Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматизації, комп'ютерно-інтегрованих технологій та систем
(повна назва)
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розроблення кросплатформеного програмного засобу для створення і симуляції електропневматичних схем
(тема)

Виконав:
здобувач 4 року навчання,
групи АКТАКІТ-22-1
Василь СТОЛЯРЕНКО
(власне ім'я, прізвище)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітньої програми Автоматизація та комп'ютерно-інтегровані технології
(повна назва освітньої програми)
Керівник доц. Оксана СИЧОВА
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2026 р.

Я, Столяренко Василь Васильович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«18 » червня 2026 р.  Василь СТОЛЯРЕНКО

Харківський національний університет радіоелектроніки

Факультет	Автоматизації, комп'ютерно-інтегрованих технологій та систем
Кафедра	Комп'ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії
Рівень вищої освіти	перший (бакалаврський)
Спеціальність	151 Автоматизація та комп'ютерно-інтегровані технології
	(код і повна назва)
Тип програми	освітньо – професійна
Освітня програма	Автоматизація та комп'ютерно-інтегровані технології
	(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри_____
(підпис)
«_____»_____2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Столяренку Василю Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення кросплатформеного програмного засобу для створення і симуляції електропневматичних схем

затверджена наказом по університету від 04.06.2026 р. № 472 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10.07.2026 р.

3. Вихідні дані до роботи

3.1 Система програмування JetBrains WebStorm

3.2 Мова програмування TypeScript

3.3 Технології Vue 3, Electron, SVG та Quasar

4. Перелік питань, що потрібно опрацювати в роботі

4.1 Вступ

4.2 Аналіз вихідних даних і літератури за темою дослідження

4.3 Проєктування структури та алгоритмів програмного засобу

4.4 Розробка архітектури та моделі даних програмного засобу

4.5 Реалізація та оптимізація кросплатформеного програмного засобу

4.6 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 14 с. формату А4

6. Консультанти розділів роботи

Найменування розділу	Керівник (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк/терміни виконання етапів роботи	Примітка
1	Отримання завдання	27.04.2026	Виконано
2	Аналіз завдання	28.04.2026	Виконано
3	Огляд літератури з теми дослідження	30.04.2026	Виконано
4	Аналіз існуючих програмних засобів моделювання	01.05.2026	Виконано
5	Формування вимог до програмного засобу	04.05.2026	Виконано
6	Проєктування архітектури та моделі даних системи	05.05.2026	Виконано
7	Розробка алгоритмів імітаційної симуляції	11.05.2026	Виконано
8	Реалізація графічного редактора схем	18.05.2026	Виконано
9	Реалізація підсистеми симуляції	01.06.2026	Виконано
10	Реалізація механізмів оптимізації програмного засобу	15.06.2026	Виконано
11	Оформлення пояснювальної записки	20.06.2026	Виконано
12	Подання кваліфікаційної роботи в ЕК	10.07.2026	Виконано

Дата видачі завдання 27.04.2026

Студент _____ Василь СТОЛЯРЕНКО
(підпис) (власне ім'я, прізвище)

Керівник роботи _____ доц. Оксана СИЧОВА
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 83 с., 7 табл., 24 рис., 2 дод., 23 джерела.

ЕЛЕКТРОПНЕВМАТИЧНА СИСТЕМА, АВТОМАТИЗАЦІЯ, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, АЛГОРИТМ СИМУЛЯЦІЇ, КРОСПЛАТФОРМЕНИЙ ПРОГРАМНИЙ ЗАСІБ, ЛОГІЧНА ОБРОБКА СИГНАЛІВ.

Об'єкт дослідження – процес проєктування та імітаційного моделювання електропневматичних систем.

Предмет дослідження – методи та алгоритми імітаційного моделювання електропневматичних схем у кросплатформеному програмному середовищі.

Мета роботи – підвищення ефективності та вдосконалення навчального процесу у сфері електропневмоавтоматики шляхом розроблення кросплатформного програмного засобу для створення електропневматичних схем і проведення їх імітаційного моделювання.

В роботі проведено аналіз процесів функціонування електропневматичних систем, здійснено формалізацію їх елементів та розроблено алгоритмічне забезпечення для імітаційної симуляції у кросплатформеному програмному середовищі. Запропоновано архітектурну модель програмного засобу та структуру моделі даних для представлення електропневматичних схем.

Програмний засіб може бути використаний для автоматизації навчального процесу під час проведення лабораторних занять з дисциплін, пов’язаних з автоматизацією та керуванням технологічними системами.

Імітаційна симуляція дозволить досліджувати принципи роботи електропневматичних схем без використання фізичного обладнання, що зменшує матеріальні витрати та підвищує безпечність навчального процесу. У подальшому програмний засіб може бути розширений для використання в системах дистанційного навчання та інтегрований у цифрові освітні платформи.

ABSTRACT

Explanatory note: 83 p., 7 tab., 24 fig., 2 appendices, 23 sources.

ELECTRO-PNEUMATIC SYSTEM, AUTOMATION, SIMULATION MODELING, SIMULATION ALGORITHM, CROSS-PLATFORM SOFTWARE TOOL, LOGICAL SIGNAL PROCESSING.

The object of research is the process of designing and simulation modeling of electro-pneumatic systems.

The subject of research is methods and algorithms for simulation modeling of electro-pneumatic circuits in a cross-platform software environment.

The purpose of the work is to improve the efficiency and enhance the educational process in the field of electro-pneumatic automation by developing a cross-platform software tool for creating electro-pneumatic circuits and performing their simulation modeling.

The work includes an analysis of the operating processes of electro-pneumatic systems, formalization of their elements, and development of algorithmic support for simulation in a cross-platform software environment. An architectural model of the software tool and a data model structure for representing electro-pneumatic circuits have been proposed.

The software tool can be used to automate the educational process during laboratory classes related to automation and control of technological systems.

Simulation modeling makes it possible to study the operating principles of electro-pneumatic circuits without the use of physical equipment, which reduces material costs and increases the safety of the educational process. In the future, the software tool can be extended for use in distance learning systems and integrated into digital educational platforms.

ЗМІСТ

Перелік скорочень 9
Вступ 10
1 Аналіз електропневматичних систем 13
1.1 Постановка задачі та аналіз технічного завдання 13
1.2 Електропневматичні системи в навчальному процесі 15
1.3 Типові структури електропневматичних схем 18
1.4 Огляд існуючих програмних засобів для моделювання електропневматичних схем 22
1.5 Висновки до першого розділу 26
2 Проєктування кросплатформеного програмного засобу 27
2.1 Вибір архітектурного підходу 27
2.2 Загальна структура програмного засобу 30
2.3 Врахування принципів теорії автоматичного керування при проєктуванні програмного засобу 34
2.4 Проєктування моделі даних електропневматичних схем 37
2.5 Висновки до другого розділу 39
3 Розробка алгоритму імітаційної симуляції електропневматичної схеми 40
3.1 Формалізація елементів електропневматичної схеми 40
3.2 Моделювання станів пневматичної частини 44
3.3 Алгоритм поширення сигналів 48
3.4 Алгоритм оновлення станів елементів 54
3.5 Висновки до третього розділу 59
4 Реалізація та оптимізація кросплатформеного програмного засобу для моделювання електропневматичних схем 60
4.1 Вибір технологічного стеку та середовища розробки 61
4.2 Реалізація графічного редактора схем 62
4.3 Програмна реалізація підсистеми симуляції 68

4.4 Оптимізація програмного засобу 72
4.5 Охорона праці під час реалізації кросплатформеного програмного засобу для моделювання електропневматичних схем 76
4.6 Висновки до четвертого розділу 79
Висновки 80
Перелік джерел посилання 81
Додаток А Код програми 84
Додаток Б Демонстраційний матеріал 98

ПЕРЕЛІК СКОРОЧЕНЬ

ЕПС – електропневматична система;

ЕПСх – електропневматична схема;

ІМ – імітаційне моделювання;

КПЗ – кросплатформений програмний засіб;

ПЗ – програмне забезпечення;

ПЛК – програмований логічний контролер;

API – Application Programming Interface;

CAD – Computer-Aided Design;

JSON – JavaScript Object Notation;

SVG – Scalable Vector Graphics;

UI – User Interface.

ВСТУП

Сучасний розвиток автоматизації технологічних процесів зумовлює широке використання електропневматичних систем у промисловості, мехатроніці та виробничих комплексах. Електропневматичні схеми застосовуються для керування виконавчими механізмами, пневматичними приводами, циліндрами та клапанами, забезпечуючи надійність, швидкодію та простоту реалізації логічних функцій керування.

У навчальному процесі закладів вищої освіти вивчення електропневматичних систем супроводжується виконанням лабораторних робіт, що передбачають аналіз схем, дослідження їх роботи та побудову алгоритмів керування. Однак використання фізичного обладнання пов’язане з матеріальними витратами, обмеженим доступом до лабораторій та підвищеними вимогами до безпеки.

У зв’язку з цим актуальним є створення програмного засобу, який дозволяє автоматизувати процес створення та імітаційного моделювання електропневматичних схем, забезпечити наочність роботи систем та розширити можливості навчального процесу.

Виходячи з цього тема роботи «Розроблення кросплатформеного програмного засобу для створення і симуляції електропневматичних схем» є актуальною.

Об'єкт дослідження – процес проєктування та імітаційного моделювання електропневматичних систем.

Предмет дослідження – методи та алгоритми імітаційного моделювання електропневматичних схем у кросплатформеному програмному середовищі.

Мета роботи – підвищення ефективності та вдосконалення навчального процесу у сфері електропневмоавтоматики шляхом розроблення кросплатформного програмного засобу для створення електропневматичних схем і проведення їх імітаційного моделювання.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– провести аналіз електропневматичних систем та існуючих програмних засобів моделювання;

– сформулювати вимоги до розроблюваного програмного засобу;

– розробити архітектурну модель кросплатформеної системи;

– спроєктувати модель даних електропневматичної схеми;

– формалізувати елементи електропневматичної системи;

– розробити алгоритм імітаційної симуляції та оновлення станів елементів.

У роботі застосовані методи системного аналізу, формалізації структурних елементів, моделювання дискретних станів та алгоритмічного проєктування.

Робота складається з чотирьох розділів, висновків та списку використаних джерел. У першому розділі проведено аналіз електропневматичних систем, розглянуто їх застосування у навчальному процесі, проаналізовано типові структури електропневматичних схем та виконано огляд існуючих програмних засобів моделювання. У другому розділі виконано проєктування кросплатформеного програмного засобу, обґрунтовано вибір архітектурного підходу, розроблено загальну структуру системи та модель даних електропневматичної схеми. У третьому розділі розроблено алгоритмічну основу імітаційної симуляції електропневматичних схем, виконано формалізацію елементів на основі моделі Element / Port / State, розглянуто моделювання станів пневматичної частини, а також реалізовано алгоритми поширення сигналів та оновлення станів елементів. У четвертому розділі розглянуто практичну реалізацію програмного засобу, вибір технологічного стеку, реалізацію графічного редактора схем, підсистеми симуляції та механізмів оптимізації архітектури й інтерфейсу користувача.

Кваліфікаційна робота виконана згідно з ДСТУ 3008 – 15 [1], керуючись навчальним посібником з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання [2] та методичними вказівками з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти [3].

Відповідно Указу президента України №722/2019 від 30.09.2019 року "Про Цілі сталого розвитку (ЦСР) України на період до 2030 року" [4], питання, що розглядаються в кваліфікаційній роботі відповідають Цілі сталого розвитку 9 "Промисловість, інновації та інфраструктура", а саме завданню 9.4 " Сприяти прискореному розвитку високо- та середньовисокотехнологічних секторів переробної промисловості, які формуються на основі використання ланцюгів "освіта – наука – виробництво" та кластерного підходу за напрямами: розвиток інноваційної екосистеми; розвиток інформаційно-телекомунікаційних технологій (ІК); застосування ІК в АПК, енергетиці, транспорті та промисловості; високотехнологічне машинобудування; створення нових матеріалів; розвиток фармацевтичної та біоінженерної галузей"

Отримані результати роботи опубліковано у Збірнику студентських наукових статей "Автоматизація та приладобудування" ("Automation and Development of Electronic Devices" ADED-2026(1)), назва статті "Програмний засіб для створення електропневматичних схем" [5].

1 АНАЛІЗ ЕЛЕКТРОПНЕВМАТИЧНИХ СИСТЕМ

1.1 Постановка задачі та аналіз технічного завдання

У сучасних системах автоматизації технологічних процесів широко застосовуються електропневматичні системи, які поєднують електричні засоби керування з пневматичними виконавчими механізмами. Такі системи використовуються для керування приводами, циліндрами, розподільниками, клапанами та іншими елементами виробничого обладнання [6].

У навчальному процесі дисциплін, пов’язаних з автоматизацією та мехатронікою, студенти виконують лабораторні роботи, що передбачають аналіз, побудову та дослідження електропневматичних схем.

Основними завданнями є:

– побудова електричної та пневматичної частин схеми;

– з’єднання функціональних елементів відповідно до заданої логіки;

– дослідження послідовності спрацювання виконавчих механізмів;

– аналіз впливу керуючих сигналів на стан системи.

Використання фізичних лабораторних стендів пов’язане з низкою обмежень: високою вартістю обладнання, необхідністю технічного обслуговування, обмеженою кількістю робочих місць та підвищеними вимогами до безпеки. Крім того, традиційні методи навчання не завжди забезпечують достатню наочність процесів, що відбуваються в системі.

Електропневматичні системи є складовою частиною сучасних автоматизованих технологічних комплексів. Вони забезпечують керування пневматичними виконавчими механізмами за допомогою електричних сигналів та реалізують логічні алгоритми функціонування виробничого обладнання. Поєднання електричної та пневматичної підсистем дозволяє створювати ефективні, надійні та відносно прості у реалізації системи керування [7].

Аналіз існуючих програмних засобів моделювання показує, що більшість з них орієнтовані на професійне використання, мають надлишковий функціонал

або складний інтерфейс, що ускладнює їх застосування в освітньому середовищі. Крім того, не всі системи забезпечують спрощену логічну симуляцію, достатню для виконання типових лабораторних завдань без застосування складних фізичних моделей.

У зв’язку з цим виникає потреба у створенні програмного засобу, який дозволяє створювати електропневматичні схеми у графічному 2D-редакторі; здійснювати з’єднання елементів відповідно до заданої структури; виконувати імітаційну симуляцію роботи системи; візуалізувати зміну станів елементів у реальному часі та використовувати систему для виконання лабораторних робіт.

Технічне завдання передбачає розробку кросплатформеного програмного засобу з реалізацією імітаційної логічної симуляції без застосування складних фізичних моделей.

Система повинна підтримувати обмежений набір елементів, достатній для покриття типових лабораторних завдань, та забезпечувати збереження і завантаження створених схем.

Основними функціональними вимогами є:

– підтримка графічного представлення електропневматичних елементів;
– реалізація механізму з’єднання портів елементів;
– формалізована модель станів;
– алгоритм поширення сигналів;
– можливість запуску, зупинки та перезапуску симуляції.

Використання дискретної моделі станів дозволяє спростити реалізацію алгоритмів симуляції, забезпечити достатню швидкодію та реалізувати наочну візуалізацію роботи електропневматичних схем у навчальному процесі.

Таким чином, постановка задачі полягає у розробці архітектурно обґрунтованого програмного засобу, який забезпечує автоматизоване створення та імітаційне моделювання електропневматичних схем для використання в навчальному процесі.

1.2 Електропневматичні системи в навчальному процесі

Електропневматичні системи є різновидом автоматизованих систем керування, у яких електричні сигнали використовуються для керування пневматичними виконавчими механізмами. Такі системи поєднують переваги електричного керування та пневматичних приводів, що забезпечує надійність, безпечність, відносну невисоку вартість, стійкість до перевантажень, простоту реалізації логічних функцій та надає можливість інтеграції з цифровими пристроями [9].

До основних складових електропневматичної системи можна віднести: джерела стисненого повітря; пневматичні виконавчі механізми (циліндри, пневмодвигуни); розподільники та клапани; електричні елементи керування (кнопки, датчики, реле, котушки електромагнітів) та з'єднувальні магістралі та керуючі лінії.

Структурно електропневматична система складається з двох взаємопов'язаних підсистем з електричної та пневматичних частин.

Електрична підсистема формує керуючі сигнали, забезпечує обробку вхідних сигналів (від кнопок, кінцевих вимикачів, датчиків положення) та впливає на електромагнітні котушки розподільників.

Пневматична підсистема реалізує фізичний рух виконавчих механізмів та переміщення робочих елементів шляхом зміни напрямку та тиску повітря в робочих камерах виконавчих механізмів.

Функціонування електропневматичної системи можна описати як послідовність переходів між дискретними станами. Стан системи визначається положенням розподільників, наявністю або відсутністю тиску в магістралях, а також положенням виконавчих механізмів. Зміна стану відбувається внаслідок подачі електричного сигналу або досягнення певного положення виконавчим елементом.

З точки зору формалізації електропневматичну систему доцільно розглядати як множину елементів, що взаємодіють через визначені з'єднання. Кожен елемент має набір вхідних та вихідних портів, множину можливих

станів та правило переходу між станами.

Структурна схема взаємодії електричної та пневматичної частин представлена на рис. 1.1.



Рисунок 1.1 – Взаємодія підсистем в електропневматичній системі

Така інтерпретація дозволяє описувати роботу системи в термінах дискретної логічної моделі, що є основою для подальшої реалізації імітаційної симуляції [9].

Електропневматичні системи широко застосовуються в автоматизованих виробничих лініях, робототехнічних комплексах, пакувальному обладнанні, системах сортування та позиціонування. Простота реалізації логіки керування та надійність пневматичних приводів зумовлюють їх поширення в промисловості.

Таким чином, електропневматичні системи є важливим об’єктом дослідження в галузі автоматизації, а їх формалізація як сукупності взаємопов’язаних елементів із дискретними станами створює підґрунтя для розробки програмних засобів імітаційного моделювання.

Електропневматичні системи займають важливе місце у підготовці

фахівців з автоматизації, мехатроніки та комп’ютерно-інтегрованих технологій. Вивчення принципів їх побудови та функціонування є складовою дисциплін, пов’язаних з автоматизацією технологічних процесів, керуванням виробничими комплексами та проектуванням мехатронних систем.

У межах навчального процесу студенти засвоюють структуру електропневматичних схем; принципи роботи пневматичних приводів; логіку керування виконавчими механізмами; взаємодію електричних та пневматичних підсистем та методи аналізу послідовності спрацювання елементів.

Лабораторні роботи, як правило, передбачають побудову схеми відповідно до заданого алгоритму, підключення елементів, перевірку правильності функціонування та дослідження поведінки системи при зміні керуючих сигналів. Такий підхід дозволяє формувати практичні навички проєктування та аналізу електропневматичних систем.

Разом з тим, традиційна організація лабораторних занять із використанням фізичних стендів має низку обмежень, а саме:

– обмежена кількість обладнання та робочих місць;
– значні витрати на придбання та обслуговування компонентів;
– складність швидкої зміни конфігурації схеми;
– ризик пошкодження елементів через помилки підключення;
– обмежені можливості візуалізації внутрішніх процесів.

Крім того, при роботі з фізичним обладнанням студент не завжди може спостерігати повну картину зміни станів усіх елементів системи, що ускладнює розуміння причинно-наслідкових зв’язків між керуючими сигналами та результатом функціонування. Використання програмних засобів імітаційного моделювання дозволяє вирішити зазначені проблеми.

Симуляція забезпечує можливість створення та редагування схем без фізичних обмежень; швидкої перевірки різних варіантів реалізації алгоритму; наочної візуалізації зміни станів елементів; багаторазового повторення експериментів; інтеграції у дистанційні та змішані форми навчання.

