# System ostrzegania przed przymrozkami

## Etap 3: Testowanie rozwiązania

Michał Kołodziejczyk, Sebastian Chmielewski, Bartłomiej Gromulski

Opiekun projektu: dr inż. Wojciech Kazubski

1 lutego 2026

## Spis treści

# 1 Wstęp

## Uzasadnienie zmiany architektury

Zrezygnowano z wykorzystania bazy InfluxDB oraz narzędzia Telegraf na rzecz autorskiego oprogramowania w języku Python. Przejście na komunikację WebSocket i przechowywanie danych w pamięci operacyjnej pozwoliło na znaczną redukcję opóźnień w systemie ostrzegania oraz zmniejszenie obciążenia sprzętowego Raspberry Pi, kolejnym powodem były gotowe dashboardy Grafany które oferują ograniczone możliwości personalizacji mapy. Własny serwer HTTP (Flask) pozwolił na stworzenie dedykowanej wizualizacji, w której kolor znaczników na mapie zmienia się dynamicznie w zależności od skomplikowanych flag alarmowych, co w Grafanie wymagałoby zaawansowanych wtyczek i konfiguracji.

## Stacja pomiarowa

Stacja pomiarowa odpowiada za zbieranie pomiarów z czujników oraz przesłanie tych danych do stacji centralnej. Opiera się na Raspberry Pi Zero. Każda stacja mierzy wilgotność, prędkość wiatru oraz temperaturę na dwóch wysokościach: DS18B20 na wysokości 2 metrów oraz BME280 na wysokości 1 metra. Wyniki pomiarów przesyłane są do stacji centralnej przez LoRa.

### 1.0.1 Wilgotność

Urządzenie mierzy wilgotność względną powietrza przy użyciu czujnika BME280 połączonego przez magistralę I2C (adres 0x76). Pomiar realizowany jest co 30 sekund. Czujnik dodatkowo kompensuje temperaturę przy obliczaniu wilgotności, co zwiększa dokładność odczytów. Uśrednione wartości z cyklu pomiarowego (domyślnie 10 próbek) są wysyłane w ramce LoRa do stacji centralnej co 5 minut.

### 1.0.2 Temperatura

Temperatura jest mierzona w dwóch punktach: na wysokości 2 metrów (czujnik DS18B20 przez magistralę 1-Wire) oraz na wysokości 1 metra (czujnik BME280 przez I2C). DS18B20 charakteryzuje się niższą bezwładnością cieplną, co pozwala na szybsze wykrywanie gwałtownych spadków temperatury. BME280 z kolei lepiej reaguje na zmiany temperatury mas powietrza. Pomiary odbywają się co 30 sekund, a uśrednione wartości są wysyłane do stacji centralnej.

### 1.0.3 Wiatr

Prędkość wiatru mierzona jest za pomocą wiatromierza podłączonego do GPIO16. Urządzenie generuje impulsy (zwarcie kontaktronu) przy każdym obrocie. Mikrokontroler zlicza impulsy w przerwaniach, a następnie przelicza je na prędkość wiatru zgodnie ze współczynnikiem kalibracyjnym: 1 Hz = 2.4 km/h. Pomiar jest ciągły, a wynik obliczany co 30 sekund i przesyłany w ramce danych.

### 1.0.4 Punkt rosy

Punkt rosy obliczany jest na stacji centralnej po odebraniu danych. Wykorzystywane są wartości temperatury i wilgotności względnej z czujnika BME280. Obliczenie odbywa się według przybliżenia wzoru Magnusa:

$$\gamma = \frac{a \cdot T}{b + T} + \ln\left(\frac{H}{100}\right)$$

$$T_d = \frac{b \cdot \gamma}{a - \gamma}$$

Gdzie: $a = 17.27$, $b = 237.7$ [°C], $T$ - temperatura [°C], $H$ - wilgotność względna [%], $T_d$ - punkt rosy [°C]. Niski punkt rosy (poniżej 0°C) w połączeniu z niską temperaturą powietrza wskazuje na ryzyko przymrozku radiacyjnego.

