodzin
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
PZEM	Nazwa rodziny tanich modułów do pomiaru napięcia, prądu, mocy i energii (PZEM-004T), odczytywanych przez UART i Modbus
RCD	Residual Current Device: wyłącznik różnicowoprądowy
REST	sposób budowania API przez zwykłe żądania HTTP (GET, POST) zamiast osobnego protokołu
RS485	Standard łącza szeregowego dla długich przewodów, używany przez liczniki energii z Modbus
SEP	Stowarzyszenie Elektryków Polskich: wydaje świadectwa kwalifikacyjne elektryków
SMS	Short Message Service: krótka wiadomość tekstowa na telefon, która dociera także bez internetu
SNMP	Simple Network Management Protocol: protokół odczytu stanu urządzeń sieciowych
UPS	Uninterruptible Power Supply: zasilacz awaryjny, podtrzymuje zasilanie urządzeń na czas zaniku napięcia z sieci

Skrót	Co znaczy
USB	Universal Serial Bus: standard portów i urządzeń podłączanych do komputera, tu pendrive'y i inne nośniki wymienne
Wi-Fi	Sieć bezprzewodowa: wygodna w biurze, ale w serwerowni dokłada kolejną możliwą przyczynę awarii
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 z jednego pliku YAML program dla płytki ESP32 i wgrywa go przez USB albo przez sieć [1].

◫◫ **ESPHome** = Oprogramowanie, które zamienia ESP32 w gotowy czujnik sieciowy bez pisania kodu od zera, a odczyty udostępnia przez sieć · **YAML** = YAML Ain't Markup Language: format plików konfiguracyjnych oparty na wcięciach · **ESP32** = Tani mikrokontroler z Wi-Fi i Bluetooth firmy Espressif: malutki komputer, z którego budujemy własne czujniki sieciowe · **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 ESPHome to 32 losowe bajty w base64. Wynik wpisujemy w sekcji `api` pliku konfiguracyjnego.

> W serwerowni Zielonej płytki z ESPHome mają Ethernet i zasilanie PoE, a nie Wi-Fi: tak działają `esp01` i `esp02` z czujnikami temperatury i wilgotności. Moduły pomiaru energii z tej strony (`energia-szafa02`, `cegi-szafa02`) są osobnymi urządzeniami o innej roli,

więc zachowują własne nazwy. Pliki poniżej używają Wi-Fi, bo pokazują wariant do prób na biurku i do domu. Moduł, który ma stać w serwerowni, dostaje Ethernet i PoE: zamiast bloku `wifi` wpisujemy blok `ethernet` jak na stronie ESPHome i urządzenia: adopcja esp01 i esp02, API albo MQTT, nazwy encji, a w sekcji `esp32` ustawiamy `board: esp32-poe`:

```
ethernet:
  type: LAN8720
  mdc_pin: GPIO23
  mdio_pin: GPIO18
  clk:
    pin: GPIO17
    mode: CLK_OUT
  phy_addr: 0
  power_pin: GPIO12
  manual_ip:
    static_ip: AdresModulu
    gateway: 10.20.10.1
    subnet: 255.255.255.0
    dns1: 10.20.10.11
```

Blok pochodzi z przykładu w dokumentacji ESPHome dla płytki Olimex ESP32-POE, a numery pinów sprawdzamy w opisie własnej płytki. Ethernet zajmuje m.in. GPIO17, GPIO21 i GPIO22, czyli piny linii szeregowej (GPIO17) i magistrali I2C (GPIO21 i GPIO22) z plików poniżej. Przy płytce z Ethernetem przenosimy je na wolne, na przykład linię szeregową na GPIO32 i GPIO33, a I2C na GPIO13 i GPIO16, jak w `esp01.yaml` na wskazanej stronie. Adres `AdresModulu` przydzielamy z puli serwerowni, bo adresy `10.20.10.70` i `10.20.10.71` należą do `esp01` i `esp02`.