Таким чином, впровадження програмного засобу для створення та симуляції електропневматичних схем сприяє підвищенню ефективності

навчального процесу, автоматизації виконання лабораторних робіт та формуванню системного мислення у студентів технічних спеціальностей.

1.3 Типові структури електропневматичних схем

Електропневматичні схеми будуються на основі поєднання електричної логіки керування та пневматичної виконавчої частини. Незважаючи на різноманіття можливих конфігурацій, у навчальній та промисловій практиці використовуються типові структурні рішення, що повторюються у більшості задач автоматизації [11].

Найпростішою структурою є схема прямого керування, у якій електричний сигнал безпосередньо подається на котушку електромагнітного розподільника, що керує пневматичним циліндром.

Типова структура прямого керування включає кнопку або вимикач; джерело електроживлення; електромагнітну котушку (соленоїд); пневматичний розподільник (наприклад, 5/2); двосторонній пневматичний циліндр; джерело стисненого повітря.

У такій схемі подача сигналу призводить до перемикання розподільника та зміни напрямку руху поршня. Структура є лінійною та не містить зворотних зв’язків.

Більш складною є структура, що містить елементи зворотного зв’язку. У цьому випадку положення виконавчого механізму контролюється кінцевими вимикачами або датчиками положення.

Типова структура, що містить елементи зворотного зв’язку включає пускову кнопку; кінцеві вимикачі положення штока; логічний блок (релейна логіка); електромагнітні котушки розподільника; двосторонній циліндр.

Наявність зворотного зв’язку дозволяє реалізувати автоматичні цикли, наприклад, висування → спрацювання датчика → повернення → зупинка. Такі схеми є основою циклічних автоматичних систем.

У багатопозиційних системах використовуються послідовні структури, де спрацювання одного виконавчого механізму є умовою запуску іншого.

Наприклад, циліндр A висувається → датчик A1 спрацьовує → запускається циліндр B → датчик B1 → повернення A → повернення B. Наведена структура вже має вигляд логічного ланцюга станів і може бути описана як дискретна система переходів.

У деяких задачах реалізується паралельне керування кількома виконавчими механізмами. У такому випадку логічний блок формує декілька незалежних керуючих сигналів, що подаються на різні розподільники. Такі схеми застосовуються у пакувальних машинах; сортувальних системах; маніпуляторах.

Незалежно від конкретної конфігурації, кожна схема містить вхідні елементи (сенсори, кнопки); логічний блок керування; електромагнітні виконавчі елементи; пневматичну частину (розподільники та приводи) та джерело енергії [12].

Систематизація типових структур дозволяє виділити повторювані закономірності побудови електропневматичних схем, що створює підґрунтя для їх формального опису в термінах множин елементів, зв’язків та станів. Такий підхід є необхідним для розробки універсальної моделі програмного симулятора.

Важливою особливістю типових структур електропневматичних схем є їх придатність до формалізації у вигляді дискретних моделей, що описуються скінченними множинами станів та переходів між ними. Такий підхід дозволяє представити функціонування системи як послідовність логічних подій, що змінюють стан окремих елементів залежно від вхідних сигналів та умов спрацювання. Формалізація у вигляді дискретної моделі значно спрощує реалізацію алгоритмів імітаційного моделювання [11].

Аналіз типових структур електропневматичних схем показує, що більшість навчальних задач базуються на обмеженому наборі елементів та повторюваних структур керування. Це дозволяє сформувати мінімальний набір компонентів, достатній для реалізації імітаційного симулятора, що покриває основні лабораторні роботи.

Узагальнені характеристики таких структур наведено в таблиці 1.1.

Таблиця 1.1 – Типові структури електропневматичних схем

Тип структури	Основні елементи	Переваги	Приклад застосування	Особливості моделювання
Пряме керування	Кнопка, джерело живлення, соленоїд, 5/2 розподільник, циліндр	Лінійна структура без зворотного зв’язку	Ручне керування циліндром	Достатньо двох станів (увімкнено/ вимкнено)
Схема зі зворотним зв’язком	Схема зі зворотним зв’язком	Кнопка, кінцеві вимикачі, реле/логіка, розподільни, циліндр	Автоматичний цикл висування-повернення	Потрібне моделювання дискретних станів та переходів
Послідовна структура	Декілька циліндрів, датчики положення, логічний блок	Послідовні залежності між елементами	Цикл A → B → A → B	Реалізація алгоритму переходів станів
Каскадна структура	Групи розподільникі, логічні елементи, кілька приводів	Ієрархічна логіка керування	Багатоступенева автоматизація	Потрібна формалізація умов переходів
Паралельна структура	Кілька розподільників, спільний логічний блок	Паралельне керування виконавцями	Сортувальні системи, маніпулятори	Одночасне оновлення станів кількох елементів
Циклічна автоматична система	Датчики, таймери (за потреби), логічний блок, виконавчі механізми	Замкнений цикл переходів	Автоматизовані виробничі операції	Реалізація циклічного алгоритму

Обмеження набору елементів дозволяє спростити реалізацію програмного засобу, зосередивши увагу на формалізації логіки взаємодії та алгоритмі оновлення станів, без необхідності моделювання складних фізичних процесів.

Мінімальний набір елементів для реалізації типових структур електропневматичних схем для реалізації більшості лабораторних завдань наведений в таблиці 1.2.

Таблиця 1.2 – Мінімальний набір елементів для реалізації типових структур електропневматичних схем

Елемент	Тип підсистеми	Функціональне призначення	Необхідність для типових структур
Джерело стисненого повітря	Пневматична	Забезпечення подачі робочого середовища	Обов’язковий для всіх схем
Розподільник 5/2	Пневматична	Керування напрямком руху повітря	Основний елемент керування циліндром
Двосторонній пневматичний циліндр	Пневматична	Реалізація механічного руху	Виконавчий механізм
Електромагнітна котушка (соленоїд)	Електрична	Перемикання розподільника	Забезпечує взаємодію підсистем
Кнопка (NO)	Електрична	Формування початкового сигналу	Для прямого керування
Кінцевий вимикач	Електрична	Формування зворотного зв’язку	Для автоматичних циклів
Джерело електроживлення	Електрична	Забезпечення живлення логіки та соленоїдів	Обов’язковий елемент

Використання зазначених компонентів забезпечує можливість моделювання лінійних, послідовних, циклічних та паралельних структур керування.

1.4 Огляд існуючих програмних засобів для моделювання електропневматичних схем

Сучасні програмні засоби моделювання електропневматичних систем можна умовно поділити на три групи:

– професійні інженерні середовища;

– освітні програмні комплекси;

– спрощені онлайн-рішення.

Більшість сучасних систем базуються на принципах імітаційного моделювання, що передбачає відтворення поведінки технічної системи шляхом дискретного оновлення її станів відповідно до заданих алгоритмів [11].

До професійних засобів належать Automation Studio, Siemens TIA Portal (у поєднанні з PLC-симуляцією), а також спеціалізовані CAD/CAE-рішення.

Такі системи забезпечують детальне фізичне моделювання; підтримку промислових стандартів; інтеграцію з контролерами та ПЛК; складні розрахункові моделі.

Однак їх основною метою є промислове застосування та проєктування реальних виробничих комплексів. Згідно з сучасними підходами до програмної інженерії, подібні системи характеризуються багаторівневою архітектурою та високою функціональною насиченістю, що ускладнює використання в навчальному процесі.

Недоліками таких середовищ є складність інтерфейсу; надлишковий функціонал для базових лабораторних задач; високі вимоги до апаратного забезпечення та ліцензійні обмеження.

Найбільш відомим освітнім середовищем є FluidSIM (Festo Didactic), що широко використовується у технічних навчальних закладах. Дане середовище поєднує графічний редактор схем; бібліотеку стандартних пневматичних компонентів; засоби імітаційної симуляції; навчальні матеріали. Симуляція в таких системах реалізується як поєднання фізичної та логічної моделей. Проте аналіз показує, що функціональність освітніх комплексів часто перевищує потреби типових лабораторних робіт, що призводить до підвищеної складності

інтерфейсу та перевантаження користувача.

Окрему групу становлять веб-орієнтовані або спрощені симулятори, що реалізують базову логіку пневматичних схем. Такі системи зазвичай базуються на спрощених дискретних моделях станів без детального фізичного розрахунку параметрів. Перевагами є доступність, простота використання та відсутність складної інсталяції. Недоліками є обмежений набір компонентів; відсутність гнучкої архітектури; неможливість адаптації під конкретну навчальну програму; відсутність модульної структури [13].

З точки зору теорії імітаційного моделювання [11], програмні засоби можна класифікувати за рівнем деталізації моделі:

– фізично орієнтовані моделі (розрахунок тиску, витрат, динаміки);

– логічні дискретні моделі (перемикання станів елементів);

– гібридні моделі.

Для реалізації навчального симулятора доцільним є використання дискретної логічної моделі, що забезпечує достатню наочність при відносній простоті реалізації. Такий підхід відповідає принципам модульної архітектури та розділення відповідальності компонентів програмної системи та дозволяє зосередити увагу на логіці функціонування електропневматичних схем без необхідності реалізації складних фізичних розрахунків.

Аналіз сучасних програмних засобів показує, що професійні середовища орієнтовані на промислове застосування та мають надлишковий функціонал; освітні комплекси є ефективними, проте обмежені ліцензійною політикою та складністю; спрощені онлайн-рішення не забезпечують достатньої гнучкості. Існуючі програмні засоби або орієнтовані на професійне промислове використання та мають надлишковий функціонал, або є спрощеними онлайн-рішеннями з обмеженими можливостями.

Таким чином, доцільною є розробка спеціалізованого кросплатформеного програмного засобу з логічною імітаційною симуляцією, оптимізованого для виконання лабораторних робіт та використання в навчальному процесі.

Порівняння основних характеристик існуючих програмних засобів для моделювання електропневматичних схем наведені в таблиці 1.3.

Таблиця 1.3 – Порівняння існуючих програмних засобів

Критерій	FluidSIM	Automation Studio	Онлайн-симулятори	Запропонований підхід
Орієнтація на навчання	Так	Частково	Частково	Так
Складність інтерфейсу	Середня	Висока	Низька	Оптимізовано
Надлишковий функціонал	Так	Так	Ні	Ні
Фізична деталізація	Висока	Висока	Низька	Логічна модель
Кросплатформеність	Обмежена	Обмежена	Так	Так
Можливість адаптації	Обмежена	Обмежена	Низька	Повна

На основі проведеного аналізу електропневматичних систем та існуючих програмних засобів моделювання сформульовано вимоги до розроблюваного кросплатформеного програмного засобу.

Програмний засіб повинен забезпечувати:

– створення електропневматичних схем у графічному 2D-редакторі;
– додавання та видалення елементів із визначеного набору компонентів;
– з’єднання портів елементів із перевіркою коректності;
– збереження та завантаження схем у структурованому форматі даних;

– запуск, зупинку та перезапуск імітаційної симуляції;

– поширення сигналів відповідно до заданого алгоритму;

– оновлення станів елементів у дискретні моменти часу;

– візуалізацію поточних станів елементів (активні лінії, положення розподільників, рух штока циліндра);

– реалізацію типових структур керування (пряме, послідовне, циклічне, паралельне).

Система повинна відповідати таким характеристикам:

– кросплатформеність (переважна підтримка Windows);

– використання 2D-графіки;

– модульна архітектура;

– логічна імітаційна симуляція без складних фізичних розрахунків;

– зручний та інтуїтивно зрозумілий інтерфейс;

– достатня швидкодія при роботі з типовими лабораторними схемами.

У межах кваліфікаційної роботи передбачаються такі обмеження:

– реалізація лише мінімального набору елементів, достатнього для виконання лабораторних робіт;

– відсутність моделювання динаміки тиску та складних фізичних процесів;

– відсутність інтеграції з реальними ПЛК або промисловими контролерами;

– використання дискретної моделі станів.

Таким чином, основною задачею є розробка архітектури програмного засобу, що забезпечує формалізоване представлення електропневматичної схеми та реалізацію алгоритму імітаційної симуляції відповідно до визначених вимог.

Запропонований підхід дозволить створити програмний засіб, орієнтований на використання у навчальному процесі, який поєднуватиме наочність, достатню функціональність та відносну простоту реалізації. Реалізація такої системи створюватиме основу для подальшого розвитку програмного засобу, розширення бібліотеки елементів та інтеграції сучасних технологій імітаційного моделювання в освітнє середовище.

1.5 Висновки до першого розділу

У першому розділі проведено аналіз електропневматичних систем як об’єкта дослідження та визначено їх роль у сучасних автоматизованих технологічних комплексах. Встановлено, що електропневматичні системи поєднують електричну логіку керування та пневматичні виконавчі механізми, що дозволяє реалізовувати ефективні та надійні алгоритми функціонування.

Проаналізовано типові структури електропневматичних схем, зокрема схеми прямого керування, структури зі зворотним зв’язком, послідовні, паралельні та циклічні конфігурації. Відображено, що більшість навчальних задач базується на обмеженому наборі стандартних компонентів та повторюваних структур керування.

Виконано огляд існуючих програмних засобів моделювання електропневматичних систем. Встановлено, що професійні інженерні середовища мають надлишковий функціонал і складність для використання в освітньому процесі, тоді як спрощені онлайн-рішення не забезпечують достатньої гнучкості та можливостей адаптації.

На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до кросплатформеного програмного засобу. Визначено, що доцільним є використання логічної дискретної моделі станів без реалізації складних фізичних розрахунків. Також обґрунтовано мінімальний набір елементів, достатній для реалізації типових лабораторних завдань.

2 ПРОЄКТУВАННЯ КРОСПЛАТФОРМЕНОГО ПРОГРАМНОГО ЗАСОБУ

2.1 Вибір архітектурного підходу

Розробка кросплатформеного програмного засобу для створення та імітаційного моделювання електропневматичних схем вимагає вибору архітектурного підходу, який забезпечує гнучкість, масштабованість та зручність супроводу системи [11].

З урахуванням специфіки задачі, а саме необхідності поєднання графічного редактора, логіки обробки користувацьких дій та алгоритму імітаційної симуляції, доцільним є використання багаторівневої (шарової) архітектури.

Такий підхід дозволяє розділити систему на окремі логічно незалежні компоненти, кожен з яких відповідає за визначений набір функцій. Це забезпечує спрощення розробки, тестування та подальшого розширення програмного засобу.

Основними причинами вибору багаторівневої архітектури є:

– необхідність розділення інтерфейсу користувача та обчислювальної логіки;

– забезпечення незалежності модулів системи;

– можливість масштабування та розширення функціоналу;

– підвищення зручності супроводу та тестування програмного забезпечення.

У межах обраного підходу система поділяється на три основні рівні:

– рівень представлення;

– рівень прикладної логіки;

– рівень моделювання (обчислювальне ядро).

Схема архітектури програмного засобу відображена на рис. 2.1.

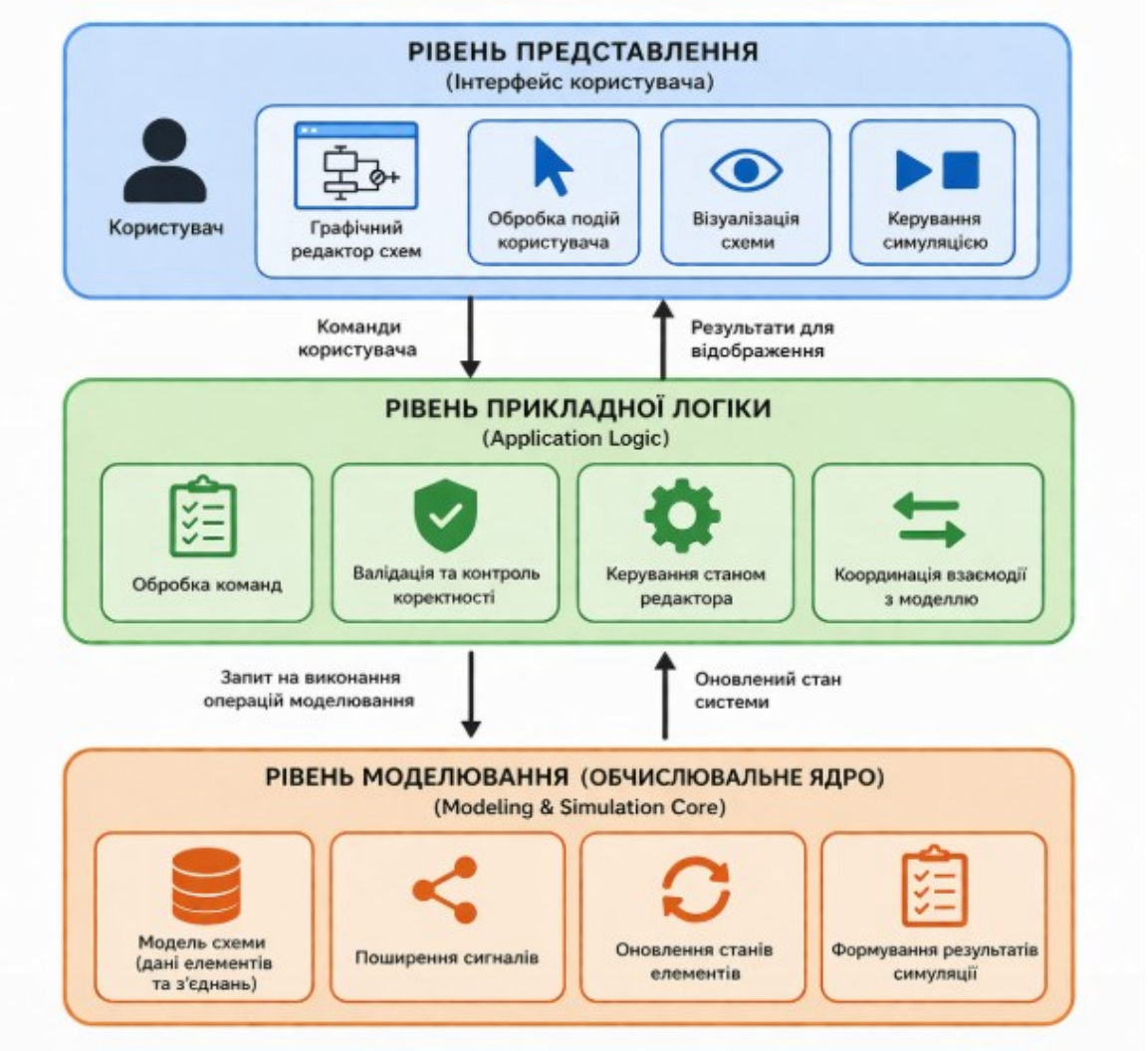


Рисунок 2.1 – Структурна схема архітектури програмного засобу

Рівень представлення відповідає за взаємодію користувача із системою та забезпечує створення, редагування та візуалізацію електропневматичних схем у графічному середовищі. На цьому рівні реалізуються елементи інтерфейсу, обробка подій користувача та відображення результатів симуляції [18].

Рівень прикладної логіки забезпечує обробку команд користувача, перевірку коректності дій, керування станом схеми та координацію взаємодії між інтерфейсом і обчислювальним ядром. Даний рівень виконує роль посередника між користувацьким інтерфейсом та моделлю системи.

Рівень моделювання є ядром програмного засобу та відповідає за представлення електропневматичної схеми у вигляді формалізованої моделі.

На цьому рівні здійснюється обробка елементів та з’єднань, поширення сигналів, оновлення станів компонентів та формування результатів імітаційної

симуляції.

