МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Web-додаток для моделювання роботи електричних схем»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-12 Астахов Дмитрій Сергійович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2022 р.

Науковий керівник _____ к.т.н., доц., Чибіряк Я.І.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2022

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. зав. кафедри ІТ

_____ С. М. Ващенко
«__» _____ 2022 р.

З А В Д А Н Н Я
на кваліфікаційну роботу магістра студентові

Астахов Дмитрій Сергійович

1 **Тема роботи** *Web-додаток для моделювання роботи електричних схем*

затверджені наказом по університету від «29» жовтня 2022 р. № _____

2 Термін здачі студентом закінченого проекту «10» грудня 2022 р.

3 Вхідні дані до проекту завдання на розробку web-додатку для моделювання роботи електричних схем

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) 1) Аналіз предметної області, 2) Постановка задачі та методи дослідження, 3) Проектування додатку, 4) Практична реалізація.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність роботи, мета та задач, дослідження аналогів, засоби реалізації, етапи реалізації, практична реалізація.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Оформлення календарного плану робіт	01.09.2022 – 10.09.2022	
2.	Визначення мети проекту	11.09.2022 – 15.09.2022	
3.	Аналіз продуктів аналогів	15.09.2022 – 20.09.2022	
4.	Визначення функціональних вимог до web-додатку	21.09.2022 – 30.09.2022	
5.	Визначення засобів реалізації	01.10. 2022 – 10.10.2022	
6.	Моделювання та проектування web-додатку	11.10.2022 – 25.10.2022	
7.	Реалізація додатку	26.10.2022 – 25.11.2022	
8.	Тестування web-додатку	26.11.2022 – 30.11.2022	
9.	Виправлення виявлених помилок	01.12.2022 – 05.12.2022	
10.	Оформлення пояснювальної записки	05.12.2022 – 10.12.2022	
11.	Здача web-додатку для моделювання роботи електричних схем	До 15.12.2022	

Студент _____ Астахов Д.С.
(підпис)

Керівник роботи _____ Чибіряк Я.І.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Web-додаток для моделювання роботи електричних схем».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел. Пояснювальна записка включає в себе 70 сторінок, 29 рисунків та 6 таблиць, 2 додатка, 31 джерело.

Об'єкт дослідження — розробка web-додатку, призначеного для імітації роботи електричних схем.

Мета роботи — розробити web-додаток для моделювання роботи електричних схем.

Предмет дослідження – web-додаток для моделювання роботи електричних схем.

Результати: при виконанні кваліфікаційної роботи магістра розроблено web-додаток, який дозволяє використовувати вже існуючі схеми, вдосконалювати їх або створювати нові електричні схеми.

Ключові слова: web-додаток, моделювання, інформаційні технології, електричні схеми, дослідження, вимірювання;

web-приложение, моделирование, информационные технологии, электрические схемы, исследования, измерения;

web application, modeling, information technology, electrical circuits, research, measurement;

ЗМІСТ

ВСТУП

На сьогоднішній день спостерігається тенденція до створення програм які використовуються для моделювання роботи електричних схем, або покращення вже існуючих програм. Це дозволяє полегшити створення цих електричних схем, тому що доволі зручно мати все що потрібно для створення електричної схеми в одній програмі.

Майже всі розробники електричних схем рано чи пізно натрапляють на необхідність швидко змоделювати схему, або перевірити якісь зміни в існуючій схемі, але не всі можуть це зробити фізично, тому що іноді якихось компонентів може не бути і на цей випадок дуже допомагають такі програми на яких можна швидко змоделювати потрібно схему і перевірити її, адже все що потрібно для тестування вже є в цій програмі. Однією із головних переваг є саме зменшення втраченого часу на розробку схеми, її тестування та виявлення потенційних проблем на ранніх побудови електричної схеми.

Основною метою цієї роботи є створення web-додатку для моделювання роботи електричних схем, з можливістю досліджувати графіка напруги та струму. Також в той же час додаток не повинен бути перевантаженим зайвим функціоналом, та повинен бути зрозумілим навіть для користувачі, які мало знайомі з електричними схемами.

Об’єктом дослідження є розробка web-додатку, призначеного для імітації роботи електричних схем.

Предметом дослідження є web-додаток для моделювання роботи електричних схем.

Новизна роботи полягає в використанні більш актуальних технологій для моделювання роботи електричних схем.

Тому обрана тема кваліфікаційної роботи магістра є актуальною, оскільки дозволяє автоматизувати проектування і тестування електричних схем.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

В наш час розвиток IT технологій стрімко йде вверх, і дуже важко уявити будь-яку сферу бізнесу без використання сучасних тенденцій інформаційних технологій (IT). IT є невід'ємною частиною життя інформаційної діяльності та кожної людини, без якою ми не зможемо уявити наше життя [1]. Відповідно до цього, інформаційні технології надають значний вплив у багатьох сферах діяльності, у тому числі й у бізнесі. Людство перейшло у новий період, у якому дуже цінується інформація та інформаційні технології [1].

Група вчених провели дослідження, щодо того як впливають інформаційні технологія на діяльність людини та ефективність використання цих технологій. В ході дослідження було виявлено, що «Нові інформаційні технології з'явилися не просто інструментом для застосування, але також процесами для розвитку [2]. Звідси випливає нове співвідношення між соціальними процесами створення та обробки символів (культура суспільства) та здатністю виробляти та розподіляти товари та послуги (продуктивні сили). Вперше в історії людська думка є прямо продуктивною силою, а не просто певним елементом продуктивної системи» [2].

З результату досліджень вчених було виявлено, що подальший розвиток IT – це лише питання часу. Все більше компаній починають впроваджувати ці технології, як норму для компанії. IT-директори високотехнологічних компаній у країнах які є лідерами за обсягом інвестицій у IT сферу, орієнтовані на майбутнє, думатимуть не лише про цифрову трансформацію, а й реалізовуватимуть ініціативи. які тісно поєднують купівельний досвід та досвід працівників [3]. Гіганти високотехнологічних компаній, такі як AMD і Intel показують, що використання актуальних технологій та мислення на майбутнє у своєму бізнесі, дає покращення не тільки комунікації між співробітниками, керівництвом, але й у досягненнях яким можуть позаздрити багато компаній [3].

1.2 Аналіз продуктів - аналогів

Існує велика кількість різних додатків для моделювання електричних схем. Є як платні так и безкоштовні додатки. На рисунку 1.1 наведено приклад однієї із найвідоміших програм для моделювання роботи електричних схем sPlan, ця програма використовується багатьма компаніями. sPlan є багатофункціональною та простою програмою, використовується для моделювання схем розводки електропроводки та трасування електронних плат [4]. Програмний пакет включає безліч готових бібліотек електронних компонентів, а також має функцію додавання власних шаблонів. Цей комплекс є платним, з чого випливає, що більшість функціоналу буде заблоковано і щоб розблокувати – треба заплатили. Вартість sPlan приблизно 50$. Є також безплатна версія, але в ній відключено функції збереження та друку файлів, тому її можна використовувати тільки щоб ознайомитися [4].

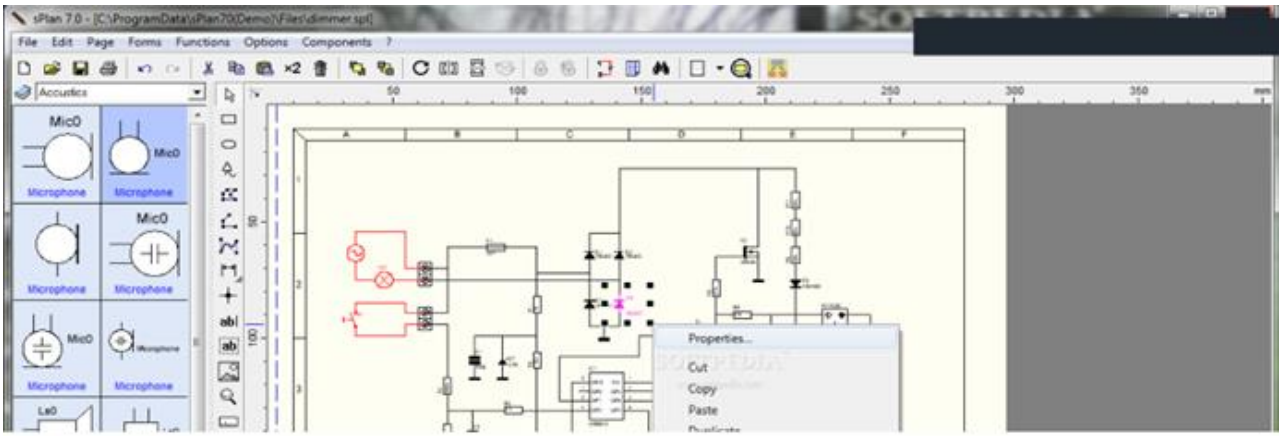

Рисунок 1.1 Приклад вигляду частини програми для моделювання роботи електричних схем sPlan

Ще один приклад, схожий до попереднього аналога - AutoCAD Electrician який зображено на рисунку 1.2. Ця програма має велику кількість вбудованих бібліотек та функцій. Є можливість створювати відразу кілька проектів зі спільним доступом різних користувачів. Але один із великих недостатків – це те, що для коректної роботи потрібно виконати безліч налаштувань, але це в подальшому значно полегшує роботу [5].

Унікальна особливість додатку полягає в наявності інтелектуальної системи, яка може аналізувати проект, відстежувати можливі помилки проектувальника та

виправляти їх [5]. Програма доволі дорога та складна, тому використовується в основному професіональними електриками. Для ознайомлення з додатком надається безплатна демоверсія на 30 діб.

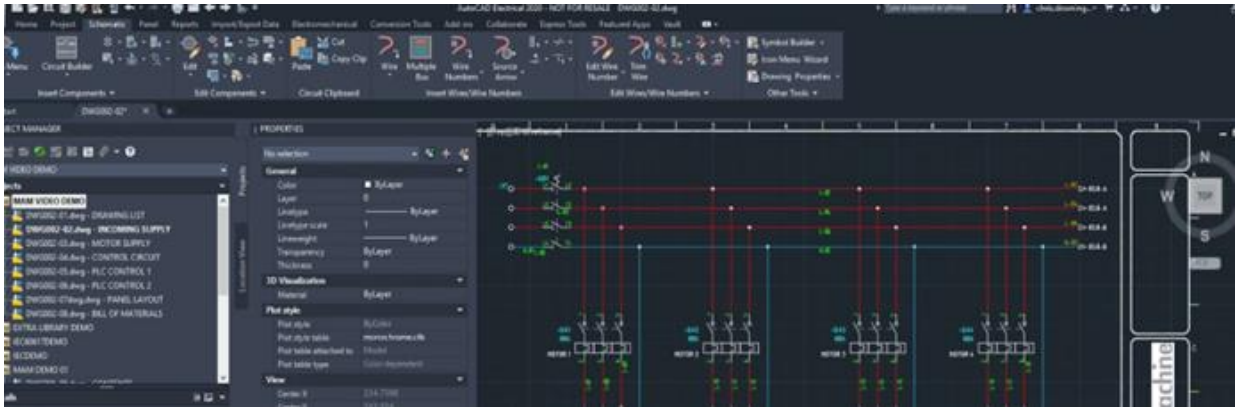

Рисунок 1.2 Приклад вигляду частини програми для моделювання роботи електричних схем AutoCAD Electrician

Схожий продукт до попереднього аналогу є XCircuit. Цей додаток на відмінно від попереднього є безкоштовним але дуже сильно уступає у функціоналі. Додаток має бібліотеку готових шаблонів популярних елементів, які можна використовувати під час складання схем [6]. Однак більш складних та рідко використовуваних елементів там немає [6]. Одним із мінусів цієї програми є складний і незвичний інтерфейс додатку, який можна освоїти тільки досвідним шляхом.

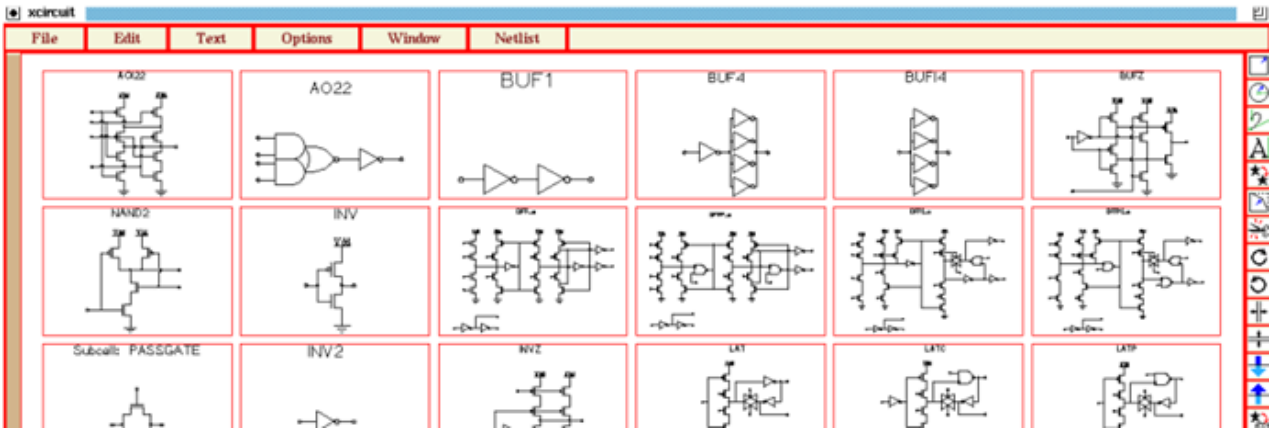

Рисунок 1.3 Приклад вигляду частини програми для моделювання роботи електричних схем XCircuit

Табличне порівняння розроблюваного web-додатку з аналогами конкурентів зображено в таблиці 1.1:

Таблиця 1.1 – Порівняння розроблюваного web-додатку з аналогами

	sPlan	AutoCAD Electrician	XCircuit	Власний Web-додаток
Доступність	В вільному доступі відстуній. Необхідне придбання ПО	В вільному доступі відстуній. Необхідне придбання ПО	У вільному доступі	У вільному доступі
Вартість придбання	50$ одноразово	235$ за місяць ліцензії	Безкоштовне	Можливість встановлення конкуренто спроможності
Призначення	Комерційне	Комерційне	Не комерційне, програма безкоштовна	Не комерційне, але з можливістью подальшого продажу
Необхідність обладнання для користувачів	Ні	Бажано	Ні	Ні

Розроблюваний web-додаток є дешевшим в розробці та використанні, программа не потребує додаткових встановлень для користувача і також не потрібно витрачати на придбання ліцензії. Більш доступний для усіх бажаючих хто хоче познайомитись з

роботою електричних схем.

Дослідження функціональних можливостей аналогів у порівняні з розроблюваним web-додатком представлено у таблиці 1.2:

Таблиця 1.2 – Порівняння функціональних можливостей розроблюваного web-додатку з аналогами

Критерії	sPlan	AutoCAD Electrician	XCircuit	Власний Web-додаток
Відображення графіків струму та напруги, а також можливість редагувати обрані графіки	Функціонал відображення графіків – відсутній	Функціонал відображення графіків - відсутній	Функціонал відображення графіків - відсутній	Функціонал відображення графіків струму та напруги на обраній електричній схемі - проісутній, також є можливість редагувати відображення графіків на обраній електричній схемі
Вбудована бібліотека створених електричних схем	Вбудована бібліотека створених електричних схем - відсутня	Вбудована бібліотека створених електричних схем - відсутня	Вбудована бібліотека створених електричних схем - відсутня	Вбудована бібліотека створених електричних схем - присутня
Можливість	Можливість	Можливість	Можливість	Можливість

змінювати швидкість симуляції електричної схеми	відсутня	відсутня	відсутня	присутня, а також існує можливість змінювати швидкість струму

На основі проведеного аналізу аналогів у таблиці 1.2 сформовано ряд функціональних вимог до власного додатку:

- Додаток повинен мати функціонал відображення графіків струму та напруги;
- Додаток повинен мати вбудовану бібліотеку створених електричних схем;
- Додаток повинен мати функціонал, який дає можливість змінювати швидкість симуляції електричної схеми.

1.3 Мета дослідження

Огляд актуального стану додатків для моделювання роботи електричних схем дозволив визначити цілі проекту.

Метою проекту є створення web-додатку для моделювання роботи електричних схем. Основне призначення додатку є створення нових або модифікування існуючих схем із бібліотеки.

Даний додаток надає ряд переваг для користувачів, а саме:

- Можливість створювати електричні схеми:
  а) на основі шаблону, обраного з базової бібліотеки схем
  б) шляхом побудови схеми з використанням бібліотечних об’єктів
- Можливість відображення графіків: графіків струму, графіків напруги, графік споживаної потужності;
- Можливість графічно відображати характеристики обраного елемента схеми;
- Можливість змінювати швидкість роботи електричної схеми;
- Збереження створеної схеми у форматі txt;
- Імпорт створеної схеми на робочу область додатку;
- Можливість додавати відображення значень на графіку такі як, частота, пікове значення, негативне пікове значення.

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Постановка задачі

На основі отриманих результатів під час аналізу було сформовано мету проекту – розробити web-додаток для моделювання роботи електричних схем.

Основна ціль проекту – створення web-додатку, призначеного для моделювання роботи електричних схем.

Для спрощення виконання поставленої задачі було прийнято рішення поділити основне завдання на підзадачі:

1. Дослідити предметну область;
2. Визначити переваги і недоліки аналогів;
3. Сформувати функціональні вимоги до додатку;
4. Обрати програмні засоби реалізації;
5. Реалізувати веб-додаток;
6. Протестувати роботу додатку.

Даний web-додаток матиме наступні можливості:

1. Можливість створювати електричні схеми:
    a. на основі шаблону, обраного з базової бібліотеки схем
    b. шляхом побудови схеми з використанням бібліотечних об’єктів
2. Можливість відображення графіків: графіків струму, графіків напруги, графік споживаної потужності;
3. Можливість графічно відображати характеристики обраного елемента схеми;
4. Можливість змінювати швидкість роботи електричної схеми;
5. Збереження створеної схеми у форматі txt;
6. Імпорт створеної схеми на робочу область додатку;
7. Можливість додавати відображення значень на графіку такі як, частота, пікове значення, негативне пікове значення.

2.2 Вибір засобів реалізації

Під час розробки web-додатку було використано різні програмні засоби. Розглянемо спочатку весь список використаних ПЗ (рис. 2.1), а саме:

1. Adobe Photoshop для створення макету;
2. IntelliJ IDEA для роботи з кодом;
3. OpenServer для розгортки сайту на локальному сервері;
4. Notepad++ для написання налаштувань та прикладів електричних схем.