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 · **PoE** = Power over Ethernet: zasilanie urządzenia tym samym kablem sieciowym, którym płyną dane · **Wi-Fi** = Sieć bezprzewodowa: wygodna w biurze, ale w serwerowni dokłada kolejną możliwą przyczynę awarii · **MQTT** = Message Queuing Telemetry Transport: lekki protokół, w którym urządzenia wysyłają krótkie wiadomości na nazwane tematy przez serwer pośredniczący · **I2C** = Inter-Integrated Circuit: prosta dwuprzewodowa magistrala, którą mały układ (np. czujnik) rozmawia z mikrokontrolerem · **openssl** = zestaw narzędzi do certyfikatów i szyfrowania · **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

PLIK KONFIGURACYJNY

Plik konfiguracyjny z modułem PZEM-004T: energia-szafa02.yaml

```
esphome:
  name: energia-szafa02
  friendly_name: Energia szafa 02

esp32:
  board: esp32dev
  framework:
    type: esp-idf

logger:

wifi:
  ssid: "NazwaSieciWifi"
  password: "HasloSieciWifi"
  ap:
    ssid: "energia-szafa02"
    password: "HasloAwaryjne"

# Home Assistant łączy się przez API. Zabbix czyta dane przez web_server (REST).
api:
  encryption:
    # Przykładowy klucz (32 bajty w base64). Własny generujemy poleceniem openssl rand -
base64 32.
    key: "7/xlYaHAQKumI9IYRpTi51i6lEbUvWknaark851jG+I="
  reboot_timeout: 0s

ota:
  - platform: esphome
    password: "HasloOta"

web_server:
  port: 80
  auth:
    username: "nazwa_usera"
    password: "HasloWebServera"
```

```
# Linia szeregowa do modułu PZEM-004T v3: 9600 bodów, 8N1
uart:
  tx_pin: GPIO17
  rx_pin: GPIO16
  baud_rate: 9600
  stop_bits: 1

modbus:

sensor:
  - platform: pzemac
    address: 1
    voltage:
      name: "napiecie_tor_a"
    current:
      name: "prad_tor_a"
    power:
      name: "moc_tor_a"
    energy:
      name: "energia_tor_a"
    frequency:
      name: "czestotliwosc_tor_a"
    power_factor:
      name: "wspolczynnik_mocy_tor_a"
    update_interval: 30s
  - platform: wifi_signal
    name: "wifi_sygnal"
    update_interval: 60s
```

Komponent `pzemac` potrzebuje sekcji `uart` i `modbus`, a domyślny adres modułu to 1 [2]. Czujnik energii podaje wartość w watogodzinach z klasą `energy` i `total_increasing`, więc Home Assistant rozpoznaje go bez dodatkowej konfiguracji [3].

□□ PLIK KONFIGURACYJNY

Plik konfiguracyjny z cęgami SCT-013: cegi-szafa02.yaml

```
esphome:
  name: cegi-szafa02
  friendly_name: Cegi szafa 02

esp32:
  board: esp32dev
  framework:
    type: esp-idf

logger:

wifi:
  ssid: "NazwaSieciWifi"
  password: "HasloSieciWifi"
  ap:
    ssid: "cegi-szafa02"
    password: "HasloAwaryjne"

api:
  encryption:
    # Przykładowy klucz (32 bajty w base64). Własny generujemy poleceniem openssl rand -
base64 32.
    key: "5ZgVEjnAZ/Mg91g+Z8LGPezsgVxqJenW+ezwaeQp1dQ="
  reboot_timeout: 0s

ota:
  - platform: esphome
    password: "HasloOta"

web_server:
  port: 80
  auth:
    username: "nazwa_usera"
    password: "HasloWebServera"

# Przetwornik ADS1115 na magistrali I2C, wejście A0 względem masy
i2c:
  sda: GPIO21
  scl: GPIO22
```

```
ads1115:
  - address: 0x48

sensor:
  - platform: ads1115
    id: wejscie_cegow
    multiplexer: 'A0_GND'
    gain: 6.144
    update_interval: never
    internal: true
  - platform: ct_clamp
    sensor: wejscie_cegow
    name: "prad_szafa02"
    update_interval: 30s
    sample_duration: 200ms
    filters:
      # 0,8 V skutecznie odpowiada 16 A (cęgi 100 A:50 mA, rezystor 100 omów).
      # Wartości kalibrujemy miernikiem cęgowym.
      - calibrate_linear:
          - 0.0 -> 0.0
          - 0.8 -> 16.0
  - platform: wifi_signal
    name: "wifi_sygnal"
    update_interval: 60s
```

Komponent `ct_clamp` liczy wartość skuteczną sygnału z przetwornika ADS1115 w oknie 200 ms, czyli w dziesięciu okresach prądu 50 Hz [4]. Filtr `calibrate_linear` zamienia wolty na ampery.

