# SCADA 2

Maciej Cebula
Piotr Merynda
Maciej Podsiadło

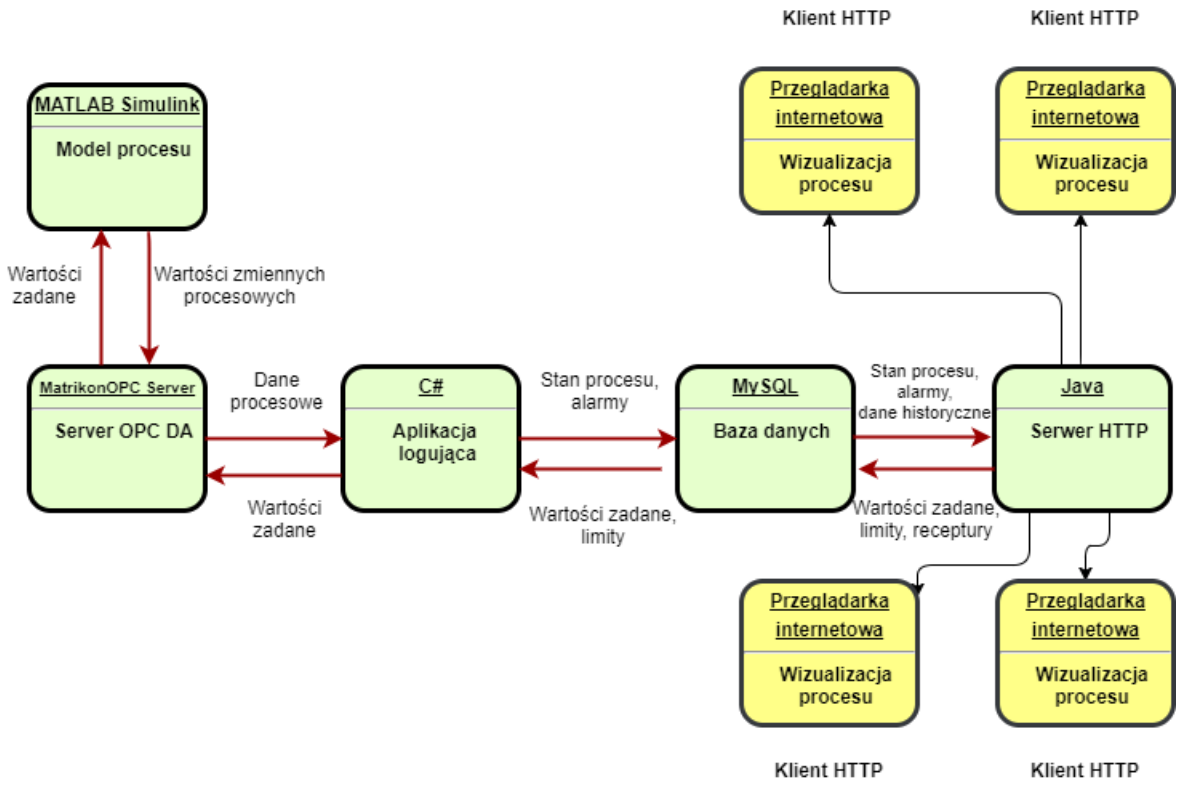
# Spis treści

# Wstęp

W dobie przemysłu 4.0 projektowane systemy mają za zadanie nie tylko sterować wybranym procesem, ale także zapewniać jego intuicyjną wizualizację, udostępniać dane innym systemom oraz inteligentnie te dane przetwarzać. Pewnego rodzajem standardem stało się włączenie systemów przemysłowych do internetu. Dzięki takiemu podejściu końcowy użytkownik może z powodzeniem podejrzeć, a nawet wysterować obiekt znajdujący się nawet na drugim końcu świata. Inżynierowie implementujący tak rozbudowane systemu natrafiają na trudności w zintegrowaniu wielu urządzeń pochodzących od różnych producentów, na których zainstalowane są różne systemy operacyjne. Wygodnym rozwiązaniem wydaję się być w takiej sytuacji wykorzystanie technologii webowej, która cieszy się ogromną popularnością.

## 1.1 Opis projektu

W niniejszym projekcie stworzono wielopoziomowy system sterowania linią dozującą napój. Na rysunku 1.1 przedstawiony został diagram obrazujący ideę działania całego systemu.



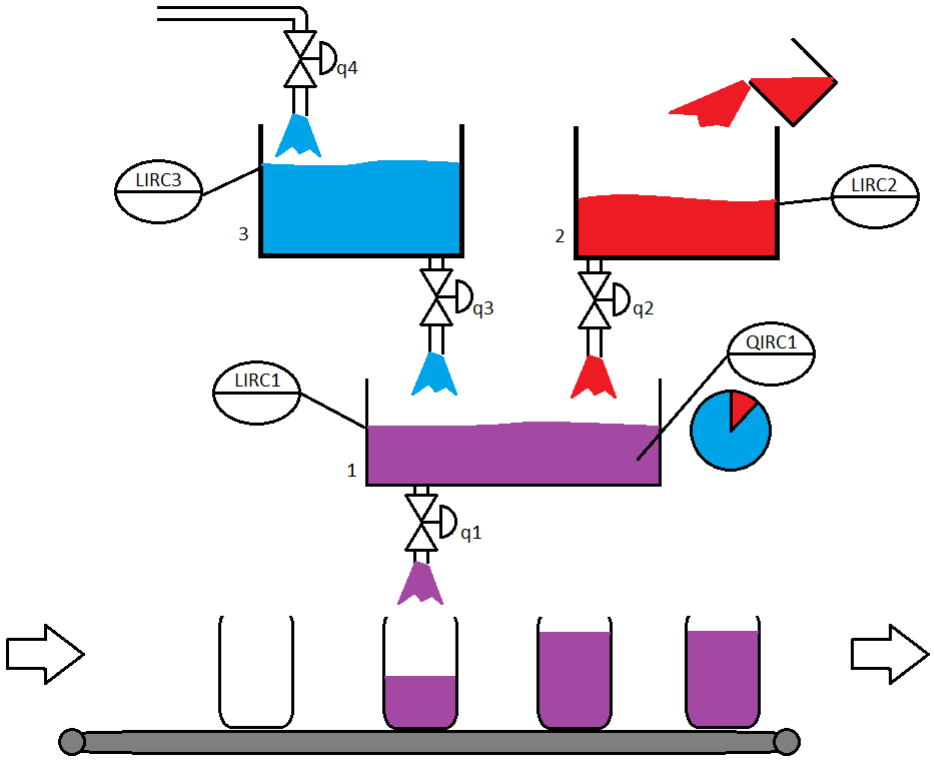

Rys. 1.1: Diagram wielopoziomowego systemu sterowania.