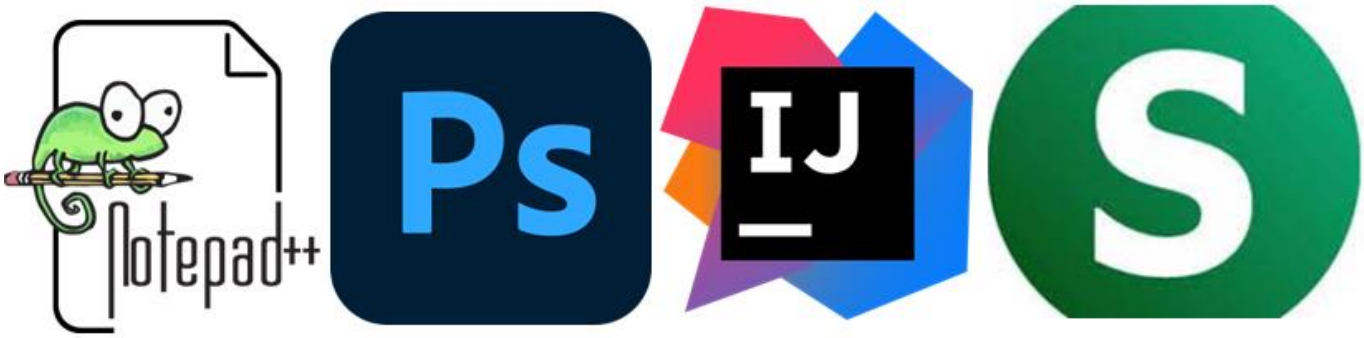

Рисунок 2.1 – Основні інструменти розробки додатку

Як основу додатку, було обрано один із найпопулярніших мов програмування – Java [7], через саме поширенність та простота у використанні. Та також через досить велику кількість матеріалів для вивчення та освоєння [8].

Для створення макету – web-додатку, та для різних графічних елементів та також для їх редагування – було використано Adobe Photoshop [9]. Можливість цього програмного засобу достатньо для редагування та створення елементів додатку.

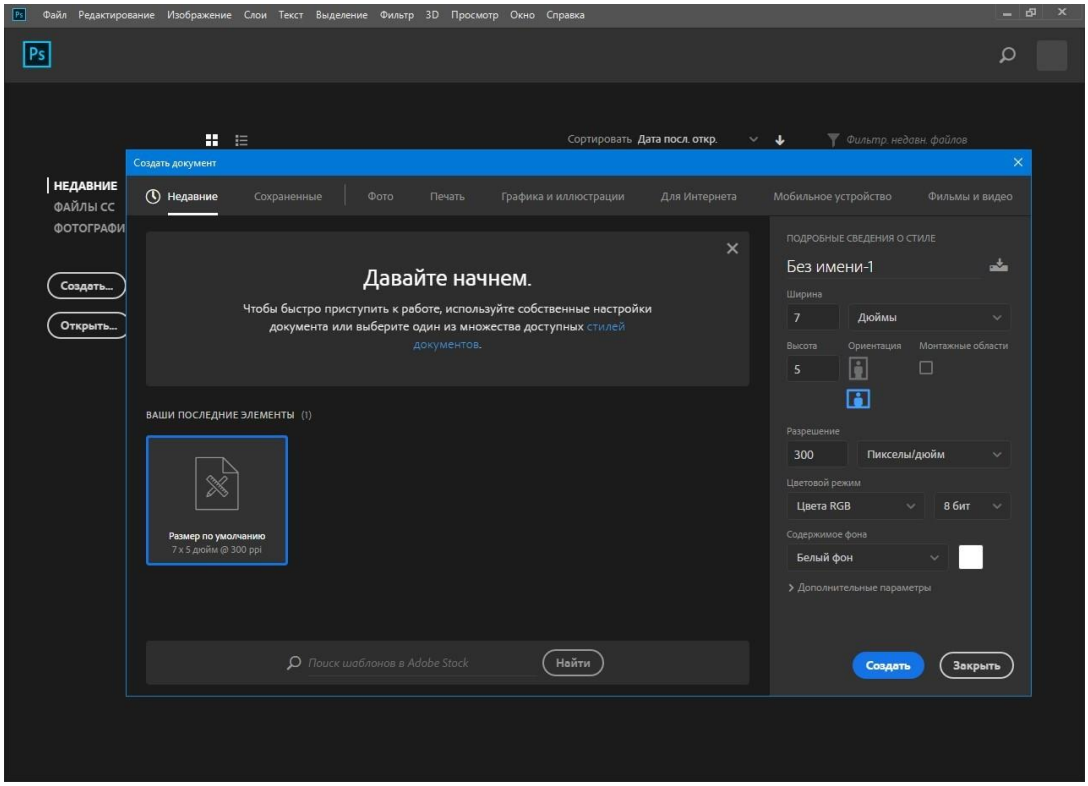

Рисунок 2.2 – Фоторедактор Adobe Photoshop

Для налаштувань проекту та створення списку з електричними схемами, було використано Notepad++, він є покращеною версію того ж самого блокноту. Notepad++ має багато корисних функцій, які спрощують роботу з написанням тексту [10].

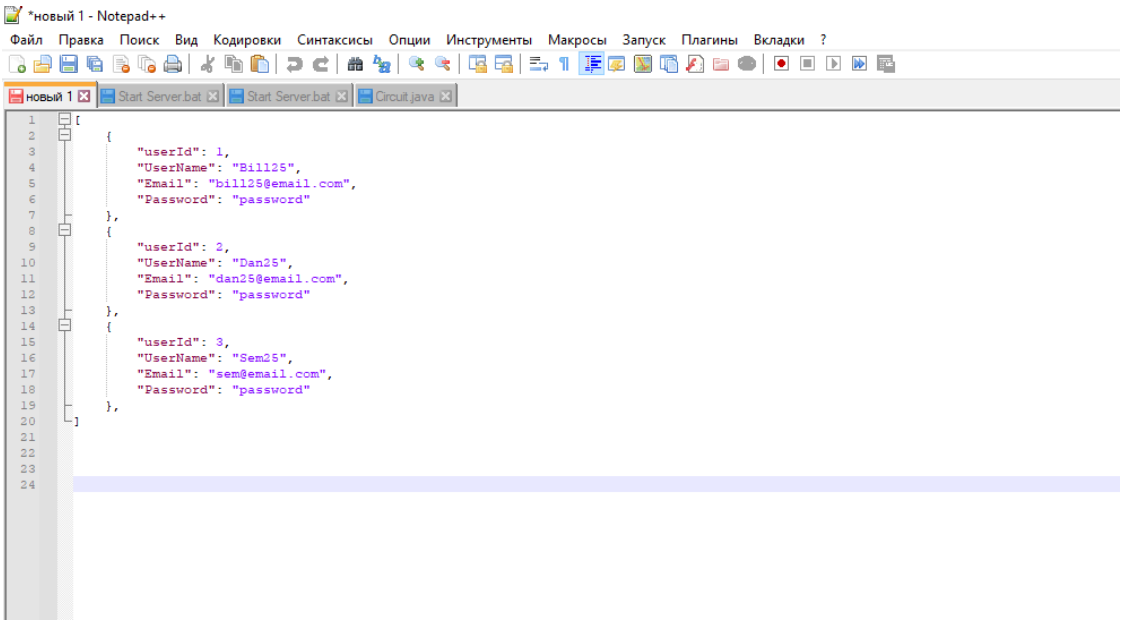

Рисунок 2.3 – Notepad++

Програма Open Server дає можливість відтворити додаток на локальному web-сервері. Open Server включає в себе безліч ретельно підібраних функцій, набір серверного ПЗ, а також ця програма є неймовірно зручною і продуманою, яка володіє потужними можливостями з адміністрування і налаштування всіх доступних користувачу компонентів (рис. 2.4) [11].

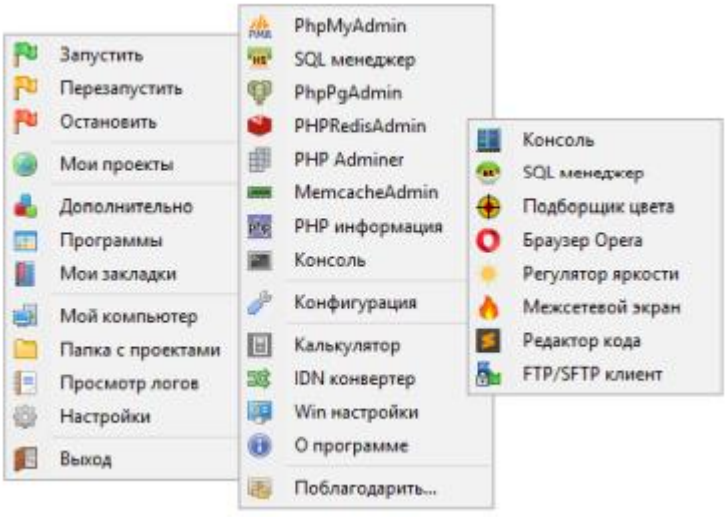

Рисунок 2.4 – Меню програми Open Server

Програма IntelliJ IDEA було використано для написання головного функціоналу, тому що цей ПЗ має безліч корисних та дуже потужних функцій які спрощують написання коду [12]. Також IntelliJ IDEA добре працює в парі з мовою програмування – Java, так як має досить велику бібліотеку плагінів, які покращають методику написання коду. Вбудований компілятор в цій програмі дуже якісно аналізує код і інколи виправляє або спрощує написання деяких функції і це дуже якісно впливає на продуктивність написання коду [13].

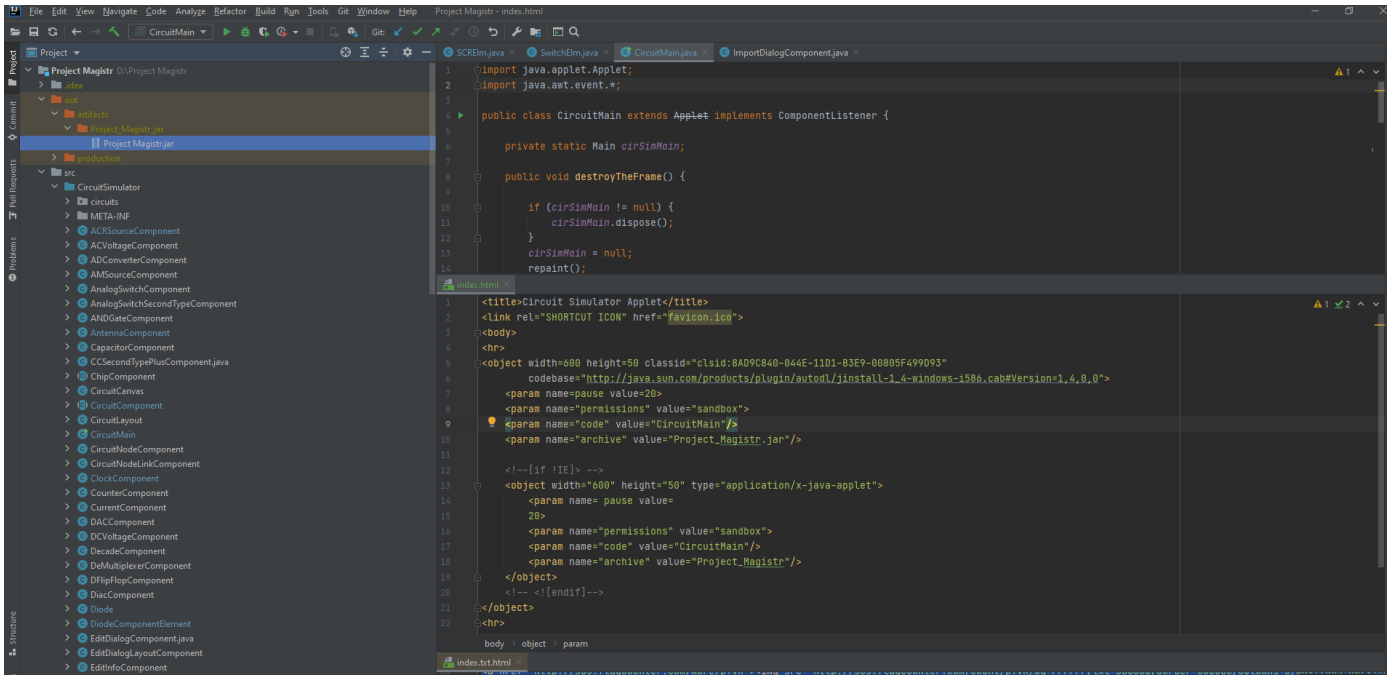

Рисунок 2.5 – IntelliJ IDEA

За допомогою цього списку обраних програмних засобів було реалізовано функціональні вимоги web-додатку:

1. створено макет;
2. розроблено функціонал;
3. локально розгорнуто web-додаток.

3 МОДЕЛЮВАННЯ РОБОТИ WEB-ДОДАТКУ

3.1 Моделювання роботи web-додатку в IDEF0

Після завершення основного етапу, а саме визначення основної мети та задачі кваліфікаційної роботи магістра було виконане фукціонувальне моделювання за методологією IDEF0. IDEF0 - це методологія графічного опису систем і процесів діяльності організації як безлічі взаємозалежних функцій. Вона дозволяє досліджувати функції організації, не пов'язуючи їх з об'єктами, що забезпечують їх реалізацію [14].

Контекстна діаграма web-додатку для моделювання роботи електричних схем та відповідно його декомпозиція, яку було реалізовано в блочному вигляді, для опису роботи створеного web-додатку, наведено на рисунках 3.1 – 3.2.

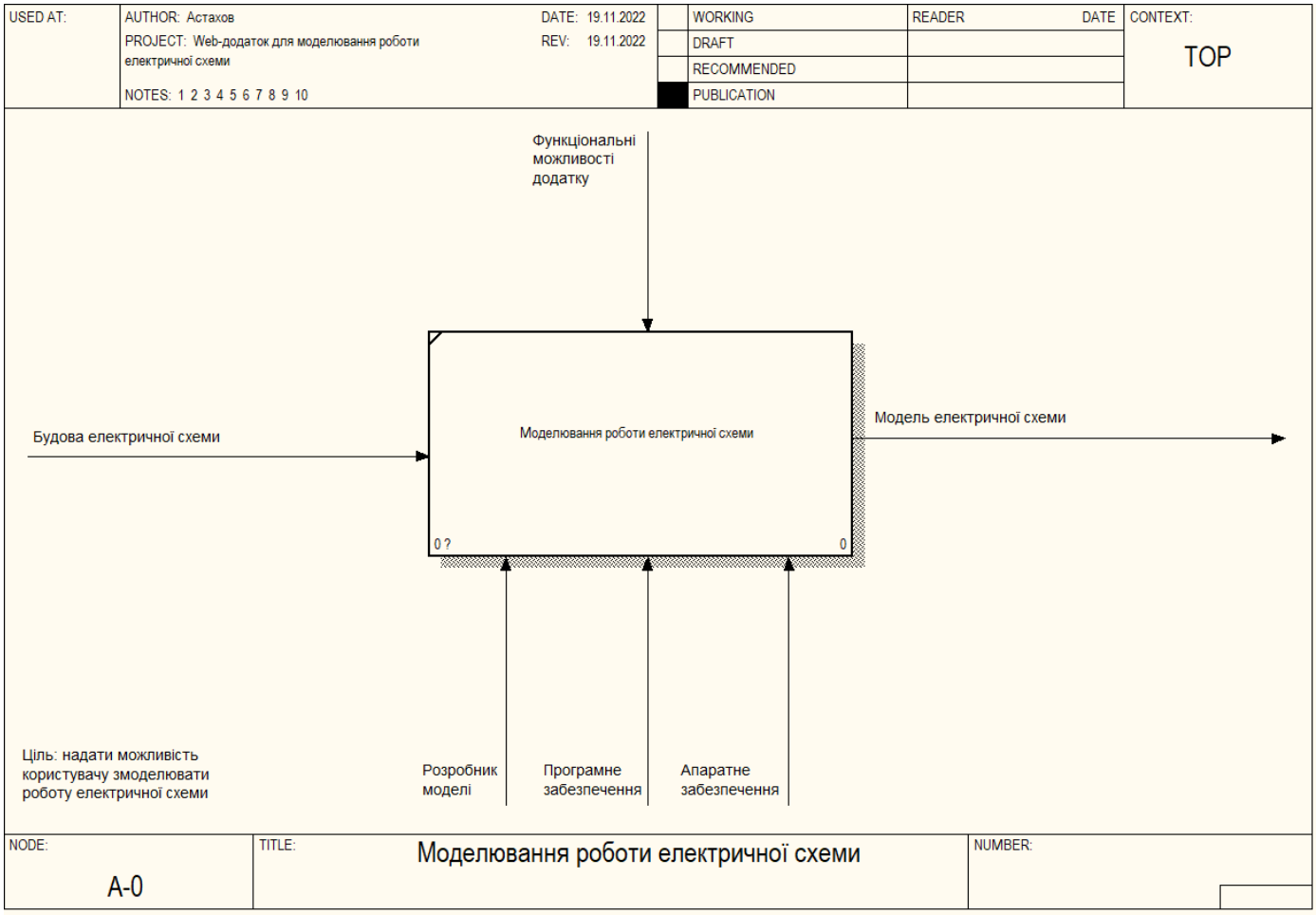

Рисунок 3.1 – Контекстна діаграмма

Декомпозиція контекстної діаграми, наглядно відображає всі основні етапи роботи створеного web-додатку. Декомпозиція містить не тільки основні блоки, але й основні матеріальні та інформаційні ресурси, які позначені за допомогою стрілок [15].