### 1.0.5 Trend zmian

System oblicza tempo zmian temperatury (tzw. cooling rate) w stopniach Celsjusza na godzinę. Dla każdej stacji zapisywana jest ostatnia pomierzona temperatura i czas pomiaru. Przy kolejnym pomiarze obliczany jest gradient:

$$\text{Trend} = \frac{T_{\text{aktualne}} - T_{\text{poprzednie}}}{\Delta t\,[\text{h}]}$$

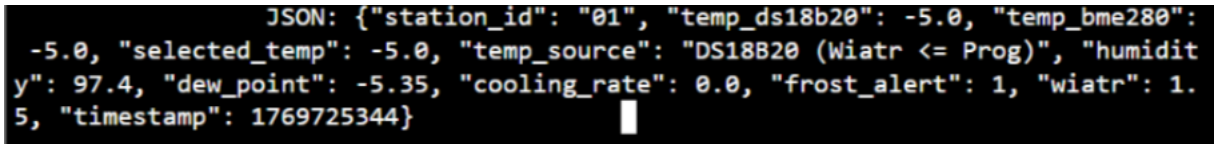
Aby wyeliminować błędy pomiarowe spowodowane zbyt częstymi aktualizacjami, trend jest obliczany tylko gdy minęło co najmniej 10 minut od ostatniego pomiaru. Gwałtowny spadek temperatury (poniżej -1.5°C/h) przy temperaturze poniżej 3.5°C jest traktowany jako sygnał ostrzegawczy.

# 2 Analiza komunikacji bezprzewodowej

### 2.0.1 Interfejsy

Moduł LoRa SX1262 komunikuje się z Raspberry Pi Zero przez interfejs SPI. Konfiguracja obejmuje: częstotliwość 868 MHz (pasmo ISM dla Europy), moc nadawania 14 dBm, Spreading Factor SF7, szerokość pasma 500 kHz oraz Coding Rate 5. Układ sterowany jest pinami GPIO: Reset (17), Busy (4), RxEn (5), TxEn (6) oraz CS (8). Wykorzystuje zewnętrzny oscylator TCXO dla stabilności częstotliwości.

### 2.0.2 Ramka




Rysunek 1: Ramka przesyłana przez lore.

Ramka danych ma stałą długość 32 bajtów bez separatorów, co upraszcza parsowanie. Struktura:

`ID(2) + TDS(6) + TBME(6) + HUM(5) + N(2) + CZAS(6) + WIATR(5)`
Przykład: `01+22.5+21.3045.210143052005.2`

- **ID** (2B): Identyfikator stacji (01-07)
- **TDS** (6B): Temperatura DS18B20 z znakiem, format ±XX.X
- **TBME** (6B): Temperatura BME280 z znakiem, format ±XX.X
- **HUM** (5B): Wilgotność względna, format XXX.X
- **N** (2B): Liczba uśrednionych próbek
- **CZAS** (6B): Czas na stacji nadawczej, format HHMMSS
- **WIATR** (5B): Prędkość wiatru [km/h], format XXX.X

Zastosowanie SF7 przy szerokości pasma 500 kHz zapewnia kompromis między zasięgiem (do kilku kilometrów w terenie otwartym) a szybkością transmisji. Ramka z dołączonym nagłówkiem i CRC zajmuje około 40 bajtów.

## 2.1 Stacja centralna

Stacja centralna odpowiada za odbiór danych przychodzących ze stacji końcowych, zapisywaniu w bazie danych i ich wizualizacji na stronie.

### 2.1.1 Odbiór danych

Moduł odbiorczy na Raspberry Pi 4B działa w trybie ciągłego nasłuchu (RX continous). Po wykryciu ramki (flaga IRQ_RX_DONE) system weryfikuje sumę kontrolną CRC. Prawidłowe ramki są odczytywane z bufora modułu i przekazywane do parsera.

Parser dekoduje pola o stałych pozycjach, konwertuje znaki ASCII na liczby zmiennoprzecinkowe i obsługuje znaczniki błędów (np. `N/A` dla niedziałającego czujnika). Następnie dane przekazywane są do logiki hybrydowej:

1. **Wybór źródła temperatury:** Jeśli wiatr $> 2.0$ m/s $\rightarrow$ BME280 (przymrozek adwekcyjny), w przeciwnym razie DS18B20 (przymrozek radiacyjny).
2. **Obliczenia:** Punkt rosy, tempo zmian temperatury.
3. **Decyzja alarmowa:** Flaga `frost_alert` ustawiana jest na 1, jeśli:
   - Temperatura $\leq$ 2.0°C (bezwzględny próg)
   - Punkt rosy $<$ 0°C i temperatura $<$ 5.0°C (suche powietrze)
   - Temperatura $\leq$ 3.5°C i trend $<$ -1.5°C/h (gwałtowny spadek)

Wynikowy JSON jest publikowany do brokera MQTT.