Взаємодія між рівнями здійснюється послідовно та дії користувача передаються з рівня представлення до рівня прикладної логіки, після чого відповідні команди надходять до рівня моделювання. Результати обчислень повертаються у вигляді оновленого стану системи та відображаються в інтерфейсі.

Обраний архітектурний підхід дозволить забезпечити чітке розмежування відповідальності між компонентами системи, підвищить її гнучкість та створить основу для подальшого розвитку програмного засобу.

При виборі архітектурного підходу має бути враховано особливості реалізації програмного засобу з використанням сучасних веб-технологій. Зокрема, застосування фреймворку Vue 3 [21] у поєднанні з середовищем Electron [22] дозволить створити настільний кросплатформений застосунок із використанням веб-інтерфейсу.

Використання компонентної архітектури Vue забезпечить розділення інтерфейсу на незалежні функціональні елементи, що відповідають за відображення окремих частин системи, таких як редактор схем, панель інструментів та область керування симуляцією. Це узгодиться із загальною багаторівневою архітектурою програмного засобу.

Для реалізації графічного редактора електропневматичних схем має бути використана технологія SVG, яка дозволить точно відображати елементи схеми та зможе забезпечити можливість їх інтерактивного редагування. SVG підтримає масштабування без втрати якості та дозволить ефективно реалізувати прив’язку з’єднань між елементами.

Рівень прикладної логіки має реалізуватися у вигляді централізованого керування станом, що забезпечить синхронізацію між інтерфейсом користувача та моделлю системи. Такий підхід дозволить уникнути розбіжностей між відображенням та фактичним станом елементів під час симуляції.

Обчислювальне ядро системи має реалізовувати алгоритми імітаційної симуляції, що базуються на дискретному поширенні сигналів між елементами електропневматичної схеми. Кожен елемент матиме власний стан, який

оновлюється відповідно до вхідних сигналів та логіки його функціонування.

Важливою перевагою обраної архітектури є можливість ізоляції обчислювального ядра від інтерфейсу користувача. Це дозволить в подальшому модифікувати алгоритми симуляції без необхідності змін у графічній частині системи, а також забезпечить можливість повторного використання ядра в інших програмних рішеннях.

Крім того, модульна структура програмного засобу дозволить розширювати перелік підтримуваних елементів електропневматичних схем шляхом додавання нових компонентів без зміни загальної архітектури системи.

2.2 Загальна структура програмного засобу

Кросплатформений програмний засіб для створення та імітаційного моделювання електропневматичних схем реалізується як багаторівнева система з модульною структурою. Такий підхід забезпечує розподіл функцій між окремими підсистемами, що підвищує гнучкість, спрощує розробку та дозволяє розширювати функціональні можливості без зміни загальної архітектури.

Загальна структура програмного засобу базується на трирівневій архітектурі, яка включає рівень представлення, рівень прикладної логіки та рівень моделювання. Взаємодія між рівнями здійснюється послідовно та забезпечує перетворення дій користувача у результати імітаційної симуляції [19].

Верхній рівень (рівень представлення) реалізує користувацький інтерфейс програмного засобу. На цьому рівні здійснюється створення та редагування електропневматичних схем, керування запуском симуляції, а також візуалізація результатів. Інтерфейс забезпечує відображення елементів схеми, з’єднань між ними та поточних станів компонентів.

Середній рівень (рівень прикладної логіки) виконує обробку команд користувача та координує роботу системи. Він забезпечує перевірку коректності дій, керування станом редактора, формування запитів до ядра моделювання та узгодження взаємодії між інтерфейсом і обчислювальним ядром. На цьому рівні реалізується контроль правильності побудови схеми та обмеження, пов’язані з

допустимими конфігураціями.

Нижній рівень (рівень моделювання) є обчислювальним ядром програмного засобу. Він відповідає за зберігання формалізованої моделі електропневматичної схеми, обробку зв’язків між елементами, поширення сигналів, оновлення станів компонентів та формування результатів симуляції. Робота цього рівня базується на алгоритмах дискретної імітаційної симуляції.

Взаємодія між рівнями здійснюється наступним чином: користувач формує дії через інтерфейс, середній рівень обробляє ці дії та передає відповідні команди до ядра моделювання, після чого результати обчислень повертаються у вигляді оновленого стану та відображаються користувачу.

Взаємодію між рівнями системи представлено на рисунку 2.2.

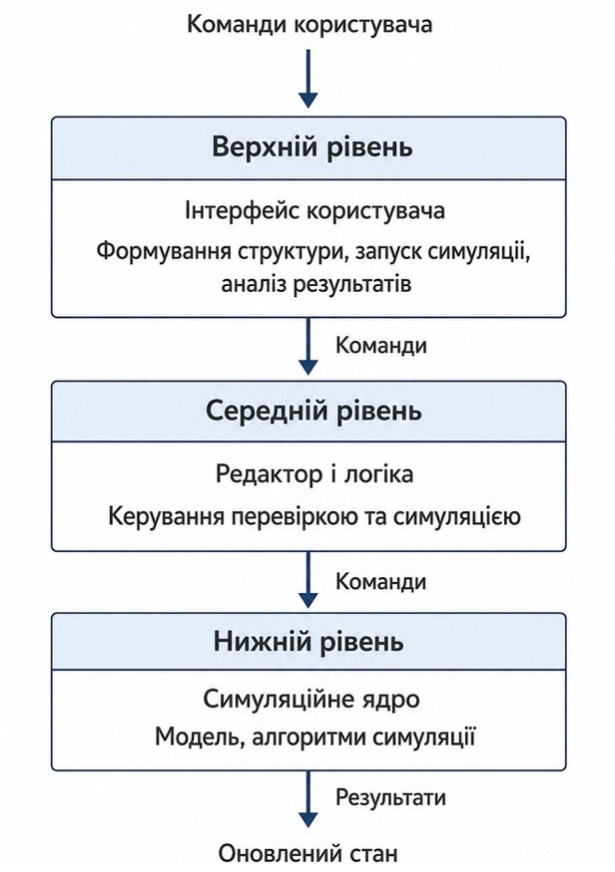


Рисунок 2.2 – Багаторівнева архітектура програмного засобу

З огляду на особливості предметної області, структура програмного засобу орієнтована на обробку дискретних подій та станів елементів електропневматичної схеми. Це зумовлює необхідність використання

централізованої моделі представлення даних, яка забезпечує узгодженість між рівнями системи.

У межах програмного засобу схема представляється у вигляді сукупності елементів та зв’язків між ними. Кожен елемент має визначений набір параметрів та станів, що змінюються у процесі симуляції. З’єднання між елементами визначають напрямок поширення сигналів та взаємодію між компонентами системи.

Керування станом системи реалізується за принципом єдиного джерела істини, при якому всі зміни відбуваються через централізовану модель даних. Такий підхід дозволяє забезпечити синхронізацію між інтерфейсом користувача та обчислювальним ядром, а також уникнути розбіжностей у відображенні результатів симуляції.

Процес функціонування програмного засобу можна подати у вигляді послідовності етапів: формування структури схеми, перевірка коректності, запуск симуляції, обчислення станів елементів та відображення результатів. Кожен з цих етапів відповідає окремій підсистемі та взаємодіє з іншими через загальну модель даних.

Важливою особливістю структури програмного засобу є підтримка розширюваності. Додавання нових типів елементів електропневматичних схем здійснюється шляхом розширення моделі даних та реалізації відповідної логіки їх обробки в ядрі симуляції без необхідності зміни інших підсистем.

Крім того, структура системи передбачає можливість повторного використання окремих модулів, зокрема ядра моделювання, що може бути застосоване в інших програмних рішеннях, пов’язаних із симуляцією технічних систем.

До складу системи доцільно включити наступні підсистеми:

– підсистема графічного редактора схем має відповідати за додавання, переміщення, видалення та з’єднання елементів на робочому полі;

– підсистема керування проєктом схеми має забезпечити створення нової схеми, збереження, завантаження та імпорт/експорт даних;

– підсистема симуляції має виконувати обчислення станів елементів,

визначає активні сигнали та оновлює результати моделювання;

– підсистема візуалізації станів має відображати користувачу поточний стан елементів та результати симуляції;

– підсистема контролю коректності має перевіряти правильність з’єднань, допустимість конфігурації та наявність помилок у схемі.

Декомпозиція підфункцій системи відображена на рис. 2.3.



Рисунок 2.3 – Декомпозиція функцій програмного засобу

2.3 Врахування принципів теорії автоматичного керування при проєктуванні програмного засобу

Електропневматичні системи належать до класу автоматизованих систем керування, у яких виконання технологічних операцій забезпечується шляхом взаємодії керуючої та виконавчої підсистем. При проєктуванні програмного засобу доцільно враховувати базові поняття теорії автоматичного керування, зокрема поняття об'єкта керування, керуючого впливу, зворотного зв'язку та стану системи.

Об'єктом керування в електропневматичній системі є виконавчий механізм, який безпосередньо виконує необхідну технологічну операцію. У більшості навчальних схем таким об'єктом виступає пневмоциліндр двосторонньої дії, що забезпечує лінійне переміщення робочого органу. У мехатронних модулях пакувальних машин саме пневматичний привід виконує функцію переміщення продукції або робочих елементів механізму [10].

Керуючий вплив являє собою сигнал, який формується системою керування та подається на виконавчий механізм. У електропневматичних системах такий вплив реалізується через електромагнітні розподільники, котушки яких отримують сигнали від кнопок, датчиків, реле або програмованого логічного контролера. У позиційних пневматичних приводах контролер обробляє інформацію від датчиків та формує сигнали керування електромагнітним розподільником, який визначає напрямок руху пневмоциліндра.

Одним із основних принципів автоматичного керування є використання зворотного зв'язку. Зворотний зв'язок забезпечує передачу інформації про поточний стан об'єкта керування до системи керування для подальшого коригування її роботи. У слідкувальних системах сигнал зворотного зв'язку формується за допомогою датчиків переміщення, швидкості або тиску, а система керування використовує його для зменшення похибки позиціонування. При цьому керуючий вплив подається на вхід приводу, а інформація про фактичне переміщення отримується з його виходу. Саме такий принцип лежить в основі

побудови систем автоматичного керування зі зворотним зв'язком.

Похибка керування визначається як різниця між заданим та фактичним значенням керованої величини та визначається за (2.1):

$$e(t) = x_{зад}(t) - x(t), \quad (2.1)$$

де e(t) – похибка керування;

$x_{зад}(t)$ – задане значення (заданий сигнал);

x(t) – фактичне значення вихідної величини.

У типових електропневматичних схемах навчального призначення функцію зворотного зв'язку виконують кінцеві вимикачі та датчики положення. Їх спрацювання використовується для формування нових керуючих сигналів та реалізації автоматичних циклів роботи. Наявність зворотного зв'язку дозволяє реалізувати послідовності виду: висування циліндра → спрацювання датчика → формування нового сигналу → повернення циліндра.

Відпрацювання керуючого сигналу системою зі зворотним зв'язком наведено на рисунку 2.4.

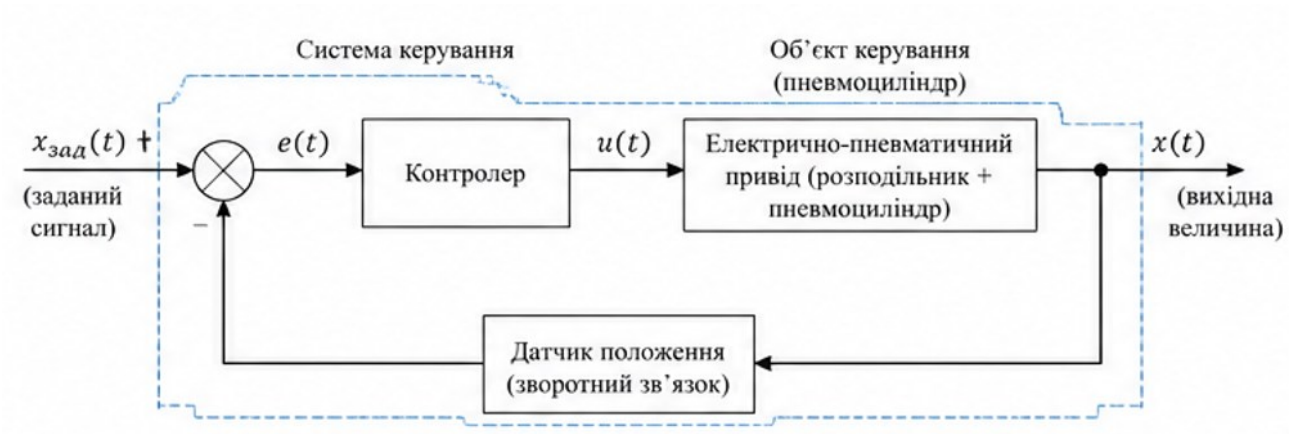


Рисунок 2.4 – Відпрацювання керуючого сигналу системою зі зворотним зв'язком

На рисунку відображений типовий процес позиціонування виконавчого механізму. Під впливом керуючого сигналу фактичне положення поступово наближається до заданого значення, а похибка керування зменшується.

Аналогічний принцип використовується у слідкувальних пневматичних системах зі зворотним зв'язком.

Стан системи визначається сукупністю параметрів, що характеризують її поточне функціонування. Для електропневматичної системи такими параметрами є положення розподільників, наявність або відсутність тиску в магістралях, стан електричних сигналів та положення виконавчих механізмів. Перехід системи з одного стану до іншого відбувається внаслідок надходження керуючих сигналів або спрацювання елементів зворотного зв'язку.

З точки зору формалізації множину можливих станів електропневматичної системи можна подати у вигляді (2.2):

$$S = \{s_1, s_2, ..., s_n\}, \quad (2.2)$$

де S – множина можливих станів системи;

s_i – окремий стан, що визначається положенням виконавчих механізмів, станом розподільників та наявністю тиску в пневматичних магістралях.

Застосування зворотного зв'язку є характерним для сучасних електропневматичних систем автоматизації, де необхідно забезпечити точне позиціонування виконавчих механізмів та узгодження фактичних параметрів роботи із заданими значеннями. Саме тому принципи зворотного зв'язку широко використовуються при побудові автоматизованих виробничих та мехатронних систем.

Під час проєктування кросплатформеного програмного засобу для моделювання електропневматичних схем зазначені принципи мають бути враховані на рівні моделі даних та алгоритмів симуляції. Електропневматична схема розглядається як множина взаємопов'язаних елементів, кожен з яких має власний стан та правила його зміни. Такий підхід дозволить представити роботу системи у вигляді послідовності дискретних переходів між станами та забезпечить коректне моделювання схем із прямим керуванням, зворотним зв'язком і циклічними алгоритмами функціонування.

2.4 Проєктування моделі даних електропневматичних схем

Проєктування моделі даних є важливим етапом розробки програмного засобу, оскільки саме вона визначає спосіб представлення електропневматичної схеми у програмному середовищі та забезпечує основу для реалізації алгоритмів імітаційної симуляції.

З урахуванням специфіки предметної області електропневматична схема розглядається як сукупність взаємопов’язаних елементів, між якими встановлюються логічні та функціональні зв’язки. Для формалізації такої структури доцільно використати графову модель, у якій елементи виступають вершинами, а з’єднання – ребрами графа.

Основними сутностями моделі даних є:

– елемент (Element);

– порт (Port);

– з’єднання (Connection).

Опис сутностей моделі даних.

Елемент (Element) є базовою структурною одиницею електропневматичної схеми. Кожен елемент має унікальний ідентифікатор, тип (наприклад, клапан, циліндр, джерело повітря, кнопка), позицію на робочому полі та внутрішній стан.

Елемент також містить набір портів, через які здійснюється взаємодія з іншими елементами.

Порт (Port) визначає точку підключення елемента до інших компонентів. Кожен порт має унікальний ідентифікатор, належить конкретному елементу та характеризується типом (електричний або пневматичний), напрямком (вхід/вихід) та роллю (наприклад, P, A, B, R для пневматичних портів).

З’єднання (Connection) описує зв’язок між двома портами різних елементів. З’єднання визначає напрямок передачі сигналу або потоку та забезпечує взаємодію між компонентами системи.

Модель даних електропневматичної системи представлена на рис. 2.5.

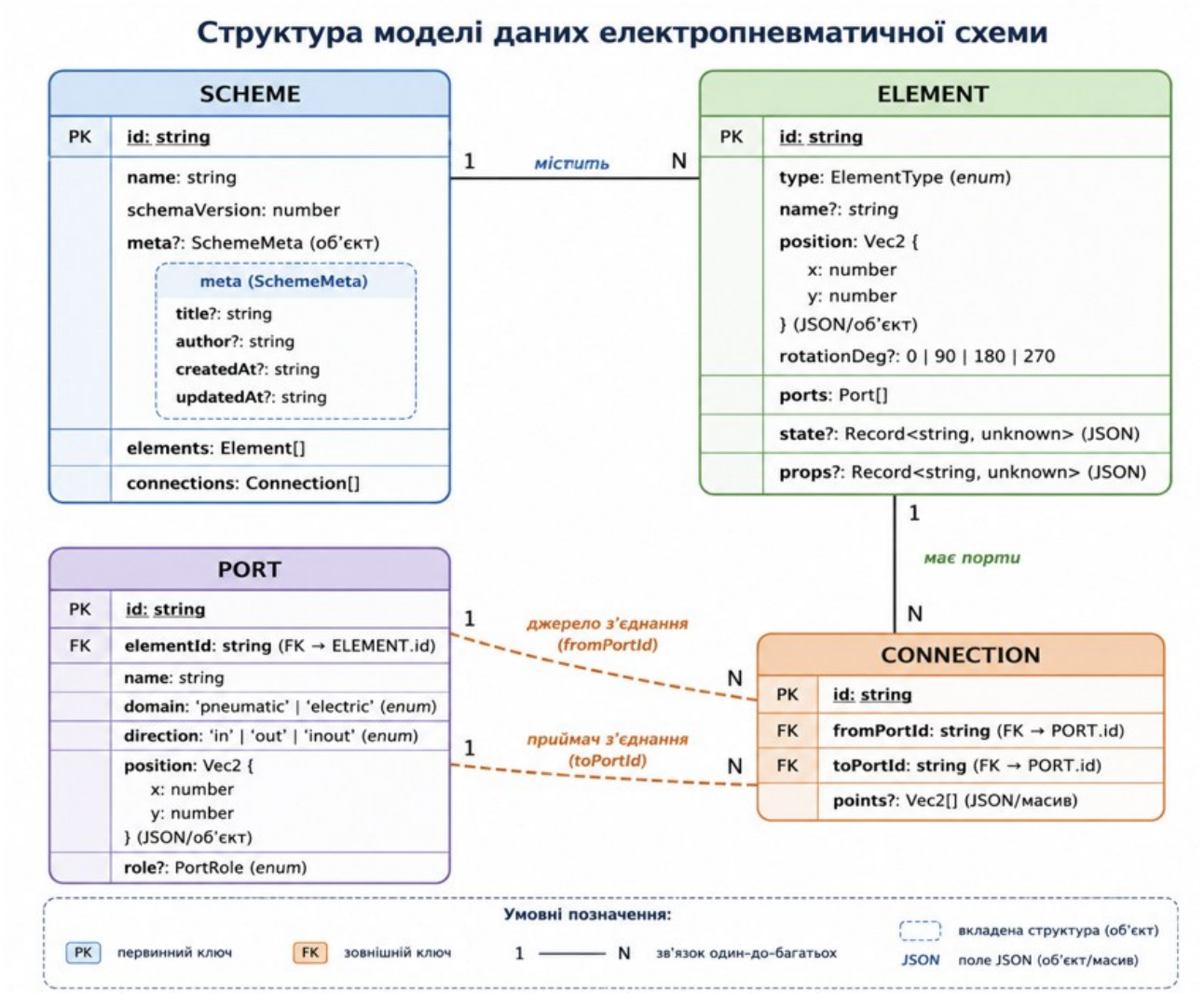


Рисунок 2.5 – Структура моделі даних електропневматичної схеми

Модель даних електропневматичної схеми може бути представлена у вигляді набору взаємопов’язаних колекцій:

– множина елементів;

– множина портів;

– множина з’єднань.