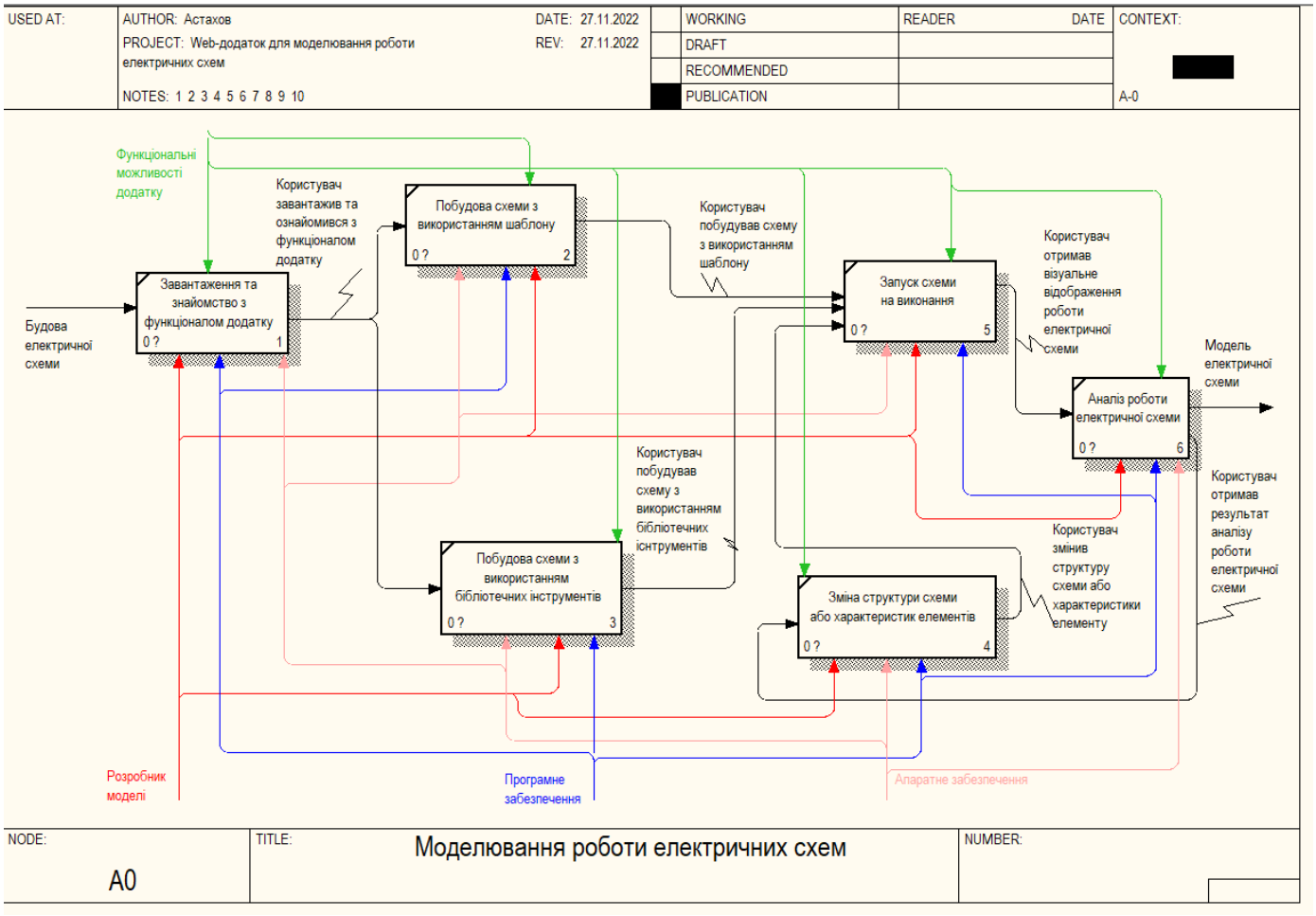

Рисунок 3.2 – Декомпозиція діаграми IDEF0

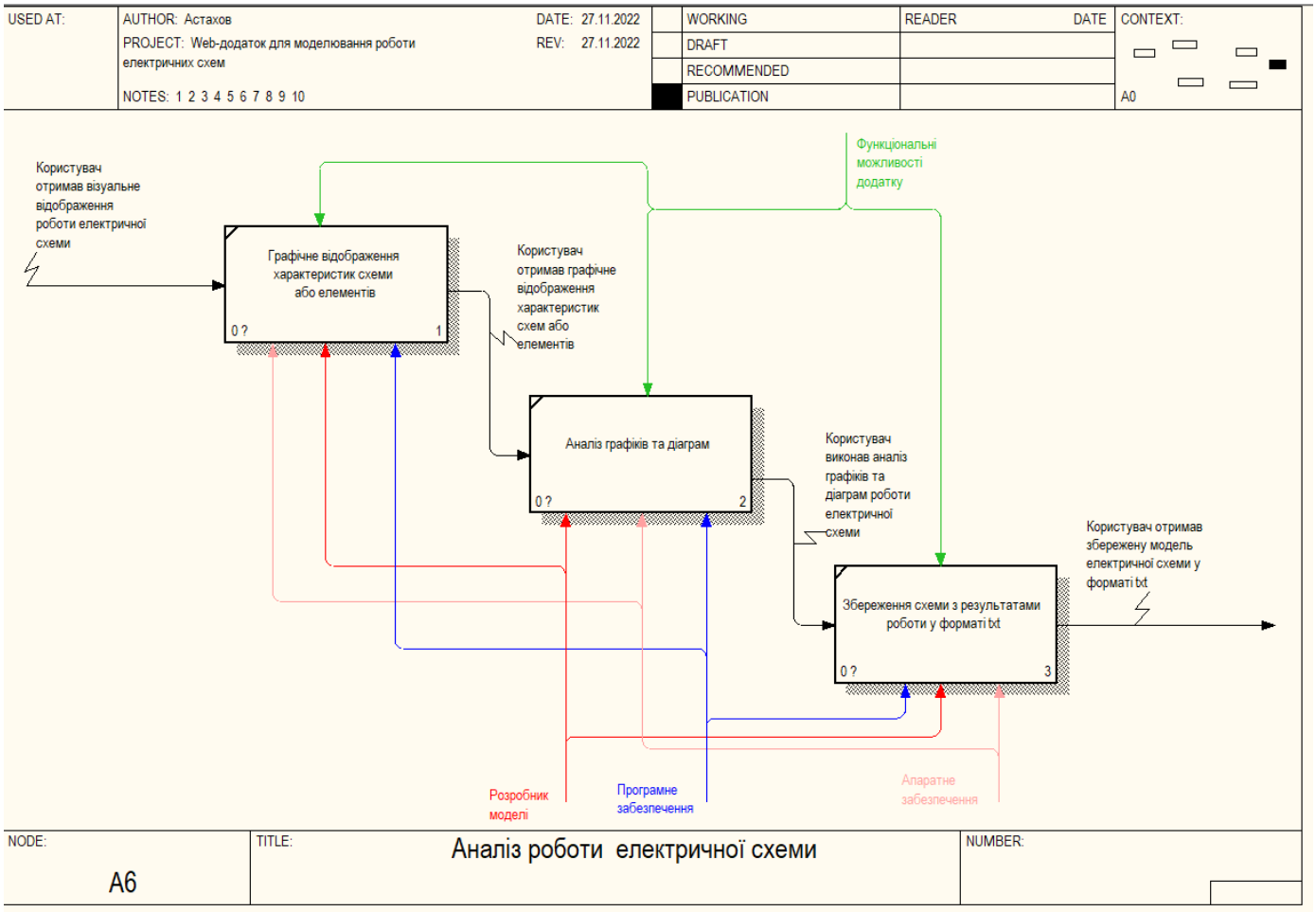

Рисунок 3.3 – Декомпозиція блоку Аналіз роботи електричної схеми

3.2 Моделювання варіантів використання

Завершенням етапу проектування web-додатку є саме розробка діаграми варіантів використання. Дана діаграмма створюється за для того, щоб описати поведінку web'-додатку, відображення користувачів та їх діяльності [16]. Діаграмма варіантів використання наведено на рисунку 3.4.

Рисунок 3.4 – Діаграма варіантів використання

Для даного web-додатку було визначено наступне:

- Користувач – має доступ до функціонала web-додатку;
- «Завантаження додатку» - дає можливість користувачу завантажити додаток;
- «Знайомство з функціоналом додатку» - дає можливість користувачу ознайомитись з функціоналом додатку;
- «Побудова електричної схеми» - дає можливість користувачу побудувати електричну схему;
- «Побудова схеми з використанням шаблону» - дає можливість користувачу побудувати схему з використанням шаблону;
- «Побудова схеми з використанням бібліотечних інструментів» - дає можливість користувачу побудувати схему використовуючи бібліотечні інструменти;
- «Запуск схеми на виконання» - дає можливість користувачу запустити виконання електричної схеми;
- «Графічне відображення електричних характеристик» - дає можливість користувачу переглянути графічне відображення електричних характеристик;
- «Аналіз графіків та діаграм електричних характеристик» - дає можливість користувачу проаналізувати графіки та діаграми електричних характеристик;
- «Зміна структури схеми» - дає можливість користувачу змінювати структуру схеми;
- «Зміна характеристик елементів схеми» - дає можливість користувачу змінювати характеристики елементів схеми;
- «Збереження схеми з результатами роботи у форматі txt» - дає можливість користувачу зберегти схему з результатами роботи у форматі txt.

4 РОЗРОБКА Web-додатку для моделювання роботи електричних схем

4.1 Короткий опис програмної реалізації

Перед тим як почати розглядати основний функціонал, розглянемо базові знання електроніки, які були використані в ході розробки програмної реалізації. Щоб імітувати роботу електричної схеми, є список правил, які потрібно дотримуватись для коректної роботи ПЗ:

1. Усі ланцюги повинні бути повними, замкнутими та мати принаймні одне джерело опору (resistor), щоб воно могло щось робити;
2. Хоча ПЗ забезпечує «коротке замикання» ланцюгів, але програма все одно буде попереджати користувача про існування ланцюга без джерела опору (resistor);
3. Замкнуте коло повинно мати принаймні один шлях, по якому електрика може безперервно текти від джерела живлення, через джерело опору і або назад до джерела живлення, або до заземлення;
4. ПЗ не дозволить ланцюгам, які не є замкнутими працювати належним чином, якщо тільки розрив не спричинений перемикачем (switch);
5. Усі з’єднання між проводами повинні закінчуватися в точці з’єднання;
6. Проводи, які перетинаються, не вважаються з’єднаними;
7. Проводи представляють собою такими, що мають незначний рівень опору. Якщо користувачу потрібен опір проводів, то він може додати резистор з відповідним малим значенням.

Розглянемо основні компоненти, які були розроблені для моделювання роботи електричних схем.

Web-додаток для моделювання роботи електричних схем дозволяє користувачеві організувати вибір компонентів які можна встановити, наприклад, перемикачі, мікросхеми, джерела живлення, тощо, у робочу схему, з’єднуючи компоненти проводами – приблизно так, як б це було зроблено при реальній розробці схеми. Нижче наведено деякі з основних компонентів та короткий опис цих компонент, які користувач може використовувати при розробці електричної схеми.

Джерело живлення

Існує чотири основні джерела живлення [17] (рис. 4.1) : 2-термінальний DC, 2-термінальний AC, 1-термінальний DC і 1-термінальний AC. Створюючи схеми в додатку, потрібно використовувати одне з цих чотирьох джерел живлення. Якщо використовується джерело з 1 терміналом, тоді потрібно також включити джерело заземлення. Однак, якщо використовується джерело з 2 терміналами, тоді можна або прив'язати до заземлення, або потрібно замкнути ланцюг за допомогою проводів, які повертаються до заземлення джерела живлення [18]. Для ланцюгів постійного струму напругу можна змінити, двічі клацнувши на джерелі живлення або навівши курсор на джерело живлення та клацнувши правою кнопкою миші.

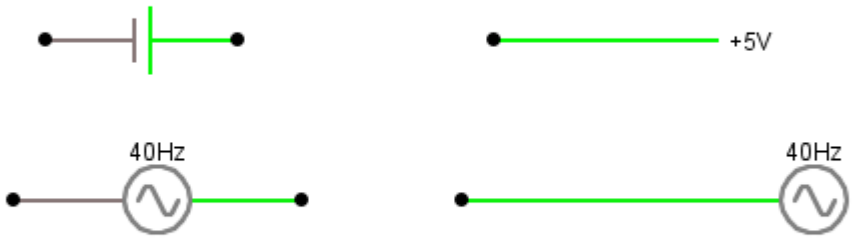

Рисунок 4.1 - Основні джерела живлення. Праві два джерела э 1-термінальними, а два зліва – є 2-термінальними

Перемикачі

Перемикачі - це механічні пристрої, які дозволяють користувачеві керувати станом електричного кола [19].

У створеному додатку є два типи перемикачів – одно позиційні вимикачі (рис. 4.2) і багато позиційні вимикачі (рис. 4.3) [20]. Для одноразових перемикачів опція «Switch» — це нормально закритий перемикач (він починає працювати в положенні

«closed» або «on»), а «Push Switch» — це нормально відкритий перемикач (він починає працювати у положенні «open» або «off»).

Рисунок 4.2 – Одно позиційні вимикачі “Switch” та “Push Switch”

Рисунок 4.3 – Багато позиційні вимикачі “Switch SPDT”

Реле

Реле – це електричні або електромеханічні пристрої, які дозволяють користувачеві перемикати стан перемикача з напругою, а не шляхом ручного маніпулювання перемикачем. Як і у перемикачів, у реле існує декілька конфігурацій, які вказують кількість ланцюгів або ліній, які можуть бути перемкнутими. Крім комутаційних схем, є також привід (електромагніт, який зазвичай називають котушкою), який робить перемикання [21].

Фізичні реле бувають різних форматів і типів (твердотільні, герконові, із замикання, тощо), але для простоти використання у створеному додатку підтримується

лише один тип . Крім того, фізичні реле мають різні варіанти напруги (5 VDC, 24 VDC, 110 VAC, тощо) і обмеження напруги та струму, на які розраховані контактори [22]. Але так само як і типи, формати – створений додаток ігнорує ці ознаки заради простоти, що означає, що будь-яке джерело живлення може використовуватися для моделювання роботи електричних схем.

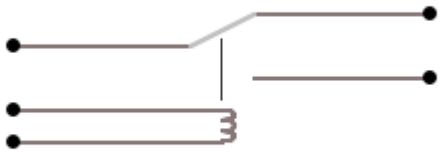

Рисунок 4.4 – Вигляд реле

Світильники (світлодіодні та лампи)

Однією з компонентів, де створений додаток має обмежений функціонал, є вихідні дані, мається на увазі саме «фізичні» виходи, наприклад, світлодіоди та лампи [23]. Якщо користувачу потрібно використовувати, наприклад, двигун, клапан, тощо, то користувач замість цього може використовувати створений функціонал світлодіодів та лампи. Оскільки і світлодіоди, і лампи насправді є не що інше, як джерела опору в ланцюзі, потрібно трохи думати про те, що використовується та як вони входять у схему, коли маємо справу з логічними схемами реле [24]. У будь-якому випадку редагування властивостей світла дозволить змінювати напругу, колір та інші засоби керування продуктивністю, якщо вони знадобляться.

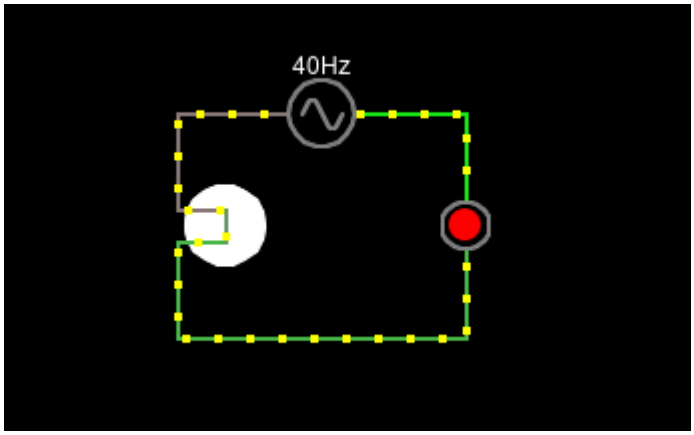

Рисунок 4.5 – Приклад вигляду роботи світильника та лампи

4.2 Огляд створеного web-додатку для моделювання роботи електричних схем

Розглянемо створений web-додаток та основні функції цього додатку.

При початку роботи з web-додатком користувачу відображається типова схема (рис. 4.6), цю схему можна видалити перейшовши в меню Circuits і вибравши із спливаючого списку - Blank Circuit, це дає користувачеві не тільки очистити робочу область додатку, але й почати розробляти власну схему з нуля (рис. 4.7).

