Programowanie sieciowe

Brama komunikacyjna dla urządzeń sensorycznych

Autorzy: *Lider - Gołębiowski Hubert, Pomiankiewicz Bartosz, Rodzik Bartłomiej*
24.04.2023

1. Treść zadania

Zaprojektuj i zaimplementuj system bramy komunikacyjnej dla podsieci do 1024 urządzeń sensorycznych. Urządzenia obsługują jedynie protokół UDP. Użytkownicy łączą się poprzez bramę używając protokołu TCP. Można wzorować się na protokole TFTP (Trivial File Transfer Protocol). Zbiór aktywnych urządzeń i użytkowników może się zmieniać. Zaproponuj sposób ich autoryzacji.

2. Założenia funkcjonalne

- Użytkownik może skorzystać z interfejsu konsolowego
- Użytkownik komunikuje się przez zdefiniowane komunikaty
- Użytkownik może odczytać zawartość rejestru urządzenia
- Użytkownik może wpisać wartość do rejestru urządzenia

3. Założenie niefunkcjonalne

- Obsługa do 1024 urządzeń sensorycznych
- Zmiana aktywnych urządzeń i użytkowników
- Użytkownik może się połączyć jednocześnie z jednym urządzeniem
- Jedno urządzenie może być obsługiwane w danym momencie przez jednego użytkownika

4. Podstawowe przypadki użycia

- Konfiguracja zdalna urządzeń
- Odczytywanie wartości z sensorów (np. system smart home)

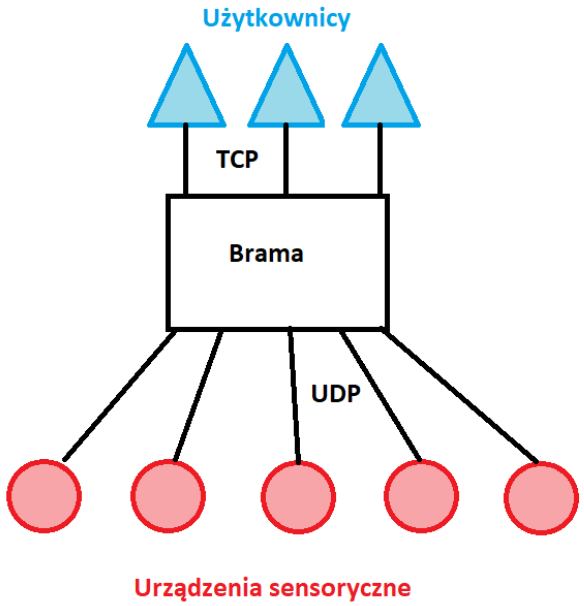
5. Analiza możliwych sytuacji błędnych i proponowana ich obsługa

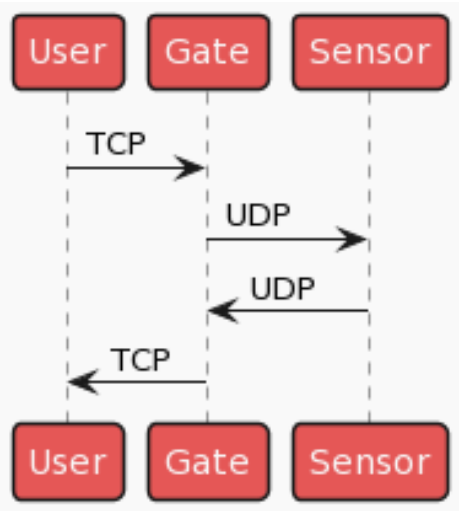
- Pakiet UDP został zgubiony
    - Ponowne wysłanie pakietu.
- Błędne wykrywanie nowych użytkowników i urządzeń
    - Śledzenie nowych użytkowników i sensorów w momencie dołączenia i rozłączenia.
- Problem, które urządzenie jest połączone z którym użytkownikiem
    - Przechowywanie par połączonych użytkowników - urządzeń, sprawdzanie listy przy przesyłaniu pakietów przez bramę.
- Błąd konwersji protokołu
    - Funkcja konwertująca będzie sprawdzała poprawność przekonwertowanego pakietu przed przesłaniem go dalej.
- Błędne komendy w przesyłanych wiadomościach
    - Sprawdzanie przesyłanych argumentów przed przesłaniem ich dalej. Zwracanie informacji o sposobie wykorzystania zaimplementowanych metod.

6. Wybrane środowisko sprzętowo-programowe

Język programowania: python 3.10
System operacyjny: Linux (docker container - bigubu)

7. Architektura rozwiązania





- Moduł odpowiadający za rejestrację dostępnych urządzeń i aktywnych użytkowników
- Moduł odpowiadający za obsługę połączenia z użytkownikiem
  - Sprawdzanie poprawności komunikatów
  - Obsługa błędów
- Moduł odpowiadający za komunikację z urządzeniem
  - Obsługa błędów
- Interfejs konsolowy dla użytkownika
- Logger zapisujący akcje wykonujące się w bramie

8. Ewentualne API

- Socket
- Struct

9. Ewentualne listy komunikatów

- Użytkownik → Brama:
  - Connect [ID sensora] - połączenie z danym urządzeniem
  - Read [rejestr] - odczytanie wartości rejestru sensora
  - Config [rejestr, wartość] - zapisanie wartości do rejestru konfiguracyjnego sensora
- Brama → Użytkownik
  - CONNECTED - udało się połączyć z urządzeniem
  - DIDREAD - zwraca odczytaną wartość
  - DIDCONFIG - udało się zapisać wartość
  - NOTAVAILBLE - urządzenie jest niedostępne
  - COULDNOTREAD - nie udało się odczytać wartości rejestru

- COULDNOTWRITE - nie udało się zapisać wartości w rejestrze
- TIMEOUT - sesja zakończona przez timeout

10. Sposób testowania

Testowanie funkcji niewymagających łączenia się za pomocą protokołów mogą być testowane lokalnie (np. przy pomocy modułu pytest).
Elementy połączeń między modułami będą testowane zarówno lokalnie (localhost - różne porty), jak i przy pomocy docker na serwerze bigubu (docelowe środowisko). Testy te będą wykonywane ręcznie przez użytkownika.

11. Wstępny podział prac w zespole

- Hubert Gołębiowski -
  - Moduł połączenia z użytkownikiem
  - Interfejs użytkownika
- Bartosz Pomiankiewicz
  - Sprawdzanie poprawności komunikatów
  - Rejestracja aktywnych użytkowników i urządzeń
- Bartłomiej Rodzik
  - Moduł połączenia z urządzeniem
  - Logger

12. Przewidywane funkcje do zademonstrowania w ramach odbioru częściowego

- Moduł sprawdzający poprawność komunikatów
- Moduł pozwalający na połączenie z bramą jednego użytkownika i jedno urządzenie