Proces mieszania cieczy oraz napełniania butelek zasymulowany został w środowisku MATLAB Simulink. Aktualny stan procesu wysyłany jest na serwer OPC. Dedykowana aplikacja loguje dane z serwera do bazy danych. Serwer HTTP zapewnia klientom wizualizację danych oraz możliwość wysterowania obiektu z poziomu dowolnej przeglądarki internetowej. Każda z wyodrębnionych na rysunku 1.1 części oprogramowania może pracować niezależnie na różnych komputerach PC, natomiast trudno wyobrazić sobie praktyczny sens takiego rozwiązania. Proponuje się, aby oprogramowanie zaznaczone kolorem zielonym pracowało po stronie serwera, oferując wielu klientom dostęp do wizualizacji bieżącego stanu systemu wraz z systemem alarmowania oraz możliwością wysterowania proces zdalnie, przez internet.

# Model Systemu

## 2.1 System zbiorników

Przedmiotem rozważań jest system 3 połączonych zbiornkiów. Dwa zbiorniki pełnią rolę rezerwy wody oraz koncentratu. W trzecim odbywa się mieszanie gotowego produktu o zadanym stężeniu. Gotowa mieszanka dozowana jest w odpowieniej ilości do pojemników na taśmie produkcyjnej. Regulacja przepływu pomiędzy zbiornikami odbywa się przy pomocy sterowanych zaworów. Poglądowy schemat procesu znajduje się na rysunku 2.1.




Rys. 2.1: Schemat procesu.

Dynamika procesu do celów symulacji została zamodelowana przy pomocy równań różniczkowych. Zależność prędkości wypływu ze zbuornika od wyskości cieczy została przybliżona zależnością pierwiastkową. Dodatkowo może być ona steowana poprzez stopień otwarcia

zaworu. Ostatecznie równiania uzyskują postać:

$$\frac{dH_3}{dt} = \frac{1}{\beta(H_3)}q_4 v_4 - \frac{1}{\beta(H_3)}q_3\sqrt{H_3}$$
$$\frac{dH_2}{dt} = \frac{1}{\beta(H_2)}K - \frac{1}{\beta(H_2)}q_2\sqrt{H_2}$$
$$\frac{dH_1}{dt} = \frac{1}{\beta(H_1)}q_3\sqrt{H_3} + \frac{1}{\beta(H_1)}q_2\sqrt{H_2} - \frac{1}{\beta(H_1)}q_1\sqrt{H_1},$$

*gdzie* :

$H_i$−poziom w $i$ -tym zbiorniku

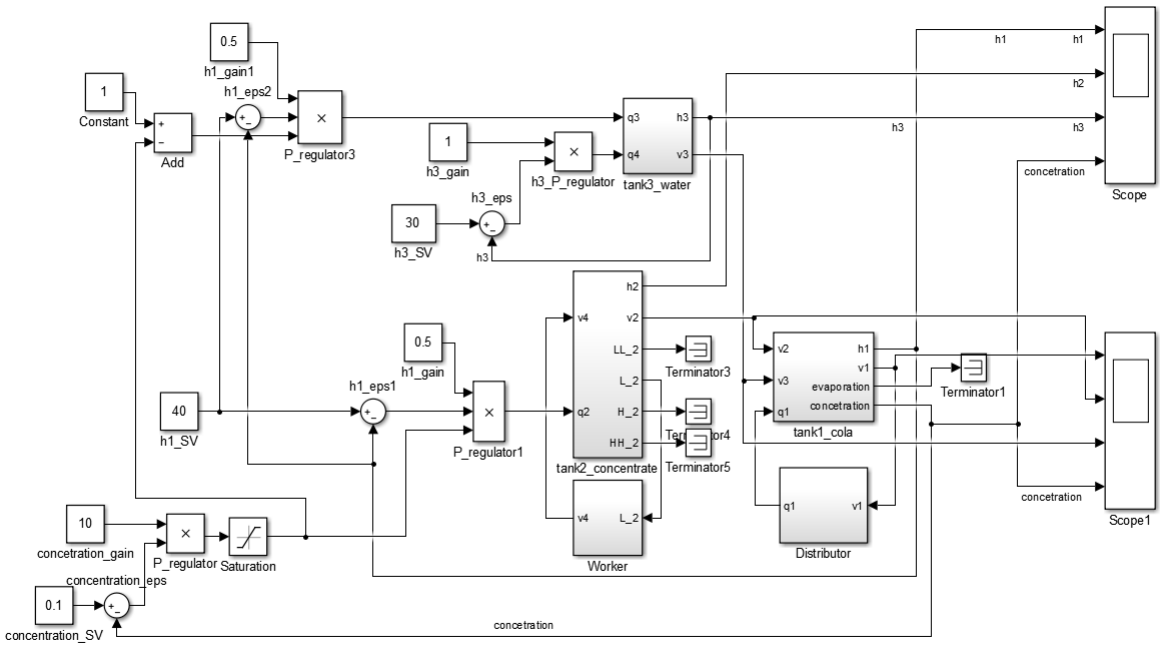
$\beta(H_i)$−powierzchnia swobodna w $i$ -tym zbiorniku dla poziomu $H_i$, w przypadku zbiorników o stałym przekroju $\beta(H_i) = const$

$q_i$−stopień otwarcia $i$ -tego zaworu

$v_4$−strumień wody zasilającej zbiornik 3

$K$−dopływ koncentratu do zbiornika 2

Na podstawie modelu matematycznego utworzono symulację w środowisku Matlab Simulink. Układ modelu symulacyjnego przedstawiony jest na rysunku 2.2.

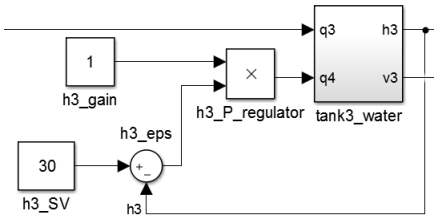



Rys. 2.2: Schemat procesu w środowisku Simulink.

## 2.2 Regulatory

### 2.2.1 Regulacja poziomu wody w zbiorniku 3

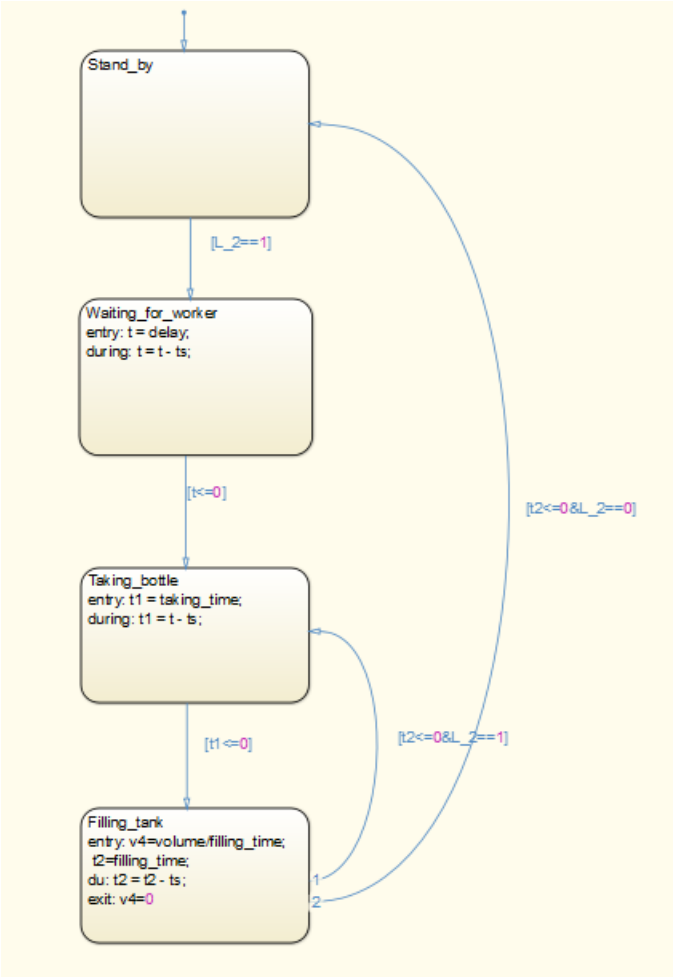
Regulacja poziomu wody w zbiorniku $H_3$ odbywa się poprzez regulator proporcjonalny sterujący stopniem otwarcia zaworu $q_4$. Wartość sterowania wyliczana jest na podstawie uchybu pomiędzy wartością zadaną h3_SV a wysokością poziomu wody w zbiorniku h3.