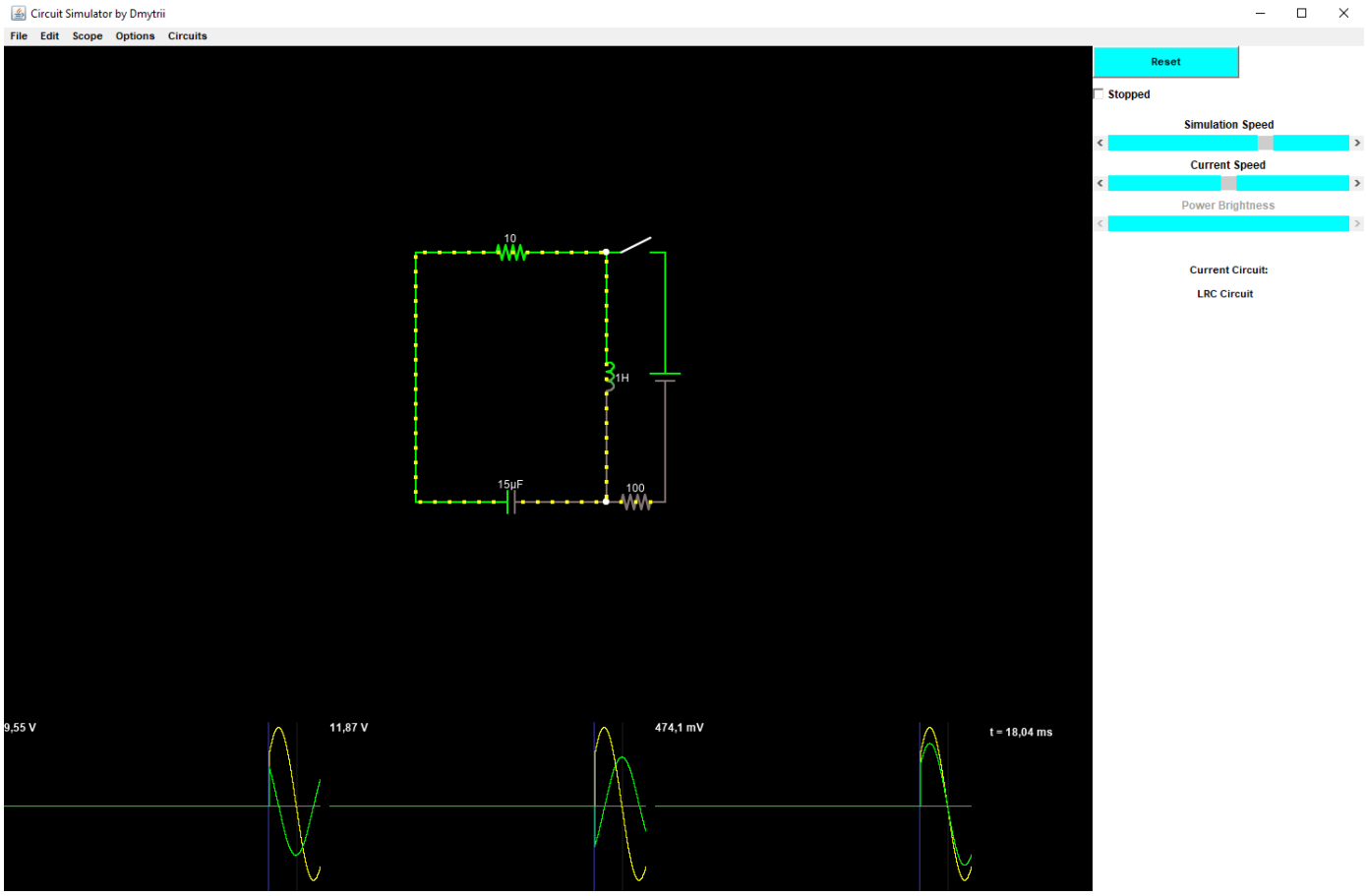

Рисунок 4.6 – Початковий екран створеного web-додатку

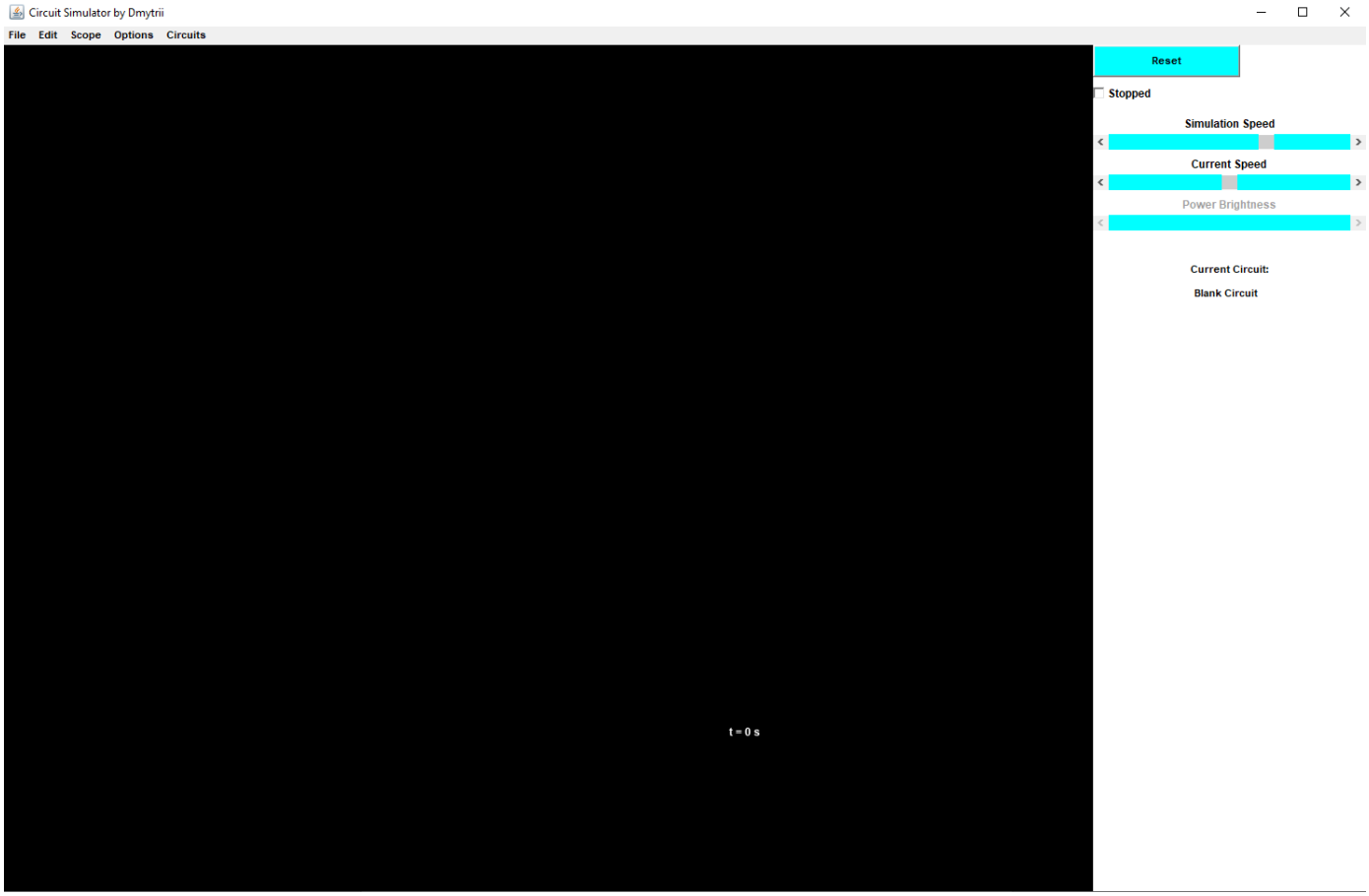

Рисунок 4.7 – Приклад вигляду Blank Circuit

На екрані створеного додатку є чотири області (рис. 4.8):

1. Меню, яке знаходиться зверху;
2. Основна область, в якій розроблено схему;
3. Область вздовж правої сторони, яка містить елементи керування симуляцією (Simulation Speed, Reset, checkbox stopped, ім'я поточної схеми, параметри змінних компонентів);
4. Коли користувач наводить курсор на компоненти, то тоді користувач зможе побачити, як вони змінюють колір на синій, а опис разом із поточним станом відображатиметься в нижньому правому куті області дизайну праворуч від області видимості.

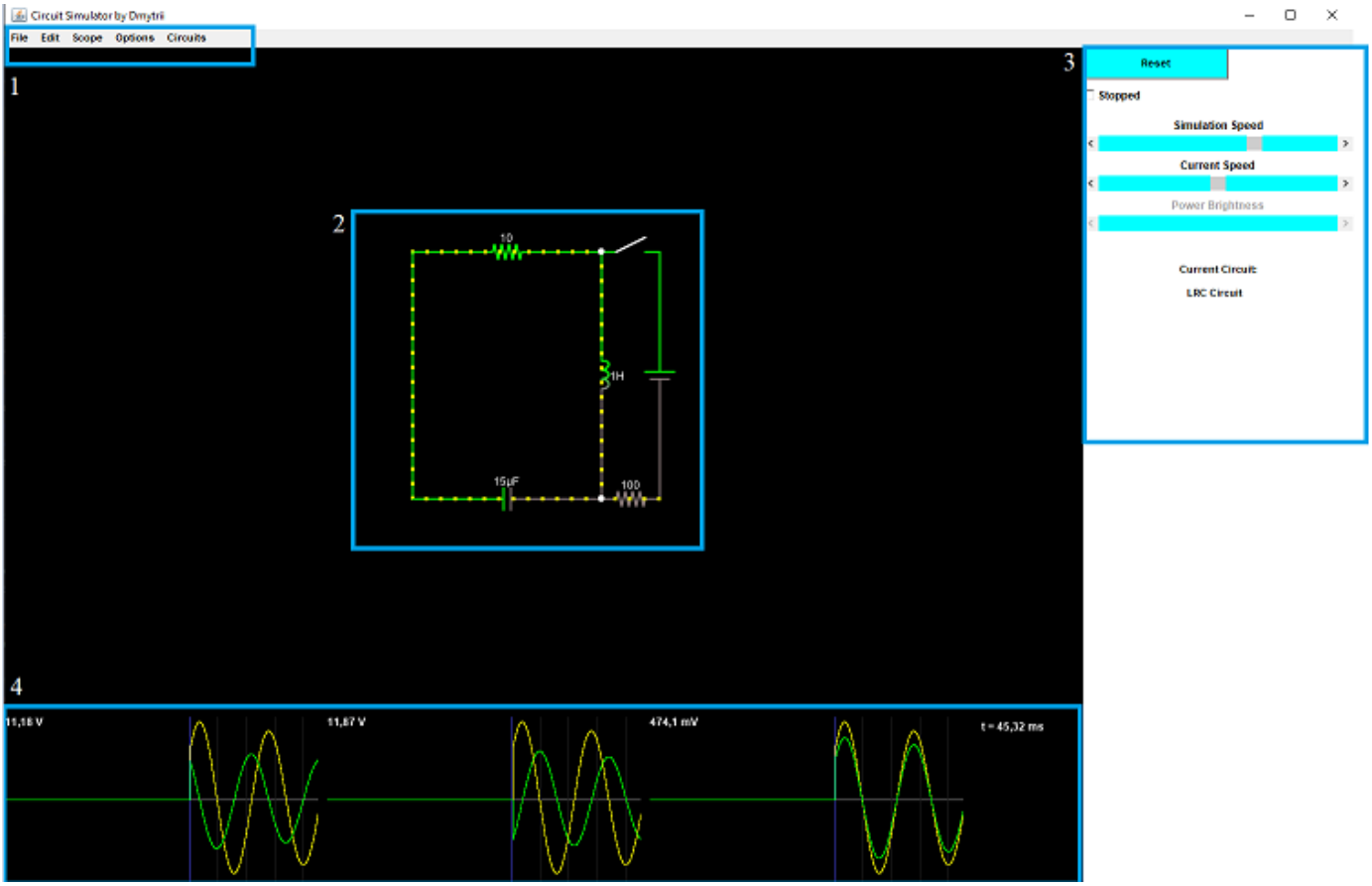

Рисунок 4.8 – Основні області створеного додатку

Розглянемо кожну з областей меню детальніше. Структура меню для створеного web-додатку відповідає традиційній логікі Windows (рис. 4.9), з універсальним меню “File” який має в собі такі функції як Import/Export за допомогою якого можна заекспортувати свою схемою у файл та зберегти собі на комп’ютері і Exit, за допомогою якого можна вийти із додатку. Меню “Edit” має стандарті функції, по типу Undo, Cut, Copy, Paste. Меню “Scope” дає можливість налаштовувати області відображення графіків, також якщо користувач натисне правою кнопкою миші по області відображення графіків, то тоді з’явиться вікно в якому користувач може редагувати цю область та змінювати деякі параметри, або взагалі видалити обраний графік (рис. 4.10). Налаштування зовнішнього вигляду та функціональності симуляції відповідає меню “Options” і останнє меню “Circuits” в якому вказані зразки готових схем, які користувач може вибрати та тестувати або модифікувати.

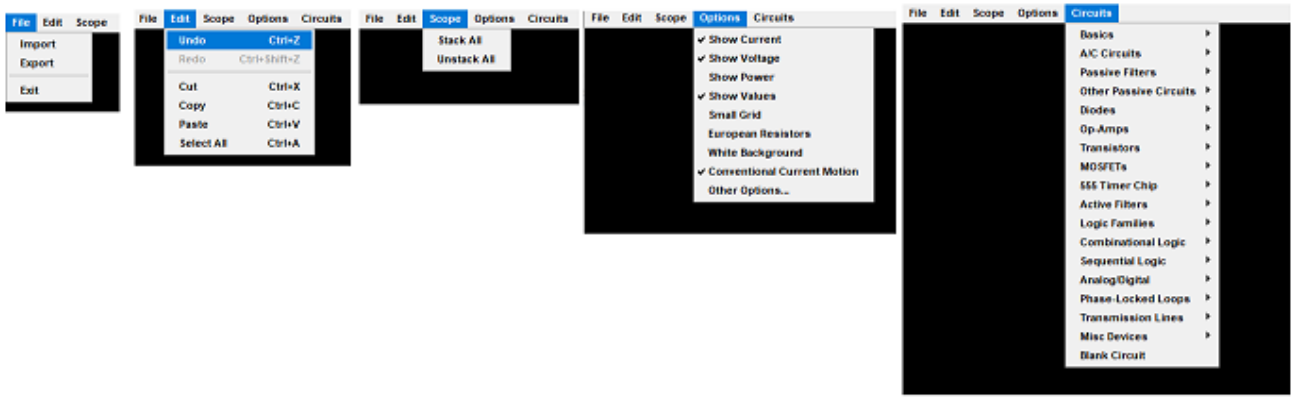

Рисунок 4.9 – Структура меню

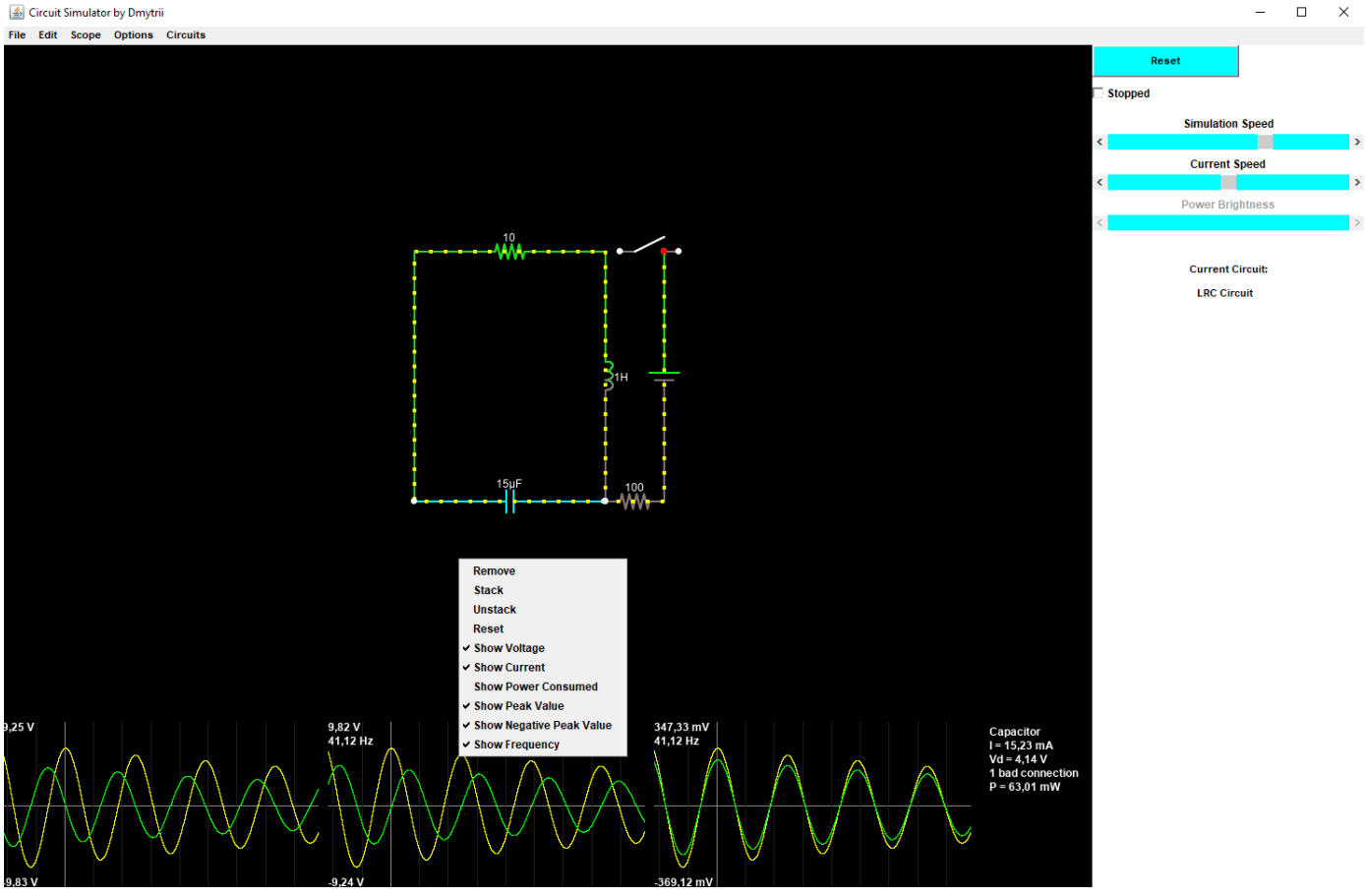

Рисунок 4.10 – Меню редагування області графіків

Коли програма працює, тоді на схемі є такі позначення – проводи які мають червоний колір означають, що вони з негативною напругою, зелені означають, що проводи з позитивною напругою, а сірі – із заземленням, рухомі жовті точки вказують на струм. Розробка та модифікація схеми може відбуватися в будь-якому стані

(прапорець є на кнопці Stopped – вимкнено, прапорця немає – увімкнена). Кнопка Reset скине останні результати симуляції роботи електричної схеми та почне її робити по-новому.

Існує деяка кількість функцій які можна виконувати за допомогою комбінацій клавіш і миші, список функцій наведено у таблиці 4.1:

Таблиця 4.1 – Список функцій, які можна виконувати за допомогою комбінацій клавіш і миші

Комбінація клавіши і миші	**Функція**
ALT + left mouse	Затиснувши вказану комбінацію, користувач може перемістити всі компоненти, які знаходяться на робочій області
CTRL + A	Натиснув на вказану комбінацію, користувач може виділити всі компоненти, які знаходяться на робочій області з подальшою можливістю редагування, також з цією функцією є можливість як вадили все обране, так і скопіювати та вставити використовуючи сталі комбінація (CTRL + C, CTRL + V, Delete)
Right mouse	Натиснув на вказану кнопку, користувачу відкриється меню з додаванням нових компонентів в робочу область
Left mouse	Затиснувши вказану кнопку, користувачу дається можливість за допомогою курсора виділити потрібні компоненти в робочій області
CTRL + right mouse	Затиснувши вказану комбінацію, користувачу дається можливість змінювати довжину компоненти

CTRL + right mouse	Ця функція також має деякі модифікація, коли користувач при зменшенні або збільшенні компоненти доторкнеться до іншої компоненти, то програма її підв’яже як до нового ланцюга і тоді користувач одночасно зможе змінювати вже дві компоненти, і так може дійти ланцюг до необмеженої кількості компонент
CTRL + left mouse	Затиснувши вказану комбінацію, користувачу дається можливість змінювати положення та розмір однієї компоненти на яку користувач натисне

Розглянемо на прикладі деякі функції. CTRL + A, як вказано в таблиці 1 – натиснувши цю комбінацію побачимо наступне (рис. 4.11), виділилася вся робоча область, і якщо, наприклад натиснути CTRL + C, а після CTRL + V, ми можемо побачити наступне – рисунок 4.12. Схема скопіювалася і після вставлення вона з’явилась поруч з попередньою схемою.

