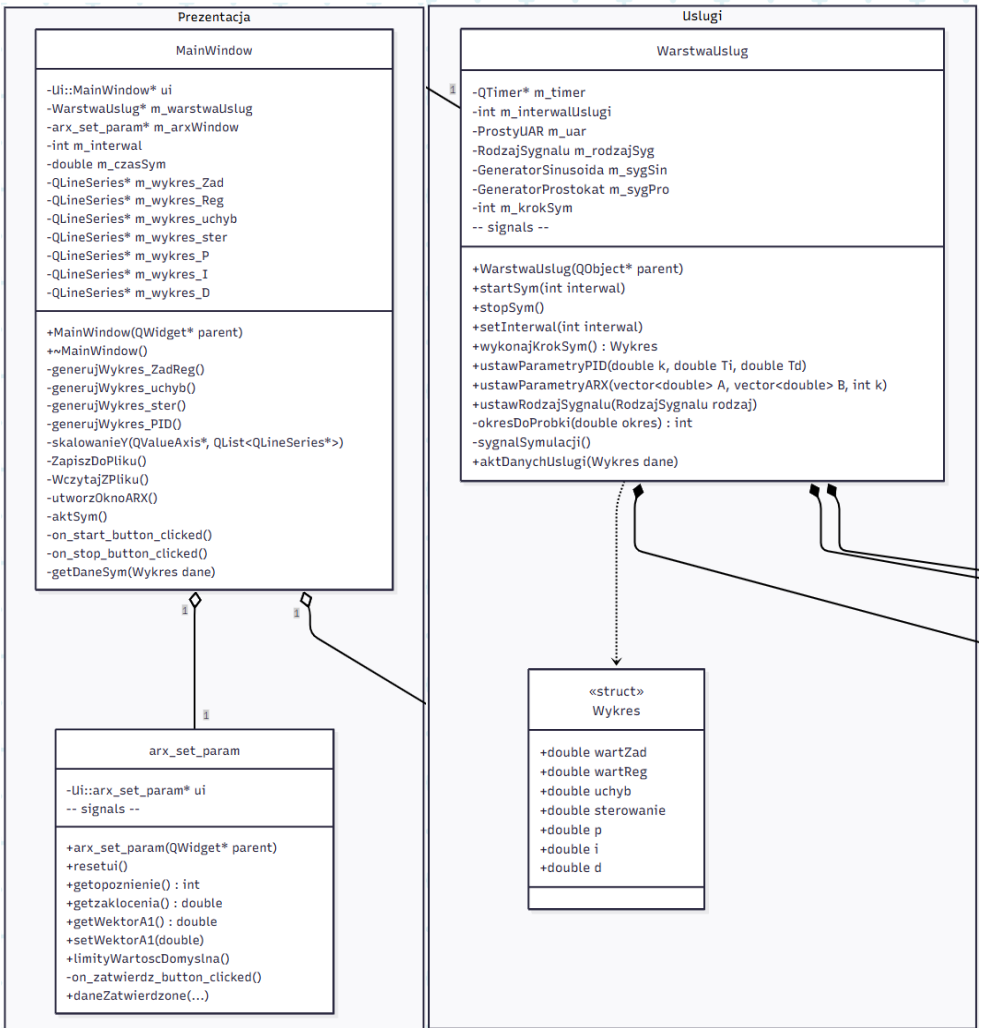
Sprawozdanie z projektu

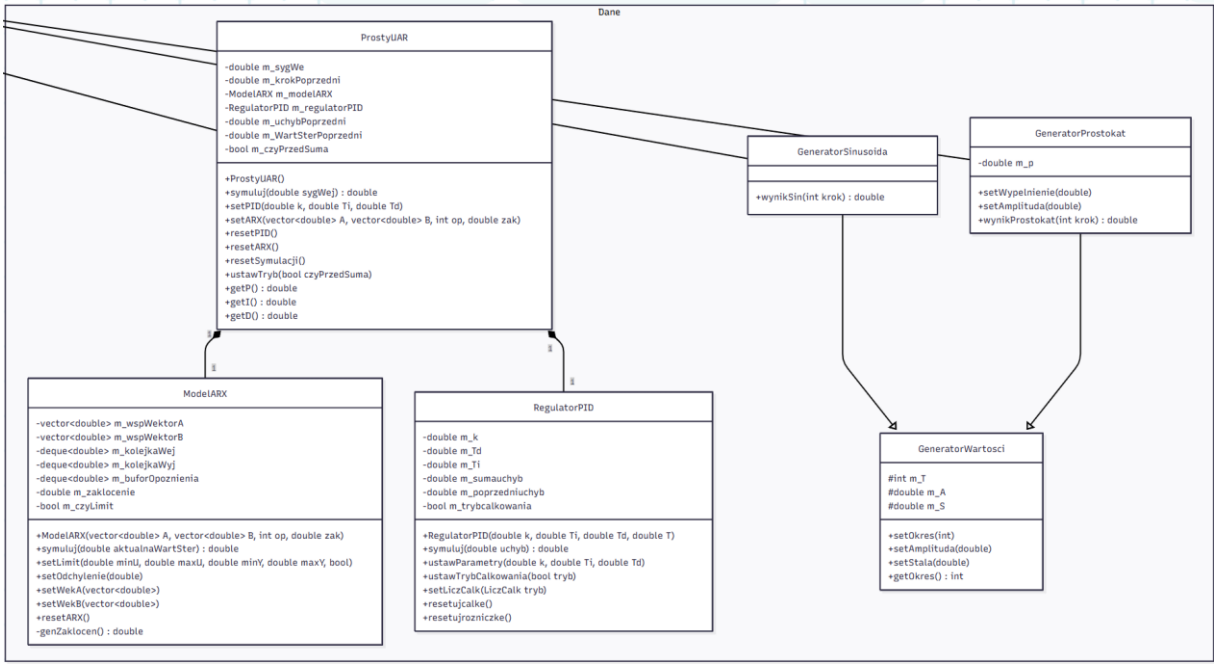
Nazwa przedmiotu				
Programowanie komputerów				
Rok akademicki	Kierunek	Prowadzący	Semestr	Sekcja
2025/26	IPpp	Dr inż. Łukasz Maliński	3	1
Skład sekcji				
Kacper Skowroński		Igor Juraszek		

1. Podział obowiązków

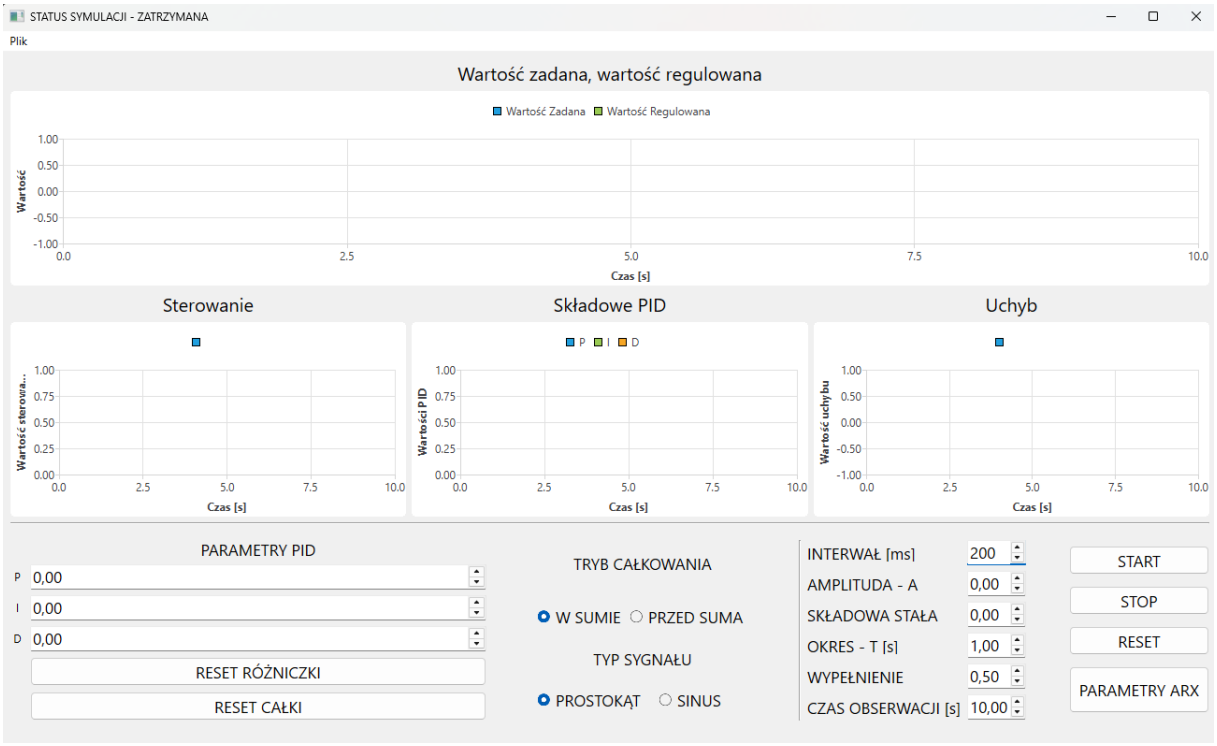
Kacper	Igor
Implementacja logiki modelu ARX, Implementacja warstwy usług, Utworzenie i generowanie wykresów, Skalowanie wykresów, Code review - zarządzanie repozytorium oraz zatwierdzanie zmian, Przygotowywanie prezentacji z etapów pracy,	Warstwa prezentacji (GUI), Implementacja regulatora PID, Zapis i odczyt konfiguracji, Testy jednostkowe, Pomoc w przygotowaniu prezentacji,