Кожен елемент містить список власних портів, а з’єднання визначають зв’язки між портами різних елементів.

Взаємозв’язки між сутностями мають наступний вигляд:

– один елемент може містити декілька портів;

– один порт належить лише одному елементу;

– одне з’єднання пов’язує два порти.

2.5 Висновки до другого розділу

У другому розділі було виконано проєктування кросплатформеного програмного засобу для створення та імітаційного моделювання електропневматичних схем.

Обґрунтовано вибір багаторівневої архітектури, яка передбачає розподіл системи на рівень представлення, рівень прикладної логіки та рівень моделювання. Такий підхід забезпечує розділення відповідальності між компонентами системи, спрощує розробку та створює можливість подальшого розширення функціональних можливостей програмного засобу.

Розроблено загальну структуру системи, у межах якої визначено основні підсистеми: графічний редактор схем, підсистему керування проєктом, підсистему симуляції, підсистему візуалізації та підсистему контролю коректності. Встановлено принципи взаємодії між підсистемами та рівнями архітектури.

Побудовано функціональну модель програмного засобу, яка відображає основні процеси створення та симуляції електропневматичних схем, а також визначає вхідні дані, керуючі впливи, механізми реалізації та результати функціонування системи.

Розроблено модель даних електропневматичної схеми, яка базується на графовому представленні та включає сутності: елемент, порт та з’єднання. Визначено структуру взаємозв’язків між цими сутностями, що дозволяє формалізувати електропневматичну схему у програмному середовищі.

3 РОЗРОБКА АЛГОРИТМУ ІМІТАЦІЙНОЇ СИМУЛЯЦІЇ ЕЛЕКТРОПНЕВМАТИЧНОЇ СХЕМИ

3.1 Формалізація елементів електропневматичної схеми

Формалізація елементів електропневматичної схеми є основою для реалізації імітаційної симуляції. У розробленій системі вона базується на уніфікованій моделі Element / Port / State, яка дозволяє описати як структуру схеми, так і її поведінку під час виконання алгоритму симуляції.

Кожен елемент схеми подається як структурована сутність:

– елемент (Element) – логічна одиниця схеми;

– порти (Port) – точки взаємодії елемента із зовнішнім середовищем;

– стан (State) – набір змінних, що визначають поточну поведінку елемента;

– runtime-стан – динамічні значення, що обчислюються під час симуляції.

Формально елемент можна представити як:

Element = {
id: string,
type: ElementType,
ports: Port[],
state: ElementState,
runtime: ElementRuntime
}

Порт є базовою одиницею взаємодії між елементами. Через порти відбувається передача сигналів (електричних або пневматичних) та тиску.

Кожен порт описується такими атрибутами:

id – унікальний ідентифікатор;

role – функціональна роль (наприклад: P, A, R, control_in);

direction – напрямок (in, out, inout);

domain – домен сигналу (pneumatic, electric, control).

Port = {

```
  id: string,
  role: string,
  direction: 'in' | 'out' | 'inout',
  domain: 'pneumatic' | 'electric' | 'control'
}
```

Загальна модель елемента відображена на рис. 3.1.

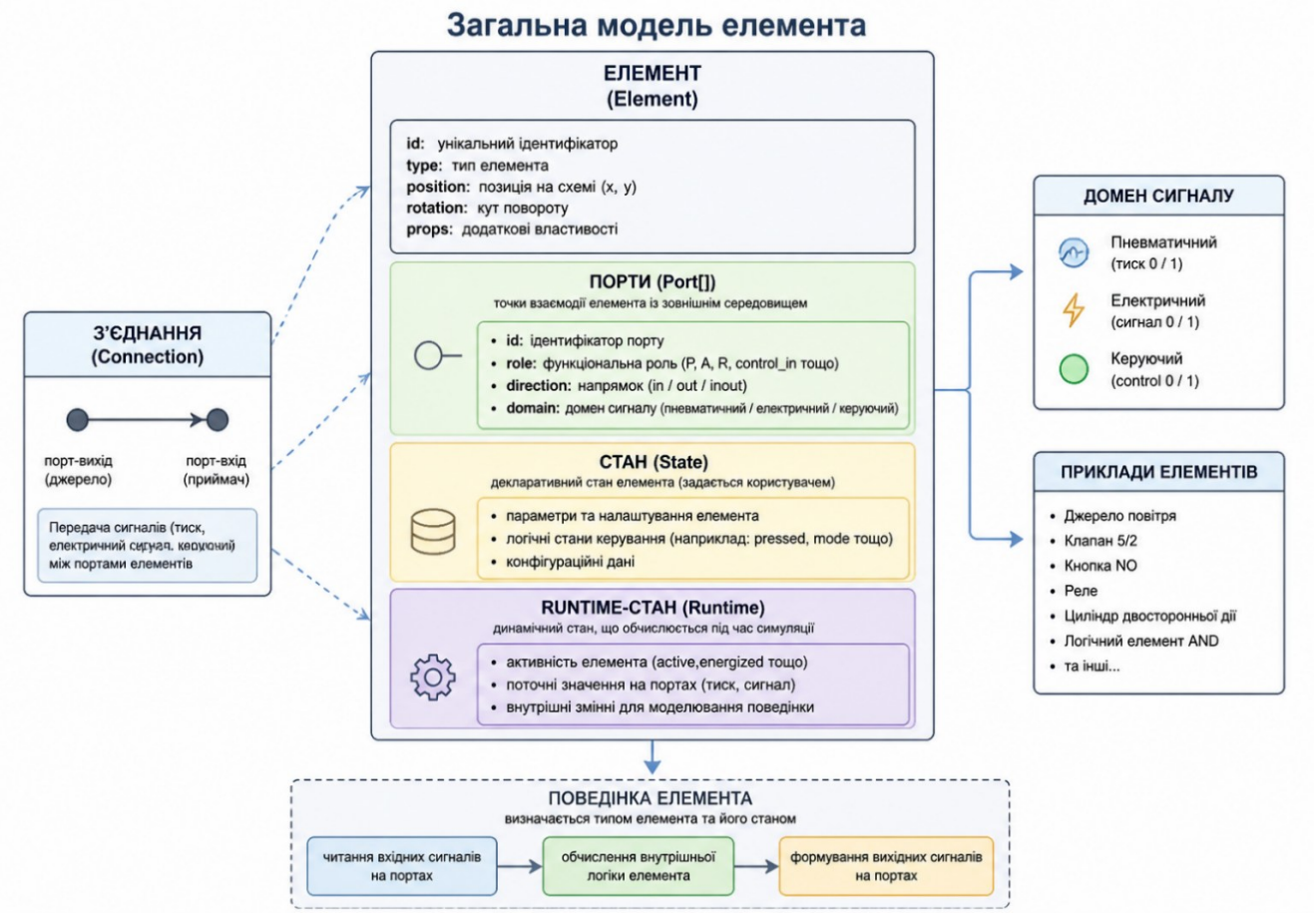


Рисунок 3.1 – Загальна модель елемента

З'єднання визначає передачу сигналу між портами різних елементів.

У процесі симуляції з'єднання використовуються для поширення тиску, передачі електричного сигналу та формування графа залежностей.

Стан елемента поділяється на два рівні декларативний стан (state), який задається користувачем або фабрикою та runtime-стан (runtime), який обчислюється під час симуляції. Такий поділ дозволяє відокремити UI-стан від логіки симуляції та реалізувати детермінований алгоритм оновлення.

У системі має бути реалізовано кілька типів елементів, що відрізняються поведінкою. Для систематизації формалізованих елементів електропневматичної схеми виконано їх класифікацію за їх функціональним призначенням та роллю у процесі симуляції. Для кожного класу визначено принцип роботи, який у програмній системі реалізується у вигляді окремого правила обробки стану або маршрутизації сигналу. Такий підхід дозволяє описувати різні елементи через єдину модель Element / Port / State та забезпечує можливість подальшого розширення бібліотеки компонентів. Результати наведено в таблиці 3.1.

Таблиця 3.1 – Класифікація елементів електропневматичної схеми

Клас елементів	Призначення	Принцип роботи	Приклад формалізації
Джерела сигналу	Формують початковий сигнал у схемі	Елемент не залежить від вхідних портів і задає сигнал на виході	OUT.pressure = runtime.on ? 1 : 0
Транзитні елементи	Передають сигнал між частинами схеми	Вхідний сигнал передається на вихід без зміни або з мінімальною обробкою	OUT.pressure = IN.pressure
Керуючі елементи	Формують сигнал залежно від дії користувача або стану механізму	Стан елемента визначає, чи буде подано вихідний сигнал	active = state.pressed
Керуючі елементи з інверсною логікою	Формують сигнал у початковому стані та вимикають його при активації	Вихідний сигнал є протилежним до стану натискання	OUT.signal = active ? IN.signal: 0
Електричні комутаційні елементи	Передають електричний сигнал за умови активації	Якщо елемент активний, сигнал з входу передається на вихід	OUT.signal = active ? IN.signal: 0

Продовження таблиці 3.1

Клас елементів	Призначення	Принцип роботи	Приклад формалізації
Електричні комутаційні елементи	Передають електричний сигнал за умови активації	Якщо елемент активний, сигнал з входу передається на вихід	OUT.signal = active ? IN.signal: 0
Перехідні електромагні тні елементи	Зв’язують електричну частину з пневматичною	Електричний сигнал переводиться у внутрішній стан збудження	energized = readElectricSignl (A1) == 1
Логічні пневматичні елементи AND	Реалізують логічну операцію “І” для пневматичних сигналів	Тиск на виході з’являється лише за наявності тиску на всіх входах	OUT.pressure = min(IN1.pressure, IN2.pressure)
Логічні пневматичні елементи OR	Реалізують логічну операцію “АБО” для пневматичних сигналів	Тиск на виході з’являється, якщо активний хоча б один вхід	OUT.pressure = max(IN1.pressur, IN2.pressure)
Розподільник	Перемикають маршрути проходження тиску між портами	Поточне положення клапана визначає активні внутрішні маршрути	activeRoutes = position == 'left' ? ['P-B', 'A-R'] : ['P-A', 'B-S']
Імпульсний розподільник	Зберігають положення після короткого керуючого сигналу	Нове положення задається пілотом, а за відсутності нового сигналу зберігається попередній стан	position = pilotLeft ? 'left' : pilotRight ? 'right': previousPosition
Виконавчі елементи	Перетворюють пневматичний сигнал у механічну дію	Положення виконавчого механізму змінюється залежно від тиску на робочих портах	rod = clamp(rod + step, 0, 1)

Продовження таблиці 3.1

Клас елементів	Призначення	Принцип роботи	Приклад формалізації
Елементи регулювання швидкості	Впливають на швидкість зміни стану виконавчого елемента	Крок переміщення визначається параметрами з'єднання або регулятора витрати	step = resolveCylinderSpeedStep(connection, defaultStep)
Елементи з пам'яттю	Зберігають попередній стан за відсутності однозначного нового сигналу	Якщо вхідні сигнали не визначають новий стан, використовується попередній runtime-стан	nextState = inputDefined ? newState : previousRuntime.state

Особливістю розроблюваної системи є підтримка dynamic ports, що дає можливість зміни набору портів залежно від стану елемента. Це використовується для клапанів з різними конфігураціями, елементів зі змінною топологією та оптимізації моделі (активні тільки релевантні порти).

Формально:

ports = resolvePorts(element.state)

3.2 Моделювання станів пневматичної частини

Моделювання станів пневматичної частини у розроблюваній системі базується на спрощеному підході, що дозволяє відтворювати логіку роботи електропневматичних схем без використання складних фізичних моделей. Основною метою є забезпечення коректного поширення тиску між елементами та визначення їхнього поточного стану у процесі симуляції.

У системі пневматичний сигнал має бути представлено у вигляді бінарної величини: 0 – відсутність тиску; 1 – наявність тиску. Це дозволить спростити алгоритм симуляції, забезпечити детермінованість обчислень та досягти високої продуктивності.

Формально: pressure ∈ {0, 1}

Кожен пневматичний порт елемента має поточне значення тиску, яке зберігається у runtime-стані:

```
portRuntime = {
  pressure: 0 | 1
}
```

Значення тиску на порту має формуватися в результаті роботи самого елемента та впливу з’єднань із іншими елементами.

Для забезпечення коректної роботи симуляції стан кожного пневматичного елемента повинен визначатися на основі значень тиску на його вхідних портах та внутрішніх правил функціонування. Наприклад, для пневматичних розподілювачів положення золотника визначає напрямок передачі тиску між портами, а для логічних елементів результат формується відповідно до реалізованої логічної функції. Таким чином, кожен елемент виконує локальне перетворення вхідних сигналів у вихідні.

Моделювання станів має виконуватися дискретно у межах окремих тактів симуляції. На кожному такті відбувається зчитування поточних значень тиску, обчислення нового стану елементів та оновлення значень на вихідних портах. Такий підхід забезпечує передбачуваність результатів моделювання та виключає неоднозначності, які можуть виникати під час одночасної зміни станів декількох взаємопов’язаних компонентів.

Отримані стани елементів та значення тиску на портах використовуються не лише для подальших розрахунків, а й для візуалізації результатів моделювання у графічному редакторі. Це дозволить користувачу спостерігати за процесом проходження пневматичних сигналів по схемі, аналізувати роботу окремих компонентів та перевіряти правильність побудованої електропневматичної системи.

Обраний підхід до моделювання дозволить абстрагуватися від складних фізичних процесів і зосередитися на логіці функціонування електропневматичної системи.

Моделювання станів пневматичної частини представлено на рисунку 3.2.

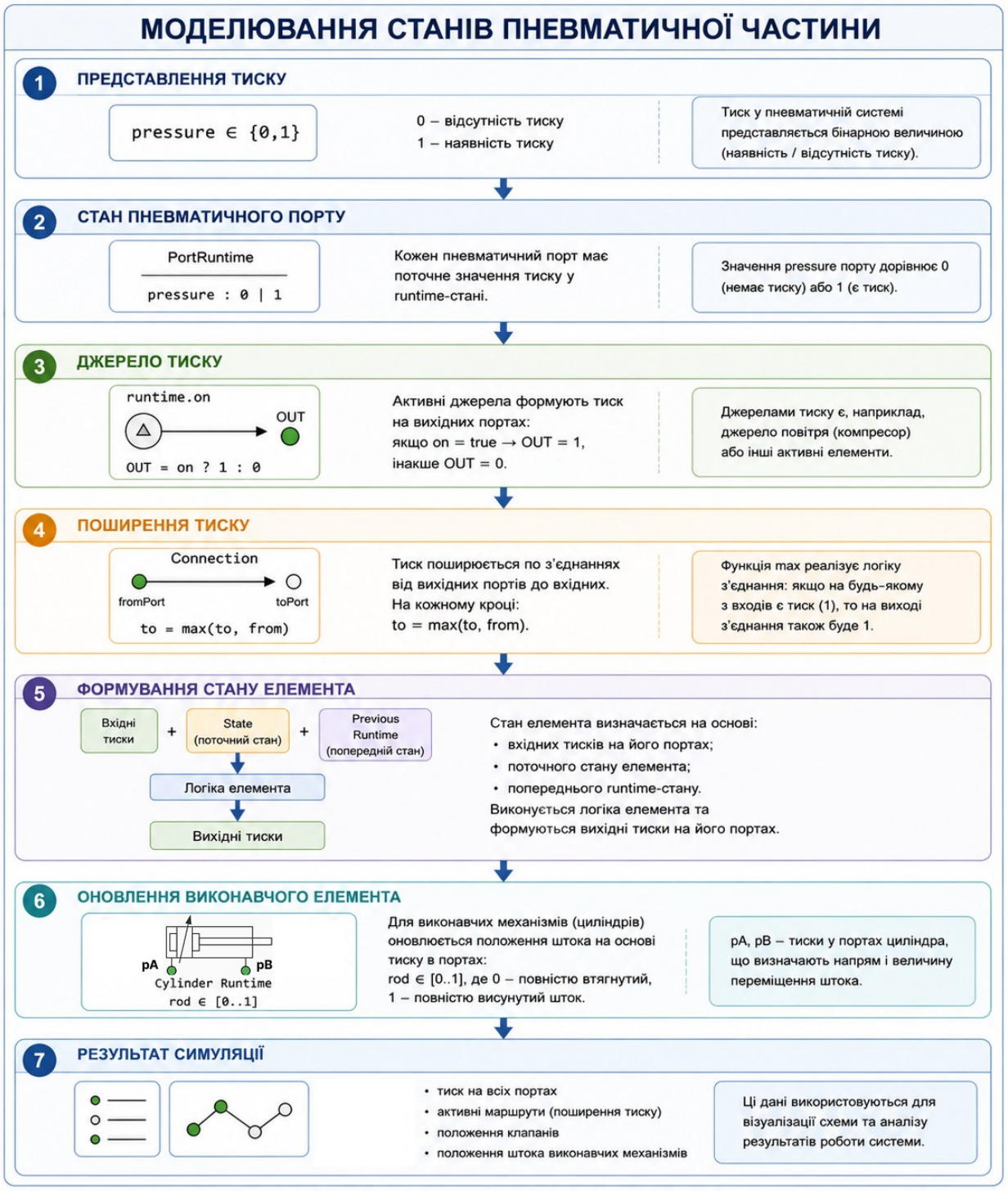


Рисунок 3.2 – Моделювання станів пневматичної частини

Моделювання станів пневматичної частини базується на дискретному представленні тиску та ітераційному алгоритмі поширення сигналів у графі схеми. Для систематизації підходу основні етапи та принципи моделювання наведено в таблиці 3.2.

Таблиця 3.2 – Основні етапи та принципи моделювання

Етап / компонент	Призначення	Вхідні дані	Принцип моделювання	Приклад формалізації
Представлення сигналу	Уніфікація пневматичного сигналу	–	Тиск подається у бінарному вигляді	pressure ∈ {0,1}
Стан порту	Збереження поточного тиску	Сигнали від з’єднань	Кожен порт має runtime-значення	port.pressure = 0
Ініціалізація	Початковий стан перед симуляцією	–	Всі порти обнуляються	for port: pressure = 0
Джерело тиску	Формування базового сигналу	runtime.on	Джерело генерує тиск незалежно від входів	OUT = on ? 1 : 0
Передача через з’єднання	Поширення тиску між елементами	from.port	Тиск передається по графу	to = max(to, from)
Логічний AND	Обробка двох входів	p1, p2	Вихід активний тільки при двох сигналах	OUT = min(p1, p2)
Логічний OR	Обробка альтернативних входів	p1, p2	Вихід активний при хоча б одному сигналі	OUT = max(p1, p2)
Маршрутизація клапана	Перемикання каналів	position	Активні тільки певні маршрути	routes = position ? R1 : R2

Продовження таблиці 3.2

Етап / компонент	Призначення	Вхідні дані	Принцип моделювання	Приклад формалізації
Передача через маршрут	Внутрішня передача в клапані	pressure[from]	Тиск передається по активному каналу	if from=1 → to=1
Імпульсний клапан	Збереження положення	pilot signals	Положення змінюється імпульсом	pos = new ? val : prev
Виконавчий елемент (циліндр)	Моделювання руху	pA, pB	Зміна положення штока	rod += step
Обмеження руху	Контроль діапазону	rod	Значення нормалізується	rod = clamp(0,1)
Швидкість руху	Вплив дроселя	connection	Крок залежить від обмеження потоку	step = resolveCylinderSpeedStep()
Ітераційне поширення	Досягнення стабільності	всі порти	Повторюється до відсутності змін	while changed
Memory-поведінка	Збереження стану	previous runtime	Використовується попередній стан	state = prev

3.3 Алгоритм поширення сигналів

Алгоритм поширення сигналів у розроблюваній системі призначений для визначення поточних значень тиску, електричних сигналів і керуючих сигналів на портах елементів схеми. Оскільки схема складається з елементів, портів і з’єднань, процес симуляції можна подати як обчислення станів у графі, де порти є вузлами, а з’єднання є ребрами між ними.