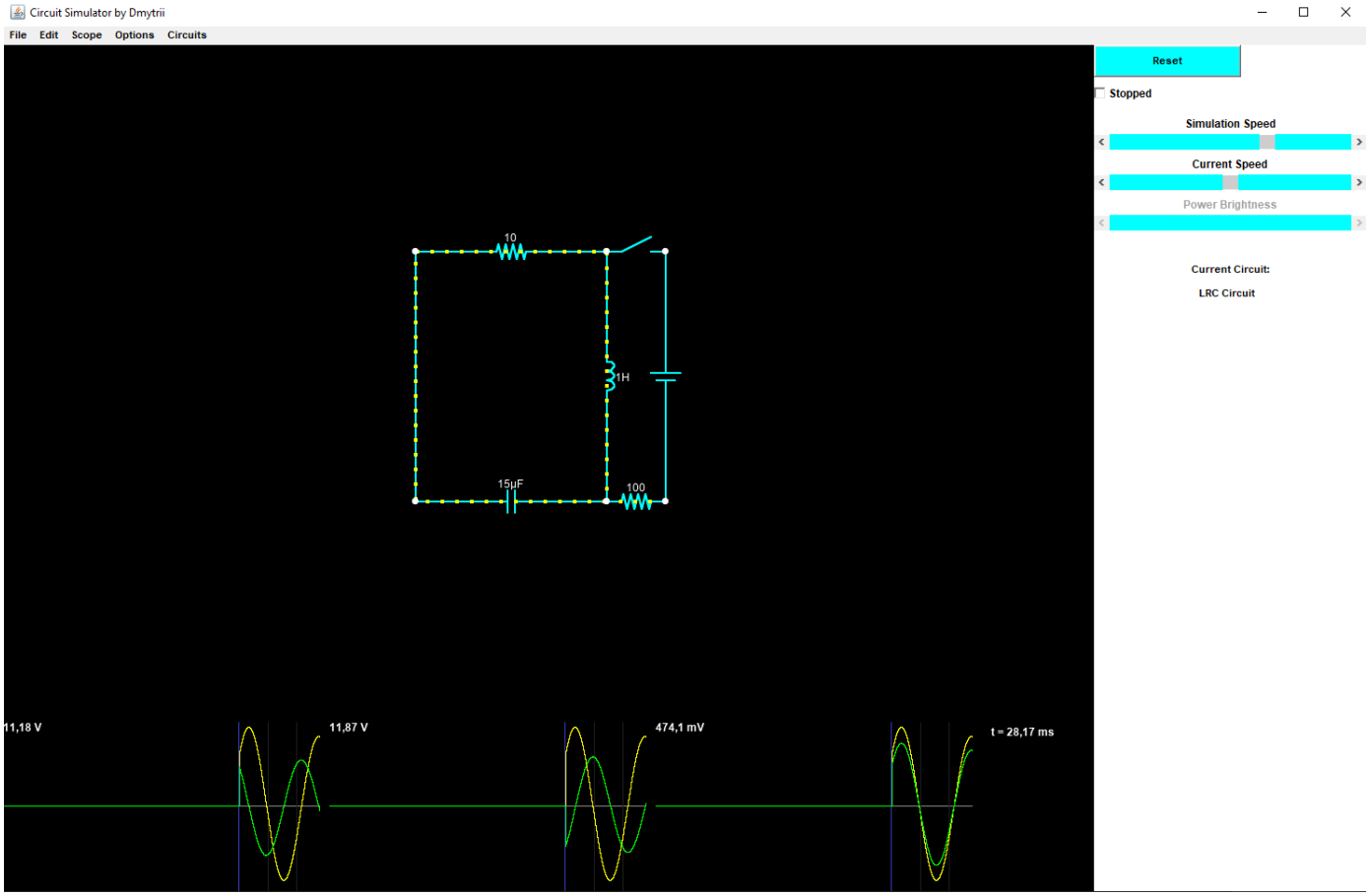

Рисунок 4.11 – Приклад вигляду виділення всіх компонентів за допомогою CTRL + A

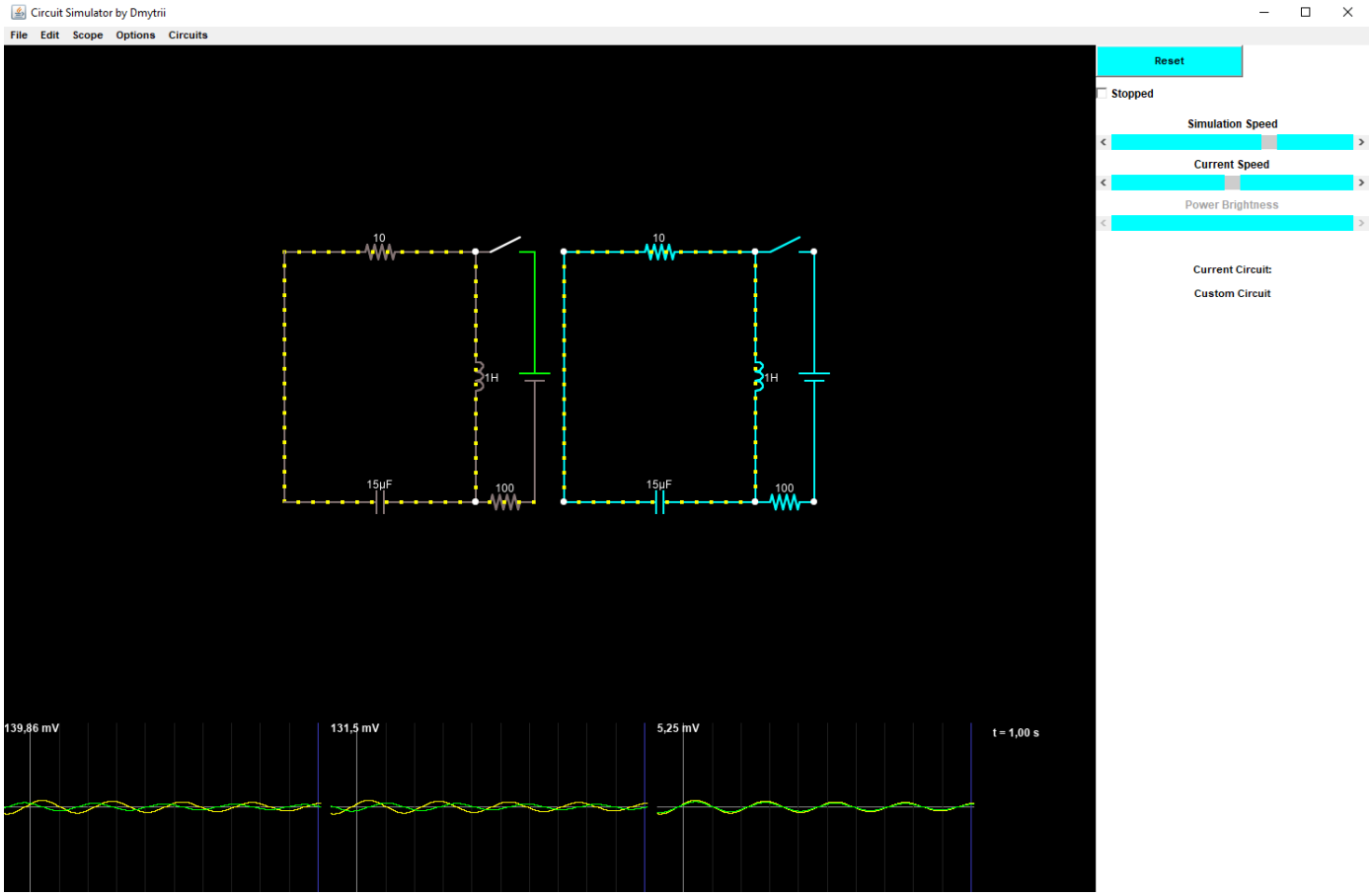

Рисунок 4.12 – Приклад копіювання та вставлення схеми в робочу область

Розглянемо приклад функції натискання кнопки Right mouse. Після натискання цієї кнопки над робочою областю користувачу з’явиться список компонентів, які він може додати до своєї схеми (рис. 4.13). Як видно з рисунку, користувач може додавати достатньо велику кількість компонентів, та редагувати їх (рис. 4.14).

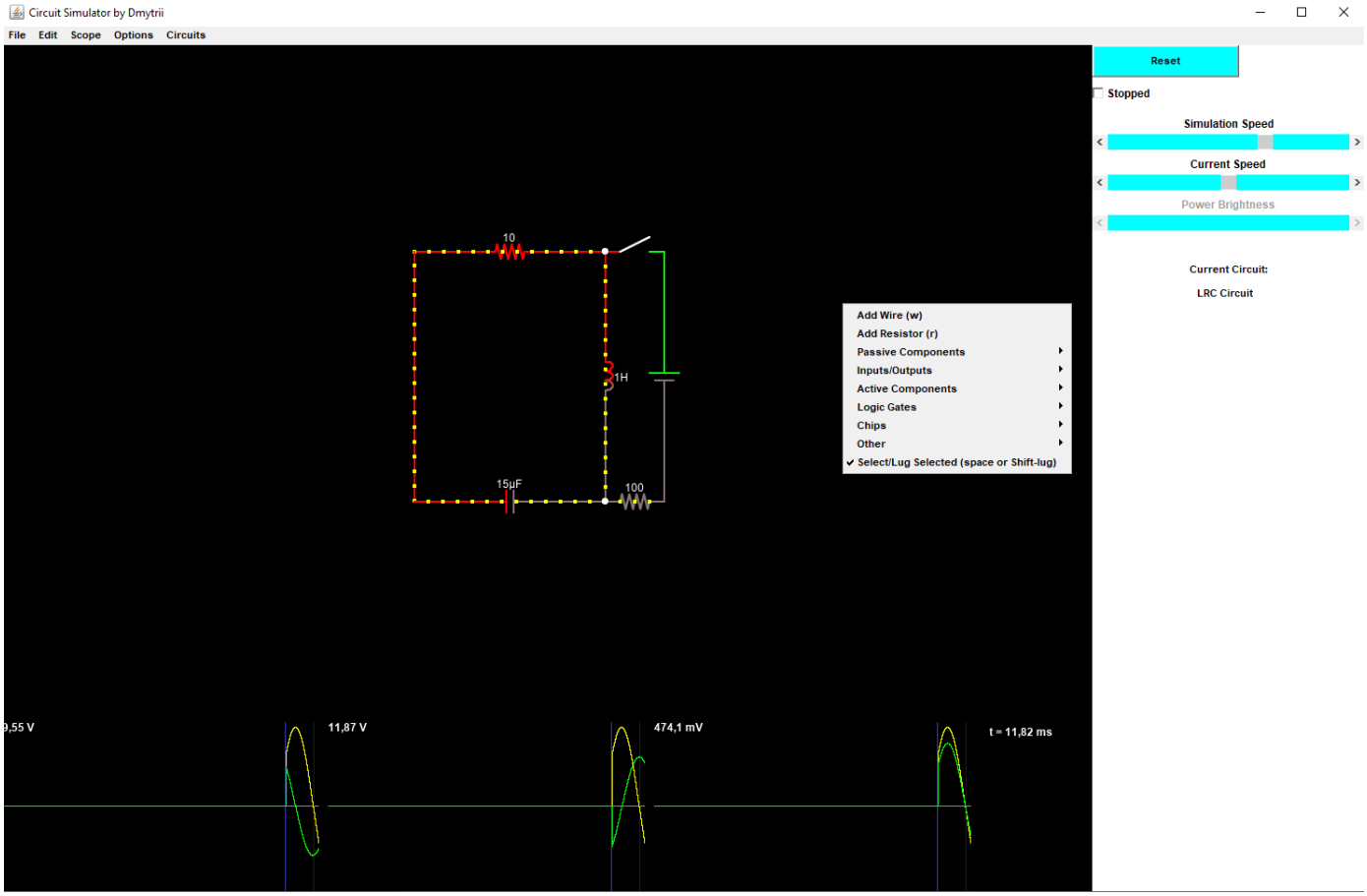

Рисунок 4.13 – Приклад вигляду списку компонентів

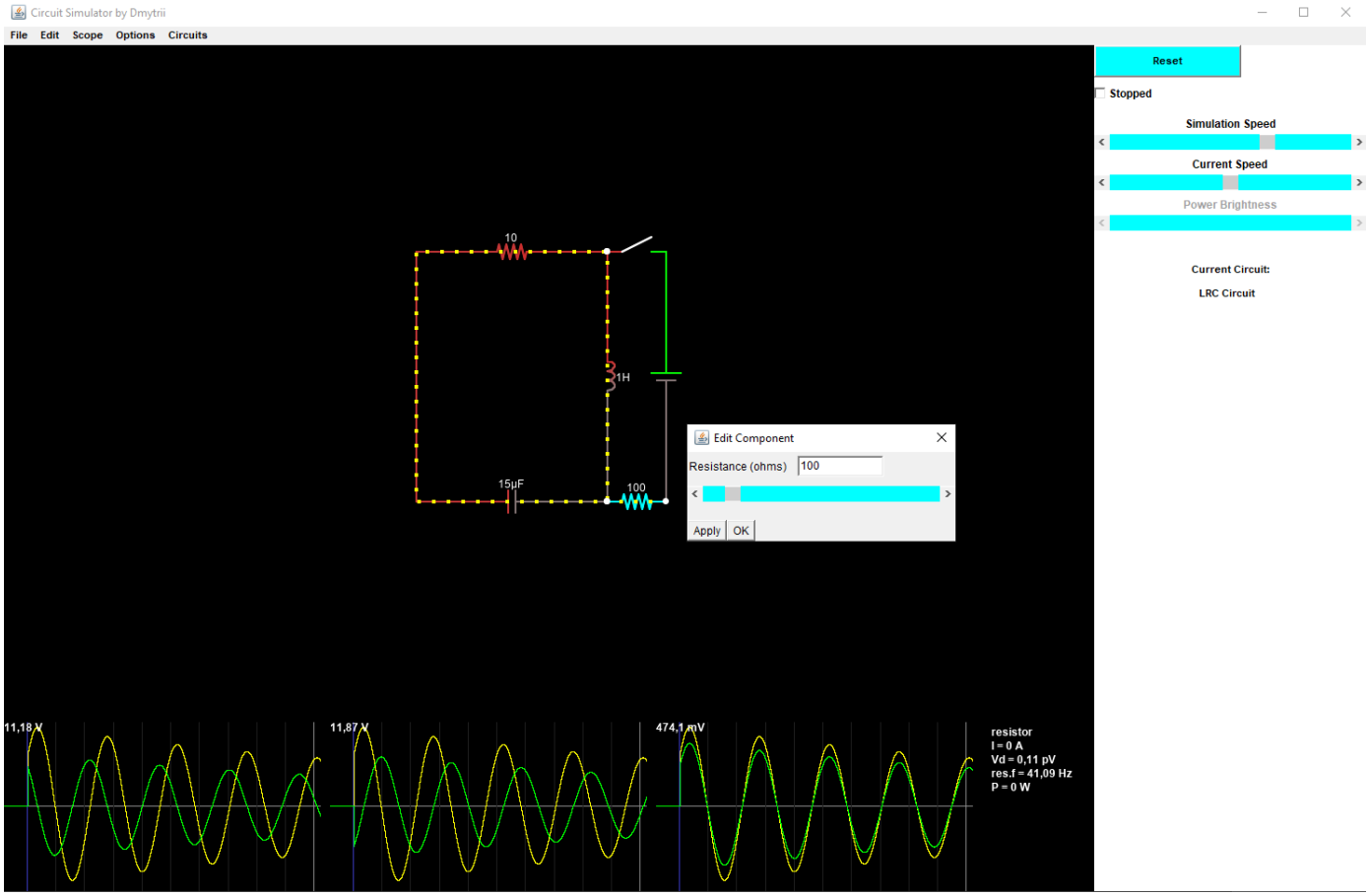

Рисунок 4.14 - Приклад вигляду редагування компоненти – Resistor

В результаті усі поставлені задачі на початку проекту на розробку web-додатку для моделювання роботи електричних схем було виконано. Тобто реалізовано:

1. Можливість створення нової електричної схеми;
2. Можливість вибору схеми із списку вже існуючих схем;
3. Можливість редагувати схему та додавати нові компоненти;
4. Можливість відслідковувати зміни напруги та струму на графіку;
5. Можливість змінювати режим роботи симуляції (збільшувати струм або збільшувати швидкість симуляції).

ВИСНОВОК

Даний дипломний проект було присвячено створенню web-додатку для моделювання роботи електричних схем.

У даному проекті було проаналізовано та вибрано форму розробки, програмні засоби реалізації. Доцільним визначено використання мови Java для реалізації основної логіки програми.

Мета досягнута, поставлені завдання виконані:

1. Досліджена предметна область.
2. Проведено аналіз аналогів.
3. Обрана стратегію розробки.
4. Програмна реалізація.

Реалізована функціональність додатку відповідає поставленим вимогам. Клієнт може створювати нові , або вдосконалювати вже існуючі електричні схеми та перевіряти їх. Всім користувачам доступна перегляд базову бібліотеку електричних схем, додавати нові елементи електричної схеми, вносити зміни до характеристик елементів електричної схеми, переглядати графіки напруги та струму, вносити зміни до відображень значень на графіку, імпорт та експорт створених схем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Вплив інформаційних технологій на розвиток бізнесу [Електронний ресурс] – Режим доступу до ресурсу: https://cyberleninka.ru/article/n/vliyanie-informatsionnyh-tehnologiy-na-razvitie-biznesa/viewer.
2. Вчені, що вивчають інформаційно-комунікаційні технології. [Електронний ресурс] – Режим доступу до ресурсу: https://studwood.net/1486575/ekonomika/uchenye_izuchayuschie_informatsionno_kommunikatsionnye_tehnologii.
3. Тенденції світового IT-ринку. [Електронний ресурс] – Режим доступу до ресурсу: https://www.tadviser.ru/.
4. ТОП-10 програм-помічників електрику. [Електронний ресурс] – Режим доступу до ресурсу: https://ds-electronics.com.ua/.
5. Best Web-Based Electrical Design Software. [Електронний ресурс] – Режим доступу до ресурсу: https://www.capterra.com/electrical-design-software.
6. Best Electrical Design Software. [Електронний ресурс] – Режим доступу до ресурсу: https://www.goodfirms.co/software.
7. Рейтинг мов програмування 2022. [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/651585/.
8. Посібник з мови програмування Java. [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/java/tutorial/.
9. Уроки Adobe Photoshop для початківців. [Електронний ресурс] – Режим доступу до ресурсу: https://vse-kursy.com/read/254-uroki-fotoshopa-dlya-nachinayuschih-besplatnye-video-dlya-zanyatii-doma.html
10. Список функцій. [Електронний ресурс] – Режим доступу до ресурсу: https://notepadpp.fandom.com/ru/wiki.
11. Документація Open Server. [Електронний ресурс] – Режим доступу до ресурсу: https://ospanel.io/docs/.
12. IntelliJ IDEA: корисні функції для розробників [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/wrike/blog/318136/.
13. IntelliJ Idea. [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/idea/.

14. Методологія IDEF0. [Електронний ресурс] – Режим доступу до ресурсу: https://stud.com.ua/87184/ekonomika/metodologiya_idef0.
15. Призначення та склад методології IDEF0. [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_2?pli=1.
16. Діаграма варіантів використання. [Електронний ресурс] – Режим доступу до ресурсу: https://flexberry.github.io/ru/fd_use-case-diagram.html.
17. Модуль живлення AC-DC. [Електронний ресурс] – Режим доступу до ресурсу:https://ua.ehpowersupply.com/info/what-is-ac-dc-power-adapter-62713563.html.
18. Блок живлення. [Електронний ресурс] – Режим доступу до ресурсу: https://elf-led.com.ua/upload/iblock/65d/Blok-zhivlennya-impulsniy-ELF_-12V_-33_3A_-400Vt-elf-0.pdf.
19. Перемикач. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki.
20. Перемикачі SPDT. [Електронний ресурс] – Режим доступу до ресурсу: https://deneb-80.ru/electronic/form_of_contacts_switch_relays_spst_spdt/.
21. Реле. [Електронний ресурс] – Режим доступу до ресурсу: https://anlan.ru/articles/ustrojstvo-i-tipy-elektricheskih-rele.
22. Реле. Типи реле. [Електронний ресурс] – Режим доступу до ресурсу: https://220volt.com.ua/news/useful/elektrika-i-svet/kakie-byvayut-tipy-rele.html.
23. Світлодіоди та лампи. [Електронний ресурс] – Режим доступу до ресурсу: https://corelamps.com/elektronika/svitlodiod/.
24. Характеристика світлодіодів. [Електронний ресурс] – Режим доступу до ресурсу: https://kindly.com.ua/harakteristiki-svitlodiodiv-spojivannia-strymy-napryga-potyjnist-i-svitloviddacha.
25. Формуйте ефективні цілі за методикою SMART за допомогою цих порад та прикладів. [Електронний ресурс] – Режим доступу до ресурсу: https://asana.com/ru/resources/smart-goals.
26. Що таке SMART мета: правила постановки. [Електронний ресурс] – Режим доступу до ресурсу: https://sendpulse.ua/ru/support/glossary/smart-goal.
27. Що таке ієрархічна структура робіт та як з її допомогою ефективно реалізувати

проект. [Електронний ресурс] – Режим доступу до ресурсу: https://netology.ua/blog/02-2022-what-is-wbs.