□□ TYLKO ODCZYT

Sprawdzenie pliku: esphome config energia-szafa02.yaml

```
esphome config energia-szafa02.yaml
```

Przy błędzie komenda wypisuje miejsce w pliku. Dla drugiego wariantu podajemy `cegi-szafa02.yaml`.

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

TYLKO ODCZYT

Wgranie programu przez USB: esphome run energia-szafa02.yaml

```
esphome run energia-szafa02.yaml
```

Pierwsze wgranie robimy kablem USB, a kolejne przez sieć. Komenda pyta o port, a po wgraniu pokazuje dziennik płytki.

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 dziennika płytki: esphome logs energia-szafa02.yaml

```
esphome logs energia-szafa02.yaml
```

Przykład:

```
[10:12:31][D][sensor]: 'napiecie_tor_a': Sending state 231.39999 V with 1 decimals of accuracy
[10:12:31][D][sensor]: 'prad_tor_a': Sending state 5.65000 A with 3 decimals of accuracy
[10:12:31][D][sensor]: 'moc_tor_a': Sending state 1305.40002 W with 1 decimals of accuracy
```

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

TYLKO ODCZYT

Odczyt wartości przez REST: curl

```
curl --user nazwa_usera http://AdresPlytki/sensor/moc_tor_a
```

Przy takim zapisie `curl` pyta o hasło, więc nie zostaje ono w historii poleceń. Serwer HTTP płytki odpowiada obiektem JSON z polem `value`, które Zabbix wyciąga w następnym kroku

[5].

Przykład:

```
{"id":"sensor/moc_tor_a","state":"1305.4 W","value":1305.4}
```

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

KLIKAMY W INTERFEJSIE

Dodanie urządzenia w Home Assistant

```
Ustawienia → Urządzenia i usługi → ESPHome → Dodaj urządzenie
```

(nazwy pozycji zależą od wersji Home Assistant; wpisujemy adres płytki i klucz API z pliku konfiguracyjnego)

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

KLIKAMY W INTERFEJSIE

Dodanie obwodu do panelu Energia w Home Assistant

```
Ustawienia → Pulpity → Energia → Dodaj zużycie → energia_tor_a
```

Panel Energia przyjmuje czujniki z klasą energii i stanem `total` albo `total_increasing`, w jednostkach Wh, kWh albo MWh [3].

kWh = Kilowatogodzina: jednostka energii, 1000 watów pracujących przez godzinę · **MWh** = Megawatogodzina: 1000 kilowatogodzin

Z życia wzięte

Z ŻYCIA WZIĘTE

Czwartek, 15:20. Przed dołożeniem serwerów GPU Tomek chce wiedzieć, ile prądu naprawdę pobiera szafa R2, ale jej listwy PDU są podstawowe, bez pomiaru. Wpisuje w arkusz moc z

> tabliczek i wychodzi mu 5 kW, a miernik cęgowy pokazuje w tym samym momencie 5,7 A na torze A i tyle samo na torze B, czyli razem około 2,6 kW. Różnica jest prawie dwukrotna, więc Ola pyta, czy da się mieć taki pomiar stale, a nie tylko wtedy, gdy ktoś przyjdzie z miernikiem.

Projekt DIY krok po kroku

Płytka ESP32 z modułem Wi-Fi i Bluetooth. Ten sam układ, który w innych stronach tej książki mierzy