### 2.1.2 MQTT

Protokół MQTT pełni rolę magistrali komunikacyjnej między odbiornikiem LoRa a serwerem HTTP. Broker Mosquitto działa na localhost (127.0.0.1:1883).

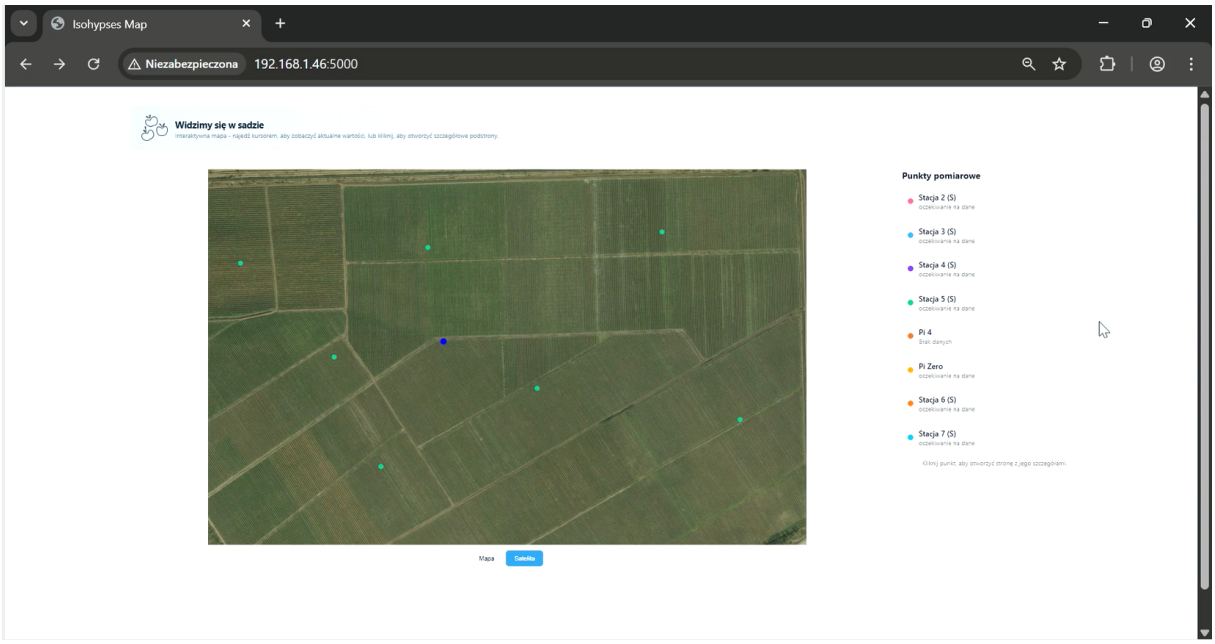
**Publisher (odbiornik_v7.py):** Publikuje przetworzone dane w formacie JSON na temat `lora/pogoda`. Przykładowa wiadomość zawiera: ID stacji, temperatury z obu czujników, wybraną temperaturę do analizy, źródło wyboru, wilgotność, punkt rosy, tempo chłodzenia, flagę alarmu oraz znacznik czasu Unix.

**Subscriber (ff.py):** Serwer HTTP subskrybuje temat `lora/pogoda` i aktualizuje struktury danych w pamięci. Każda nowa wiadomość MQTT jest mapowana na odpo-

wiedni indeks stacji (0-7) i zapisywana w kolejce historycznej (72 ostatnie pomiary). Równocześnie dane są rozgłaszane do wszystkich podłączonych klientów WebSocket.

### 2.1.3 Serwer HTTP

Serwer WWW oparty na Flask działa na porcie 5000 (dostępny w sieci lokalnej na adresie 0.0.0.0:5000). Obsługuje następujące trasy:




Rysunek 2: screen strony internetowej

- `GET /` → Strona główna z mapą wszystkich stacji
- `GET /<nazwa_stacji>` → Strona szczegółowa stacji z wykresami
- `GET /api/history/<index>` → JSON z historią pomiarów (72 ostatnie wartości)
- `GET /api/values` → JSON z aktualnymi wartościami wszystkich stacji

Strona główna (`index.html`) wyświetla interaktywną mapę z pozycjami 8 stacji. Po najechaniu kursorem na punkt wyświetla się tooltip z aktualnymi danymi. Kliknięcie otwiera stronę szczegółową (`point.html`) z wykresami temperatury, wilgotności i wiatru w czasie. Przy wykryciu alarmu przymrozkowego (`frost_alert=1`) kropka na mapie zmienia kolor na czerwony, a na stronie szczegółowej pojawia się widoczny komunikat ostrzegawczy. Klikając na kropkę można także podejrzeć na wykresach aktualny stan pomiarów odebrany przez punkt.