Поширення сигналів має виконуватися дискретно, тобто у вигляді окремих кроків симуляції. На кожному кроці система зчитує поточні стани елементів, визначає вихідні сигнали елементів, передає сигнали через з’єднання, оновлює runtime-стани елементів та формує новий стан схеми для відображення в редакторі.

У межах одного кроку симуляції сигнал не передається безпосередньо «миттєво» через усю схему одним оператором. Для цього використовується ітеративне поширення, яке дозволяє врахувати послідовні зв’язки між елементами.

Для виконання симуляції схема інтерпретується як граф:

– вершини графа (порти елементів);

– ребра графа (з’єднання між портами);

– значення у вершинах (поточний сигнал або тиск);

– правила елементів (локальні функції перетворення сигналів).

Формально з’єднання можна подати так:

```
Connection = {
  fromElementId,
  fromPortId,
  toElementId,
  toPortId
}
```

Кожен порт має значення сигналу відповідно до свого домену:

```
PortSignal = {
  pressure?: 0 | 1,
  electric?: 0 | 1,
  control?: 0 | 1
}
```

Алгоритм поширення сигналів представлений на рис. 3.3.

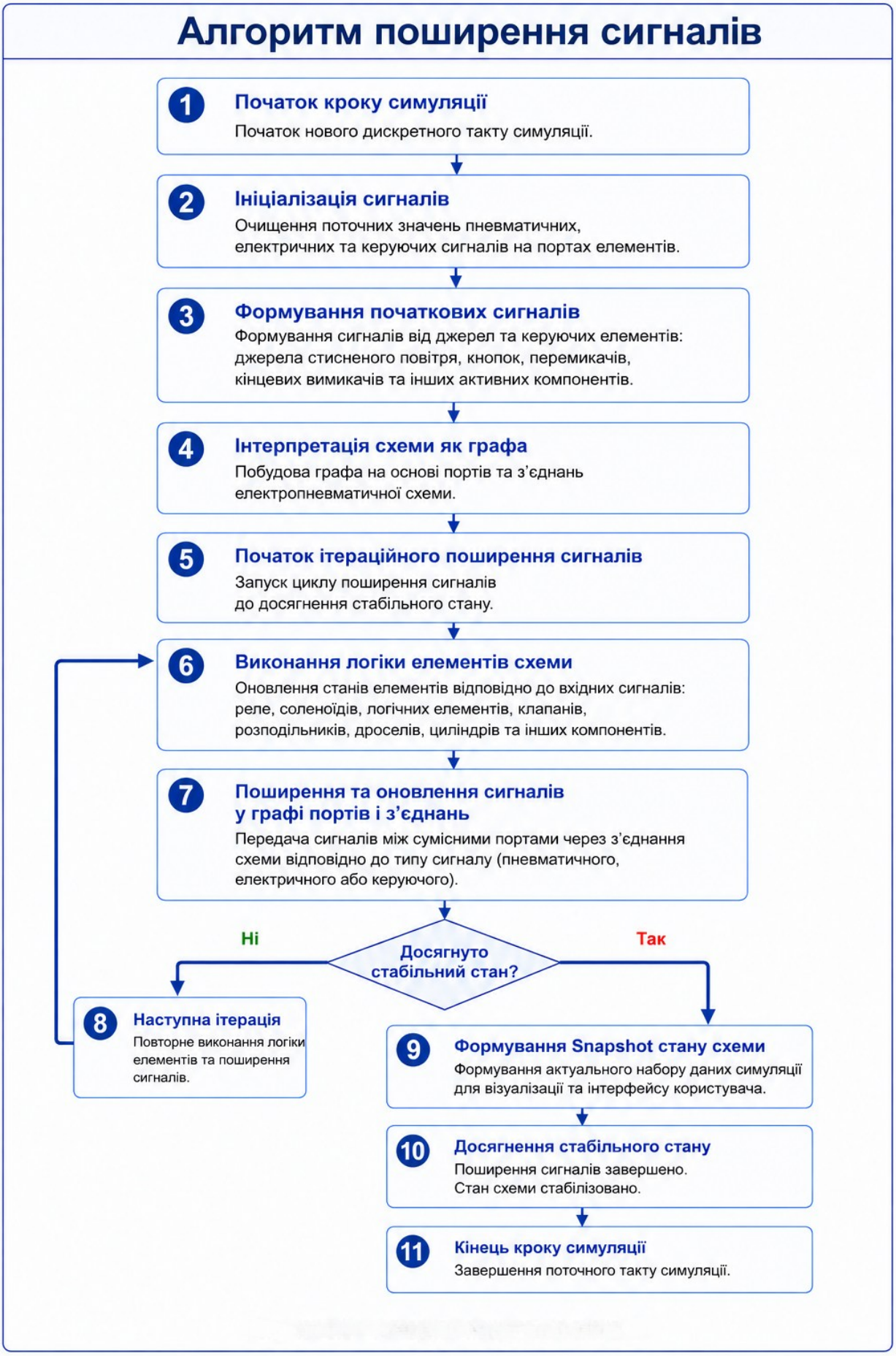


Рисунок 3.3 – Алгоритм поширення сигналів

У системі використовується поділ сигналів на домени:

– пневматичний домен (передача тиску);

– електричний домен (передача електричних сигналів);

– керуючий домен (передача логічних команд керування).

Такий поділ потрібний для того, щоб уникнути некоректного з’єднання різнорідних сигналів. Наприклад, пневматичний тиск не повинен безпосередньо передаватися в електричний порт, а електричний сигнал не повинен трактуватися як тиск. Передача сигналу дозволяється лише тоді, коли домени портів сумісні:

```
if fromPort.domain == toPort.domain:
   transferSignal(fromPort, toPort)
else:
   ignoreConnection()
```

На початку кожного кроку симуляції значення сигналів на портах скидаються до нульового стану:

```
for each port:
   signal[port] = 0
```

Після цього активні джерела формують початкові сигнали. Наприклад, джерело стисненого повітря встановлює тиск на вихідному порту:

```
if airSource.runtime.on == true:
   signal[airSource.OUT].pressure = 1
```

Також на цьому етапі враховуються елементи, які формують електричні або керуючі сигнали на основі свого state, наприклад кнопки.

Кожен елемент має власне правило формування вихідних сигналів. Це правило залежить від типу елемента, стану елемента, вхідних сигналів та попереднього runtime-стану. Наприклад, для нормально відкритої кнопки:

```
if button.pressed == true:
   OUT.electric = 1
else:
   OUT.electric = 0
```

Для логічного пневмоклапана AND:

```
OUT.pressure = min(IN_LEFT.pressure, IN_RIGHT.pressure)
```

Для імпульсного клапана 5/2 спочатку визначається положення золотника, а потім активні маршрути:

```
if pilot_left == 1 and pilot_right == 0:
    position = 'left'
else if pilot_right == 1 and pilot_left == 0:
    position = 'right'
else:
    position = previousPosition
if position == 'left':
    routes = [P -> B, A -> R]
else:
    routes = [P -> A, B -> S]
```

Після формування вихідних сигналів елементів виконується передача сигналів через з’єднання. Для бінарної моделі використовується операція максимального значення:

```
toSignal = max(toSignal, fromSignal)
```

Для пневматичного тиску це означає: якщо хоча б один підключений вихід подає тиск, на вхідному порту приймача також буде тиск.

Окремо враховується маршрутизація всередині елементів, зокрема клапанів. Клапан не просто передає тиск між зовнішніми з’єднаннями, а визначає, які його внутрішні канали активні у поточному стані.

Наприклад, для клапана 5/2:

```
if position == 'left':
    activeRoutes = [
        P -> B,
        A -> R
    ]
if position == 'right':
    activeRoutes = [
        P -> A,
        B -> S
```

]

Після визначення активних маршрутів тиск передається між портами клапана:

```
for each route in activeRoutes:
    if pressure[route.from] == 1:
        pressure[route.to] = 1
```

Таким чином, маршрути є внутрішніми з’єднаннями елемента, які залежать від його стану.

Оскільки вихід одного елемента може бути входом іншого, поширення сигналів виконується ітеративно. На кожній ітерації система перевіряє, чи з’явилися нові значення сигналів на портах.

```
changed = true
while changed:
    changed = false
    for each element:
        oldSignals = copy(signals)
        evaluateElement(element, signals)
        if signals != oldSignals:
            changed = true
    for each connection:
        oldSignals = copy(signals)
        propagateConnection(connection, signals)
        if signals != oldSignals:
            changed = true
```

Цикл завершується тоді, коли нові сигнали більше не з’являються. Це означає, що для поточного кроку симуляції схема досягла стабільного стану.

Для деяких елементів поточний стан залежить від попереднього. До таких елементів належать, наприклад, імпульсні клапани та OR-клапани з memory-поведінкою. У цьому випадку алгоритм не скидає внутрішній стан елемента повністю, а використовує попереднє значення runtime:

```
if inputSignal defines new state:
```

```
    runtime.position = newPosition
else:
    runtime.position = runtime.previousPosition
```

Це дозволяє моделювати фіксацію положення клапана після короткого керуючого імпульсу.

Результатом виконання алгоритму поширення сигналів є набір актуальних значень:

- тиск на пневматичних портах;
- електричні сигнали на електричних портах;
- керуючі сигнали;
- положення клапанів;
- положення штока циліндра;
- стани активності елементів.

Ці дані використовуються для оновлення runtime-станів, візуалізації схеми, відображення активних маршрутів та анімації виконавчих елементів.

Алгоритм поширення сигналів забезпечує зв’язок між окремими елементами електропневматичної схеми. Завдяки представленню схеми як графа портів і з’єднань, а також використанню фазового та ітеративного підходу, система дозволяє визначати поточні значення тиску та сигналів у всій схемі.

Запропонований алгоритм є достатнім для реалізації навчальної імітаційної симуляції електропневматичних схем, оскільки дозволяє моделювати роботу джерел тиску, кнопок, логічних елементів, клапанів і циліндрів без складних фізичних розрахунків. Алгоритм має детермінований характер і забезпечує досягнення стабільного стану системи за скінченну кількість ітерацій.

3.4 Алгоритм оновлення станів елементів

Алгоритм оновлення станів елементів призначений для перерахунку внутрішніх параметрів схеми після поширення сигналів. Якщо алгоритм поширення сигналів визначає, які значення тиску або електричного сигналу присутні на портах, то алгоритм оновлення станів визначає, як ці сигнали

впливають на поведінку конкретних елементів.

У розроблюваній системі кожен елемент має два рівні стану:

– state – декларативний стан елемента;

– runtime – динамічний стан, що оновлюється під час симуляції.

state використовується для збереження параметрів, які задаються користувачем або фабрикою елемента. Наприклад, для кнопки це може бути pressed, а для циліндра це початкове положення або крок руху.

runtime використовується для збереження результатів симуляції: активності елемента, положення клапана, тиску на входах, вихідного тиску, положення штока тощо.

Алгоритм оновлення станів елементів забезпечує перехід від значень сигналів на портах до конкретної поведінки елементів схеми. Завдяки поділу на декларативний state і динамічний runtime система може одночасно зберігати параметри елементів та обчислювати їхній поточний стан під час симуляції. Такий підхід дозволяє моделювати кнопки, реле, соленоїди, логічні пневматичні елементи, розподільники та циліндри в межах єдиного алгоритму. Він також забезпечує основу для візуалізації результатів симуляції у SVG-редакторі.

Оновлення станів виконується після того, як на поточному кроці симуляції були визначені значення сигналів на портах.

Загальна послідовність має такий вигляд:

– скидання сигналів на портах;

– формування сигналів джерел;

– поширення сигналів через з’єднання;

– визначення активних маршрутів елементів;

– оновлення runtime-станів елементів;

– передача оновлених станів у UI.

Таким чином, стан елемента оновлюється не ізольовано, а на основі вже обчисленої картини сигналів у схемі.

Оновлення станів елементів в процесі симуляції представлений на рис. 3.4.

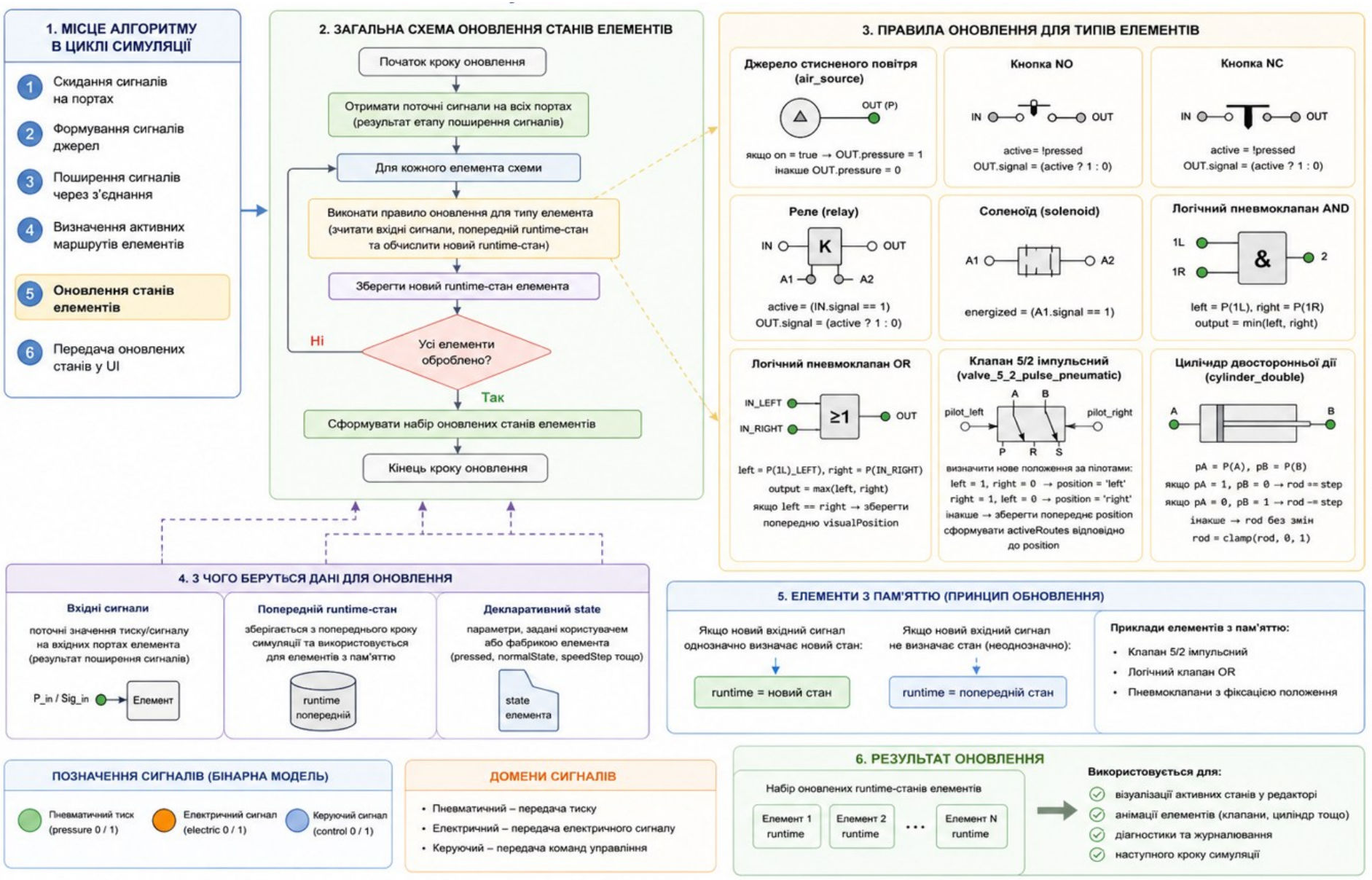


Рисунок 3.4 – Стани елементів в процесі симуляції

Алгоритм оновлення станів елементів визначає перехід від значень сигналів на портах до внутрішніх runtime-станів елементів. Для кожного типу елемента використовується власне правило оновлення, що базується на вхідних сигналах, декларативному стані та попередньому runtime-станів.

Основні етапи алгоритму наведено в таблиці 3.3.

Таблиця 3.3 – Основні етапи та принципи моделювання

Етап / компонент	Призначення	Вхідні дані	Принцип оновлення	Приклад формалі-зації
Зчитування сигналів	Отримання актуальних значень на портах	Signals [ports]	Для кожного елемента читаються значення на входах	inputs = readInput Signals(el ement)
Доступ до попереднього стану	Підтримка memory-поведінки	previousRu ntime	Використовується попередній runtime-стан	prev = runtime[el ement.id]
Оновлення джерела	Формування базового стану	state.on	Джерело визначає тиск незалежно від входів	OUT = on ? 1 : 0
Оновлення кнопки NO	Формування сигналу	state.pressed	Активність дорівнює натисканню	active = pressed
Оновлення кнопки NC	Інверсна логіка	state.pressed	Активність протилежна натисканню	active = !pressed
Оновлення реле	Комутація сигналу	IN.signal	Активується при наявності сигналу	active = IN == 1
Оновлення соленоїда	Формування стану котушки	A1.signal	Визначення збудження	energized = A1 == 1
Логічний AND	Обробка пневматичних сигналів	p1, p2	Вихід тільки при двох сигналах	OUT = min(p1, p2)

Продовження таблиці 3.3

Етап / компонент	Призначення	Вхідні дані	Принцип оновлення	Приклад формалі-зації
Логічний OR	Альтернативна логіка	p1, p2, prev	Вихід при одному сигналі + memory	OUT = max(p1, p2)
Оновлення положення клапана	Визначення стану розподільника	pilot signals, prev	Перемикання або збереження стану	pos = new ? val : prev
Формування маршрутів	Визначення каналів тиску	position	Активні тільки певні маршрути	routes = positionMap[pos]
Оновлення циліндра	Моделювання руху	pA, pB, prev.rod	Зміна положення штока	rod += step * direction
Обмеження руху	Контроль меж	rod	Значення в межах 0..1	rod = clamp(rod 0, 1)
Врахування швидкості	Моделювання дроселя	connection	Крок залежить від обмеження	step = resolveCylinderSpeedStep()
Елементи з пам’яттю	Збереження стану	prev, input	Якщо немає нового сигналу стан не змінюється	state = input ? new : prev
Запис нового стану	Оновлення runtime	nextRuntime	Збереження результату	runtime[element.id] = next
Формування результату	Підготовка для UI	runtime, signals	Дані передаються у відображення	return SimulationResult

3.5 Висновки до третього розділу

У третьому розділі виконано формалізацію елементів електропневматичної схеми на основі моделі Element / Port / State, що забезпечує уніфіковане представлення компонентів та поділ стану на декларативний і runtime-рівні.

Розглянуто моделювання пневматичної частини з використанням бінарної моделі тиску, яка забезпечує достатню точність для навчальних задач при низькій обчислювальній складності.

Розроблено алгоритм поширення сигналів на основі графа портів і з’єднань із урахуванням доменів сигналів та ітеративного досягнення стабільного стану, а також механізм маршрутизації тиску в клапанах.