Rys. 2.3: Schemat regulatora poziomu wody $H_3$.

### 2.2.2 Regulacja poziomu koncentratu w zbiorniku 2

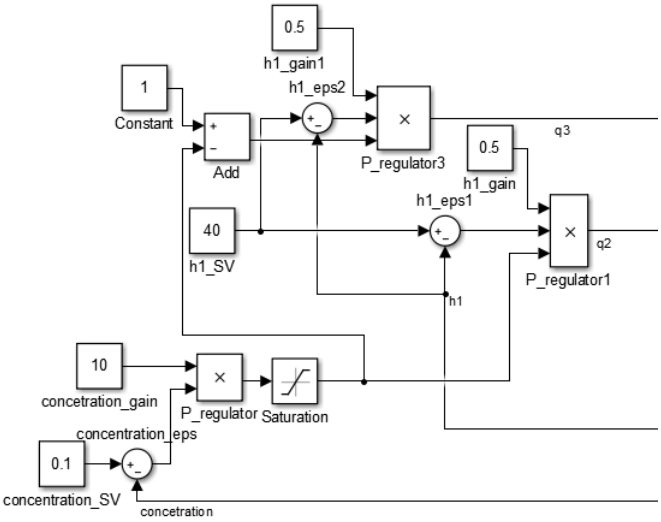
Regulacja poziomu koncentratu w zbiornkiu 2 odbywa się poprzez dolewanie dodatkowych porcji substancji przez pracownika po zgłoszeniu przez system komunikatu o niskim poziomie cieczy w zbiorniku. Aby zamodelować działanie pracownkia, który posiada pewną zwłokę w wykonywaniu działań oraz potrzebuje czasu na przemieszczenie się z dodatkową porcją koncentratu, użyto maszyny stanów. Określono czas reakcji pracownika na komunikat systemowy, czas potrzebny do zabrania kolejnej porcji, oraz czas potrzebny na uzupełnienie koncentratu. Schemat działania pokazano na rysunku 2.4.




Rys. 2.4: Schemat maszyny stanów imitującej działanie pracownika

### 2.2.3 Regulacja dawkowania wody i koncentratu do zbiornika mieszającego

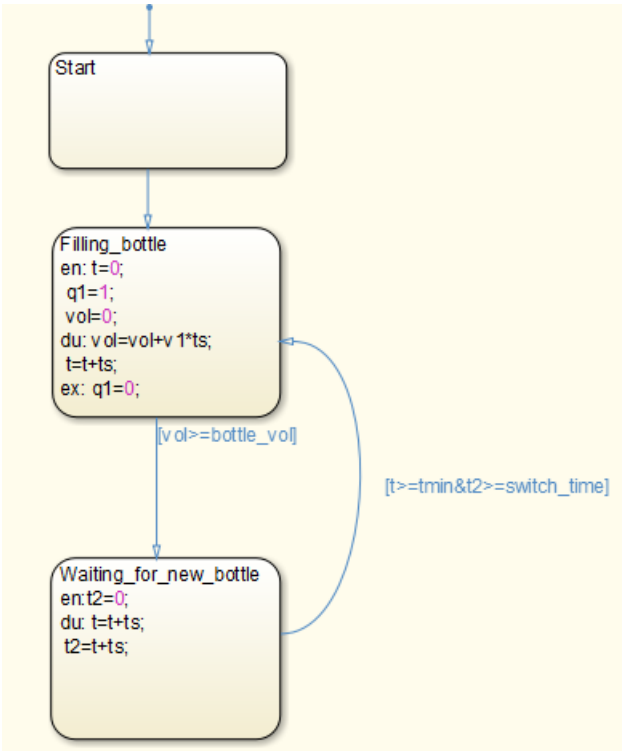
Regulacja przepływu wody oraz koncentratu do zbiornika mieszającego odbywa się na podstawie pomiaru wysokości cieczy w zbiorniku 1 oraz na podstawie pomiaru stężenia koncentratu w gotowym produkcie. Regulacja odbywa się w taki sposób aby utrzymać zadany poziom w zbiorniku oraz zadane stężenie gotowego produktu. Schemat działania regulatorów przedstawiony został na rysunku 2.5. W celu utrzymania zadanych wartości stężenia oraz poziomu $H_1$, wykorzystano regulację proporcjonalną dla stopnia otwarcia zaworów $q_2$ i $q_3$ oraz czynnik proporcjonalny od uchybu stężenia przełączający udział każdego ze sterowań.




Rys. 2.5: Schemat regulatora stężenia i poziomu wody $H_1$.

### 2.2.4 Regulacja dawkowania gotowej mieszanki ze zbiornika mieszającego

Regulator odpowiedzialny za napełnianie pojemników gotową mieszanką działa poprzez maksymalne otwarcie wypływu ze zbiornika 1 aż do całkowitego napełnienia się naczynia. Po całkowitym napełnieniu zawór zostaje zamknięty aż do nadejścia kolejnego pojemnika do napełnienia. Jego działanie zostało zaimplementowane przy użyciu maszyny stanów której schemat widnieje na rysunku 2.6.




Rys. 2.6: Schemat maszyny stanów imitującej działanie dystrubutora

# Serwer HTTP

## 3.1 Opis działania

W celu umożliwienia komunikacji aplikacji wizualizującej z bazą danych napisano w języku *JAVA* serwer HTTP. Dane pomiędzy serwerem i aplikacją wymieniane są za pomocą protokołu HTTP, wykorzystując w tym celu zapytania typu *GET* i *POST*. Wszystkie żądania są utożsamione z odpowiednim adresem url i odpowiednio przetwarzane po stronie serwerowej. Aby zapewnić jak największą responsywność aplikacji oraz synchroniczne odświeżanie danych każdy request po stronie serwera przetwarzany jest w osobnym wątku.
Do napisania serwera wykorzystano następujące biblioteki *JAV-wy*:

1. **RXJava** - biblioteka zapewniająca mechanizmy asynchronicznego przetwarzania danych [4],
2. **JDBI** - biblioteka zapewniająca połączenie z bazą danych [3],
3. **AKKA-HTTP** - framework do implementacji serwera HTTP [2],
4. **Javax mail** - biblioteka do wysyłania emaili.