### 2.1.4 WebSocket

Z każdym połączonym do serwera hostem otwierane jest połączenie WebSocket umożliwiające przesłanie do wyświetlenia nowych danych bez konieczności przesyłania całego HTTP.

# 3 Badania funkcjonalne i logika systemu

Zweryfikowano poprawność działania zaimplementowanej logiki hybrydowej, która automatycznie dobiera źródło danych w zależności od warunków atmosferycznych (siły wiatru). Testy potwierdziły następujące zachowanie systemu:

- **Przymrozek Radiacyjny (Wiatr $<$ 2.0 m/s):** System poprawnie przełączał się na analizę danych z czujnika DS18B20 (2m wysokości). Sensor ten cechuje się niższą bezwładnością cieplną, co pozwala na szybsze wykrycie gwałtownego spadku temperatury w bezwietrzne noce.
- **Przymrozek Adwekcyjny (Wiatr $>$ 2.0 m/s):** Przy wykryciu ruchu mas powietrza powyżej progu, system bazował na odczytach z czujnika BME280 (1m wysokości), reagując na zmiany temperatury niesione przez wiatr.

W logach systemowych (format JSON) zaobserwowano poprawne flagi sterujące (np. `"frost_alert": 1`) oraz wyliczone parametry pochodne, takie jak punkt rosy i tempo chłodzenia.

## 3.1 Architektura i wizualizacja danych

Testy potwierdziły poprawność przesyłu danych w zaprojektowanej architekturze:

`Czujniki` $\rightarrow$ `LoRa` $\rightarrow$ `Gateway (Obliczenia)` $\rightarrow$ `MQTT` $\rightarrow$ `WebSocket` $\rightarrow$ `Klient WWW`

Aplikacja webowa mapuje pozycje węzłów na sztucznie stworzonym terenie na mapie terenu i aktualizuje ich status w czasie rzeczywistym.

# 4 Analiza komunikacji przewodowej

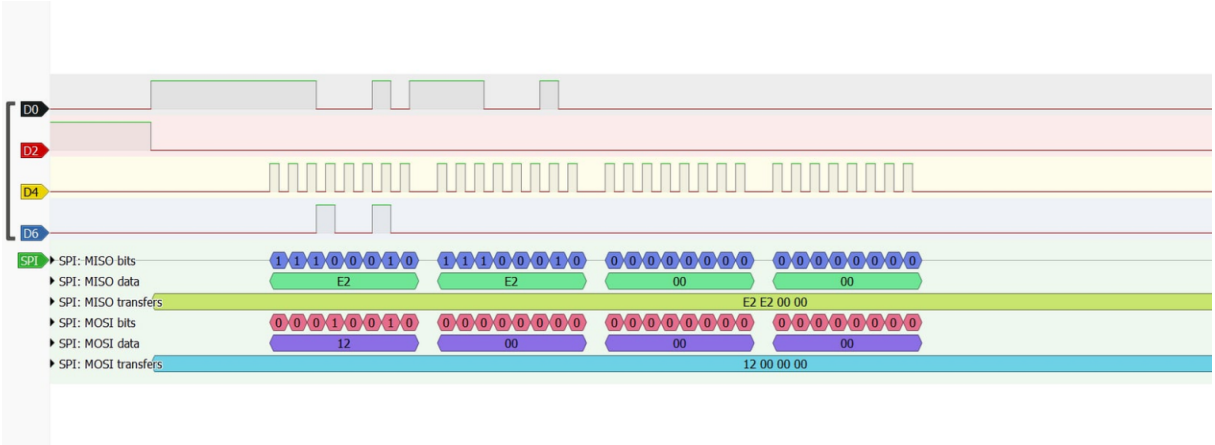
Zgodnie z wymaganiami przeprowadzono analizę sygnałów na magistralach komunikacyjnych przy użyciu analizatora stanów logicznych. Pozwoliło to na weryfikację poprawności działania sterowników sprzętowych.