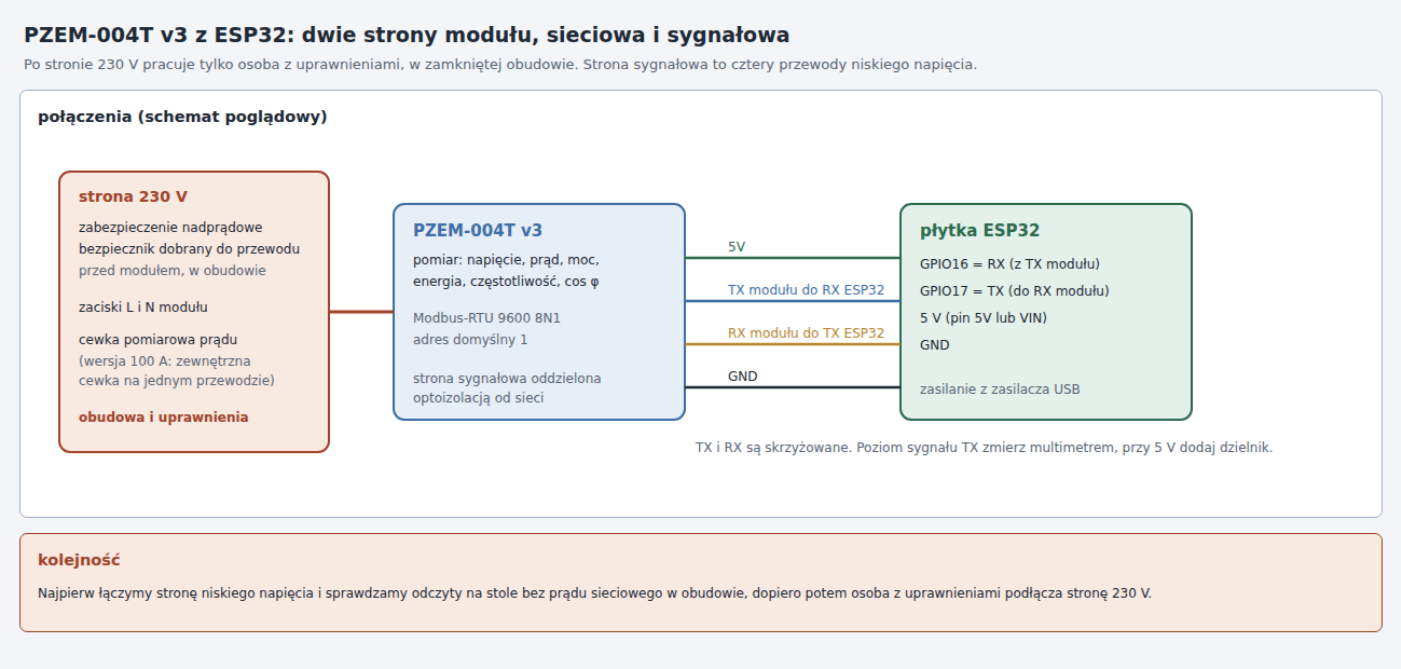
Płytka ESP32 z modułem Wi-Fi i Bluetooth. Ten sam układ, który w innych stronach tej książki mierzy temperaturę i wilgotność, tu odczytuje licznik energii. Źródło: Wikimedia Commons (autor i wolna licencja podane na stronie pliku).

Mierzymy tor A szafy R2. Przy 2,6 kW cała szafa pobiera 11,3 A, więc w razie awarii toru B tor A niesie cały prąd. Wersja modułu PZEM na 10 A z przewodem przez bocznik odpada, bo prąd może przekroczyć jej zakres, i wybieramy wersję 100 A z cewką, którą zakłada się na jeden przewód [6].

Lista części

Część	Po co	Uwagi
Płytka ESP32 (np. DevKit)	odczyt i wysyłka danych	Wi-Fi lub, jeśli odczyty wypadają, wersja z Ethernetem
PZEM-004T v3 z cewką 100 A	pomiar napięcia, prądu, mocy, energii, częstotliwości, współczynnika mocy	zakres 80 do 260 V, dokładność 0,5 procent [6]
Zasilacz 5 V z oznaczeniem CE	zasilanie strony niskiego napięcia	gotowy, nie własnej roboty
Bezpiecznik z oprawą	zabezpieczenie linii napięciowej modułu	dobrany przez elektryka do przewodu
Obudowa zamknięta, plastikowa	ochrona przed dotykiem	z przepustami kablowymi
Listwa zaciskowa, przewody	połączenie strony 230 V	przekrój dobiera elektryk
Miernik cęgowy	wzorzec do kalibracji	z aktualnym świadectwem wzorcowania
Przewody sygnałowe	linia TX, RX, 5 V, GND	krótkie, skręcone

Schemat połączeń



Rysunek 1. PZEM-004T v3 z ESP32: po lewej strona 230 V (zabezpieczenie, zaciski, cewka prądowa), w środku moduł, po prawej płytka ESP32. Cztery przewody sygnałowe 5V, TX, RX i GND są skrzyżowane, a strona sygnałowa modułu jest oddzielona optoizolacją od sieci. (kliknięcie otwiera pełny rozmiar)

Moduł wymaga zewnętrznego zasilania 5 V po stronie sygnałowej [6]. Linia TX modułu idzie do RX płytki, a RX modułu do TX płytki. Przed podłączeniem mierzymy multimetrem poziom sygnału TX modułu: ESP32 pracuje na 3,3 V, więc przy 5 V dodajemy dzielnik rezystorowy albo konwerter poziomów [7].