Описано алгоритм оновлення станів елементів, який визначає їх runtime-стани на основі сигналів, з урахуванням типу елемента, пам’яті та впливу параметрів з’єднань.

4 РЕАЛІЗАЦІЯ ТА ОПТИМІЗАЦІЯ КРОСПЛАТФОРМЕНОГО ПРОГРАМНОГО ЗАСОБУ ДЛЯ МОДЕЛЮВАННЯ ЕЛЕКТРОПНЕВМАТИЧНИХ СХЕМ

Розроблений програмний засіб Electro Pneumatic Simulation System призначений для створення та імітаційного моделювання електропневматичних схем у інтерактивному графічному середовищі. Система забезпечує побудову схем із використанням бібліотеки електричних і пневматичних компонентів, автоматизовану маршрутизацію з'єднань, запуск покрокової симуляції та візуалізацію станів елементів у реальному часі. Програмний засіб реалізовано як кросплатформений десктопний застосунок на основі сучасних вебтехнологій.

Вікно «Про програму» Electro Pneumatic Simulation System відображено на рис. 4.1.

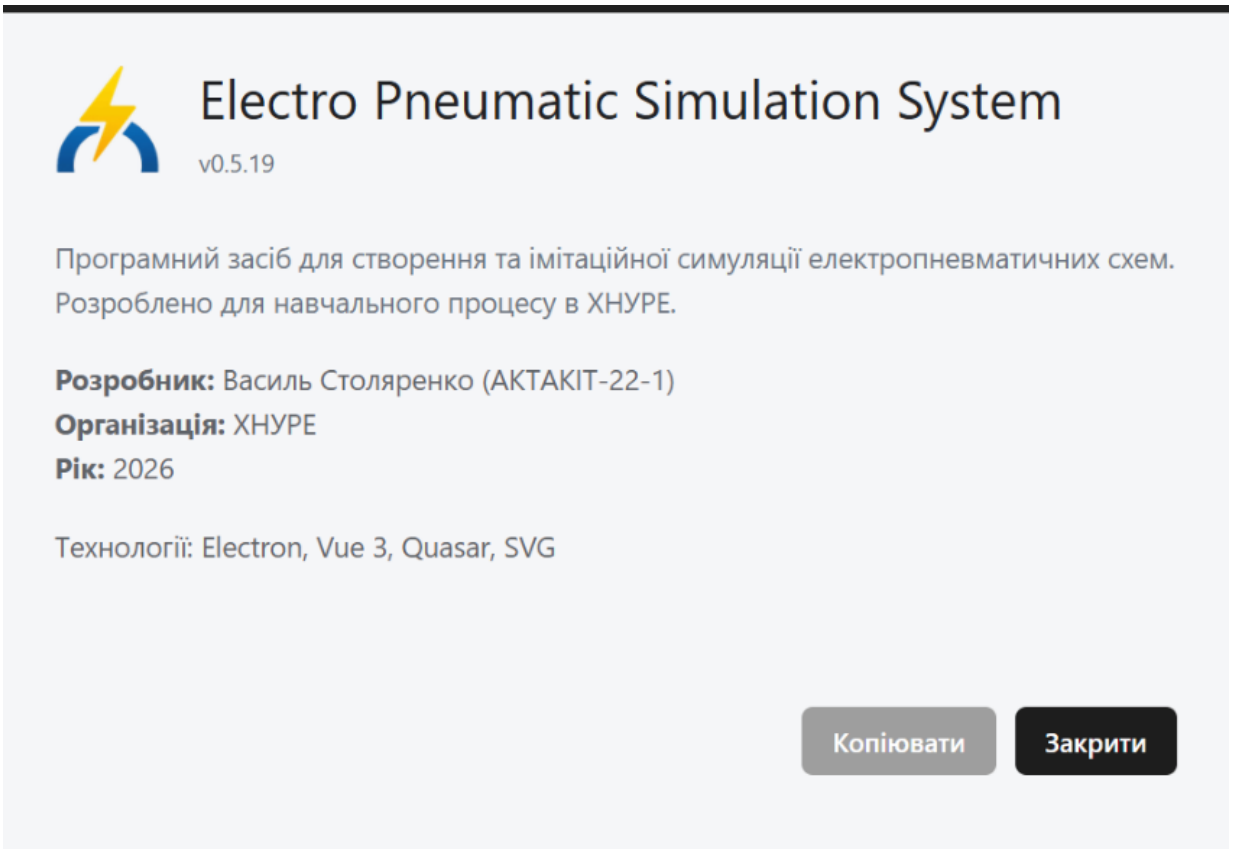


Рисунок 4.1 – Вікно «Про програму» програмного засобу Electro Pneumatic Simulation System

4.1 Вибір технологічного стеку та середовища розробки

Для реалізації програмного засобу створення та імітаційного моделювання електропневматичних схем було обрано сучасний стек веб-технологій, що забезпечив модульність архітектури, зручність супроводу, швидку ітеративну розробку та можливість розгортання у форматі desktop-застосунку на основі єдиного програмного коду.

Основою користувацького інтерфейсу є фреймворк Vue 3, який реалізував компонентний підхід до побудови програмних систем. Такий підхід дозволив структуровано розділити інтерфейс на незалежні функціональні частини (робоче полотно редактора, інспектор елементів, панелі керування, службові вікна), а реактивна модель даних забезпечила автоматичне оновлення відображення при зміні стану схеми та результатів симуляції.

Для побудови desktop-застосунку використано середовище Electron [22], що дозволило використовувати веб-технології для створення нативно-подібного програмного продукту. У поточній конфігурації проєкту налаштовано цільові збірки для Windows і macOS; архітектура рішення також дозволяє розширити конфігурацію для підтримки Linux без принципової зміни логіки застосунку. Інтерфейсний шар реалізовано з використанням Quasar, який розширив можливості Vue 3 і надав набір готових UI-компонентів. Це спростило реалізацію меню, панелей інструментів, діалогових вікон, сповіщень і загальної поведінки інтерфейсу в desktop-середовищі.

Для формалізованого опису структури електропневматичної схеми використовувався TypeScript. Статична типізація застосовується до сутностей елементів, портів, з’єднань, метаданих і станів виконання, що знизило ризик логічних помилок, підвищило надійність симуляційних алгоритмів і покращило підтримуваність коду.

Графічне представлення схеми побудовано на основі SVG, що забезпечило масштабованість без втрати якості, коректну інтерактивну взаємодію з елементами, а також наочну візуалізацію з’єднань і активних станів компонентів у процесі моделювання.

Для централізованого керування станом застосунку використано Pinia. Такий підхід забезпечив узгодженість даних між редактором, підсистемою симуляції, інспектором елементів і механізмами відображення результатів. Збереження та завантаження схем реалізовано у форматі JSON із підтримкою валідації та міграції версій структури даних, що забезпечило простоту серіалізації, зручність імпорту/експорту, а також можливість подальшого розширення формату без втрати сумісності.

Отже, обраний стек технологій відповідає вимогам до інтерактивного інженерного програмного засобу, який забезпечує модульну структуру, підтримку імітаційного моделювання, якісну графічну візуалізацію та перспективу подальшого функціонального розвитку системи.

4.2 Реалізація графічного редактора схем

Графічний редактор схем є основною складовою користувацького інтерфейсу програмного засобу та призначений для створення, редагування та візуалізації електропневматичних схем. Реалізація редактора базується на використанні технології SVG, що забезпечує масштабоване відображення графічних елементів, підтримку інтерактивної взаємодії та можливість динамічного оновлення станів компонентів під час симуляції.

Редактор реалізує робоче поле, на якому користувач може додавати елементи електропневматичної схеми, змінювати їх положення, виконувати з’єднання між портами та запускати імітаційну симуляцію. Всі елементи схеми представлені у вигляді окремих SVG-компонентів, що відповідають їх функціональному призначенню та графічному позначенню.

Головне вікно графічного редактора електропневматичних схем відображено на рис. 4.2.

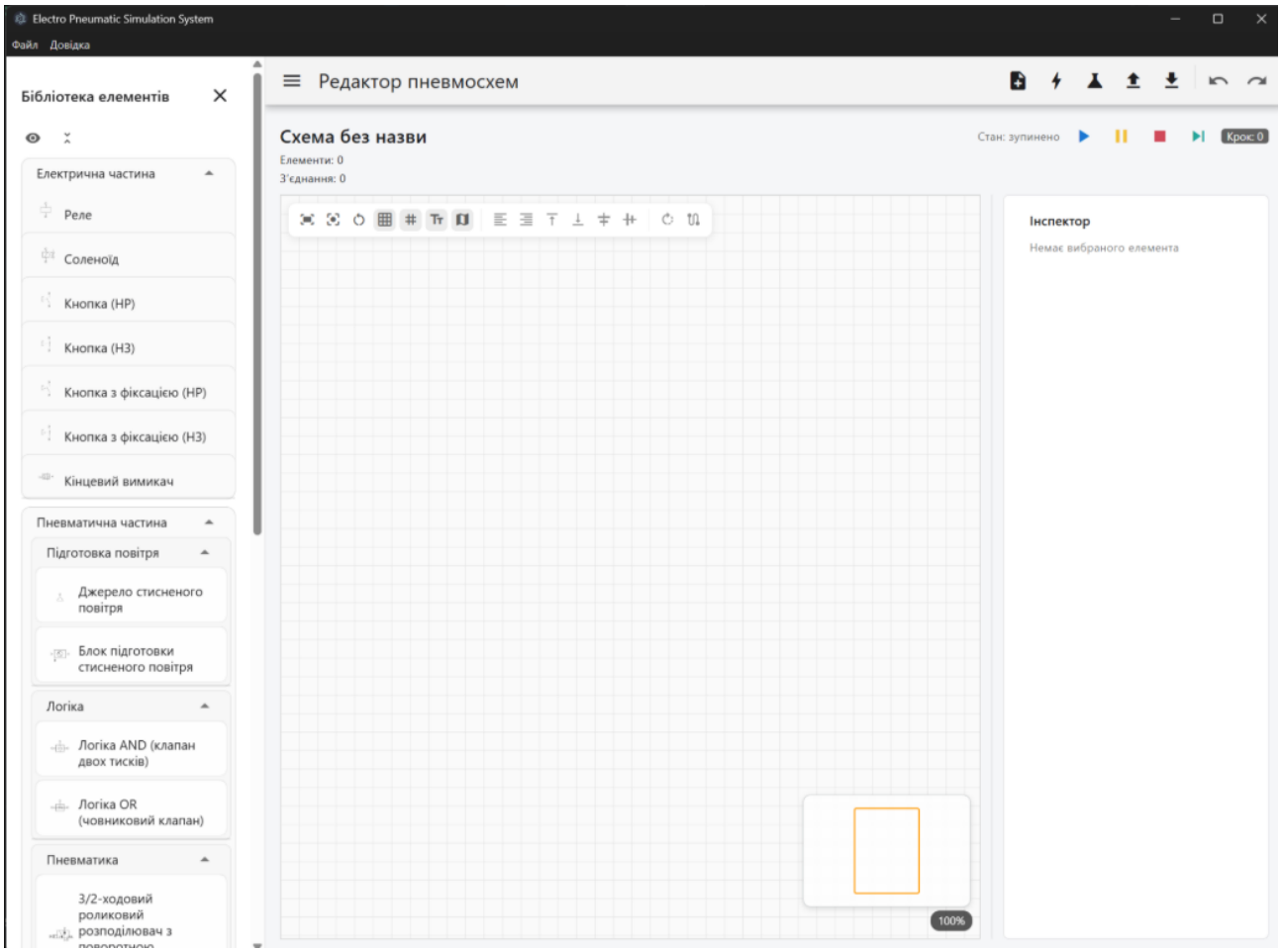


Рисунок 4.2 – Головне вікно графічного редактора електропневматичних схем

Для підвищення зручності та швидкості роботи з графічним редактором у програмному засобі реалізовано систему гарячих клавіш. Вона забезпечує швидкий доступ до основних функцій керування схемою без необхідності використання контекстних меню або панелей інструментів.

Система підтримує операції видалення елементів, скасування та повторення дій (undo/redo), масштабування робочого поля, увімкнення сітки та прив’язки, центрування схеми, копіювання, вставлення та дублювання елементів. Додатково реалізовано механізми групового виділення та переміщення декількох компонентів одночасно.

Наявність окремого вікна довідки з переліком гарячих клавіш (рис. 4.3) спрощує освоєння функціональних можливостей редактора та підвищує ефективність взаємодії користувача із системою під час створення та редагування електропневматичних схем.

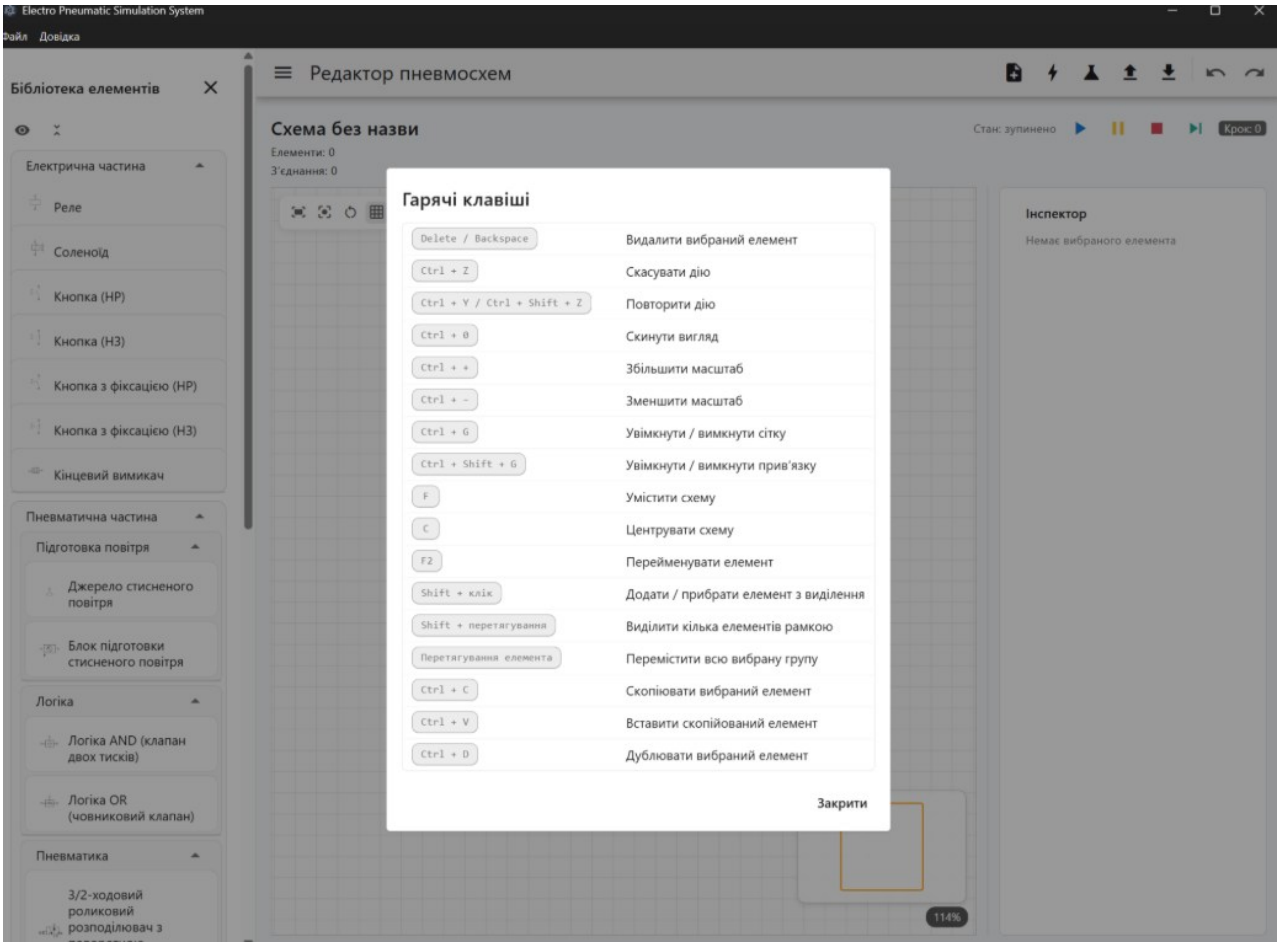


Рисунок 4.3 – Вікно довідки гарячих клавіш графічного редактора

Основою роботи редактора є централізована модель даних, яка містить інформацію про елементи, порти та з’єднання. Будь-яка дія користувача, зокрема переміщення елементів, створення з’єднань або зміна параметрів, призводить до оновлення моделі даних та автоматичного оновлення графічного представлення схеми.

Для реалізації механізму побудови з’єднань використовується прив’язка до портів елементів. Користувач може створювати з’єднання шляхом вибору початкового та кінцевого порту, після чого система автоматично формує маршрут між ними. З’єднання будуються ортогонально, тобто складаються з горизонтальних і вертикальних сегментів. Такий підхід забезпечує високу читабельність схеми та відповідає способу побудови технічних електропневматичних креслень.

Основний алгоритм побудови маршруту реалізовано у функції

buildOrthogonalRoute(...) (рис. 4.4), яка формує шлях з'єднання з урахуванням напрямку виходу з порту, проміжних точок та можливих перешкод.

orthogonalPath.ts

```
793  export function buildOrthogonalRoute(  Vasyl
794    source: OrthogonalRouteEndpoint,
795    target: OrthogonalRouteEndpoint,
796    options: BuildOrthogonalPathOptions = {},
797  ): Vec2[] {
798    const preference : OrthogonalRoutePreference  = options.preference ?? 'auto';
799    const bendSpacing : number  = clampBendSpacing(options.bendSpacing);
800    const sourceSide : OrthogonalPortSide | undefined  = options.sourceSide ?? source.side;
801    const targetSide : OrthogonalPortSide | undefined  = options.targetSide ?? target.side;
802    const portExitLength : number  = clampPositiveLength(options.portExitLength, DEFAULT_PORT_EXIT_LENGTH);
803    const portEntryLength : number  = clampPositiveLength(options.portEntryLength, DEFAULT_PORT_ENTRY_LENGTH);
804    const obstacleClearance : number  = clampObstacleClearance(options.obstacleClearance);
805    const obstacles : OrthogonalObstacle[]  = options.obstacles ?? [];
806
807    const sourcePoint = { ...source.point };
808    const targetPoint = { ...target.point };
809    const sourceExitPoint : Vec2  = offsetAlongSide(sourcePoint, sourceSide, portExitLength);
810    const targetEntryPoint : Vec2  = offsetAlongSide(targetPoint, targetSide, portEntryLength);
811    const baseIntermediateRoute : Vec2[]  = buildOrthogonalPath(sourceExitPoint, targetEntryPoint, {
812      preference,
813      bendSpacing,
814    });
815    const intermediateRoute : Vec2[]  = getObstacleAwareIntermediateRoute(
816      sourceExitPoint,
817      targetEntryPoint,
818      baseIntermediateRoute,
819      obstacles,
820      obstacleClearance,
821    );
822
823    const candidateRoute : Vec2[]  = [
824      sourcePoint,
825      sourceExitPoint,
826      ...intermediateRoute,
827      targetEntryPoint,
828      targetPoint,
829    ];
830    const fallbackRoute : Vec2[]  = [
831      sourcePoint,
832      sourceExitPoint,
833      ...baseIntermediateRoute,
834      targetEntryPoint,
835      targetPoint,
836    ];
837
838    return finalizeSafeRoute(candidateRoute, fallbackRoute, obstacles);
839  }
840
841  export function buildOrthogonalPath(from: Vec2, to: Vec2, options: BuildOrthogonalPathOptions = {}): Vec2[] {  Vasyl
842    const preference : OrthogonalRoutePreference  = options.preference ?? 'auto';
```

Рисунок 4.4 – Реалізація алгоритму побудови маршруту

Для підтримки коректного графічного представлення реалізовано механізм обходу перешкод. Якщо базовий маршрут проходить через область іншого елемента схеми, система автоматично перебудовує шлях з'єднання. Для цього виконується перевірка перетину сегментів маршруту з графічними межами елементів. Реалізація механізму обходу перешкод на рис. 4.5.

orthogonalPath.ts

```
441
442  function routeIntersectsAnyObstacle(points: Vec2[], obstacles: OrthogonalObstacle[]): boolean {  Vasyl
443    for (let i : number  = 0; i < points.length - 1; i += 1) {
444      const start : Vec2  = points[i]!;
445      const end : Vec2  = points[i + 1]!;
446
447      for (const obstacle of obstacles) {
448        if (segmentIntersectsObstacle(start, end, obstacle)) {
449          return true;
450        }
451      }
452    }
453
454    return false;
455  }
456
```

Рисунок 4.5 – Реалізація механізму обходу перешкод

У випадку виявлення перешкоди система формує декілька альтернативних маршрутів обходу та обирає найкоротший коректний маршрут із мінімальною кількістю згинів. Реалізація механізму перебудови маршруту представлена на рис. 4.6.

orthogonalPath.ts

```
711  function getObstacleAwareIntermediateRoute(  Vasyl
765    const validCandidates : { index: number; points: Vec2[] }[]  = candidates
766      .map((candidate : Vec2[] , index : number ) : { index: number; points: Vec2[] }  => ({
767        index,
768        points: dedupePathPoints(candidate),
769      }))
770      .filter((candidate : { index: number; points: Vec2[] } ) : boolean  => !routeIntersectsAnyObstacle(candidate.points, obstacles));
771
772    if (validCandidates.length === 0) {
773      return baseRoute;
774    }
775
```

Рисунок 4.6 – Реалізація механізму перебудови маршруту

Для підтримки коректного вигляду схеми після переміщення елементів реалізовано механізм автоматичного перерахунку маршруту з’єднання (рис. 4.7). Після зміни сцени система порівнює реконструйований маршрут із новим автоматично побудованим маршрутом та обирає найбільш придатний варіант.