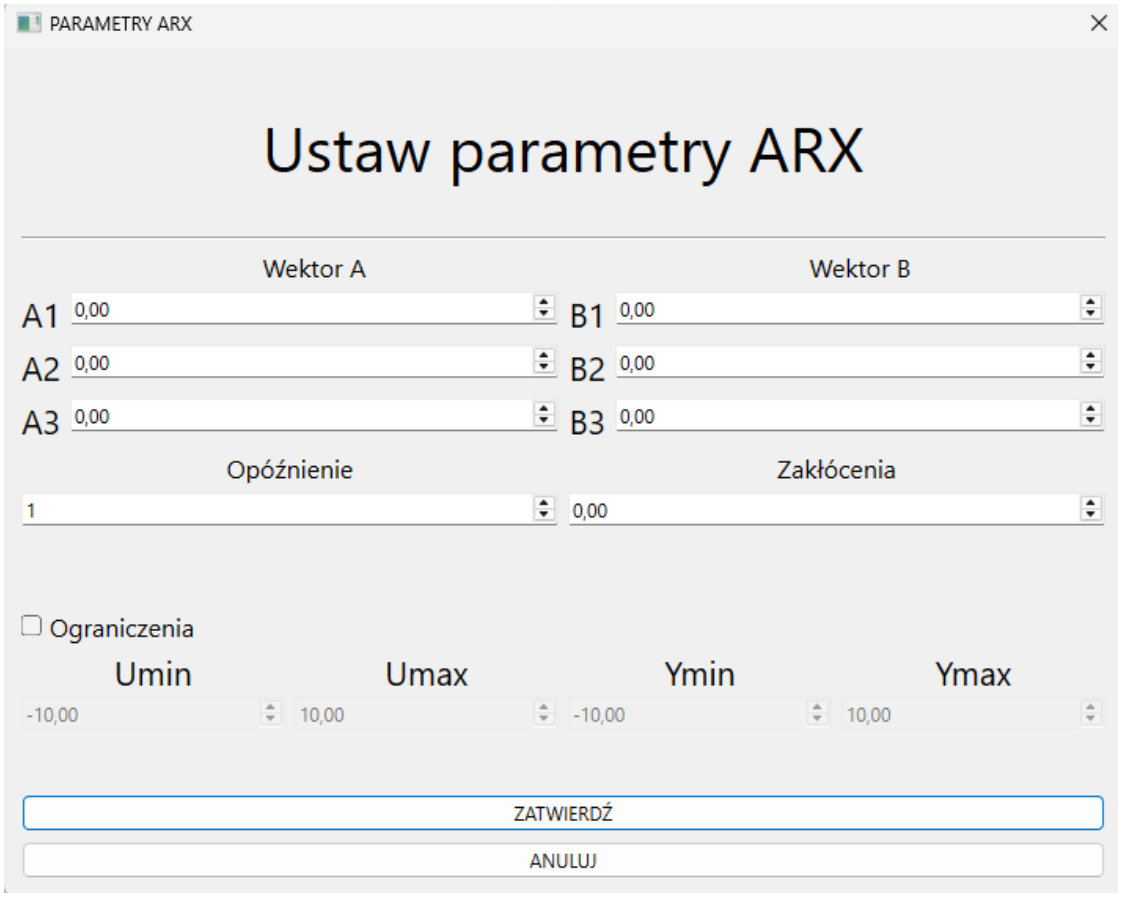
2. Schemat UML





3. Końcowy wygląd GUI





4. Historia rozwoju aplikacji

Rozwój aplikacji przebiegał w kilku etapach, podczas których ewoluowały zarówno założenia architektoniczne, jak i wizualne:

1. Implementacja logiki Back-end:

   Co zrobiono: Etap wstępny polegał na implementacji czystej logiki w C++ bez użycia bibliotek graficznych. Stworzono podstawowe klasy modelu ARX oraz regulatora PID.

   Cel: Weryfikacja poprawności algorytmów sterowania i równań różnicowych za pomocą testów jednostkowych w izolowanym środowisku.

2. Wdrożenie architektury trójwarstwowej:

   Zmiana: Wprowadzono klasę Warstwy Usług, która stała się pośrednikiem między logiką a interfejsem.

   Przyczyna: Konieczność odseparowania obliczeń symulacyjnych od GUI, co umożliwiło łatwiejsze zarządzanie cyklem życia obiektów i późniejszą integrację z Qt.