## 3.2 Uruchomienie serwera

W momencie pisania sprawozdania w repozytorium  nie udostępniono jeszcze skompilowanej wersji flików źródłowych. Dlatego też w celu uruchomienia aplikacji na komputerze wymagane jest:

1. zainstalowanie Jav-y w wersji 8
2. zainstalowanie dowolnego środowiska programistycznego *Intellij, Eclipse itp.*
3. skompilowanie i uruchomienie całego projektu dostępnego pod adresem *https://github.com/maciekc/SCADA2*

# Baza danych

## 4.1 Wstęp

Aby móc wizualizować przebieg pracy systemu wszystkie najważniejsze dane z punktu widzenia automatyki są logowane w bazie danych. Do tego celu wykorzystano bazę danych typu *MYSQL* i środowisko *MYSQL Workbench*, zainstalowane na jednym z komputerów, którym stworzono całą strukturę przedstawioną na rysunku 4.1. Dostęp poszczególnych węzłów systemu do bazy danych zapewniony jest poprzez połączenie wszystkich komputerów w lokalną sieci.

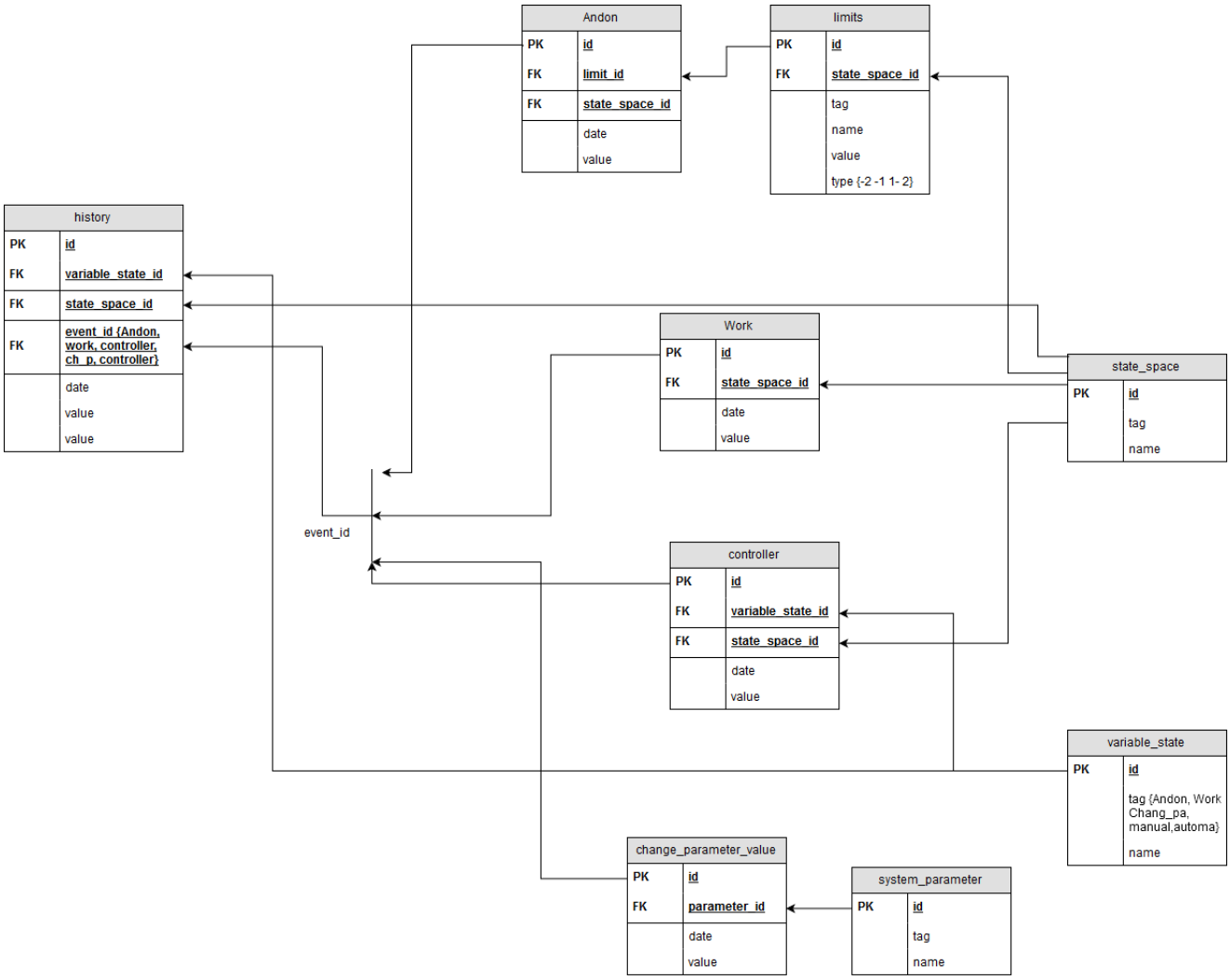
## 4.2 Struktura bazy

Zaprezentowana na rysunku baza danych na strukturę relacyjną aby zapewnić możliwość ewentualnej rozbudowy sytemu o inne składowe. Z racji na to, że baza służy głównie do logowania pracy systemu podzielono ją na następujące części:

1. główną tabele **history**, w której logowane są wszystkie zdarzenia w kolejności chronologicznej,
2. tabelę **Andon**, w której zapisywane są wszystkie sytuacje alarmowe,
3. tabelę **Work**, do której logowany jest stan normalnej pracy systemu,
4. tabelę **change_parameter**, w której zapisywane są wszelkie zmiany wartości parametrów systemu,
5. tabelę **controller**, która służy do logowania pracy regulatorów.

Dodatkowo:

1. w tabeli **state_space** zdefiniowano wszystkie zmienne stanu występujące w systemie,
2. tabele **variable_state** definiuje wszystkie możliwe stany danej zmiennej np. Work, Andon, Change parameter itd.
3. tabela **limits** definiuje poszczególne limity i przypisuje je do odpowiednich zmiennych stanu z tabeli **state_space**,
4. w tabeli **system_parameter** zdefiniowane są pozostałe parametry systemu takie jak: nastawy poszczególnych regulatorów, wartości zadane itp.




Rys. 4.1: Struktura bazy danych.

## 4.3 Opis dostępu do bazy

Dane dotyczące parametrów systemu w poszczególnych chwilach czasu są logowane przez serwer OPC i następnie prezentowane w aplikacji.