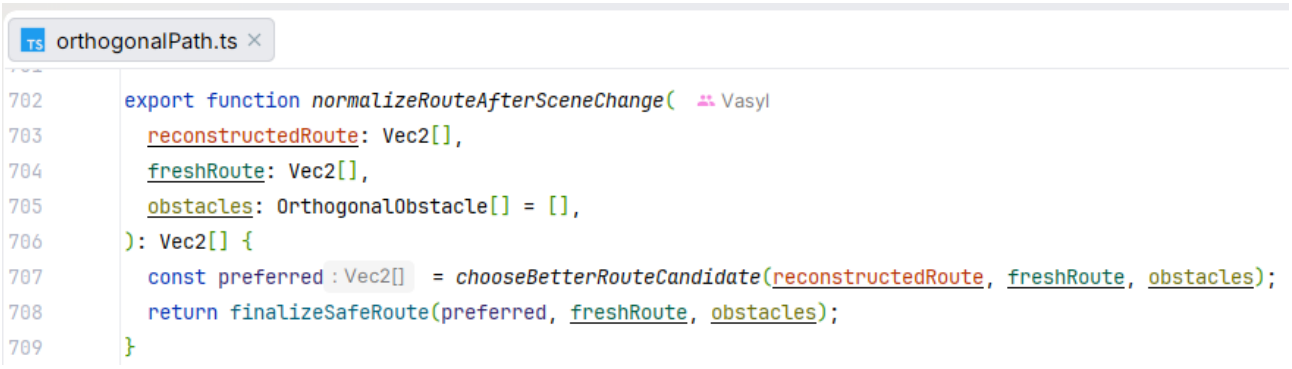


Рисунок 4.7 – Реалізація механізму автоматичного перерахунку маршруту

Додатково реалізовано перевірку некоректних самоз’єднань. Якщо користувач намагається створити з’єднання порту із самим собою, така ситуація визначається валідатором як помилка типу self-loop.

У графічному редакторі також реалізовано підтримку перетинів між з’єднаннями. Якщо одна лінія перетинає іншу без логічного з’єднання, у точці перетину формується спеціальний графічний місток, що дозволяє відрізнити звичайне перетинання від реального з’єднання елементів.

Під час побудови маршруту виконується очищення надлишкових сегментів та запобігання утворенню зайвих петель і самоперетинів. Це дозволяє зберігати акуратний вигляд схеми та зменшує кількість візуального шуму при роботі зі складними багатокомпонентними структурами.

Для підвищення зручності роботи реалізовано підтримку масштабування робочого поля, переміщення області перегляду, групового виділення елементів, копіювання компонентів та інтерактивного редагування схеми. Додатково в редакторі реалізовано систему гарячих клавіш, що забезпечує швидкий доступ до основних функцій керування. Користувач може виконувати видалення елементів, скасування та повторення дій, масштабування робочого поля,

увімкнення сітки та прив’язки, копіювання й вставлення елементів, а також групове переміщення компонентів за допомогою комбінацій клавіш. Реалізація гарячих клавіш підвищує швидкість взаємодії з редактором та наближує поведінку системи до професійних CAD-подібних середовищ проектування.

Під час симуляції редактор виконує функцію візуалізації поточних станів елементів, зокрема активних з’єднань, положення розподільників, станів соленоїдів та руху штока пневматичного циліндра.

Особливістю реалізації редактора є використання компонентного підходу, при якому кожен тип елемента реалізується як окремий незалежний SVG-компонент. Це спрощує розширення бібліотеки елементів та дозволяє додавати нові компоненти без зміни загальної структури програмного засобу.

Реалізований графічний редактор забезпечує інтерактивне створення електропневматичних схем, автоматизовану побудову ортогональних з’єднань, підтримку візуалізації станів елементів та створює основу для реалізації підсистеми імітаційної симуляції.

4.3 Програмна реалізація підсистеми симуляції

Підсистема симуляції призначена для імітаційного моделювання роботи електропневматичних схем та забезпечує оновлення станів елементів відповідно до структури з’єднань і логіки функціонування системи. Реалізація базується на дискретному покроковому алгоритмі, при якому стан схеми оновлюється через фіксовані часові інтервали.

Основою підсистеми є модуль SimulationEngine, який виконує послідовне оновлення станів елементів, обробку сигналів та формування результату кроку симуляції. Такий підхід забезпечує відокремлення обчислювальної логіки від інтерфейсу користувача та підтримує модульність архітектури програмного засобу.

Візуалізація процесу імітаційної симуляції електропневматичної схеми відображена на рис. 4.8.

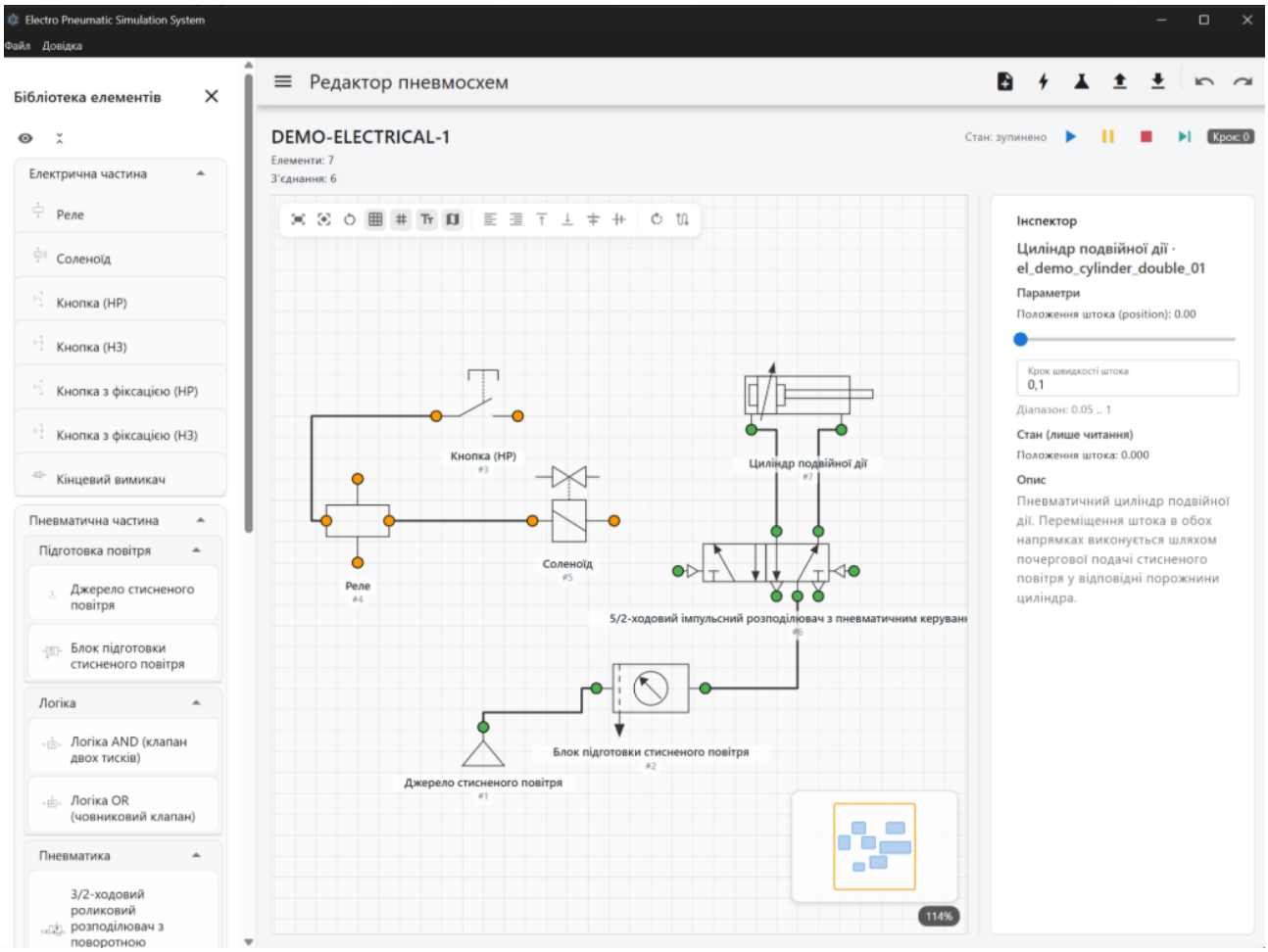


Рисунок 4.8 – Візуалізація процесу імітаційної симуляції електропневматичної схеми

Для представлення поточного стану використовується runtime-модель (контекст симуляції), що містить номер такту, активні сигнали на портах та внутрішні стани елементів. Після кожного кроку формується структура SimulationSnapshot, яка передається до шару візуалізації для відображення активних портів, з'єднань та змінених станів елементів.

Оновлення станів виконується фазово. У поточній реалізації виділено фази sources, control, electrical, routing, actuators, що дозволяє послідовно обробляти різні типи сигналів та виконавчих механізмів.

Додатково використовується механізм стабілізації сигналів, зокрема повторні проходи для електричної підсистеми. Це забезпечує стабілізацію поширення взаємозалежних сигналів і підвищує детермінованість симуляції.

Реалізація узагальненого циклу виконання кроку симуляції представлена на рис. 4.9.

orthogonalPath.ts | simulationEngine.ts

```
17  export class SimulationEngine {  Vasyl +1
30    step(): SimulationSnapshot {  Vasyl +1
31      const resetAndInitContext : (ctx: SimulationContext) => void  = (ctx: SimulationContext): void ⇒ this.resetAndInitContext(ctx);  Vasyl
32      const applyPressureSources : (ctx: SimulationContext, scheme: Scheme, ...  = (ctx: SimulationContext, scheme: Scheme, graph: ReturnType<typeof buildGraph>): PortId[] ⇒  Vasyl
33        this.applyPressureSources(ctx, scheme, graph);
34      const applyControlSignals : (ctx: SimulationContext, scheme: Scheme, ...  = (ctx: SimulationContext, scheme: Scheme, graph: ReturnType<typeof buildGraph>): PortId[] ⇒  Vasyl
35        this.applyControlSignals(ctx, scheme, graph);
36      const applyElectricalSignals : (ctx: SimulationContext, scheme: Scheme, ...  = (ctx: SimulationContext, scheme: Scheme, graph: ReturnType<typeof buildGraph>): PortId[] ⇒  Vasyl
37        this.applyElectricalSignals(ctx, scheme, graph);
38      const applyValveRouting : (ctx: SimulationContext, scheme: Scheme, ...  = (ctx: SimulationContext, scheme: Scheme, graph: ReturnType<typeof buildGraph>): PortId[] ⇒  Vasyl
39        this.applyValveRouting(ctx, scheme, graph);
40      const updateActuators : (ctx: SimulationContext, scheme: Scheme, ...  = (ctx: SimulationContext, scheme: Scheme, graph: ReturnType<typeof buildGraph>): void ⇒  Vasyl
41        this.updateActuators(ctx, scheme, graph);
42
43      return runSimulationTick({
44        ctx: this.ctx,
45        getScheme: this.getScheme,
46        resetAndInitContext,
47        initRuntimeState,
48        syncInteractiveRuntimeState,
49        applyPressureSources,
50        applyControlSignals,
51        applyElectricalSignals,
52        applyValveRouting,
53        updateActuators,
54        isScaffoldElementType,
55      });
56    }
57
58    reset(): void {  Vasyl Stoliarenko
59      this.ctx = this.createContext();
60    }
61
62    setSolenoidEnergized(solenoidId: string, energized: boolean): void {  Vasyl +1
63      setSolenoidEnergized(this.ctx, solenoidId, energized);
64    }
65
66    getSolenoidEnergized(solenoidId: string): boolean {  Vasyl +1
67      return getSolenoidEnergized(this.ctx, solenoidId);
68    }
69
70    private resetAndInitContext(ctx: SimulationContext): void {  Vasyl
71      ctx.tick += 1;
72      ctx.portSignals.clear();
73    }
74
75    private applyPressureSources(ctx: SimulationContext, scheme: Scheme, graph: ReturnType<typeof buildGraph>): PortId[] {  Vasyl
76      return evaluatePhaseBySimulator(ctx, scheme, graph,  phase: 'sources');
77    }
78
```

Рисунок 4.9 – Реалізація узагальненого циклу виконання кроку симуляції

Для поширення сигналів між елементами використовується графова модель схеми, у якій порти та з’єднання утворюють орієнтовану структуру взаємодії компонентів. Граф формується динамічно на основі поточної структури схеми перед виконанням кожного кроку симуляції. Під час обчислень система аналізує зв’язки між портами та визначає активні маршрути для пневматичного, керуючого та електричного контурів.

Синхронізація симуляційного ядра з графічним редактором реалізується через simulation-store. Після кожного кроку оновлені дані стають доступними компонентам інтерфейсу, що дозволяє у реальному часі відображати активні з’єднання, положення розподільників, стани соленоїдів та рух виконавчих механізмів.

Для керування процесом моделювання реалізовано режими запуску, паузи, зупинки та покрокового виконання симуляції (рис. 4.10).

orthogonalPath.ts | simulationEngine.ts | simulationStore.ts

```
import { defineStore } from 'pinia';
import { ref } from 'vue';
import { useSchemeStore } from '../stores/schemeStore';
import { SimulationEngine } from './simulationEngine';
import type { SimulationSnapshot, SimulationStatus } from './types';
import { validateScheme } from '../validation/schemeValidator';

const SIMULATION_TICK_INTERVAL_MS = 200;

export const useSimulationStore : StoreDefinition<'simulation', Pick<{ stat... = defineStore( id: 'simulation', () : { status: Ref<SimulationStatus, Simulatio... => {
  const schemeStore : Store<'scheme', SchemeState, {}, { normal... = useSchemeStore();
  const engine = new SimulationEngine(() : { id: string; name: string; schemaVersion... => schemeStore.scheme);

  const status : Ref<SimulationStatus, SimulationStatus> = ref<SimulationStatus>( value: 'stopped');
  const tick : Ref<number, number> = ref( value: 0);
  const snapshot : Ref<{ tick: number; activePortIds: string... = ref<SimulationSnapshot | null>( value: null);
  const selectedSolenoidId : Ref<string | null, string | null> = ref<string | null>( value: null);
  let loopTimer: ReturnType<typeof setInterval> | null = null;

  function clearLoop(): void {...}

  function ensureLoopStarted(): void {...}

  function start(): void {  Vasyl Stoliarenko +1
    status.value = 'running';
    ensureLoopStarted();
  }

  function stop(): void {  Vasyl Stoliarenko +1
    clearLoop();
    status.value = 'stopped';
    tick.value = 0;
    snapshot.value = null;
    engine.reset();
  }

  function pause(): void {  Vasyl Stoliarenko +1
    clearLoop();
    status.value = 'paused';
  }

  function runTick(): void {...}

  function stepOnce(): void {  Vasyl
    if (status.value === 'running') {
      return;
    }

    runTick();
  }
```

Рисунок 4.10 – Реалізація режиму запуску, паузи, зупинки та покрокового виконання симуляції

Такий підхід забезпечує не лише демонстраційний режим роботи системи, а й можливість покрокового аналізу спрацювання елементів у навчальному процесі.

Перед виконанням кроку симуляції застосовується валідація структури схеми. У випадку виявлення помилок конфігурації запуск обчислень блокується, а система формує діагностичну інформацію про помилки та попередження.

Модульність підсистеми підтримується за рахунок розділення на simulation engine, store-рівень керування, графовий модуль поширення сигналів та типізовані структури runtime/snapshot-станів. Логіка кожного типу елемента реалізується незалежно через окремі обробники, що дозволяє розширювати бібліотеку компонентів без зміни базової архітектури системи.

Така організація підсистеми симуляції забезпечує модульність, стабільність роботи, достатню швидкодію та можливість подальшого розширення алгоритмів моделювання без зміни базової архітектури програмного засобу.

4.4 Оптимізація програмного засобу

Під час розробки програмного засобу особливу увагу приділено оптимізації архітектури, механізмів візуалізації та алгоритмів симуляції. Ключовою метою було забезпечення стабільної роботи системи під час взаємодії з багатокомпонентними електропневматичними схемами, а також підвищення зручності використання редактора в навчальному процесі.

Одним із базових рішень стала побудова системи на основі незалежних функціональних модулів: графічного редактора, ядра симуляції, системи валідації, модуля маршрутизації з’єднань і централізованого керування станом. Такий підхід зменшує зв’язність між підсистемами, спрощує супровід коду та створює передумови для подальшого розширення функціональності.

На рисунку 4.11 представлено приклад роботи програмного засобу після реалізації механізмів оптимізації графічного редактора та підсистеми симуляції. Схема містить електричну та пневматичну частини, ортогональні з’єднання між елементами, автоматичний обхід графічних перешкод і візуалізацію активних станів компонентів.