Program

Plik `energia-szafa02.yaml` z Fiszki wgrywamy komendą `esphome run`. W dzienniku po minucie powinny pojawić się odczyty. Brak odczytów oznacza zwykle zamienione przewody TX i RX albo brak zasilania 5 V modułu.

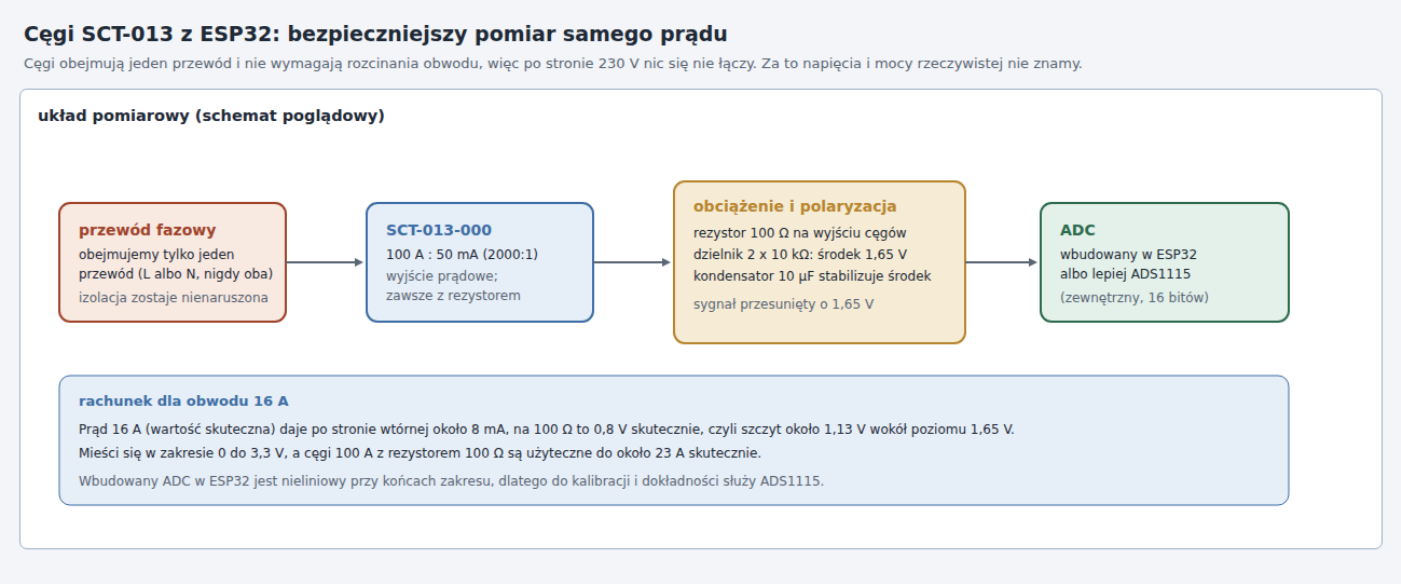
Kalibracja i porównanie z wzorcem

Moduł jest fabrycznie kalibrowany, ale porównujemy go z miernikiem cęgowym, który ma świadectwo wzorcowania. Mierzymy przy trzech obciążeniach, na przykład przy mocy 25, 50 i 100 procent typowego poboru szafy, i zapisujemy różnicę. Rozbieżność powyżej 1 procenta wymaga poprawki w filtrze `multiply` albo wyjaśnienia przyczyny. Producent podaje dokładność 0,5 procent [6].

Obudowa i miejsce montażu

Moduł z zabezpieczeniem i płytką ESP32 montujemy w zamkniętej obudowie przy rozdzielnicy, nie wewnątrz szafy z serwerami. Obudowę opisujemy: „Pomiar energii, tor A szafy R2, 230 V". Prowadzimy kabel sygnałowy do sieci Ethernet albo ustawiamy Wi-Fi z zapasem sygnału (sprawdzamy `wifi_sygnal`).