28. Lavrov, E., Siryk, O., Chybiriak, Y., Danilova, L., Nahornyi, V., & Vakal, S. (2021). A model for the organization of adaptive dialogue interaction 'man-computer' taking into account the requirements of reliability and efficiency. Paper presented at the 2021 IEEE 4th International Conference on Advanced Information and Communication Technologies, AICT 2021 - Proceedings, 31-35. doi:10.1109/AICT52120.2021.9628939
29. Що таке діаграми Ганта? [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/agile/project-management/gantt-chart.
30. Ризики проекту: аналіз, оцінка та стратегії управління. [Електронний ресурс] – Режим доступу до ресурсу: https://skill setter.io/blog/risk-management-us.
31. Lavrov, E., Chybiriak, Y., Siryk, O., Logvinenko, V., & Zakharova, A. (2022). Training of specialists for adaptive management. techniques for teaching computer analysis of automated production systems in the FlexSim environment. Paper presented at the CEUR Workshop Proceedings, , 3104 106-118
32. Lavrov, E. A., Siryk, O. E., Chybiriak, Y. I., Zolkin, A. L., & Sedova, N. A. (2022). Human-centered management in polyergatic information systems. multi-criteria distribution of functions between operators. Paper presented at the IOP Conference Series: Earth and Environmental Science, , 1049(1) doi:10.1088/1755-1315/1049/1/012020
33. Lavrov, E., Siryk, O., Kirichenko, I., Barchenko, N., & Chybiriak, Y. (2021). The methodology of managed functional networks for organizing effective and adaptive human-machine dialogue in automated systems. Paper presented at the CEUR Workshop Proceedings, , 3013 428-437.

ДОДАТОК А

ПЛАНУВАННЯ РОБІТ

А.1 Ідентифікація ідеї проекту

З розвитком комп’ютерної техніки в основі яких є електричні схеми з’явилась велика тенденція до створення програм які використовуються для моделювання роботи електричних схем, або покращення вже існуючих програм. Це дозволяє полегшити створення цих електричних схем, тому що доволі зручно мати все що потрібно для створення електричної схеми в одній програмі.

В сучасному світі кожна компанія, яка працює з електротехнікою має при собі власні або високотехнологічні програми, які полегшують створення та тестування електричної схеми перед тим як створити її фізично. Для компаній яка працює з електротехнікою дуже важливо мати власні або використовувати існуючі додатки для моделювання роботи електричних схем з ряду причин:

- спрощення створення електричної схеми;
- тестування електричної схеми;
- перевірка сумісності різних компонентів, яку можна перевірити за допомогою графіків напруги та струму;
- автоматизація та прискорення створення електричної схеми.

А.2 Деталізація мети методом SMART

Метою дипломного проекту є «Web-додаток для моделювання роботи електричних схем». Створений web-додаток спрямований на покращення та спрощення моделювання роботи електричних схем [25-26]. Результати деталізації мети проекту методом SMART розміщенні у табл. А.1.

Таблиця А.1 – Деталізація мети методом SMART

Specific (конкретна)	Розробити web-додаток для моделювання роботи електричних схем.
Measurable (вимірювання)	Результатом роботи web-додатка для моделювання роботи електричних схем є оцінка користувача.
Achievable (досяжна, узгоджена)	Ціль даного проекту «Web-додаток для моделювання роботи електричних схем» є цілковито досяжною, оскільки розробник додатку володіє усіма необхідними знаннями для його створення.
Relevant (реалістична)	Для реалізації web-додатку, який є продуктом проекту є всі необхідні технічні тапрограмні засоби.
Time-framed (обмежена в часі)	Web-додаток розроблявся з обмеженням у часі на основі термінами виконання дипломного проекту.

А.3 Планування змісту структури ІТ – проекту

Під час планування розробки web-додатку було створена WBS та OBS структури. WBS (Work Break Structure) – метод розбиття великої мети на точні кроки, який дозволяє побудувати шлях до досягнення результату та успішно виконати робочий чи особистий проект [27]. Як правило, на верхньому рівні вказується сам проект, під ним – основні результати, кожен з яких, у свою чергу, деталізується, тобто наступний рівень завжди менший за попередній за обсягом робіт і, як правило, включає 2 і більше пакетів робіт. При цьому в різних гілках WBS може бути різна кількість рівнів, залежно від потрібного ступеня деталізації [28].

WBS-структура представлена на рисунку А.1

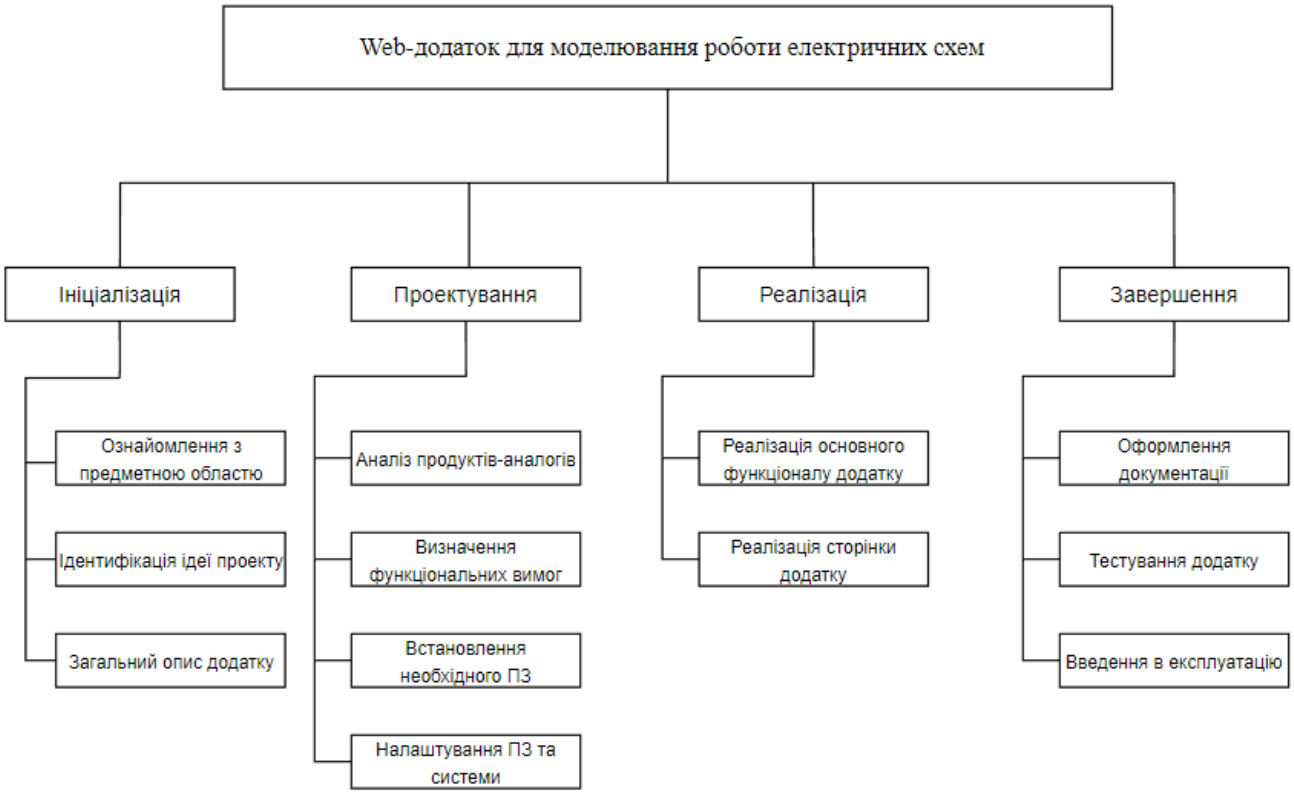

Рисунок А.1 – WBS-структура Web-додатку для моделювання роботи електричних схем

На основі створеної WBS-структури необхідно також створити OBS-структуру. OBS (Organization Breakdown Structure) – ця структура використовується для

відображення того, які організаційні підрозділи виконують ті чи інші роботи [29]. Метою розробки OBS було визначення складу та розподіл обов'язків виконавців робіт, що входять до WBS-структури.

OBS-структура представлена на рисунку А.2

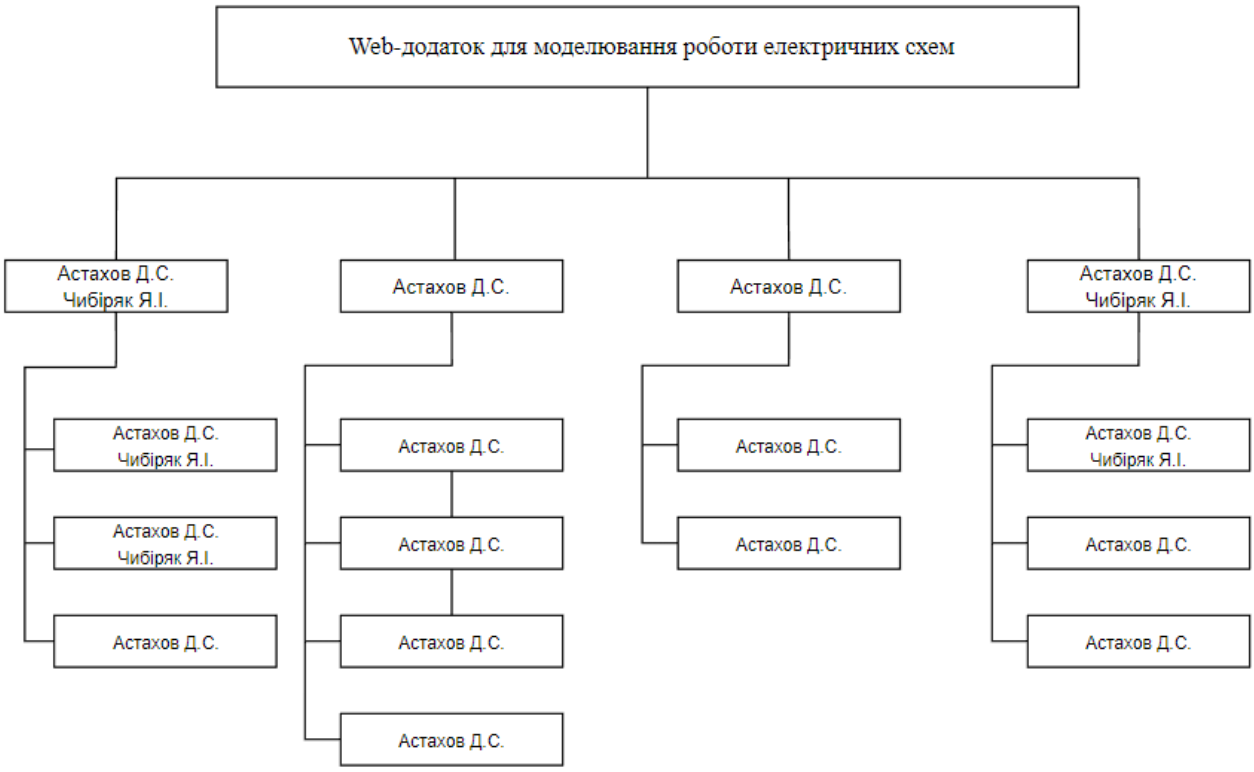

Рисунок А.2 – OBS-структура Web-додатку для моделювання роботи електричних схем

А.4 Побудова календарного графіку виконання ІТ - проекту

Для побудови календарного графіку виконання ІТ – проекту, було використано діаграму Ганта (Gantt Chart) - це інструмент управління проектами, що ілюструє те, як виконується запланована робота з часом. Діаграма Ганта може включати дати початку та завершення завдань, контрольні точки, залежності між завданнями, виконавців та не тільки. [30].

Діаграма Ганта з усіма етапами розробки web-додатку для моделювання роботи електричних схем представлена на рисунку А.3.

Назва задачі	Тривалість	Початок роботи	Кінець роботи
Ініціалізація	10	9 сент. 2022 г.	19 сент. 2022 г.
Ознайомлення з предметною областю	4	9 сент. 2022 г.	13 сент. 2022 г.
Ідентифікація ідеї проекту	2	9 сент. 2022 г.	11 сент. 2022 г.
Загальний опис додатку	4	9 сент. 2022 г.	13 сент. 2022 г.
Проектування	7	20 сент. 2022 г.	27 сент. 2022 г.
Аналіз продуктів-аналогів	1	20 сент. 2022 г.	21 сент. 2022 г.
Визначення функціонувальних вимог	2	20 сент. 2022 г.	22 сент. 2022 г.
Встановлення необхідного ПЗ	2	23 сент. 2022 г.	25 сент. 2022 г.
Налаштування ПЗ та системи	2	25 сент. 2022 г.	27 сент. 2022 г.
Релізація	25	28 сент. 2022 г.	23 окт. 2022 г.
Реалізація основного функціоналу додатку	24	28 сент. 2022 г.	22 окт. 2022 г.
Реалізація сторінки додатку	1	21 окт. 2022 г.	22 окт. 2022 г.
Завершення	15	31 окт. 2022 г.	15 нояб. 2022 г.
Оформлення документації	10	31 окт. 2022 г.	10 нояб. 2022 г.
Тестування додатку	3	10 нояб. 2022 г.	13 нояб. 2022 г.
Введення в експуплуатацію	2	13 нояб. 2022 г.	15 нояб. 2022 г.

Назва проекту	Тривалість роботи	Початок роботи	Кінець роботи
Web-додаток для моделювання роботи електричних схем	**67**	**09.сент.22**	**15.нояб.22**

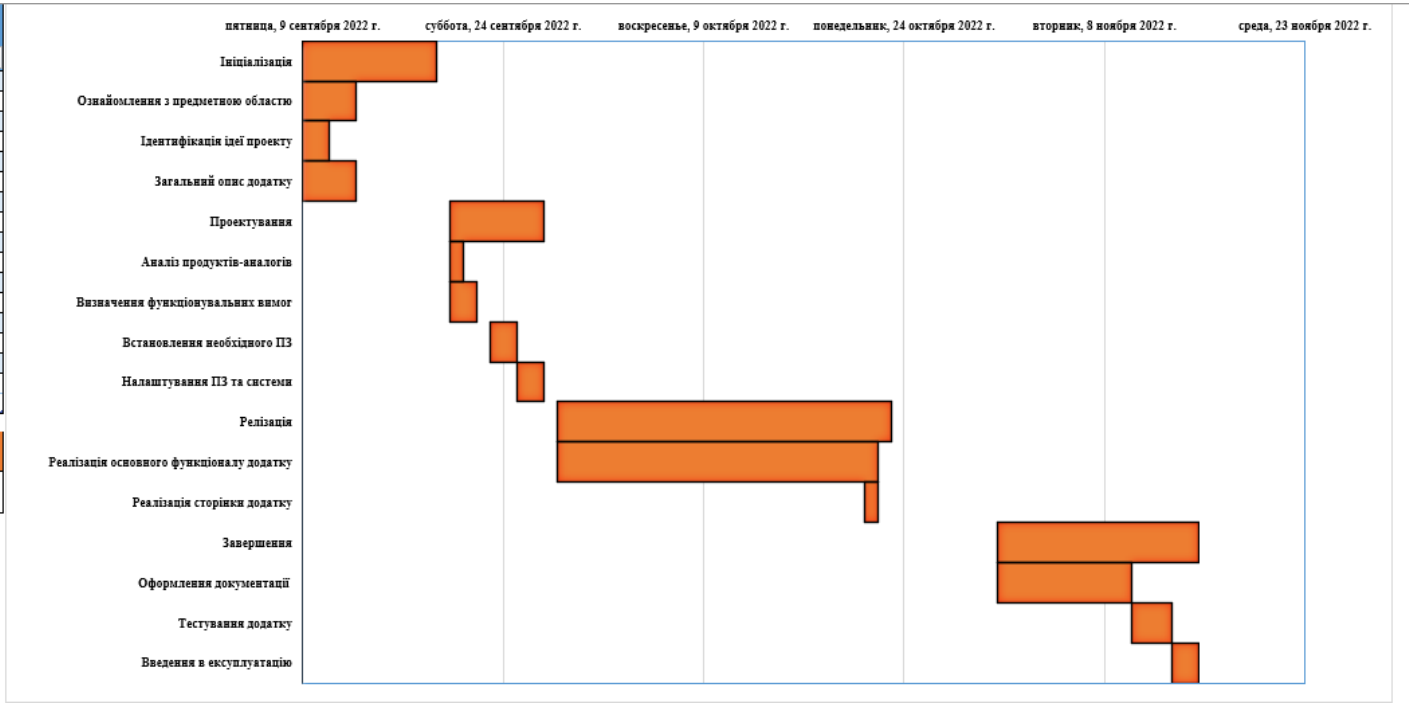

Рисунок А.3 – Діаграма Ганта

А.5 Планування ризиків проекту

Перш ніж почати планувати ризики проекту, потрібно визначитись с терміном «Ризик» - це негативні події, які можуть статися та вплинути на проект. Наприклад, держава випустить новий закон чи розробник тимчасово не зможе працювати над проектом [31].

Більшість частину ризиків при плануванні проектів можливо передбачити. Для даного проекту було виділено наступні ризики:

R1 – Неточність технічного завдання (ТЗ);

R2 – Зміна ТЗ в ході розробки додатку;

R3 – Відставання від календарного плану;

R4 – Зміна дат виконання роботи;

R5 – Збої системи та програмного забезпечення (ПЗ);

R6 – Низька якість тестування;

R7 – Людський фактор;

R8 – Зміна вимог.

Таблиця А.2 – Класифікація ризиків

№	Назва ризику	Ймовірність	Величина втрат
1	Неточність ТЗ	2	4
2	Зміна ТЗ в ході розробки додатку	3	4
3	Відставання від календарного плану	4	3
4	Зміна дат виконання роботи	2	5
5	Збої системи та ПЗ	1	4
6	Низька якість тестування	2	4
7	Людський фактор	4	4
8	Зміна вимог	1	5

На основі таблиці А.2, було побудовано матрицю ризиків, яка представлена в таблиці А.3.

Таблиця А.3 – Матриця ризиків

Ймовірність						
	5					
	4			3	7	
	3			7	2	
	2				1; 6	4
	1				5	8
		1	2	3	4	5
Величина втрат						

На основі створеної матриці ризиків було виконано оцінку рівню ризика для кожного із ризиків в проекті. Результат оцінки рівню ризика представлено в таблиці А.4.