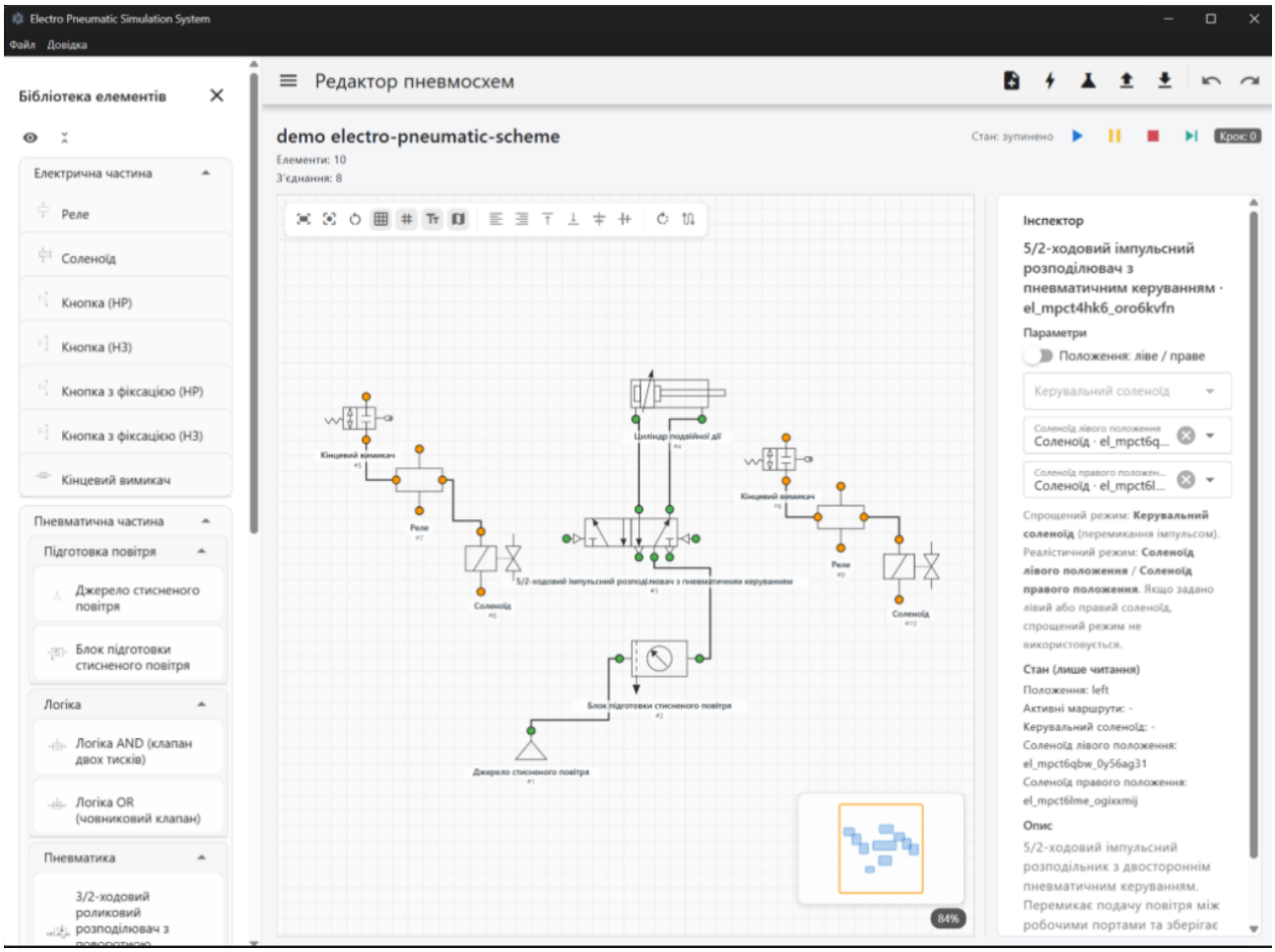


Рисунок 4.11 – Візуалізація оптимізованої електропневматичної схеми у графічному редакторі

У правій частині інтерфейсу відображається інспектор елементів, який забезпечує перегляд параметрів та поточного стану вибраного компонента. Для підвищення продуктивності використано SVG-візуалізацію, автоматизовану маршрутизацію з’єднань і механізм покрокового оновлення станів під час симуляції.

Реалізовані механізми оптимізації дозволяють забезпечити читабельність схеми, стабільність роботи редактора та достатню швидкодію при взаємодії з багатокомпонентними електропневматичними структурами.

Для оптимізації обчислень у симуляції застосовано фазовий алгоритм оновлення станів (рис. 4.12). Розділення процесу на послідовні етапи (sources, control, electrical, routing, actuators) дає змогу уникати конфліктів між взаємопов’язаними елементами та забезпечує детермінованість результатів.

orthogonalPath.ts | simulationEngine.ts | simulationStore.ts | types.ts

```
7  export type SimulationPhase = 'sources' | 'control' | 'electrical' | 'routing' | 'actuators';
8
```

Рисунок 4.12 – Реалізація фазового алгоритму оновлення станів

Для підвищення продуктивності графічного редактора використано SVG-візуалізацію з компонентним підходом. Кожен елемент подається як окремий SVG-компонент, що мінімізує обсяг перерендерингу під час змін стану схеми та дозволяє оновлювати лише ті частини інтерфейсу, які були змінені в результаті симуляції.

Оптимізація побудови з'єднань забезпечується автоматизованою ортогональною маршрутизацією. Алгоритм формує маршрут між портами, виконує обхід графічних перешкод, усуває надлишкові сегменти та підтримує автоматичний перерахунок після зміни структури схеми. Реалізація механізму автоматичного перерахунку після зміни структури схеми представлена на рис. 4.13.

orthogonalPath.ts

```
793  export function buildOrthogonalRoute(  Vasyl
815    const intermediateRoute : Vec2[]  = getObstacleAwareIntermediateRoute(
816      sourceExitPoint,
817      targetEntryPoint,
818      baseIntermediateRoute,
819      obstacles,
820      obstacleClearance,
821    );
822
```

Рисунок 4.13 – Реалізація механізму автоматичного перерахунку після зміни структури схеми

Якщо маршрут перетинає графічну область іншого елемента, система автоматично генерує альтернативні варіанти та відбирає лише валідні траєкторії.

Для зменшення візуального шуму реалізовано очищення маршрутів, яке прибирає зайві згини, петлі та надлишкові сегменти. Додатково впроваджено механізм графічних містків у місцях перетину ліній, що підвищує читабельність складних схем.

Оптимізація взаємодії користувача з редактором досягнута завдяки підтримці гарячих клавіш, механізмів undo/redo, масштабування робочого поля, групового виділення та інтерактивного переміщення елементів. Це зменшує кількість повторюваних дій і підвищує ефективність роботи зі схемами.

Для підвищення стабільності перед запуском симуляції виконується валідація структури схеми. Перевірка коректності з'єднань, типів портів і конфігурації елементів дає змогу виявляти помилки до початку обчислень і запобігати некоректній роботі алгоритмів. Реалізація механізму валідації структури схеми представлена на рис. 4.14.

schemeValidator.ts

```
168  export function validateScheme(scheme: Scheme): SchemeValidationResult {  Vasyl Stoliarenko +1
271    for (const connection of scheme.connections) {
272      if (connection.fromPortId === connection.toPortId) {
273        pushError(result, {
274          code: 'CONNECTION_SELF_LOOP',
275          message: `Connection ${connection.id} forms a self-loop on port ${connection.fromPortId}.`,
276          connectionId: connection.id,
277          portId: connection.fromPortId,
278        });
279      }
```

Рисунок 4.14 – Реалізація механізму валідації структури схеми

Узгодженість між ядром симуляції та графічним інтерфейсом забезпечується snapshot-моделлю стану. Після кожного кроку формується структура SimulationSnapshot, що містить інформацію про активні порти, з'єднання, змінені елементи та діагностичні повідомлення.

Додатково реалізовані механізми оптимізації інтерфейсу користувача, зокрема підтримка гарячих клавіш, групового редагування елементів,

масштабування робочого поля та автоматичного оновлення графічного представлення після змін структури схеми. Це дозволяє підвищити ефективність взаємодії користувача з редактором, зменшити кількість повторюваних операцій та забезпечити більш комфортну роботу з електропневматичними схемами у процесі навчального моделювання.

Загалом реалізовані підходи до оптимізації забезпечують належну швидкодію, стабільність і зручність використання програмного засобу під час роботи з електропневматичними схемами різної складності.

Модульна архітектура та використання сучасних вебтехнологій формують надійну основу для подальшого функціонального розширення програмного засобу та розвитку алгоритмів імітаційного моделювання.

4.5 Охорона праці під час реалізації кросплатформеного програмного засобу для моделювання електропневматичних схем

На робочому місці розробника можуть виникати небезпечні та шкідливі фактори, такі як високий рівень шуму, несприятливі мікрокліматичні умови, недостатнє освітлення, присутність шкідливих речовин, підвищені електромагнітні випромінювання радіочастот, висока напруга електромережі, статична електрика тощо. Робота з ПК також характеризується високим рівнем психічної та фізичної напруги. Тривалий вплив несприятливих виробничих факторів, які не відповідають нормативам, може призвести до зростання захворюваності, зокрема ураження органів зору, опорно-рухового апарату та нервової системи. Тому дослідження умов праці програміста є важливим кроком для запобігання негативним наслідкам дії цих факторів.

Організація робочого місця. Приміщення, в якому знаходиться робоче місце програміста, має бути не меншою за 9 $м^2$ на працівника. Для розробки кросплатформеного програмного засобу для моделювання електропневматичних схем слід забезпечити розробника достатнім простором для роботи. Природне, так і штучне освітлення має бути оптимальним для зменшення навантаження на очі. Робоче місце обладнане робочім столом, стільцем та персональним

комп'ютером, що складається з монітора, системного блоку, клавіатури та миші.

Електроживлення ПК здійснюється від трифазної чотирипровідної мережі змінного струму з напругою 220 В, частотою 50 Гц і заземленою нейтраллю. Відповідно до НПАОП 40.1-1.21-98, приміщення класифікується як таке, що не належить до категорії з підвищеною небезпекою, оскільки відсутні фактори, які б створювали підвищену або особливу небезпеку.

Для забезпечення безпечних умов праці необхідно вжити низку організаційних і технічних заходів. Відповідно до НПАОП 40.1-1.32-01, для захисту від ураження електричним струмом у приміщенні має бути використана система занулення.

Робота оператора ПК за енерговитратами відноситься до категорії легких робіт.

В таблиці 4.1 наведені оптимальні параметри мікроклімату в приміщеннях, де виконуються роботи операторського типу [23].

Таблиця 4.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22 – 24 °С; 40 – 60 %; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23 – 25 °С; 40 – 60 %; 0,1 – 0,2 м/с

Виміряні за допомогою приладів температура та вологість у лабораторії відповідають вказаним у таблиці для теплого періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції.

Лабораторія, де виконується робота, має наступні характеристики:

– площа приміщення – 9 м^2 (3м × 3 м);

– обладнання – стіл з ПК і периферією – 1 шт.

Приміщення, відповідно до ДНАОП 0.00-1.31-99, має забезпечувати 6 м^2 площі та 20 м^3 об’єму на одне окреме робоче місце з персональним комп'ютером [23]. Площа приміщення 9 м^2 та об’єм 22,5 м^3, на кожне робоче місце приходиться 9 м^2 площі і об’єм 22,5 м^3, тобто вимога виконана.

Приміщення з ПК повинні мати природне і штучне освітлення відповідно до ДБН В.25-28-2006 «Природне і штучне освітлення». Природне світло повинно проникати через бічні світлові прорізи, зорієнтовані, як правило, на північ або північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче 1,5 %.

Рівень загального штучного освітлення приміщення можна перевірити за допомогою методу питомої потужності, викладеної в [23].

Розрахункова формула методу (4.1):

$$W = \frac{W_{\Sigma}}{S}, \tag{4.1}$$

де W – питома потужність, Вт/м^2;

S – площа приміщення, м^2;

W_{Σ} – загальна потужність освітлювальної установки Вт, яка розраховується за формулою (4.2):

$$W_{\Sigma} = W_{св} \cdot n_{св}, \tag{4.2}$$

де $W_{св}$ – потужність одного світильника, Вт;

$n_{св}$ – кількість світильників в приміщенні.

Проведені розрахунки та предсталені на (4.3, 4.4):

$$W_{\Sigma} = 100 \cdot 4 = 400 \text{ Вт}, \tag{4.3}$$

$$W = \frac{400}{48} = 8{,}33 \text{ Вт/м}^2. \quad (4.4)$$

Питомої потужності 8.33 Вт/м^2 по таблиці Б.3 із [23] відповідає освітленість в 250 лк при мінімальній допустимої освітленості 300 лк.

Отже, для створення сприятливих зорових умов в лабораторії необхідно збільшити кількість світильників або замінити лампи в світильниках на більш потужні.

Під час експлуатації персонального комп'ютера існує ризик виникнення пожежі внаслідок несправності електрообладнання, короткого замикання або перегріву електронних компонентів. Сучасні ПК мають високу щільність розташування електронних компонентів, що сприяє виділенню великої кількості тепла під час проходження електричного струму через проводи та кабелі. Це може призвести до оплавлення ізоляції. Для відведення тепла використовуються системи вентиляції та кондиціювання повітря, які, при постійному використанні, самі можуть стати джерелом підвищеної пожежної небезпеки

4.6 Висновки до четвертого розділу

У четвертому розділі було розглянуто практичну реалізацію кросплатформеного програмного засобу для створення та імітаційного моделювання електропневматичних схем. Обґрунтовано вибір технологічного стеку та реалізовано інтерактивний графічний редактор із підтримкою автоматизованої маршрутизації з’єднань, бібліотеки елементів і системи керування схемою. Реалізовано підсистему симуляції на основі дискретного покрокового алгоритму з фазовим оновленням станів елементів та візуалізацією результатів моделювання у реальному часі. Додатково розглянуто підходи до оптимізації архітектури, алгоритмів маршрутизації та механізмів взаємодії користувача із системою. Окрему увагу приділено питанням охорони праці під час використання програмного засобу.

ВИСНОВКИ

В кваліфікаційній роботі вирішено задачу розроблення кросплатформеного програмного засобу для створення та імітаційного моделювання електропневматичних схем.

У процесі виконання роботи проведено аналіз електропневматичних систем, їх типових структур та особливостей використання у навчальному процесі, а також виконано огляд існуючих програмних засобів моделювання. На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до програмного засобу та обґрунтовано використання багаторівневої модульної архітектури.

У роботі розроблено модель даних електропневматичної схеми на основі сутностей Element, Port та Connection, що забезпечує формалізоване представлення структури схеми та є основою для реалізації алгоритмів симуляції. Розроблено алгоритм імітаційної симуляції, який базується на дискретному поширенні сигналів між елементами схеми, фазовому оновленні станів та графовому представленні з’єднань. Реалізовано механізми обробки електричних, пневматичних та керуючих сигналів.

Практичним результатом роботи став програмний засіб Electro Pneumatic Simulation System, реалізований із використанням Vue 3, Electron, TypeScript та SVG. Система забезпечує створення електропневматичних схем у графічному редакторі, автоматизовану побудову з’єднань, запуск симуляції та візуалізацію станів елементів у реальному часі.

Розроблений програмний засіб може бути використаний у навчальному процесі для виконання лабораторних робіт, дослідження принципів функціонування електропневматичних систем та підвищення наочності навчання у сфері автоматизації та мехатроніки.

Перспективами подальшого розвитку є розширення бібліотеки елементів, підтримка складніших алгоритмів моделювання та реалізація більш детальної фізичної моделі пневматичних процесів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008–2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Документація. Введ. 2015. К.: Держстандарт України, 2015. – 31 с.

2. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка освітньої програми [Електронний ресурс] : навч. посібник / І. Ш. Невлюдов, О. І. Филипенко, О. В. Токарєва, С. П. Новоселов, О. В. Сичова. ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків : Видавництво Іванченка І. С., 2025. – 164 с.

3. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти денної і заочної форми навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд.: І.Ш. Невлюдов, О.І. Филипенко, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2023. – 64 с.

4. Указ президента України №722/2019 "Про Цілі сталого розвитку (ЦСР) України на період до 2030 року " від 30.09.2019 р. Режим доступу: https://www.president.gov.ua/documents/7222019-29825.

5. Автоматизація та приладобудування ("Automation and Development of Electronic Devices" ADED-2026) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2026. – Вип. 1. – 237 с.

6. Невлюдов І.Ш. Основи наукових досліджень: Навч. посібник / І.Ш. Невлюдов, Ю.М. Олександров, А.О. Андрусевич, О.О. Чала. – Prague, OKTAN PRINT, 2024. – 468 с.

7. Невлюдов І.Ш. Електропневмоавтоматичні приводи в автоматизованих системах керування : навч. посіб. / І.Ш. Невлюдов, Л.О. Кривопляс-Володіна, С.П. Новоселов, О.В. Сичова ; М-во освіти і науки України, Харків. нац. ун-т радіоеклектроніки. – Харків : ХНУРЕ, 2021. – 292 с. – ISBN 978-966-659-332-3;

DOI: 10.30837/978-966-659-332-3.

8. Невлюдов І.Ш. Пневматичні пристрої та засоби автоматизації мехатронних систем: Навчальний посібник / І.Ш. Невлюдов, Л.О. Кривопляс-Володіна, С.П. Новоселов, О.В. Сичова. – Харків: ФОП Панов А.М., 2020. – 256 с. DOI: 10.30837/978-617-7859-58-0. ISBN 978-617-7859-58-0.

9. І.Ш. Невлюдов, О.В. Токарєва. Теорія автоматичного управління : навч. посіб. для студентів спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології / І.Ш. Невлюдов, О.В. Токарєва; Харків. нац. ун-т радіоелектроніки. - Харків: Панов А.М., 2020. – 240 с.

10. Положення про академічну доброчесність [Електронний ресурс]: Наказ ХНУРЕ від 02 лютого 2021 р. No 50. Режим доступу: https://nure.ua/wp–content/uploads/Main_Docs_NURE/polozhennja– pro– akademi– chnu–dobrochesnist.pdf.

11. Zhao L. Pneumatic servo systems analysis: control and application in robotic systems / L. Zhao, Y. Xia, H. Yang, J. Zhang. – Springer, 2022. – 320 p.

12. Merzouki R. Intelligent mechatronic systems: modeling, control and automation / R. Merzouki, A. Samantaray, P. Pathak. – Springer, 2021. – 410 p.

13. Liu J. Modeling and control of electro-pneumatic systems / J. Liu, H. Wang. – Wiley, 2023. – 356 p.

14. Bolton W. Pneumatic and hydraulic systems / W. Bolton. – 2nd ed. – Elsevier, 2021. – 304 p.

15. Parambath J. Electro-Pneumatics and Automation Fluid Power Educational Series: Pneumatic Book Series, Vol. 3 / J. Parambath. – Amazon Digital Services LLC - KDP Print US, 2020. – 80 p. ISBN 9798652544386.

16. Parambath J. Practical Book: Electro-pneumatics - Basic Level: Industrial Hydraulics and Pneumatics Practical Book, Vol. 2 / J. Parambath. – Amazon Digital Services LLC - KDP Print US, 2020. – 50 p. ISBN 9798677042935.

17. Piccuillo C. Pneumatic Systems: Principles And Maintenance: Pneumatic Books / C. Piccuillo. – Independently Published, 2021. – 124 p. ISBN 9798543642955

18. Industrial Automation News for the Next Generation [Електронний ресурс]. – Режим доступу: https://library.automationdirect.com/.

19. Quasar Framework Documentation [Електронний ресурс]. – Режим доступу: https://quasar.dev/

20. Sommerville I. Software Engineering. – 10th ed. – Pearson, 2016. – 816 p.

21. Vue.js Documentation [Електронний ресурс]. – Режим доступу: https://vuejs.org/guide/.

22. Electron Documentation [Електронний ресурс]. – Режим доступу: https://www.electronjs.org/docs.

23. Комплекс навчально-методичного забезпечення навчальної дисципліни «Організація керування умовами праці» підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [Електронний ресурс] / ХНУРЕ; розроб.: Т. Є. Стиценко, Г. В. Пронюк, Н. М. Сердюк. – Харків, 2017. – 108 с.