## 4.1 Interfejs SPI – Moduł LoRa i Raspberry Pi Zero

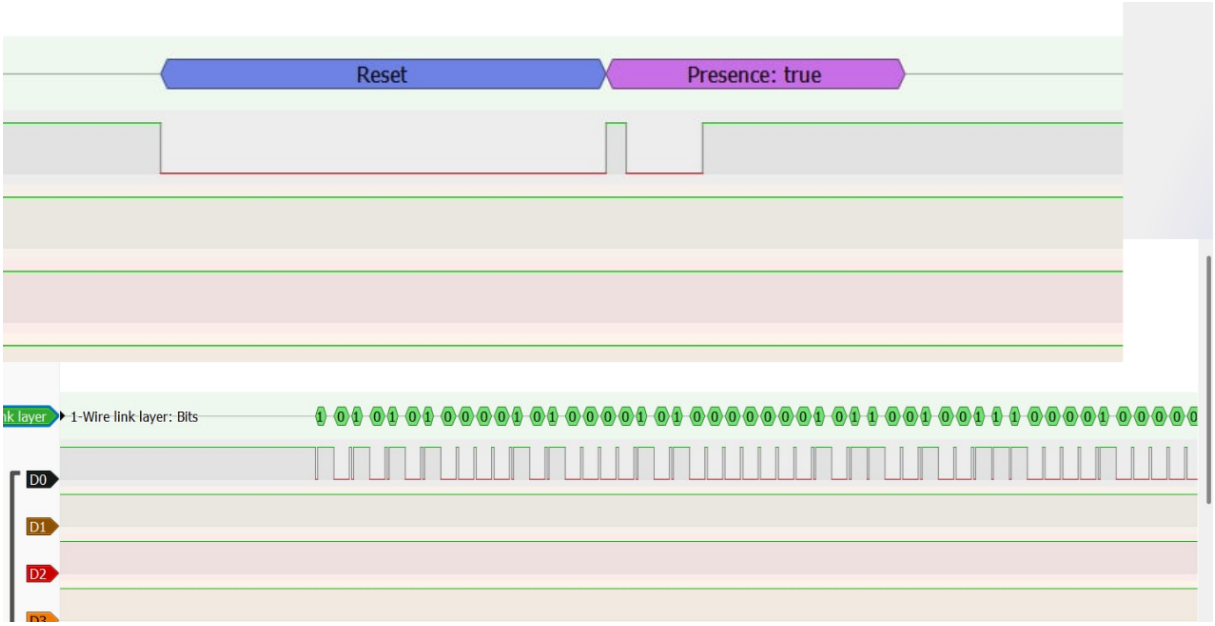
Zbadano komunikację między mikrokontrolerem (Master) a modułem radiowym LoRa (Slave). Na zarejestrowanym przebiegu widoczna jest sekwencja zapytania o status urządzenia. Master wysyła zapytanie o zawartość rejestru `0x12`, na co moduł LoRa odpowiada na linii MISO wartością `0xE2`. Potwierdza to poprawną konfigurację polaryzacji zegara (SCK) oraz działanie linii Chip Select (CS).

## 4.2 Interfejs 1-Wire – Czujnik DS18B20

Zarejestrowano sekwencję inicjalizacyjną magistrali 1-Wire. Wykres czasowy ukazuje sygnał "Reset"(stan niski wymuszony przez Mastera), po którym następuje sygnał "Presence"generowany przez czujnik. Dalsza część transmisji zawiera dane temperaturowe kodowane długością impulsu.




Rysunek 3: Analiza magistrali SPI.