Таблиця А.4 – Оцінка рівню ризиків

№	Назва ризику	Ймовірність	Величина втрат	Рівень ризику
1	Неточність ТЗ	2	4	Середній
2	Зміна ТЗ в ході розробки додатку	2	3	Середній
3	Відставання від календарного плану	4	3	Середній
4	Зміна дат виконання роботи	2	5	Середній
5	Збої системи та ПЗ	2	4	Середній
6	Низька якість тестування	2	4	Середній
7	Людський фактор	3	3	Високий
8	Зміна вимог	1	5	Середній

ДОДАТОК Б

Лістинг файлу CurrentComponent.java

```
import java.awt.Graphics;
import java.awt.Point;
import java.awt.Polygon;
import java.util.StringTokenizer;

public class CurrentComponent extends CircuitComponent {
   double currentValue;
   double value = 0.1D;
   Polygon polygon;
   Point point1;
   Point point2;
   Point centerPoint;

   public CurrentComponent(int x, int y) {
      super(x, y);
      this.currentValue = this.value;
   }

   public CurrentComponent(int a, int b, int c, int d, int e, StringTokenizer tokenizer) {
      super(a, b, c, d, e);

      try {
         this.currentValue = Double.parseDouble(tokenizer.nextToken());
      } catch (Exception var8) {
         this.currentValue = this.value;
      }

   }

   String orgs() {
      String var10000 = super.orgs();
      return var10000 + " " + this.currentValue;
   }

   public int getOrgType() {
      return 105;
   }

   void setPointsForNods() {
      super.setPointsForNods();
      this.calculatePotentials(26);
      this.point1 = this.intermediatePoint(this.pointsComponents, this.pointsComponents2, 0.25D);
      this.polygon = this.calculateArrow(this.centerPoint, intermediatePoint, 4.0D, 4.0D);
   }

   void draw(Graphics graphics) {
      int i = 12;
      this.draw2Leads(graphics);
      this.setCurrentVoltageColor(graphics, (this.currentValueVolts[0] + this.currentValueVolts[1]) / 2.0D);
      this.setPowerColor(graphics, false);
      this.setBoxCircuit(this.firstPoint, this.secondPoint, (double)i);
      this.doDots(graphics);
      if (cirSimMain.showValuesCheckItem.getState()) {
         String shortUnitText = getShortUnitText(this.currentValue, "A");
         if (this.pointsDx == 0 || this.pointsDy == 0) {
            this.drawOurValues(graphics, shortUnitText, (double)i);
```

```
            }
        }

        this.drawPosts(graphics);
    }

    void label() {
        this.currentPowerOfThisComponent = this.currentValue;
        cirSimMain.stampCurrentSource(this.nodes[0], this.nodes[1], this.currentPowerOfThisComponent);
    }

    public EditInfoComponent getEditInfoAboutComponent(int i) {
        return i == 0 ? new EditInfoComponent("Current (A)", this.currentValue, 0.0D, 0.1D) : null;
    }

    public void setEditValue(int i, EditInfoComponent editInfo) {
        this.currentValue = editInfo.value;
    }

    void getInfoAboutComponent(String[] arr) {
        arr[0] = "current source";
        this.getBasicInfoAboutComponent(arr);
    }

    double getCurrentVoltageDifference() {
        return this.currentValueVolts[1] - this.currentValueVolts[0];
    }
}
```

Лістинг файлу ChipComponent.java

```
public abstract class ChipComponent extends CircuitComponent {
    public ChipComponent.Pin[] points;
    public int checkX;
    public int checkY;
    public boolean checkLast;
    public final int CONST_1 = 0;
    public final int CONST_2 = 1;
    public final int CONST_3 = 2;
    public final int CONST_4 = 3;

    abstract void setupPoints();

    public ChipComponent(int a, int b) {
        super(a, b);
        if (this.checkCurrentChipParams()) {
            this.points1 = this instanceof DecadeComponent ? 10 : 4;
        }

        this.checkIfOfNoDiagonal = true;
        this.setupPoints();
        this.setSize(cirSimMain.smallGridCheckItem.getState() ? 1 : 2);
    }

    public ChipComponent(int a, int b, int c, int d, int e, StringTokenizer stringTokenizer) {
        super(a, b, c, d, e);
        if (this.checkCurrentChipParams()) {
            this.points1 = Integer.parseInt(stringTokenizer.nextToken());
            this.checkIfOfNoDiagonal = true;
        }
```

```
      this.setupPoints();
      this.setSize((e & 1) != 0 ? 1 : 2);

      for(int i = 0; i != this.getCurrentNumberOfPoints(); ++i) {
         if (this.points[i].state) {
            this.currentValueVolts[i] = Double.parseDouble(stringTokenizer.nextToken());
            this.points[i].value = this.currentValueVolts[i] > 2.5D;
         }
      }

   }

   boolean checkCurrentChipParams() {
      return false;
   }

   public void setSize(int a) {
      this.sizeA = a;
      this.value1A = 8 * a;
      this.value2A = this.value1A * 2;
      this.flags &= -2;
      this.flags |= a == 1 ? 1 : 0;
   }

   void draw(Graphics graphics) {
      this.drawCurrentChip(graphics);
   }

   void drawCurrentChip(Graphics graphics) {
      Font font = new Font("BOLD", 0, 12 * this.sizeA);
      graphics.setFont(font);
      FontMetrics graphicsFontMetrics = graphics.getFontMetrics();

      int i;
      for(i = 0; i != this.getCurrentNumberOfPoints(); ++i) {
         ChipComponent.Pin pin = this.points[i];
         this.setCurrentVoltageColor(graphics, this.currentValueVolts[i]);
         Point pointA = pin.post;
         Point pointB = pin.stub;
         drawThickLine(graphics, pointA, pointB);
         pin.currentCount = this.updateNumberOfPoints(pin.current, pin.currentCount);
         this.drawPoints(graphics, pointB, pointA, pin.currentCount);
         if (pin.bleb) {
            graphics.setColor(cirSimMain.printableCheckItem.getState() ? Color.white : Color.black);
            drawThickCircle(graphics, pin.blebX, pin.blebY, 1);
            graphics.setColor(lightGrayColor);
            drawThickCircle(graphics, pin.blebX, pin.blebY, 3);
         }

         graphics.setColor(whiteColor);
         int stringWidth = graphicsFontMetrics.stringWidth(pin.text);
         graphics.drawString(pin.text, pin.pointText.x - stringWidth / 2, pin.pointText.y +
graphicsFontMetrics.getAscent() / 2);
         if (pin.line) {
            int i1 = pin.pointText.y - graphicsFontMetrics.getAscent() / 2;
            graphics.drawLine(pin.pointText.x - stringWidth / 2, i1, pin.pointText.x + stringWidth / 2, i1);
         }
      }

      graphics.setColor(this.needsHighlight() ? selectColor : lightGrayColor);
```

```
    drawThickPolygon(graphics, this.straightPointsX, this.straightPointsY, 4);
    if (this.pointxX != null) {
        graphics.drawPolyline(this.pointxX, this.pointyY, 3);
    }

    for(i = 0; i != this.getCurrentNumberOfPoints(); ++i) {
        this.drawPost(graphics, this.points[i].post.x, this.points[i].post.y, this.nodes[i]);
    }

}

public void lug(int a, int b) {
    b = cirSimMain.snapGrid(b);
    this.y = this.b2 = b;
    this.a2 = cirSimMain.snapGrid(a);
    this.setPointsForNods();
}

public void setPointsForNods() {
    if (this.a2 - this.x > this.checkX * this.value2A && this == cirSimMain.lugComponent) {
        this.setSize(2);
    }

    int x1 = this.x + this.value2A;
    int y1 = this.y;
    int x2 = x1 - this.value1A;
    int y2 = y1 - this.value1A;
    int x3 = this.checkX * this.value2A;
    int y3 = this.checkY * this.value2A;
    this.straightPointsX = new int[]{x2, x2 + x3, x2 + x3, x2};
    this.straightPointsY = new int[]{y2, y2, y2 + y3, y2 + y3};
    this.setBoxCircuit(x2, y2, this.straightPointsX[2], this.straightPointsY[2]);

    for(int a = 0; a != this.getCurrentNumberOfPoints(); ++a) {
        ChipComponent.Pin pin = this.points[a];
        switch(pin.side) {
        case 0:
            pin.setPoint(x1, y1, 1, 0, 0, -1, 0, 0);
            break;
        case 1:
            pin.setPoint(x1, y1, 1, 0, 0, 1, 0, y3 - this.value2A);
            break;
        case 2:
            pin.setPoint(x1, y1, 0, 1, -1, 0, 0, 0);
            break;
        case 3:
            pin.setPoint(x1, y1, 0, 1, 1, 0, x3 - this.value2A, 0);
        }
    }

}

public Point getPost(int a) {
    return this.points[a].post;
}

abstract int getNumberOfVoltageSource();

public void setCurrentVoltageSource(int a, int b) {
    for(int h = 0; h != this.getCurrentNumberOfPoints(); ++h) {
        ChipComponent.Pin pin = this.points[h];
```

```
            if (pin.output && a-- == 0) {
                pin.voltageSource = b;
                return;
            }
        }

        System.out.println("Set Voltage Source failed for " + this);
    }

    void label() {
        for(int a = 0; a != this.getCurrentNumberOfPoints(); ++a) {
            ChipComponent.Pin pin = this.points[a];
            if (pin.output) {
                cirSimMain.setCurrentVoltageSource(0, this.nodes[a], pin.voltageSource);
            }
        }

    }

    void execute() {
    }

    void doStep() {
        int a;
        ChipComponent.Pin pin;
        for(a = 0; a != this.getCurrentNumberOfPoints(); ++a) {
            pin = this.points[a];
            if (!pin.output) {
                pin.value = this.currentValueVolts[a] > 2.5D;
            }
        }

        this.execute();

        for(a = 0; a != this.getCurrentNumberOfPoints(); ++a) {
            pin = this.points[a];
            if (pin.output) {
                cirSimMain.updateCurrentVoltageSource(pin.voltageSource, pin.value ? 5.0D : 0.0D);
            }
        }

    }

    void reload() {
        for(int a = 0; a != this.getCurrentNumberOfPoints(); ++a) {
            this.points[a].value = false;
            this.points[a].currentCount = 0.0D;
            this.currentValueVolts[a] = 0.0D;
        }

        this.checkLast = false;
    }

    String orgs() {
        StringBuilder s = new StringBuilder(super.orgs());
        if (this.checkCurrentChipParams()) {
            s.append(" ").append(this.points1);
        }

        for(int i = 0; i != this.getCurrentNumberOfPoints(); ++i) {
            if (this.points[i].state) {
```

```
            s.append(" ").append(this.currentValueVolts[i]);
        }
    }

    return s.toString();
}

void getInfoAboutComponent(String[] arr) {
    arr[0] = this.getCurrentChipName();
    int a = 1;

    for(int i = 0; i != this.getCurrentNumberOfPoints(); ++i) {
        ChipComponent.Pin p = this.points[i];
        if (arr[a] != null) {
            arr[a] = arr[a] + "; ";
        } else {
            arr[a] = "";
        }

        String t = p.text;
        if (p.line) {
            t = t + "'";
        }

        if (p.clock) {
            t = "Clk";
        }

        arr[a] = arr[a] + t + " = " + getCurrentVoltageText(this.currentValueVolts[i]);
        if (i % 2 == 1) {
            ++a;
        }
    }

}

public void setCurrent(int a, double b) {
    for(int i = 0; i != this.getCurrentNumberOfPoints(); ++i) {
        if (this.points[i].output && this.points[i].voltageSource == a) {
            this.points[i].current = b;
        }
    }

}

String getCurrentChipName() {
    return "chip";
}

public boolean getConnection(int a1, int a2) {
    return false;
}

public boolean hasCurrentGroundConnection(int a1) {
    return this.points[a1].output;
}

public EditInfoComponent getEditInfoAboutComponent(int i) {
    EditInfoComponent editInfo;
    if (i == 0) {
        editInfo = new EditInfoComponent("", 0.0D, -1.0D, -1.0D);
```

```
        editInfo.checkbox = new Checkbox("Flip X", (this.flags & 1024) != 0);
        return editInfo;
    } else if (i == 1) {
        editInfo = new EditInfoComponent("", 0.0D, -1.0D, -1.0D);
        editInfo.checkbox = new Checkbox("Flip Y", (this.flags & 2048) != 0);
        return editInfo;
    } else {
        return null;
    }
}

public void setEditValue(int i, EditInfoComponent editInfo) {
    if (i == 0) {
        if (editInfo.checkbox.getState()) {
            this.flags |= 1024;
        } else {
            this.flags &= -1025;
        }

        this.setPointsForNods();
    }

    if (i == 1) {
        if (editInfo.checkbox.getState()) {
            this.flags |= 2048;
        } else {
            this.flags &= -2049;
        }

        this.setPointsForNods();
    }

}

class Pin {
    private Point post;
    private Point stub;
    private Point pointText;
    public int pos;
    public int side;
    public int voltageSource;
    public int blebX;
    public int blebY;
    private final String text;
    public boolean line;
    public boolean bleb;
    public boolean clock;
    public boolean output;
    public boolean value;
    public boolean state;
    public double currentCount;
    public double current;

    public Pin(int a, int b, String str) {
        this.pos = a;
        this.side = b;
        this.text = str;
    }

    private void setPoint(int px, int py, int dx, int dy, int dax, int day, int sx, int sy) {
        if ((ChipComponent.this.flags & 1024) != 0) {
```

```
            dx = -dx;
            dax = -dax;
            px += ChipComponent.this.value2A * (ChipComponent.this.checkX - 1);
            sx = -sx;
         }

         if ((ChipComponent.this.flags & 2048) != 0) {
            dy = -dy;
            day = -day;
            py += ChipComponent.this.value2A * (ChipComponent.this.checkY - 1);
            sy = -sy;
         }

         int xa = px + ChipComponent.this.value2A * dx * this.pos + sx;
         int ya = py + ChipComponent.this.value2A * dy * this.pos + sy;
         this.post = new Point(xa + dax * ChipComponent.this.value2A, ya + day * ChipComponent.this.value2A);
         this.stub = new Point(xa + dax * ChipComponent.this.value1A, ya + day * ChipComponent.this.value1A);
         this.pointText = new Point(xa, ya);
         if (this.bleb) {
            this.blebX = xa + dax * 10 * ChipComponent.this.sizeA;
            this.blebY = ya + day * 10 * ChipComponent.this.sizeA;
         }

         if (this.clock) {
            ChipComponent.this.pointxX = new int[3];
            ChipComponent.this.pointyY = new int[3];
            ChipComponent.this.pointxX[0] = xa + dax * ChipComponent.this.value1A - dx *
ChipComponent.this.value1A / 2;
            ChipComponent.this.pointyY[0] = ya + day * ChipComponent.this.value1A - dy *
ChipComponent.this.value1A / 2;
            ChipComponent.this.pointxX[1] = xa;
            ChipComponent.this.pointyY[1] = ya;
            ChipComponent.this.pointxX[2] = xa + dax * ChipComponent.this.value1A + dx *
ChipComponent.this.value1A / 2;
            ChipComponent.this.pointyY[2] = ya + day * ChipComponent.this.value1A + dy *
ChipComponent.this.value1A / 2;
         }

      }
   }
}
```

Лістинг файлу Diode.java

```
public class Diode {
   int[] nodes;
   Main sim;
   public double leakage = 1.0E-14D;
   double valueT;
   double valueD;
   double fDrop;
   double voltageZ;
   double offsetZ;
   double lastVoltageDiff;
   double voltageCrit;

   public Diode(Main main) {
      this.sim = main;
      this.nodes = new int[2];
   }
```

```
    void setup(double v, double v1) {
        this.fDrop = v;
        this.voltageZ = v1;
        this.valueD = Math.log(1.0D / this.leakage + 1.0D) / this.fDrop;
        this.valueT = 1.0D / this.valueD;
        this.voltageCrit = this.valueT * Math.log(this.valueT / (Math.sqrt(2.0D) * this.leakage));
        if (this.voltageZ == 0.0D) {
            this.offsetZ = 0.0D;
        } else {
            double i = -0.005D;
            this.offsetZ = this.voltageZ - Math.log(-(1.0D + i / this.leakage)) / this.valueD;
        }

    }

    void reset() {
        this.lastVoltageDiff = 0.0D;
    }

    double limitStep(double valueNew, double voltageD) {
        double arg;
        double v0;
if (valueNew < 0.0D && this.offsetZ != 0.0D) {
            valueNew = -valueNew - this.offsetZ;
            voltageD = -voltageD - this.offsetZ;
            if (valueNew > this.voltageCrit && Math.abs(valueNew - voltageD) > this.valueT + this.valueT) {
                if (voltageD > 0.0D) {
                    arg = 1.0D + (valueNew - voltageD) / this.valueT;
                    if (arg > 0.0D) {
                        valueNew = voltageD + this.valueT * Math.log(arg);
                        v0 = Math.log(1.0E-6D / this.leakage) * this.valueT;
                        valueNew = Math.max(v0, valueNew);
                    } else {
                        valueNew = this.voltageCrit;
                    }
                } else {
                    valueNew = this.valueT * Math.log(valueNew / this.valueT);
                }

                this.sim.converged = false;
            }

            valueNew = -(valueNew + this.offsetZ);
        }

        return valueNew;
    }

    void stamp(int n0, int n1) {
        this.nodes[0] = n0;
        this.nodes[1] = n1;
        this.sim.stampNonCapital(this.nodes[0]);
        this.sim.stampNonCapital(this.nodes[1]);
    }

    void doStep(double voltageDiff) {
        if (Math.abs(voltageDiff - this.lastVoltageDiff) > 0.01D) {
            this.sim.converged = false;
        }
```

```
        voltageDiff = this.limitStep(voltageDiff, this.lastVoltageDiff);
        this.lastVoltageDiff = voltageDiff;
        double eval;
        double geq;
        if (!(voltageDiff >= 0.0D) && this.voltageZ != 0.0D) {
            eval = this.leakage * this.valueD * (Math.exp(voltageDiff * this.valueD) + Math.exp((-voltageDiff -
this.offsetZ) * this.valueD));
            geq = this.leakage * (Math.exp(voltageDiff * this.valueD) - Math.exp((-voltageDiff - this.offsetZ) *
this.valueD) - 1.0D) + eval * -voltageDiff;
            this.sim.stampConductance(this.nodes[0], this.nodes[1], eval);
            this.sim.stampCurrentSource(this.nodes[0], this.nodes[1], geq);
        } else {
            eval = Math.exp(voltageDiff * this.valueD);
            if (voltageDiff < 0.0D) {
                eval = 1.0D;
            }

            geq = this.valueD * this.leakage * eval;
            double nc = (eval - 1.0D) * this.leakage - geq * voltageDiff;
            this.sim.stampConductance(this.nodes[0], this.nodes[1], geq);
            this.sim.stampCurrentSource(this.nodes[0], this.nodes[1], nc);
        }

    }

    double calculateCurrentVoltage(double voltageDiff) {
        return !(voltageDiff >= 0.0D) && this.voltageZ != 0.0D ? this.leakage * (Math.exp(voltageDiff * this.valueD) -
Math.exp((-voltageDiff - this.offsetZ) * this.valueD) - 1.0D) : this.leakage * (Math.exp(voltageDiff * this.valueD) -
1.0D);
    }
}
```