## 4.4 Serwer OPC

### 4.4.1 Wprowadzenie

Systemy typu PLC – SCADA są powszechnie wdrażane poczynając od przemysłu chemicznego, a kończąc na automatyce budynków. Stanowią pewną normę w nowoczesnych zakładach oraz fabrykach. W związku z bogatą ofertą producentów aparatury automatyzacji powstał problem komunikacji pomiędzy różnymi komponentami. Firmy promowały swoje rodzime protokoły przemysłowe, co wymuszało na końcowych użytkownikach stosowanie sprzętu pochodzącego od tego samego producenta w obrębie całego obiektu. Przełomem okazało się wprowadzenie otwartego rozwiązania – standardu OPC. OPC jest standardem umożliwiającym komunikację pomiędzy sterownikami (najczęściej PLC) a oprogramowaniem SCADA. Bazuje na modelu klient-serwer, przy czym strona serwera zaimplementowana jest w oprogramowaniu dostarczanym przez producentów sterowników, natomiast klientem - aplikacja wykorzystująca udostępniane dane.

### 4.4.2 MatrikonOPC

Dla celów zintegrowania omawianego systemu zastosowano testowy serwer udostępniony przez firmę Matrikon służący do celów niekomercyjnych. Dostawca oferuje serwer OPC (OPC Simulation Server), jak również oprogramowanie do zarządzania odczytywanymi danymi (OPC Explorer).

### 4.4.3 Logowanie OPC - MySQL

Kolejny element stworzonego wielopoziomowego systemu sterowania stanowi aplikacja logująca aktualne dane pochodzące z serwera OPC do procesowej bazy danych. Aplikacja napisana została w języku C#. Do komunikacji z serwerem użyto open sourcową bibliotekę TitaniumAS.Opc (https://github.com/titanium-as/TitaniumAS.Opc.Client). Serwer lokalizowany jest jedynie po nazwie, co, zgodnie z ideą OPC, znacząco przyspiesza proces integracji. Aplikacja odczytuje bieżące pomiary z serwera OPC z zadaną przez użytkownika częstotliwością, a następnie loguje je w bazie danych MySQL. Podczas tej operacji aktualne wartości porównywane są z progami alarmowymi zadanymi przez użytkownika. Ewentualne alarmy zapisywane są do bazy danych. Podstawową częstotliwością odpytywania jest 1 sekunda, tak jak to ma miejsce w większości systemów typu SCADA. Poszczególne pomiary identyfikowane są po tagach jakie zostały im nadane w momencie inicjalizacji w serwerze OPC. Zapis do bazy danych odbywa się według konwencji narzuconej w momencie jej zaprojektowania. Aplikacja zapewnia mechanizmy przechwytywania zgłaszanych błędów, w szczególności błędów połączenia z serwerem oraz MySQL. Użytkownik informowany jest o napotkanym błędzie. Program dokonuje niezbędnej konwersji sposobu zapisu liczb zmiennoprzecinkowych, zamienia separator ‘,’ na ‘.’ bez czego zapis do bazy danych nie byłby możliwy.

Warto nadmienić, że użytkownik operujący na systemie Windows nie musi instalować, żadnego dodatkowego oprogramowania w celu uruchomienia omawianej aplikacji logującej.

# Aplikacja

## 5.1 Opis działania aplikacji

Głównym założeniem aplikacji wizualizującej pracę systemu było zaprezentowanie pracy sytemu w sposób przejrzysty i uporządkowany. Dodatkowo aplikacja powinna umożliwiać zmianę wybranych parametrów systemy oraz zapewniać jednoczesny dostęp dla wielu użytkowników. Biorąc pod uwagę wymienione wymagania zdecydowano się na napisanie aplikacji webowej wykorzystując do tego celu framework *Angular 4 - https://angular.io/*.

## 5.2 Opis okien

Aplikacja do wizualizacji pracy systemu podzielona jest na 5 ekranów.

1. **Schemat** - przedstawia schemat rozpatrywanej instalacji,
2. **Statystyki** - przedstawia najważniejsze dane opisujące aktualny stan instalacji takie jak: poziomu cieczy w każdym ze zbiorników, aktualne stężenie, a także przebiegi czasowe tych sygnałów zaprezentowane za pomocą wykresów,
3. **Obiekt** - prezentuje szczegółowe dane dotyczące obiektu regulacji oraz daje możliwość zmieniania limitów w każdym ze zbiorników.
4. **Regulator** - przedstawia przebiegi czasowe sygnałów sterowania, a także daje możliwość zmieniania nastaw regulatorów,
5. **Logi** - daje możliwość generowania raportów z wybranych stanów pracy systemu tzn: zwykła praca, stany alarmowe, zmiany wartości parametrów systemu.

Każdy z ekranów na składa się z trzech głównych części 5.1:

1. Pasek menu - umożliwia przechodzenie pomiędzy wszystkimi oknami aplikacji,
2. **Panel danych** - w jego obszarze wyświetlane są dane dotyczące poszczególnych funkcjonalności systemu,
3. **Pasek wiadomości** - Wyświetlane są na nim najważniejsze wiadomości dotyczące pracy systemu takie jak np. przekroczenie limitów.

## 5.3 Alarmy

Każdy zbiornik ma zdefiniowany zestaw czterech limitów poziomu cieczy tzn.

1. Limit minimalny krytyczny,
2. Limit minimalny,
3. Limit maksymalny,
4. Limit maksymalny krytyczny

Wartości wyżej wymienionych limitów mogą być monitorowane oraz zmieniane przy wykorzystaniu ekranu *Obiekty*. Każda zmiana wartości jest automatycznie zapisywana w bazie danych i propagowana do regulatora.

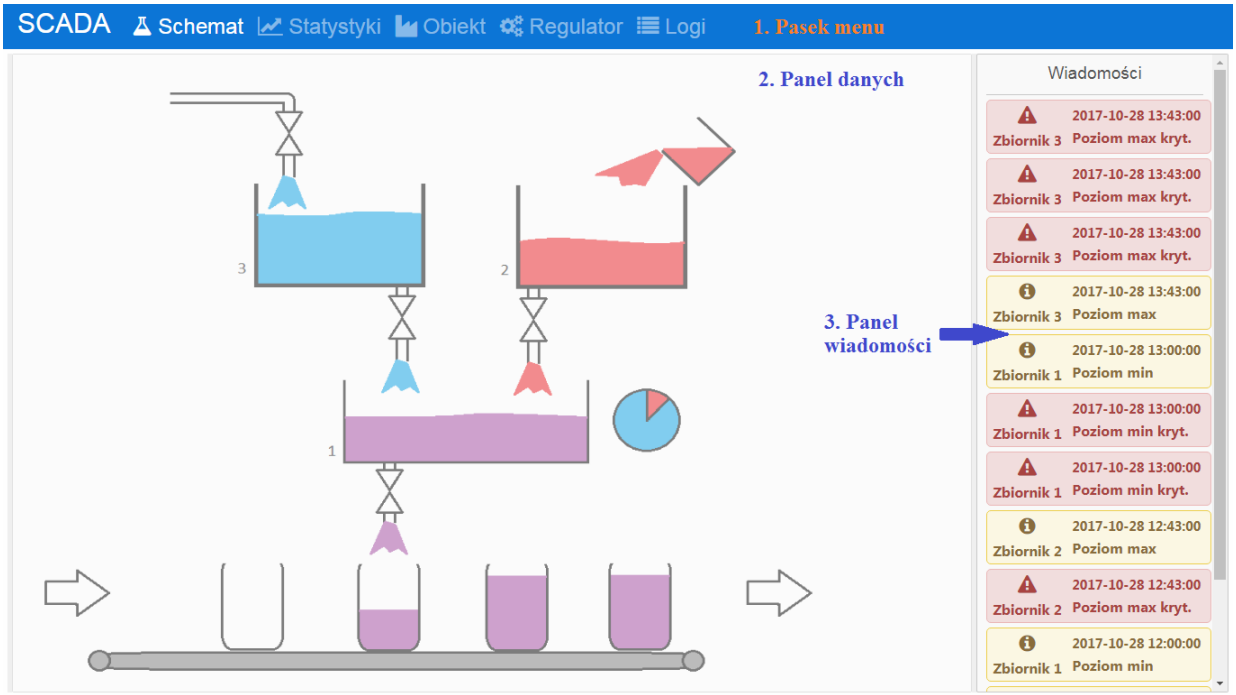
W aplikacji zaimplementowano również propagację alarmów. W momencie przekroczenia dowolnego ze zdefiniowanych limitów pod wskazany adres wysyłany jest email informujący o tym zdarzeniu.