Drugi wariant: cęgi SCT-013 i ADS1115



Rysunek 2. Cęgi SCT-013 z ESP32: cęgi obejmują jeden przewód, rezystor 100 omów zamienia prąd cęgów na napięcie, dzielnik 2 x 10 kΩ ustawia środek na 1,65 V, a przetwornik ADS1115 mierzy sygnał. Warto zauważyć rachunek: 16 A daje około 0,8 V skutecznie. (kliknięcie otwiera pełny rozmiar)

Cęgi SCT-013-000 przekształcają prąd 100 A na 50 mA (przekładnia 2000 do 1) i nie mają wbudowanego rezystora obciążającego. Cęgi zawsze pracują z rezystorem, bo otwarty obwód wtórny może mieć niebezpiecznie wysokie napięcie. Z rezystorem 100 Ω prąd 16 A (wartość skuteczna) daje około 0,8 V, czyli szczyt około 1,13 V wokół poziomu 1,65 V, co mieści się w zakresie 0 do 3,3 V. Dla większych prądów rezystor trzeba zmniejszyć, bo przy 100 Ω zakres kończy się około 23 A skutecznie.

Taki pomiar jest prostszy i bezpieczniejszy: po stronie 230 V cęgi obejmują izolowany przewód i nie rozcinamy obwodu. Nadal jednak zakładamy je na pojedynczy przewód w rozdzielnicy, więc robi to

osoba z uprawnieniami. Za to nie znamy napięcia ani współczynnika mocy, a moc rzeczywista wymagałaby dodatkowego pomiaru napięcia. Wbudowany przetwornik ADC w ESP32 jest nieliniowy przy końcach zakresu, dlatego używamy zewnętrznego ADS1115 [4].

Trzeci wariant: licznik z Modbus

Dla szafy trójfazowej albo większego obwodu wybieramy gotowy licznik z interfejsem RS485 (na przykład Eastron SDM) albo układ z rodziny ATM90E3x. ESPHome obsługuje oba [8][9]. Licznik podłączamy przez konwerter RS485 do linii szeregowej płytki.

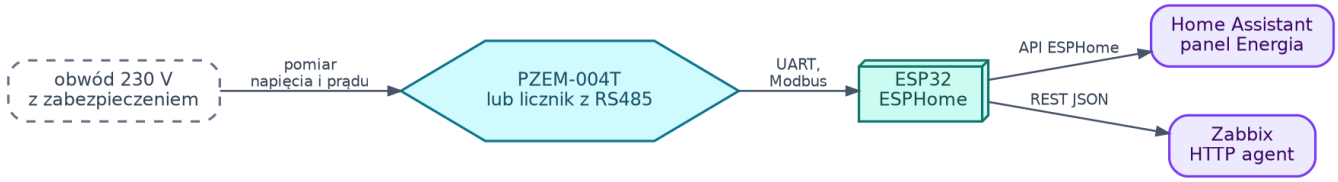
```
uart:
  tx_pin: GPIO17
  rx_pin: GPIO16
  baud_rate: 9600
  stop_bits: 1

modbus:
  flow_control_pin: GPIO4

sensor:
  - platform: sdm_meter
    address: 1
    phase_a:
      voltage:
        name: "napiecie_l1"
      current:
        name: "prad_l1"
      active_power:
        name: "moc_l1"
    total_power:
      name: "moc_calkowita"
    update_interval: 30s
```

Parametry linii (prędkość, adres) sprawdzamy w instrukcji licznika, bo niektóre modele mają fabrycznie 2400 bodów [8].

Integracja z monitoringiem



Rysunek 3. Droga danych: obwód 230 V z zabezpieczeniem, moduł pomiarowy, ESP32 z ESPHome i dwa odbiorcy: Home Assistant przez API oraz Zabbix przez REST. Warto zauważyć, że ESP32 trzyma tylko bieżące wartości, a historię przechowują systemy po prawej. (kliknięcie otwiera pełny rozmiar)

Home Assistant służy do podglądu i panelu Energia, a Zabbix na `mon01` do alarmów. Oba korzystają z tej samej płytki. Home Assistant łączy się przez natywne API (z kluczem z pliku konfiguracyjnego), a Zabbix pyta serwer HTTP płytki elementem typu HTTP agent, tak jak w czujnikach temperatury [5].

Host i elementy danych w Zabbixie

Host nazywamy `energia-szafa02`, dodajemy do grupy `Pomiary zasilania` i nie dodajemy interfejsu agenta. Na hoście ustawiamy makra: `{$CZUJNIK.ADRES}` (adres płytki), `{$CZUJNIK.UZYTKOWNIK}` i `{$CZUJNIK.HASLO}` (typ Secret text).