Лістинг файлу AnalogSwitchSecondTypeComponent.java

```
public class AnalogSwitchSecondTypeComponent extends AnalogSwitchComponent {
    private final int anInt = 16;
    private Point[] switchPosts;
    private Point[] switchPoles;
    private Point switchPoint;

    public AnalogSwitchSecondTypeComponent(int a, int b) {
        super(a, b);
    }

    public AnalogSwitchSecondTypeComponent(int a, int b, int c, int d, int e, StringTokenizer stringTokenizer) {
        super(a, b, c, d, e, stringTokenizer);
    }

    public void setPointsForNods() {
        super.setPointsForNods();
        this.calculatePotentials(32);
        this.switchPosts = this.newPointArray(2);
        this.switchPoles = this.newPointArray(2);
        this.intermediatePoint2(this.pointsComponents, this.pointsComponents2, this.switchPoles[0],
this.switchPoles[1], 1.0D, 16.0D);
        this.intermediatePoint2(this.firstPoint, this.secondPoint, this.switchPosts[0], this.switchPosts[1], 1.0D, 16.0D);
        this.switchPoint = this.intermediatePoint(this.firstPoint, this.secondPoint, 0.5D, 16.0D);
    }

    public int getCurrentNumberOfPoints() {
```

```
    return 4;
}

public void draw(Graphics graphics) {
    this.drawSwitch(graphics);
    graphics.setColor(lightGrayColor);
    int position = this.open ? 1 : 0;
    drawThickLine(graphics, this.pointsComponents, this.switchPoles[position]);
    this.updateNumberOfPoints();
    this.drawPoints(graphics, this.firstPoint, this.pointsComponents, this.currentCount);
    this.drawPoints(graphics, this.switchPoles[position], this.switchPosts[position], this.currentCount);
    this.drawPosts(graphics);
}

public Point getPost(int a) {
    return a == 0 ? this.firstPoint : (a == 3 ? this.switchPoint : this.switchPosts[a - 1]);
}

public int getOrgType() {
    return 160;
}

public void calculateCurrent() {
    if (this.open) {
        this.currentPowerOfThisComponent = (this.currentValueVolts[0] - this.currentValueVolts[2]) / this.r_on;
    } else {
        this.currentPowerOfThisComponent = (this.currentValueVolts[0] - this.currentValueVolts[1]) / this.r_on;
    }

}

public void label() {
    cirSimMain.stampNonCapital(this.nodes[0]);
    cirSimMain.stampNonCapital(this.nodes[1]);
    cirSimMain.stampNonCapital(this.nodes[2]);
}

public void doStep() {
    this.open = this.currentValueVolts[3] < 2.5D;
    if ((this.flags & 1) != 0) {
        this.open = !this.open;
    } else if (this.open) {
        cirSimMain.stampResistor(this.nodes[0], this.nodes[2], this.r_on);
        cirSimMain.stampResistor(this.nodes[0], this.nodes[1], this.r_off);
    } else {
        cirSimMain.stampResistor(this.nodes[0], this.nodes[1], this.r_on);
        cirSimMain.stampResistor(this.nodes[0], this.nodes[2], this.r_off);
    }

}

public boolean getConnection(int a1, int a2) {
    return a1 != 3 && a2 != 3;
}

public void getInfoAboutComponent(String[] arr) {
    arr[0] = "Analog Switch (SPDT)";
    arr[1] = "I = " + getCurrentText1(this.getCurrent());
}

private void drawSwitch(Graphics graphics) {
```

```
        this.setBoxCircuit(this.firstPoint, this.secondPoint, 16.0D);
        this.setCurrentVoltageColor(graphics, this.currentValueVolts[0]);
        drawThickLine(graphics, this.firstPoint, this.pointsComponents);
        this.setCurrentVoltageColor(graphics, this.currentValueVolts[1]);
        drawThickLine(graphics, this.switchPoles[0], this.switchPosts[0]);
        this.setCurrentVoltageColor(graphics, this.currentValueVolts[2]);
        drawThickLine(graphics, this.switchPoles[1], this.switchPosts[1]);
    }
}
```

Лістинг файлу DiacComponent.java

```
public class DiacComponent extends CircuitComponent {
    double checkResistanceOn;
    double checkResistanceOff;
    double breakDown;
    double holdCurrent;
    boolean state;
    Point point1;
    Point point2;

    public DiacComponent(int a, int b) {
        super(a, b);
        this.checkResistanceOff = 1.0E9D;
        this.checkResistanceOn = 1000.0D;
        this.breakDown = 1000.0D;
        this.holdCurrent = 0.001D;
        this.state = false;
    }

    public DiacComponent(int a, int b, int xb, int yb, int f, StringTokenizer tokenizer) {
        super(a, b, xb, yb, f);
        this.checkResistanceOn = Double.parseDouble(tokenizer.nextToken());
        this.checkResistanceOff = Double.parseDouble(tokenizer.nextToken());
        this.breakDown = Double.parseDouble(tokenizer.nextToken());
        this.holdCurrent = Double.parseDouble(tokenizer.nextToken());
    }

    boolean ifFlagCapital() {
        return true;
    }

    public int getOrgType() {
        return 203;
    }

    String orgs() {
        String var10000 = super.orgs();
        return var10000 + " " + this.checkResistanceOn + " " + this.checkResistanceOff + " " + this.breakDown + " " +
this.holdCurrent;
    }

    void setPointsForNods() {
        super.setPointsForNods();
        this.calculatePotentials(32);
        this.point1 = new Point();
        this.point2 = new Point();
    }

    void draw(Graphics graphics) {
```

```
    this.setBoxCircuit(this.firstPoint, this.secondPoint, 6.0D);
    this.draw2Leads(graphics);
    this.setPowerColor(graphics, true);
    this.doDots(graphics);
    this.drawPosts(graphics);
}

void calculateCurrent() {
    double value = this.currentValueVolts[0] - this.currentValueVolts[1];
    if (this.state) {
        this.currentPowerOfThisComponent = value / this.checkResistanceOn;
    } else {
        this.currentPowerOfThisComponent = value / this.checkResistanceOff;
    }

}

void startRelapse() {
    double value = this.currentValueVolts[0] - this.currentValueVolts[1];
    if (Math.abs(this.currentPowerOfThisComponent) < this.holdCurrent) {
        this.state = false;
    }

    if (Math.abs(value) > this.breakDown) {
        this.state = true;
    }

}

void doStep() {
    if (this.state) {
        cirSimMain.stampResistor(this.nodes[0], this.nodes[1], this.checkResistanceOn);
    } else {
        cirSimMain.stampResistor(this.nodes[0], this.nodes[1], this.checkResistanceOff);
    }

}

void label() {
    cirSimMain.stampNonCapital(this.nodes[0]);
    cirSimMain.stampNonCapital(this.nodes[1]);
}

void getInfoAboutComponent(String[] arr) {
    arr[0] = "spark gap";
    this.getBasicInfoAboutComponent(arr);
    arr[3] = this.state ? "on" : "off";
    arr[4] = "Ron = " + getUnitText(this.checkResistanceOn, Main.ohmString);
    arr[5] = "Roff = " + getUnitText(this.checkResistanceOff, Main.ohmString);
    arr[6] = "Vbrkdn = " + getUnitText(this.breakDown, "V");
    arr[7] = "Ihold = " + getUnitText(this.holdCurrent, "A");
}

public EditInfoComponent getEditInfoAboutComponent(int i) {
    if (i == 0) {
        return new EditInfoComponent("On resistance (ohms)", this.checkResistanceOn, 0.0D, 0.0D);
    } else if (i == 1) {
        return new EditInfoComponent("Off resistance (ohms)", this.checkResistanceOff, 0.0D, 0.0D);
    } else if (i == 2) {
        return new EditInfoComponent("Breakdown voltage (volts)", this.breakDown, 0.0D, 0.0D);
    } else {
```

```
            return i == 3 ? new EditInfoComponent("Hold current (amps)", this.holdCurrent, 0.0D, 0.0D) : null;
        }
    }

    public void setEditValue(int i, EditInfoComponent editInfo) {
        if (editInfo.value > 0.0D && i == 0) {
            this.checkResistanceOn = editInfo.value;
        }

        if (editInfo.value > 0.0D && i == 1) {
            this.checkResistanceOff = editInfo.value;
        }

        if (editInfo.value > 0.0D && i == 2) {
            this.breakDown = editInfo.value;
        }

        if (editInfo.value > 0.0D && i == 3) {
            this.holdCurrent = editInfo.value;
        }

    }
}
```

Лістинг файлу CircuitLayout.java

```
public class CircuitLayout implements LayoutManager {
    public CircuitLayout() {
    }

    public void addLayoutComponent(String name, Component component) {
    }

    public void removeLayoutComponent(Component c) {
    }

    public Dimension preferredLayoutSize(Container target) {
        return new Dimension(1700, 1300);
    }

    public Dimension minimumLayoutSize(Container target) {
        return new Dimension(200, 200);
    }

    public void layoutContainer(Container container) {
        Insets containerInsets = container.getInsets();
        int target1 = container.getSize().width - containerInsets.left - containerInsets.right;
        int i1 = target1 * 8 / 10;
        int var10001 = containerInsets.top + containerInsets.bottom;
        int target2 = container.getSize().height - var10001;
        container.getComponent(0).setLocation(containerInsets.left, containerInsets.top);
        container.getComponent(0).setSize(i1, target2);
        int widthOfTheLine = target1 - i1;
        i1 += containerInsets.left;
        int h = containerInsets.top;

        for(int i = 1; i < container.getComponentCount(); ++i) {
            Component containerComponent = container.getComponent(i);
            if (containerComponent.isVisible()) {
                Dimension d = containerComponent.getPreferredSize();
```

```
            if (containerComponent instanceof Scrollbar) {
               d.width = widthOfTheLine;
            }

            if (containerComponent instanceof Choice && d.width > widthOfTheLine) {
               d.width = widthOfTheLine;
            }

            if (containerComponent instanceof Label) {
               h += d.height / 5;
               d.width = widthOfTheLine;
            }

            containerComponent.setLocation(i1, h);
            containerComponent.setSize(d.width, d.height);
            h += d.height;
         }
      }

   }
}
```

Лістинг файлу FMComponent.java

```
public class FMComponent extends CircuitComponent {
   static final int FLAG_COS = 2;
   double carrierFrequency;
   double signalFrequency;
   double maxVoltage;
   double frequencyTimeZero;
   double deviation;
   double lastTime = 0.0D;
   double functionX = 0.0D;
   final int circleSize = 17;

   public FMComponent(int a, int b) {
      super(a, b);
      this.deviation = 200.0D;
      this.maxVoltage = 5.0D;
      this.carrierFrequency = 800.0D;
      this.signalFrequency = 40.0D;
      this.reload();
   }

   public FMComponent(int a, int b, int c, int d, int f, StringTokenizer tokenizer) {
      super(a, b, c, d, f);
      this.carrierFrequency = Double.parseDouble(tokenizer.nextToken());
      this.signalFrequency = Double.parseDouble(tokenizer.nextToken());
      this.maxVoltage = Double.parseDouble(tokenizer.nextToken());
      this.deviation = Double.parseDouble(tokenizer.nextToken());
      if ((this.flags & 2) != 0) {
         this.flags &= -3;
      }

      this.reload();
   }

   public int getOrgType() {
      return 201;
   }
```

```
    public String orgs() {
        String var10000 = super.orgs();
        return var10000 + " " + this.carrierFrequency + " " + this.signalFrequency + " " + this.maxVoltage + " " +
this.deviation;
    }

    public void reload() {
        this.frequencyTimeZero = 0.0D;
        this.currentCount = 0.0D;
    }

    int getCurrentNumberOfPoints() {
        return 1;
    }

    void label() {
        cirSimMain.setCurrentVoltageSource(0, this.nodes[0], this.voltSource);
    }

    void doStep() {
        cirSimMain.updateCurrentVoltageSource(this.voltSource, this.getVoltage());
    }

    double getVoltage() {
        double deltaT = cirSimMain.t - this.lastTime;
        this.lastTime = cirSimMain.t;
        double signalAmplitude = Math.sin(6.283185307179586D * (cirSimMain.t - this.frequencyTimeZero) *
this.signalFrequency);
        this.functionX += deltaT * (this.carrierFrequency + signalAmplitude * this.deviation);
        double value = 6.283185307179586D * this.functionX;
        return Math.sin(value) * this.maxVoltage;
    }

    public void draw(Graphics graphics) {
        this.setBoxCircuit(this.firstPoint, this.secondPoint, 17.0D);
        this.setCurrentVoltageColor(graphics, this.currentValueVolts[0]);
        drawThickLine(graphics, this.firstPoint, this.pointsComponents);
        this.setFont(graphics);
        String s = "FM";
        this.drawTextInCenter(graphics, s, this.a2, this.b2, true);
        this.drawWaveform(graphics, this.secondPoint);
        this.drawPosts(graphics);
        this.currentCount = this.updateNumberOfPoints(-this.currentPowerOfThisComponent, this.currentCount);
        if (cirSimMain.lugComponent != this) {
            this.drawPoints(graphics, this.firstPoint, this.pointsComponents, this.currentCount);
        }

    }

    void drawWaveform(Graphics graphics, Point center) {
        graphics.setColor(this.needsHighlight() ? selectColor : Color.gray);
        this.setPowerColor(graphics, false);
        int xc = center.x;
        int yc = center.y;
        drawThickCircle(graphics, xc, yc, 17);
        this.setUpBox(xc - 17, yc - 17, xc + 17, yc + 17);
    }

    void setPointsForNods() {
        super.setPointsForNods();
```

```
    this.pointsComponents = this.intermediatePoint(this.firstPoint, this.secondPoint, 1.0D - 17.0D / this.pointsD);
}

double getCurrentVoltageDifference() {
    return this.currentValueVolts[0];
}

boolean hasCurrentGroundConnection(int a1) {
    return true;
}

int getNumberOfVoltageSource() {
    return 1;
}

double getCurrentPower() {
    return -this.getCurrentVoltageDifference() * this.currentPowerOfThisComponent;
}

void getInfoAboutComponent(String[] arr) {
    arr[0] = "FM Source";
    arr[1] = "I = " + getCurrentText(this.getCurrent());
    arr[2] = "V = " + getCurrentVoltageText(this.getCurrentVoltageDifference());
    arr[3] = "cf = " + getUnitText(this.carrierFrequency, "Hz");
    arr[4] = "sf = " + getUnitText(this.signalFrequency, "Hz");
    arr[5] = "dev =" + getUnitText(this.deviation, "Hz");
    arr[6] = "Vmax = " + getCurrentVoltageText(this.maxVoltage);
}

public EditInfoComponent getEditInfoAboutComponent(int i) {
    if (i == 0) {
        return new EditInfoComponent("Max Voltage", this.maxVoltage, -20.0D, 20.0D);
    } else if (i == 1) {
        return new EditInfoComponent("Carrier Frequency (Hz)", this.carrierFrequency, 4.0D, 500.0D);
    } else if (i == 2) {
        return new EditInfoComponent("Signal Frequency (Hz)", this.signalFrequency, 4.0D, 500.0D);
    } else {
        return i == 3 ? new EditInfoComponent("Deviation (Hz)", this.deviation, 4.0D, 500.0D) : null;
    }
}

public void setEditValue(int i, EditInfoComponent editInfo) {
    if (i == 0) {
        this.maxVoltage = editInfo.value;
    }

    if (i == 1) {
        this.carrierFrequency = editInfo.value;
    }

    if (i == 2) {
        this.signalFrequency = editInfo.value;
    }

    if (i == 3) {
        this.deviation = editInfo.value;
    }

}

public void setFont(Graphics graphics) {
```

```
        Font font = new Font("BOLD", 0, 12);
        graphics.setFont(font);
        graphics.setColor(this.needsHighlight() ? selectColor : whiteColor);
        this.setPowerColor(graphics, false);
    }
}
```

Лістинг файлу LogicInputComponent.java

```
public class LogicInputComponent extends SwitchElm {

    public LogicInputComponent(int a, int b) {
        super(a, b, false);
        this.hightV = 5.0D;
        this.lowV = 0.0D;
    }

    public LogicInputComponent(int a, int b, int c, int d, int f, StringTokenizer tokenizer) {
        super(a, b, c, d, f, tokenizer);

        try {
            this.hightV = Double.parseDouble(tokenizer.nextToken());
        } catch (Exception var8) {
            this.hightV = 5.0D;
            this.lowV = 0.0D;
        }

        if (this.ifIsTernary()) {
            this.posCount = 3;
        }

    }

    boolean ifIsTernary() {
        return (this.flags & 1) != 0;
    }

    boolean isNumeric() {
        return (this.flags & 3) != 0;
    }

    public int getOrgType() {
        return 76;
    }

    public String orgs() {
        String var10000 = super.orgs();
    }

    int getCurrentNumberOfPoints() {
        return 1;
    }

    void setPointsForNods() {
        super.setPointsForNods();
    }

    void draw(Graphics graphics) {
        Font font = new Font("BOLD", 0, 20);
        graphics.setFont(font);
```

```
    graphics.setColor(this.needsHighlight() ? selectColor : whiteColor);
    String s = this.position == 0 ? "L" : "H";
    if (this.isNumeric()) {
        s = this.position.makeConcatWithConstants<invokedynamic>(this.position);
    }

    this.drawTextInCenter(graphics, s, this.a2, this.b2, true);
    drawThickLine(graphics, this.firstPoint, this.pointsComponents);
    this.updateNumberOfPoints();
    this.drawPoints(graphics, this.firstPoint, this.pointsComponents, this.currentCount);
    this.drawPosts(graphics);
}

void setCurrent(int vs, double c) {
    this.currentPowerOfThisComponent = -c;
}

void label() {
    double v = this.position == 0 ? this.lowV : this.hightV;
    if (this.ifIsTernary()) {
        v = (double)this.position * 2.5D;
    }

    cirSimMain.setCurrentVoltageSource(0, this.nodes[0], this.voltSource, v);
}

int getNumberOfVoltageSource() {
    return 1;
}

double getCurrentVoltageDifference() {
    return this.currentValueVolts[0];
}

void getInfoAboutComponent(String[] arr) {
    arr[0] = "logic input";
    arr[1] = this.position == 0 ? "low" : "high";
    if (this.isNumeric()) {
        arr[1] = this.position.makeConcatWithConstants<invokedynamic>(this.position);
    }

    arr[1] = arr[1] + " (" + getCurrentVoltageText(this.currentValueVolts[0]) + ")";
    arr[2] = "I = " + getCurrentText(this.getCurrent());
}

boolean hasCurrentGroundConnection(int i) {
    return true;
}

public EditInfoComponent getEditInfoAboutComponent(int i) {
    if (i == 0) {
        EditInfoComponent editInfoComponent = new EditInfoComponent("", 0.0D, 0.0D, 0.0D);
        editInfoComponent.checkbox = new Checkbox("Momentary Switch", this.momentary);
        return editInfoComponent;
    } else if (i == 1) {
        return new EditInfoComponent("High Voltage", this.hightV, 10.0D, -10.0D);
    } else {
        return i == 2 ? new EditInfoComponent("Low Voltage", this.lowV, 10.0D, -10.0D) : null;
    }
}
```

```
    public void setEditValue(int i, EditInfoComponent editInfo) {
        if (i == 0) {
            this.momentary = editInfo.checkbox.getState();
        }

        if (i == 1) {
            this.hightV = editInfo.value;
        }

        if (i == 2) {
            this.lowV = editInfo.value;
        }

    }

    int getLabel() {
        return 105;
    }
}
```