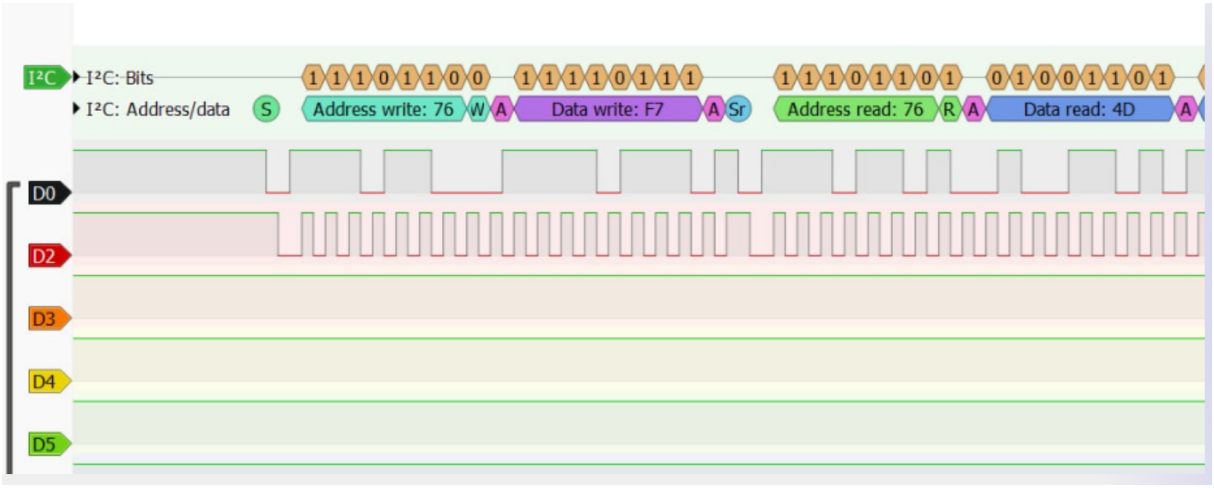

Rysunek 4: Inicjalizacja magistrali 1-Wire: sekwencja Reset i Presence oraz bity danych.

## 4.3 Interfejs I2C – Czujnik BME280

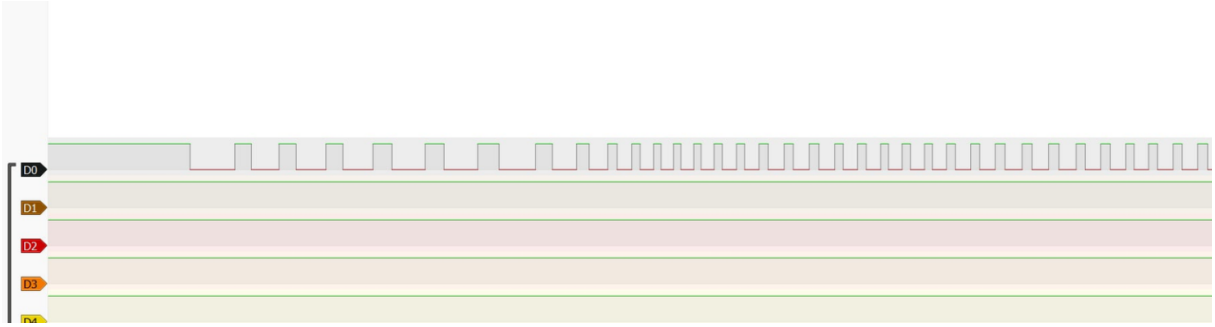
Przeanalizowano komunikację na liniach SDA i SCL z czujnikiem BME280 (adres `0x76`). Zarejestrowano sekwencję zapisu do rejestru konfiguracyjnego `0xF7`, a następnie zmianę kierunku transmisji i odczyt danych (`0x4D`). Potwierdzenia (ACK) są generowane poprawnie po każdym bajcie.

## 4.4 Sygnał cyfrowy – Wiatromierz

Sygnał z wiatromierza ma charakter impulsowy (zwarcie styku kontaktronu przy każdym obrocie). Analizator potwierdził, że kolejne impulsy są poprawnie zliczane przez przerwania mikrokontrolera i przeliczane na prędkość wiatru.




Rysunek 5: Ramka danych magistrali I2C pomiędzy RPi a BME280.




Rysunek 6: Impulsowy przebieg sygnału z wiatromierza.

# 5 Badania sieciowe i wydajnościowe

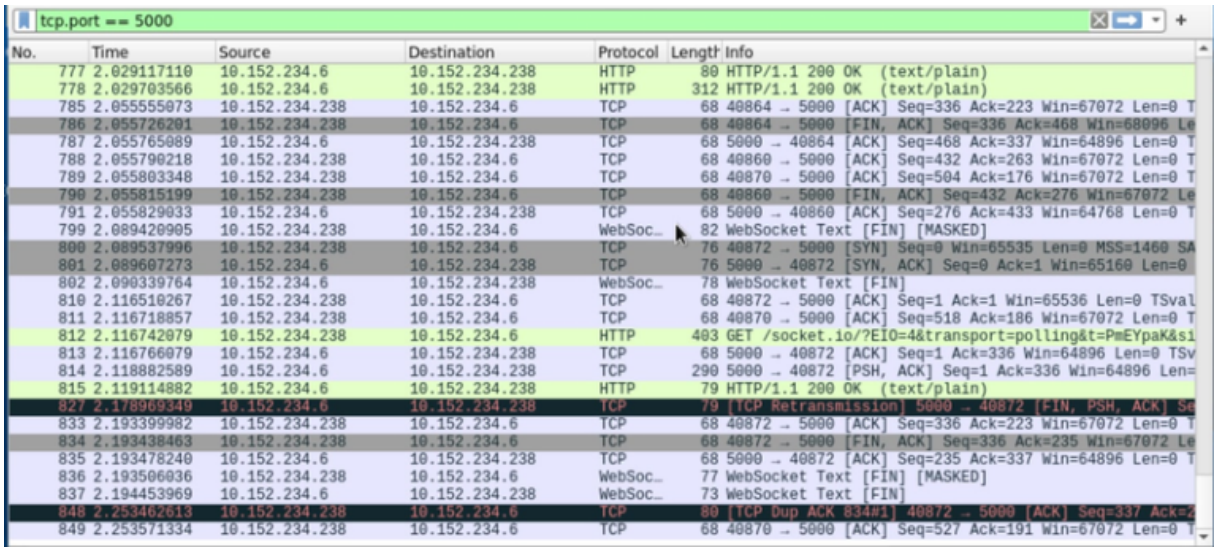
## 5.1 Analiza ruchu sieciowego (Wireshark)