Nazwa	Klucz	URL	Przetwarzanie wstępne	Jednostka
Napięcie toru A	`energia.napiecie.tor.a`	`http://{$CZUJNIK.ADRES}/sensor/napiecie_tor_a`	JSONPath `$.value`	V
Prąd toru A	`energia.prad.tor.a`	`http://{$CZUJNIK.ADRES}/sensor/prad_tor_a`	JSONPath `$.value`	A
Moc toru A	`energia.moc.tor.a`	`http://{$CZUJNIK.ADRES}/sensor/moc_tor_a`	JSONPath `$.value`	W
Energia toru A	`energia.energia.tor.a`	`http://{$CZUJNIK.ADRES}/sensor/energia_tor_a`	JSONPath `$.value`	Wh

W każdym elemencie ustawiamy uwierzytelnianie podstawowe z makrami, interwał 1 minuty i typ informacji liczba zmiennoprzecinkowa. Energia rośnie narastająco, więc na wykresach pokazujemy jej przyrost w wybranym okresie.

Wyzwalacze

Makra progów ustawiamy na hoście: `{$PRAD.OSTRZEZENIE}` równe 12,8 (80 procent obwodu 16 A), `{$NAPIECIE.MIN}` równe 207 i `{$NAPIECIE.MAX}` równe 253. Granice napięcia to 230 V plus minus 10 procent, co odpowiada typowym wymaganiom jakości napięcia w sieci publicznej [10].

Prąd toru A powyżej 80 procent obwodu (poziom Warning):

```
min(/energia-szafa02/energia.prad.tor.a,5m)>{$PRAD.OSTRZEZENIE}
```

Napięcie poza zakresem przez dłuższy czas (poziom Warning):

```
min(/energia-szafa02/energia.napiecie.tor.a,5m)>{$NAPIECIE.MAX}
or max(/energia-szafa02/energia.napiecie.tor.a,5m)<{$NAPIECIE.MIN}
```

Zanik napięcia na torze A (poziom High):

```
max(/energia-szafa02/energia.napiecie.tor.a,1m)<100
```

Brak danych z miernika (poziom High):

```
nodata(/energia-szafa02/energia.moc.tor.a,5m)=1
```

Alarm „zanik napięcia" mówi, że tor A przestał dostawać prąd, czyli serwery pracują na samym torze B bez zapasu. Takiego alarmu UPS nie pokaże, bo UPS widzi wejście, a nie obwód szafy.

Alarmy wysyłamy pocztą i SMS jak dla pozostałych poziomów, co opisuje strona Serwerownia Zielonej: gdzie stoją czujniki i dokąd płyną alarmy.

Test progów robimy bez ryzyka dla serwerów: obniżamy makro `{$PRAD.OSTRZEZENIE}` poniżej bieżącego prądu i sprawdzamy, czy alarm dochodzi, a potem przywracamy 12,8.

Bezpieczeństwo i ograniczenia

- Każdy montaż po stronie 230 V wykonuje osoba z uprawnieniami SEP. My nie dotykamy zacisków, bezpiecznika ani przewodów sieciowych.
- Moduł, bezpiecznik i płytka są w zamkniętej obudowie. Obwód ma własne zabezpieczenie nadprądowe, a obudowa jest opisana.
- Zasilacz 5 V to gotowy zasilacz z oznaczeniem CE, a nie wlutowany układ własnej roboty.
- Własny miernik nie ma świadectwa wzorcowania. Nie używamy go do rozliczeń z operatorem ani wynajmującym, a do porównań wewnętrznych tak.
- Montaż nie zastępuje przeglądów instalacji i pomiarów ochronnych (rezystancja izolacji, pętla zwarcia, RCD), które wykonuje osoba z uprawnieniami [11].
- Montaż w serwerowni uzgadniamy z zarządcą budynku i ubezpieczycielem, bo urządzenie własnej roboty może wpływać na warunki polisy i gwarancji.
- Serwer HTTP płytki domyślnie używa uwierzytelniania podstawowego, a dokumentacja zapowiada zmianę domyślnego schematu na digest w wersji 2027.1.0 [5]. Dostęp do portu 80 płytki ograniczamy do adresu `mon01`.

