Wstępna analiza funkcjonalna aplikacji „Bardzo mini SCADA”

# 1. Koncepcja i cele projektu

Stworzenie prostego systemu monitorowania i wizualizacji danych.

Projekt składa się z dwóch podstawowych elementów:

1. **Serwer** – generuje i udostępnia w sposób ciągły dane pomiarowe przez sieć (np. poprzez TCP/IP).
2. **Klient** – odbiera dane z serwera i zapewnia ich wizualizację w postaci wykresów.

Aplikacja powinna być:

* **Łatwa w uruchomieniu i obsłudze**
* **Rozszerzalna** – możliwość dodawania własnych metod przetwarzania danych (w formie pluginów lub parsera wyrażeń matematycznych).
* **Elastyczna** w zakresie prezentacji danych – skalowanie osi, zmiana kolorów, rodzajów linii, siatki itp.

# 2. Opis działania

1. **Serwer**
   * Uruchamiany jako osobna aplikacja
   * W określonych odstępach czasu wysyła losowe dane pomiarowe
   * Może przyjmować prostą konfigurację
2. **Klient**
   * Po uruchomieniu umożliwia wybranie adresu i portu serwera.
   * Klient łączy się z serwerem, a następnie cyklicznie odbiera dane pomiarowe.
   * Dane są gromadzone w wewnętrznym buforze – tak, by można było:
     + wyświetlić wykres w czasie rzeczywistym,
     + przeglądać dane historyczne (ewentualnie w ograniczonym zakresie, np. ostatnie 1000 próbek).
   * Przed wyświetleniem użytkownik może zastosować wybrane przetwarzanie danych
   * Dane są wyświetlane na wykresie w czasie rzeczywistym. Użytkownik może dostosować parametry widoku:
     + zakres osi X i osi Y,
     + siatkę (włączenie/wyłączenie, gęstość),
     + kolory linii, rodzaj linii,
3. **Buforowanie i opcje historyczne**
   * Klient przechowuje w pamięci pewną liczbę ostatnio odebranych próbek (np. 1000 czy 5000) w celu umożliwienia przewinięcia wykresu.
   * Po osiągnięciu limitu starsze dane są usuwane w trybie FIFO (first in, first out).
4. **Struktura klas (szkic)**
   * **DataServer**
     + odpowiada za generowanie i wysyłanie danych (odpowiedzialny za socket, wysyłanie w pętli).
   * **DataClient**
     + odpowiada za nawiązywanie połączenia z serwerem i odbiór danych.
     + udostępnia metody do buforowania danych i podstawowe API dla modułu wizualizacji.
   * **DataProcessor** (interfejs lub klasa bazowa)
     + w przypadku wtyczek: klasa bazowa, z której dziedziczą konkretne filtry/przetwarzania (np. FilterSmooth).
     + w przypadku parsera: klasa zawierająca mechanizm interpretacji wyrażeń matematycznych.
   * **PlotWidget** (moduł wyświetlania)
     + zarządza renderowaniem wykresów.
     + umożliwia konfigurację wyglądu (kolory, zakresy, siatka).
   * **MainWindow** (w przypadku interfejsu GUI)
     + obsługuje interakcje użytkownika (menu, przyciski, panele konfiguracji).

# Szkic interfejsu użytkownika

-----------------------------------------------------------

| [Menu] [Połącz z serwerem] [Konfiguracja wtyczek] ... |

-----------------------------------------------------------

| | |

| Lista | Wykres danych (PlotWidget) |

| dostępnych |------------------------------------------- |

| filtrów | [Opcje wyświetlania] |

| | - Skala osi X, Y |

| | - Kolor linii, rodzaj linii |

| | - Włącz/wyłącz siatkę |

---------------------------------------------------------

| Status: Połączono / Brak połączenia |

---------------------------------------------------------

# 4. Technologia i środowisko

* **Kompilator / Środowisko**:
  + **MinGW**.
  + IDE: **Qt Creator**
* **Biblioteki**:
  + Do obsługi sieci: standardowe gniazda (sockets) w C++ lub rozwiązania z Qt (QTcpServer).
  + Biblioteka do rysowania wykresów:
    - **QChart** lub **QCustomPlot**
  + W przypadku parsera – exprtk
* **System operacyjny**: Windows