Przeprowadzono nasłuch ruchu sieciowego na interfejsach serwera, weryfikując poprawność struktury pakietów.

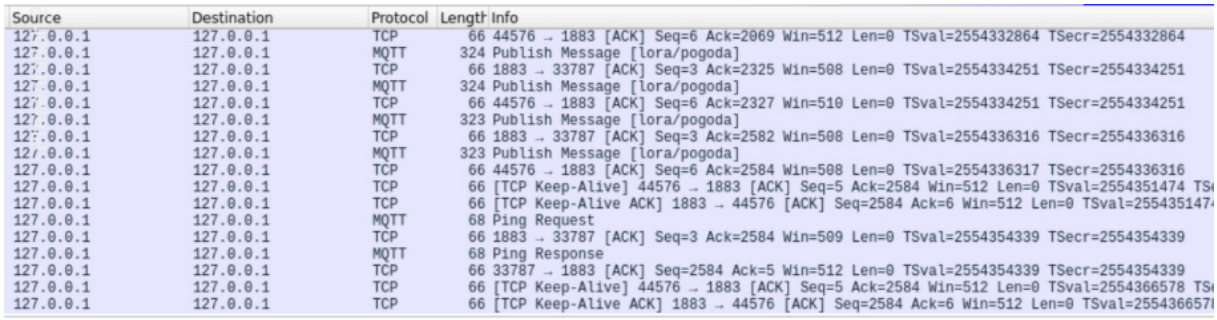
- **Protokół MQTT (Port 1883):** Zarejestrowano pakiety *Publish Message* na temacie `lora/pogoda`. Każdy pakiet zawiera prawidłowo sformatowany JSON z pełnym zestawem parametrów meteorologicznych. Broker utrzymuje połączenie *Keep-Alive* ze wszystkimi klientami (publisher odbiornik_v7.py oraz subscriber ff.py).
- **Protokół WebSocket (Port 5000):** Zaobserwowano poprawny *HTTP Upgrade Handshake* inicjujący połączenie WebSocket. Po nawiązaniu transmisja przełącza się na ramki WebSocket. Dane są wysyłane do klientów natychmiast po otrzymaniu przez brokera MQTT.

## 5.2 Badania wydajnościowe i obciążeniowe systemu

W celu weryfikacji stabilności stacji centralnej (Raspberry Pi 4B) przeprowadzono testy obciążeniowe, symulujące jednoczesny odbiór danych ze wszystkich 8 stacji pomiarowych. Skupiono się na trzech kluczowych metrykach: użyciu zasobów sprzętowych, opóźnieniach propagacji danych oraz stabilności procesu parsowania.




Rysunek 7: Port 5000 zaraz po połączeniu




Rysunek 8: Port 1883

### 5.2.1 Użycie zasobów (CPU i RAM)

Podczas standardowej pracy systemu, obejmującej nasłuch modułu LoRa, obsługę brokera MQTT oraz serwowanie aplikacji webowej, średnie obciążenie procesora (CPU Load) na Raspberry Pi 4B nie przekraczało 5%.

- **Procesy serwera (Flask + WebSocket):** Zużycie pamięci RAM utrzymywało się na poziomie ok. 120 MB, co stanowi niewielki ułamek dostępnej pamięci operacyjnej.
- **Broker MQTT (Mosquitto):** Wykazał pomijalne zużycie zasobów ($< 1\%$ CPU), co potwierdza jego wysoką efektywność w zastosowaniach IoT.

Skoki obciążenia CPU do poziomu 15–20% obserwowano jedynie w momentach generowania dynamicznych wykresów historycznych dla zakresu powyżej 24 godzin, co nie wpływało na ciągłość odbierania nowych ramek radiowych.