## 5.4 Regulatory

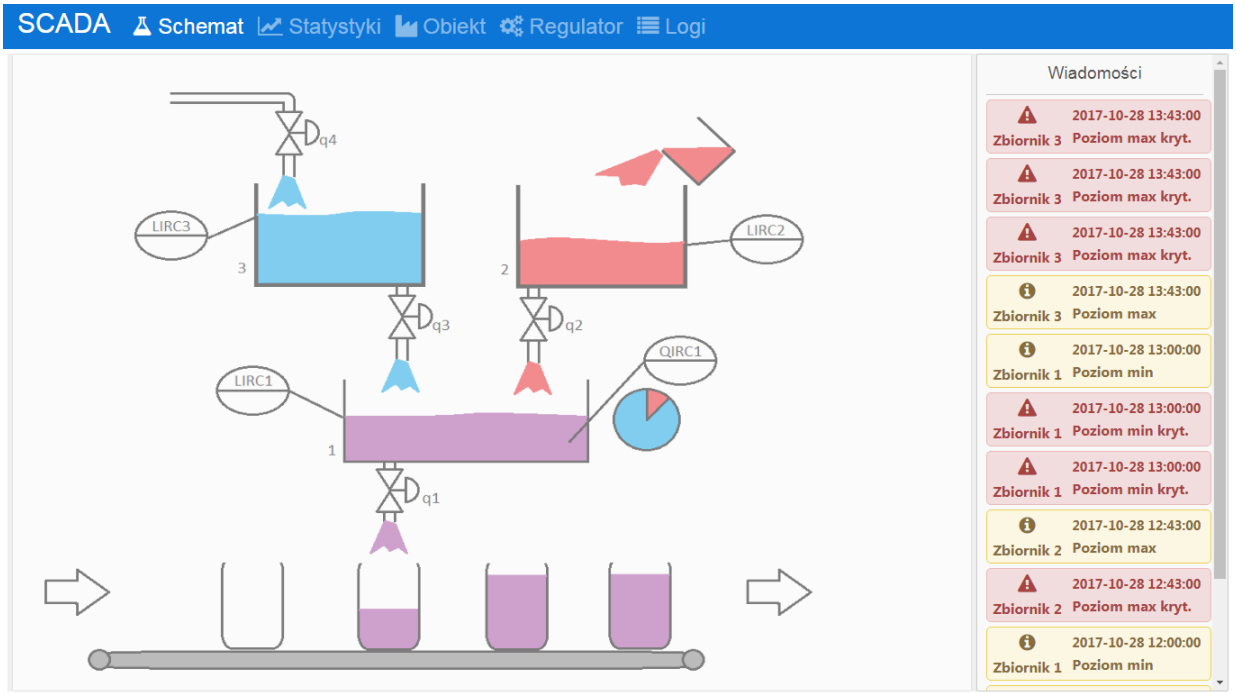
Aplikacja daje możliwość operatorowi na zdalną zmianę parametrów regulatorów za pomocą ekranu *Regulator*. Podobnie jak w przypadku alarmów każdorazowa zmiana dowolnego parametru jest propagowana do regulatora i zapisywana w bazie danych.

## 5.5 Wygląd aplikacji

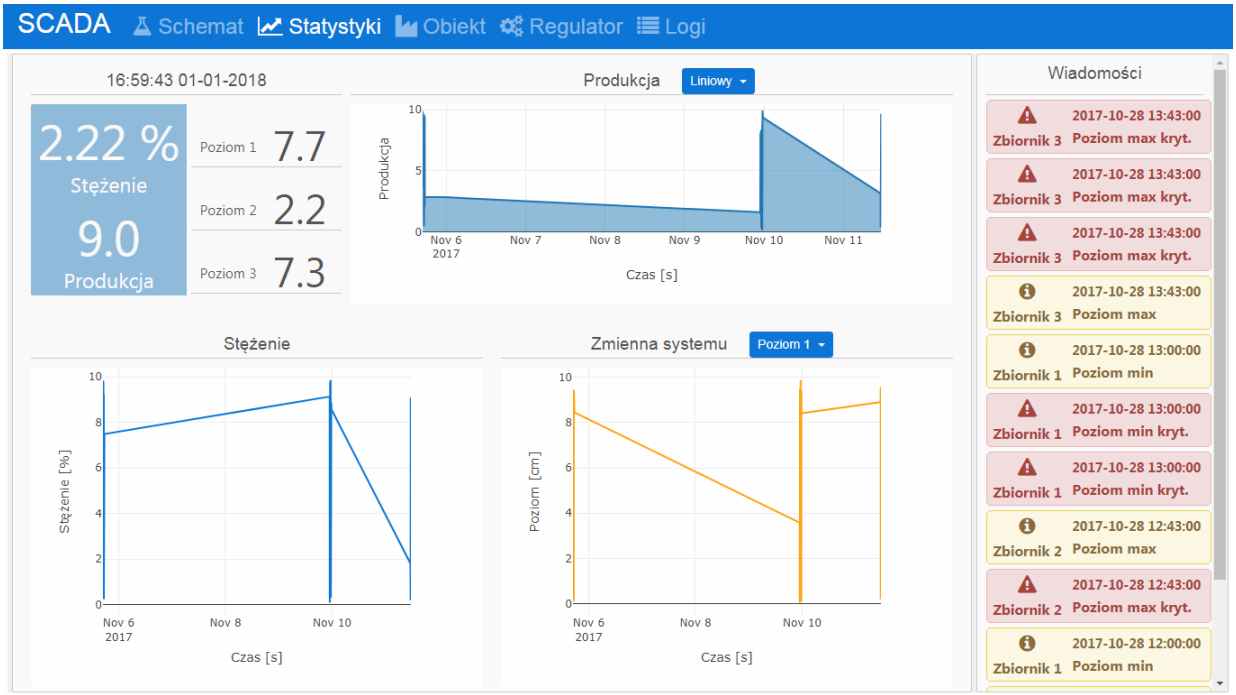
Na poniższych rysunkach przedstawiono screen-y z działania opisywanej aplikacji dla każdego z dostępnych ekranów.