3. Integracja z frameworkiem Qt i debugowanie:

   Problem i zmiana: Po przeniesieniu kodu do środowiska Qt napotkano problemy z inicjalizacją zmiennych (opisane w sekcji trudności), co wymusiło poprawki w konstruktorach klas logiki.

   Efekt: Stabilizacja działania aplikacji w trybie Release.

4. Ewolucja Interfejsu Użytkownika (GUI):

   Zmiana: Odeszliśmy od pierwotnych założeń kwadratowego układu okna głównego na rzecz układu panoramicznego.

   Przyczyna: Uznano, że panoramiczny układ wykresów jest znacznie bardziej czytelny dla użytkownika i pozwala na lepszą analizę przebiegów w czasie rzeczywistym. Finalna wersja aplikacji jest efektem dostosowania UI do ergonomii pracy z wykresami

5. Napotkane trudności

Błąd opóźnienia transportowego w modelu ARX.

Problem: Podczas testów skoku jednostkowego zauważono, że model reagował o jeden krok symulacji za wcześnie - reakcja następowała natychmiastowo, ignorując zadane opóźnienie k = 1.

Rozwiązanie: Przeanalizowano kolejność operacji w metodzie symuluj(). Błąd wynikał z aktualizacji bufora historii przed pobraniem wartości do obliczeń. Zmieniono logikę na sekwencję: najpierw odczyt wartości z końca bufora opóźnienia (dla obliczeń), a dopiero potem wstawienie nowej próbki sterowania na początek bufora.

Testy jednostkowe w Qt.

Problem: Testy jednostkowe, które przechodziły poprawnie w środowisku Visual Studio, zaczęły zgłaszać błędy po przeniesieniu do Qt. Objawiało się to losowym działaniem limitów (nasycenia) w regulatorze PID i modelu ARX, mimo że w kodzie nie wprowadzano zmian logicznych.

Rozwiązanie: Debugowanie wykazało, że zmienne typu bool (np. m_czyLimit) nie były jawnie inicjalizowane w konstruktorze. W trybie Debug kompilator często zerował pamięć (false), natomiast w Release zawierała ona losowe wartości ("śmieci"), co aktywowało limity w nieoczekiwanych momentach. Dodanie listy inicjalizacyjnej w konstruktorze, rozwiązało problem i testy osiągnęły wynik 24/24.

Problem z niepełnymi wykresami.

Problem: Błędna logika obliczania próbek, wykres rysował się tylko na końcu a reszta osi była pusta. Wykres usuwał stare punkty zbyt szybko przez co widoczne było tylko 2,5 sekundy na 10 sekundowej osi.

Rozwiązanie: Zmiana sposobu obliczania próbek. Zastosowaliśmy wzór, który oblicza limit próbek w zależności od aktualnego interwału, dzięki czemu wykres jest zawsze wyświetlany w całej szerokości.

Problem pamięci regulatora po resecie.

Problem: Po zresetowaniu wykresów i ponownym uruchomieniu symulacji, regulator PID startował z danymi z poprzedniej sesji. Działo się tak, ponieważ zmienna przechowująca sumę uchybów nie była zerowana. Powodowało to gwałtowny skok sterowania na starcie nowej symulacji.

Rozwiązanie: Rozbudowa metody resetSymulacji(). Oprócz czyszczenia serii danych na wykresach, dodano wywołanie metody w warstwie usług, która resetuje wewnętrzne zmienne stanu regulatora.

6. Podsumowanie - czego się nauczyliśmy

Kacper	Igor
Nauczyłem się analizować cudzy kod, wykrywać potencjalne konflikty i dbać o spójność stylu w całym projekcie przed scaleniem gałęzi. Zrozumiałem, jak kluczowe jest, aby warstwa usług nie tylko przekazywała dane, ale też zarządzała cyklem życia obiektów symulacyjnych, izolując je od interfejsu użytkownika. Problem z niestabilnością w trybie Release nauczył mnie, że w C++ nie można polegać na domyślnym zachowaniu kompilatora i należy inicjalizować wszystkie zmienne składowe klas.	Praktyczne zastosowanie architektury trójwarstwowej - umiejętność separacji warstwy prezentacji (GUI) od logiki i danych, Projektowanie interfejsów graficznych (GUI) - tworzenie responsywnych interfejsów w bibliotece Qt, ze szczególnym uwzględnieniem wizualizacji danych w czasie rzeczywistym, Tworzenie testów jednostkowych - weryfikacja poprawności kluczowych algorytmów sterowania i logiki aplikacji poprzez pisanie testów, Integracja warstw aplikacji - łączenie logiki obliczeniowej z interfejsem użytkownika z wykorzystaniem mechanizmu sygnałów i slotów oraz zarządzanie przepływem danych między obiektami.