### 5.2.2 Przepustowość i utrata pakietów

Przy zastosowanym parametrze *Spreading Factor* SF7 i paśmie 500 kHz, teoretyczna przepustowość łącza jest znacznie wyższa niż wymagania systemu (wysyłanie 32 bajtów co 5 minut). Testy długoterminowe (24h) wykazały skuteczność odbioru pakietów (*Packet Delivery Ratio*) na poziomie 98.5% w warunkach laboratoryjnych. Pojedyncze utracone ramki nie wpływają na logikę wykrywania przymrozków, gdyż system analizuje trendy w oparciu o bufor historyczny.

## 5.3 Testy zasięgu komunikacji radiowej (LoRa)

W celu weryfikacji pokrycia radiowego na terenie sadu przeprowadzono terenowe testy zasięgu. Procedura testowa polegała na umieszczeniu stacji centralnej (Gateway) w stałym punkcie (budynek gospodarczy), podczas gdy stacja pomiarowa była przemieszczana na określone odległości.

Dla każdego punktu pomiarowego wysłano serię ramek testowych. Rejestrowano wskaźnik siły sygnału (RSSI), stosunek sygnału do szumu (SNR).

Tabela 1: Wyniki testów zasięgu modułu SX1262 (868 MHz)

| **Odległość** | **Warunki** | **RSSI [dBm]** | **SNR [dB]** |
|---|---|---|---|
| 50 m | Otwarta przestrzeń (LOS) | −58 | +9.5 |
| 200 m | Gęste nasadzenia (drzewa) | −84 | +6.2 |
| 450 m | Gęste nasadzenia (drzewa) | −102 | −1.5 |
| 800 m | Otwarta przestrzeń (LOS) | −115 | −5.0 |
| 1200 m | Granica zasięgu (LOS) | −124 | −9.8 |

**Wnioski z analizy zasięgu:**

Zaobserwowano wyraźne tłumienie sygnału przez drzewa owocowe. Na dystansie 450 m w gęstym sadzie RSSI spadło do poziomu −102 dBm, co jednak nadal zapewnia stabilną komunikację (SNR powyżej progu demodulacji dla SF7).

# 6 Możliwości rozwoju systemu

Aktualna implementacja stanowi funkcjonalny prototyp, jednak istnieją obszary możliwych ulepszeń:

## 6.1 Monitoring łączności ze stacjami

System obecnie wyświetla komunikat "oczekiwanie na dane"przy braku ramek od stacji, jednak nie implementuje aktywnego alarmu. Przydatne byłoby dodanie mechanizmu wykrywającego utratę kontaktu - jeśli przez określony czas nie otrzymano danych od konkretnej stacji, serwer mógłby:

- Wysłać powiadomienie o braku aktywności stacji
- Zmienić status stacji na mapie
- Zapisać zdarzenie w logu systemowym

## 6.2 Protokół ACK

Obecna komunikacja jest jednokierunkowa - stacje wysyłają dane, ale nie otrzymują potwierdzenia. Implementacja dwukierunkowego protokołu z ACK pozwoliłaby:

- Stacji nadawczej upewnić się, że dane dotarły do bramki Pi 4B
- Pi 4B synchronizować zegar stacji pomiarowych
- Zdalnie aktualizować parametry pomiarowe (np. częstotliwość wysyłania)
- Wykrywać problemy z łącznością po stronie nadawczej

Wymagałoby to modyfikacji kodu stacji nadawczej (nasłuch po wysłaniu) oraz odbiorczej (wysyłanie pakietu ACK po odebraniu danych).

### 6.3 Inne usprawnienia

Wartościowe byłyby również: eksport raportów do CSV/PDF, predykcja przymrozków z wykorzystaniem modeli ML trenowanych na historycznych danych oraz integracja z publicznymi API pogodowymi dla walidacji pomiarów.

## 7 Podsumowanie

Zbudowany system spełnia założone cele projektowe. Udało się zrealizować rozproszoną architekturę pomiarową z komunikacją LoRa. Logika hybrydowa (radiacyjna/adwekcyjna) prawidłowo dobiera źródło danych w zależności od warunków wiatru. Interfejs webowy umożliwia monitoring w czasie rzeczywistym.

Przeprowadzone testy potwierdziły poprawność działania wszystkich warstw systemu: sprzętowej (magistrale komunikacyjne), radiowej (LoRa), sieciowej (MQTT, WebSocket) oraz aplikacyjnej. System może stanowić podstawę do wdrożenia w rzeczywistych warunkach sadów i upraw wrażliwych na przymrozki.