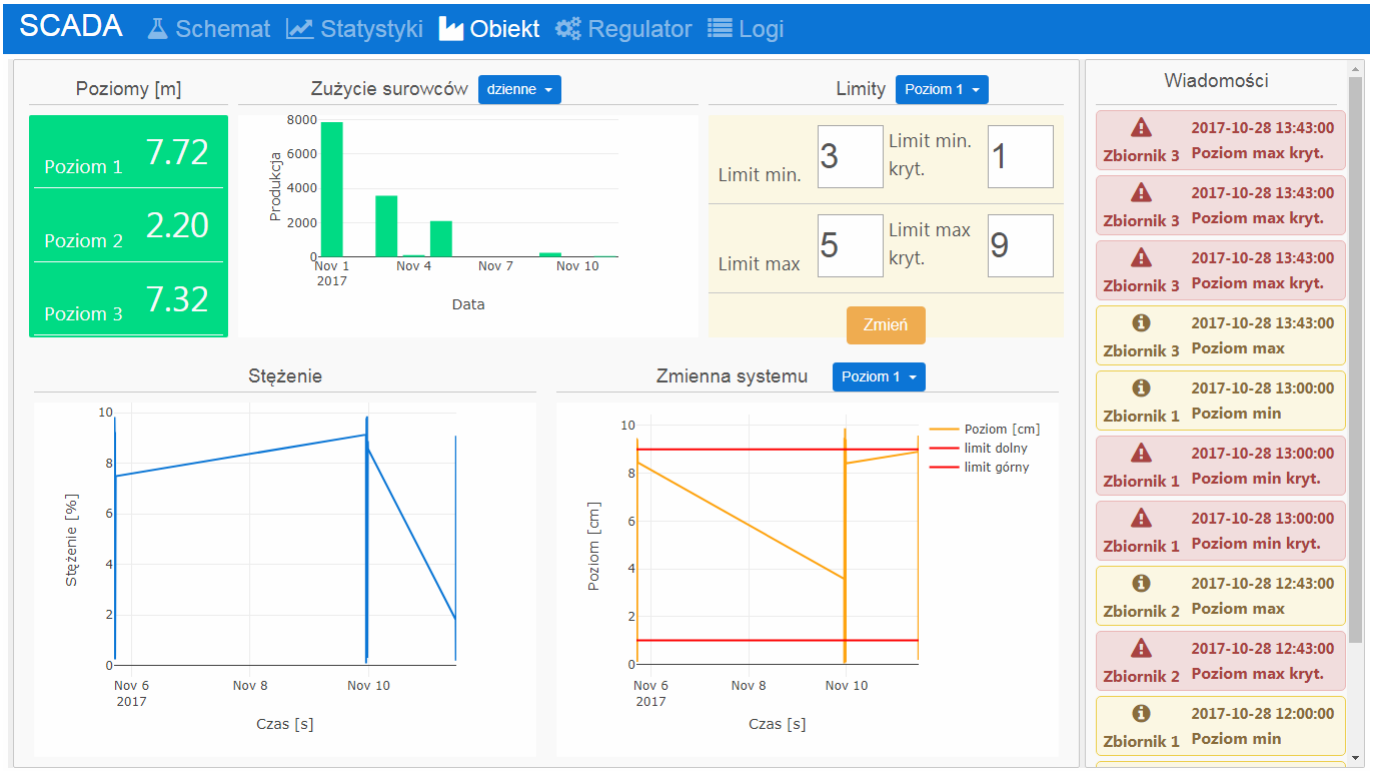

Rys. 5.1: Budowa ekranu.



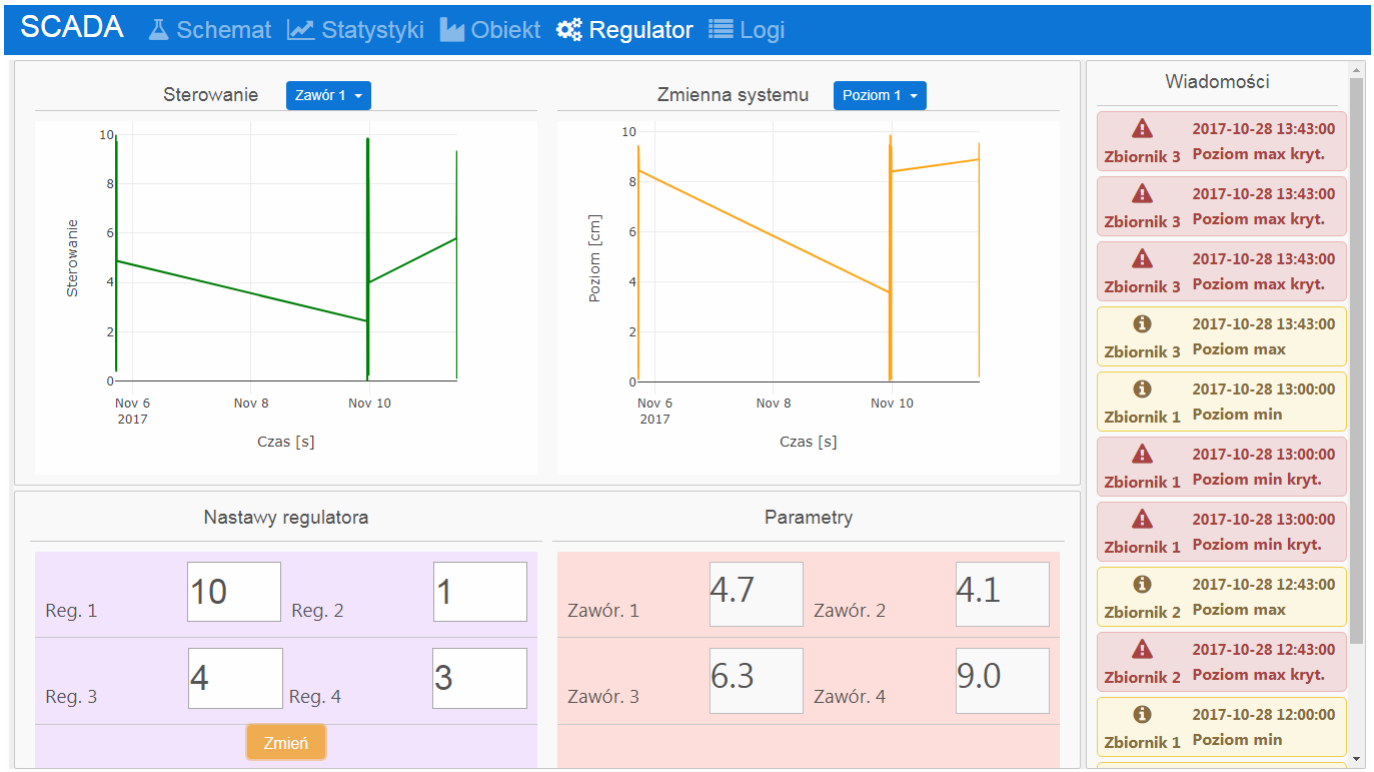

Rys. 5.2: Schemat systemu.