- Wi-Fi w serwerowni bywa niestabilne za metalowymi szafami. Jeśli odczyty wypadają, przechodzimy na Ethernet.
- Pomiar poboru jednego toru nie pokazuje obciążenia drugiego. Tory A i B mierzymy osobno albo przez PDU z pomiarem.

> Więcej możliwości albo gotowe rozwiązanie od ręki (out of the box): listwa PDU z pomiarem na każdym gnieździe i dostępem przez SNMP, UPS z kartą sieciową, który sam raportuje obciążenie i czas pracy, albo licznik z interfejsem Modbus (RS485) na szynie DIN w rozdzielnicy. W dużych centrach danych stosuje się szynoprzewody z pomiarem w każdej gałęzi, system BMS i DCIM, który łączy pomiary mocy z temperaturą i planem szaf, oraz pomiar na poziomie każdego serwera.

Częste błędy

Objaw	Przyczyna	Co robimy
Brak odczytów z modułu PZEM	zamienione TX i RX, brak 5 V, zły adres Modbus	sprawdzamy połączenia i `esphome logs`, adres domyślny to 1
Odczyty „NaN" lub zera	moduł bez napięcia sieciowego albo bez cewki na przewodzie	sprawdzamy zasilanie modułu i założenie cewki (osoba z uprawnieniami)
Prąd wyższy lub niższy o kilka procent	brak kalibracji, cewka założona na oba przewody	porównujemy z miernikiem cęgowym, cewka obejmuje jeden przewód
Odczyt prądu z cęgów skacze	wbudowany ADC ESP32 jest nieliniowy	używamy ADS1115 i kalibrujemy filtrem
Płytka restartuje się co 15 minut	brak klienta API (Home Assistant)	ustawiamy `reboot_timeout: 0s` w sekcji `api`
Zabbix pokazuje niewspierany element	błędna ścieżka JSONPath albo zła nazwa czujnika w adresie	sprawdzamy `curl` i nazwę czujnika
Alarm napięcia miga	próg za blisko normalnej wartości	rozszerzamy okno czasu albo próg

□□ Ćwiczenia

1. Szafa pobiera 3 kW przy 230 V. Który moduł PZEM-004T v3 wybieramy: 10 A czy 100 A z cewką?

Odpowiedź

3000 W dzielone przez 230 V daje 13 A, więc wersja 10 A odpada. Wybieramy wersję 100 A z cewką.

1. Cęgi 100 A:50 mA z rezystorem 100 Ω mierzą obwód 25 A. Czy zmieścimy się w zakresie przetwornika?

Odpowiedź

25 A to 12,5 mA po stronie wtórnej, a na 100 Ω daje to 1,25 V skutecznie, czyli szczyt 1,77 V przy środku 1,65 V. Wychodzi powyżej 3,3 V, więc nie. Zmniejszamy rezystor albo wybieramy cęgi o innej przekładni.

1. Alarm „zanik napięcia toru A" włączył się, a UPS nie zgłasza żadnego problemu. Co to znaczy?

Odpowiedź

UPS pracuje poprawnie, ale tor A szafy nie ma prądu: wyłączył się wyłącznik obwodu albo uszkodzona jest listwa PDU. Serwery pracują na torze B bez zapasu, więc usuwamy przyczynę natychmiast.

Powiązane: Monitoring zasilania i łagodne wyłączanie: UPS, NUT, Zabbix, Planowanie zasilania serwerowni: bilans mocy, zapas, rozbudowa, Termometry w praktyce: wybór, montaż i własny czujnik (DIY), Prąd: wielkości i zależności

Źródła:

1. ESPHome, dokumentacja: esphome.io
2. ESPHome, komponent pzemac: esphome.io
3. Home Assistant, panel Energia: home-assistant.io
4. ESPHome, komponent ct_clamp: esphome.io
5. ESPHome, web_server: esphome.io
6. Peacefair, PZEM-004T v3, karta katalogowa: narodmon.com
7. MycilaPZEM, uwagi o poziomach sygnałów i podłączeniu do ESP32: mathieu.carbou.me
8. ESPHome, komponent sdm_meter: esphome.io
9. ESPHome, komponent atm90e32: esphome.io
10. Power Quality Blog, „Standard EN 50160": powerquality.blog
11. Prawo budowlane, art. 62: lexlege.pl

Revision #16
Created 2026-10-03 11:23:32 UTC by Tomasz Konieczny
Updated 2026-10-26 14:19:49 UTC by Tomasz Konieczny