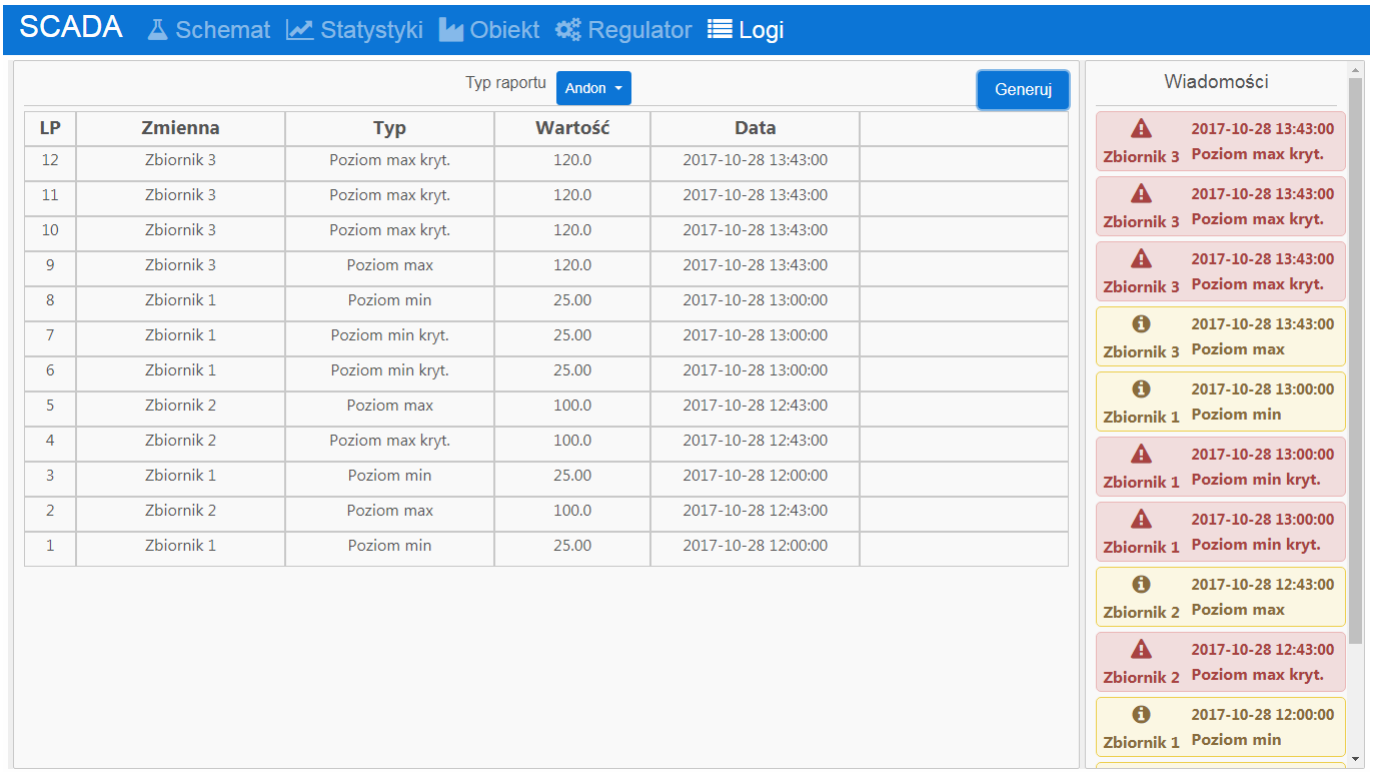

Rys. 5.3: Statystyki systemu.




Rys. 5.4: Ekran przedstawiający stan instalacji.




Rys. 5.5: Ekran przestawiający pracę regulatorów.




Rys. 5.6: Ekran do generowania raportów.

## 5.6 Uruchomienie aplikacji

Do uruchomienie aplikacji należy:

1. na wybranym komputerze zainstalować serwer *Node.js - https://nodejs.org/en/*
2. pobrać kod źródłowy projektu ze strony *https://github.com/maciekc/SCADA2*
3. w konsoli systemu *Windows* należy przejść do głównego folderu projektu *SCADA-app* i wywołać procedurę **npm start**
4. od tej pory aplikacja będzie dostępna pod adresem *www.localhost:4200*.

# Podsumowanie i wnioski

Głównym celem projektu było stworzenie aplikacji umożliwiającej zdalny podgląd i zarządzanie wybranym procesem przemysłowym. Na wstępnym etapie zdecydowano, że obiektem, którego praca będzie nadzorowana zostanie model instalacji trzech połączonych ze sobą zbiorników. Przyjęto, że celem pracy układu sterowania w rozważanym systemie będzie nadzór nad mieszaniem cieczy w jednym ze zbiorników i kontrola poziomów w każdym z osobna. Na tej bazie zaproponowano układ sterujący opisany w rozdziale 2.
Uwzględniając specyfikę wybranego systemu sterowania zdecydowano się na napisanie aplikacji umożliwiającej podgląd wybranych zmiennych w czasie rzeczywistym jak i również zmianę wybranych parametrów. Aby umożliwić korzystanie z programu przez wielu użytkowników, oraz zapewnić łatwość obsługi, aplikacja napisana została w formie webowej, uruchamianej na dowolnej przeglądarce internetowej. Dodatkowo chciano uwzględnić w architekturze całego systemu elementy charakterystyczne dla tego typu aplikacji tzn. komunikację sterownika z obiektem czy logowanie danych procesowych w bazie danych.
W trakcie pracy wyniknęło kila problemów związanych głównie z integracją poszczególnych węzłów systemu. Największy z nich dotyczył zapisu poszczególnych danych w serwerze OPC, tak aby mogły być one dostępne w programie *Simulink* oraz logowania parametrów systemu w bazie danych. Początkowo zakładano, że serwer *HTTP* będzie odpowiedzialny za komunikacje z serwerem OPC, jednak w wyniku problemów z dostępnością biblioteki umożliwiającej poprawne nawiązanie komunikacji z OPC zdecydowano się napisać tę część systemu w języku *C#*.
Podsumowując, wszystkie założenia dotyczące zostały finalnie zrealizowane. Realizacja niniejszego problemu była okazją do wykorzystania wiedzy zdobytej na przestrzeni całych studiów. Wykorzystane zostało nie tylko doświadczenie w projektowaniu układów regulacji w środowisku *Matlab/Simulik* ale również umiejętności programowania w językach typu *JAVA, C#, Typescript*. Cennym doświadczeniem okazało się również zaprojektowanie struktury bazy danych, tak aby odzwierciedlała najważniejsze funkcjonalności systemu.

# Bibliografia

[1] *https://angular.io/docs* dokumanetacja framework-u Angular

[2] *https://doc.akka.io/docs/akka-http/current/index.html* dokumentacja framework-u Akka-http.

[3] *http://jdbi.org/* - dokumentacja biblioteki JDBI

[4] *http://reactivex.io/* - dokumentacja biblioteki RXJava.