**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY**

**OOP PROJECT REPORT**

**ELECTRICAL CIRCUIT SIMULATOR**

Nguyen Quang Anh 20176684

Bui Huu Thanh Cong 20205176

Nguyen Hoang Minh 20226057

Le Ngoc Viet 20205175

|  |  |
| --- | --- |
| **Instructor:** | PhD. Tran The Hung |
| **Subject:** | OOP-IT3110E |
| **School:** | School of Information and Communication technology |
| **HA NOI, 06/2024** | |

**OVERVIEW**

The objective of this project is to develop a user-friendly electrical circuit simulator that allows users to construct and analyze electrical circuits. This simulator will enable users to visually build circuits using various electrical components and perform analyses such as calculating voltages, currents, and power within the circuit.

Our team (Group 31):

1. Nguyen Quang Anh 20176684: Diagram designer.
2. Bui Huu Thanh Cong 20205176: Presenter.
3. Nguyen Hoang Minh 20226057: Team leader, coding & moderator.
4. Le Ngoc Viet 20205175: Report & slide designer.

**TABLE OF CONTENTS**

[CHAPTER 1: INTRODUCTION 1](#_Toc169964319)

[1.1 Project Overview 1](#_Toc169964320)

[1.2 Objective 1](#_Toc169964321)

[CHAPTER 2: DIAGRAMS 2](#_Toc169964322)

[2.1 Use Case diagram 2](#_Toc169964323)

[2.2 Class diagram 2](#_Toc169964324)

[2.2.1 element package 3](#_Toc169964328)

[2.2.2 complex\_number package: 3](#_Toc169964329)

[2.2.3 voltage\_source package 3](#_Toc169964330)

[2.2.4 circuit package 3](#_Toc169964331)

[2.2.5 simulator package 4](#_Toc169964332)

[CHAPTER 3: IMPLEMENTATION DETAILS 5](#_Toc169964333)

[3.1 Core functionality 5](#_Toc169964335)

[3.2 User interface (UI) 5](#_Toc169964336)

[3.3 Calculation and analysis 5](#_Toc169964337)

[CHAPTER 4: TESTING AND VALIDATION 7](#_Toc169964338)

[4.1 Testing strategies 7](#_Toc169964340)

[4.2 Test cases 7](#_Toc169964341)

[CONCLUSION 8](#_Toc169964342)

[REFERENCES 9](#_Toc169964343)

**LIST OF FIGURES**

[Figure 1: Use Case diagram. 2](#_Toc169963814)

[Figure 2: General class diagram. 3](#_Toc169963815)

[Figure 3: Program's main window. 5](#_Toc169963816)

[Figure 4: Calculation results. 6](#_Toc169963817)

# CHAPTER 1: INTRODUCTION

## Project Overview

The central objective of our project is to develop a robust application capable of simulating a wide range of electrical circuits. This application is designed to empower users to design and construct their own circuit configurations, offering an interactive platform for both learning and experimentation. Users will have the ability to select and place various components, such as resistors, capacitors, and inductors, as well as to choose between AC (alternating current) and DC (direct current) sources to build their circuits.

One of the key features of the application is its capability to compute and display essential electrical parameters, including voltage (U), current (I), and resistance (R) for each component within the circuit. This functionality is crucial for users to understand the behaviour and characteristics of their designed circuits. Additionally, the simulator provides a visual representation of the circuit configuration, helping users to better grasp the spatial and functional relationships between different components.

By supporting both AC and DC circuits, the simulator offers a comprehensive and versatile tool that can be used for a variety of educational and practical purposes. In educational settings, it serves as an effective teaching aid, allowing students to visualize and experiment with circuit concepts in a controlled environment. For practical applications, the simulator can assist hobbyists, engineers, and researchers in prototyping and analysing circuit designs before physical implementation.

Overall, this application aims to bridge the gap between theoretical knowledge and practical application, enhancing the learning experience and providing valuable insights into electrical circuit design and analysis.

## Objective

• Provide a user-friendly interface for designing electrical circuits.

• Enable real-time calculations of circuit parameters.

• Support both series and parallel circuit configurations.

• Visualize the designed circuits and display calculated values.

• Educate users on the principles of circuit design and analysis.

# CHAPTER 2: DIAGRAMS

## 2.1 Use Case diagram

* Below is the Use Case diagram of our topic:

A diagram of a company

Description automatically generated

Figure 1: Use Case diagram.

* Explanation:
  + **Actor**: User is the actor.
  + **Choose Circuit Type**: Users select the type of circuit they want to create (Serial or Parallel).
  + **Add Circuit Element**: Users can add different elements to the circuit, such as resistors, inductors, and capacitors.
  + **Delete Circuit Element**: Users can remove an element from the circuit.
  + **Specify Value of Element**: Users specify the values for each added element.
  + **Choose Type of Source**: Users can select the type of power source (AC or DC).
  + **Specify Value of Source**: Users can define the value for the selected source.
  + **Submit**: Submit the circuit design for analysis.
  + **Reset**: Reset the current circuit design.

## 2.2 Class diagram

* Below is the general class diagram for this Java project:

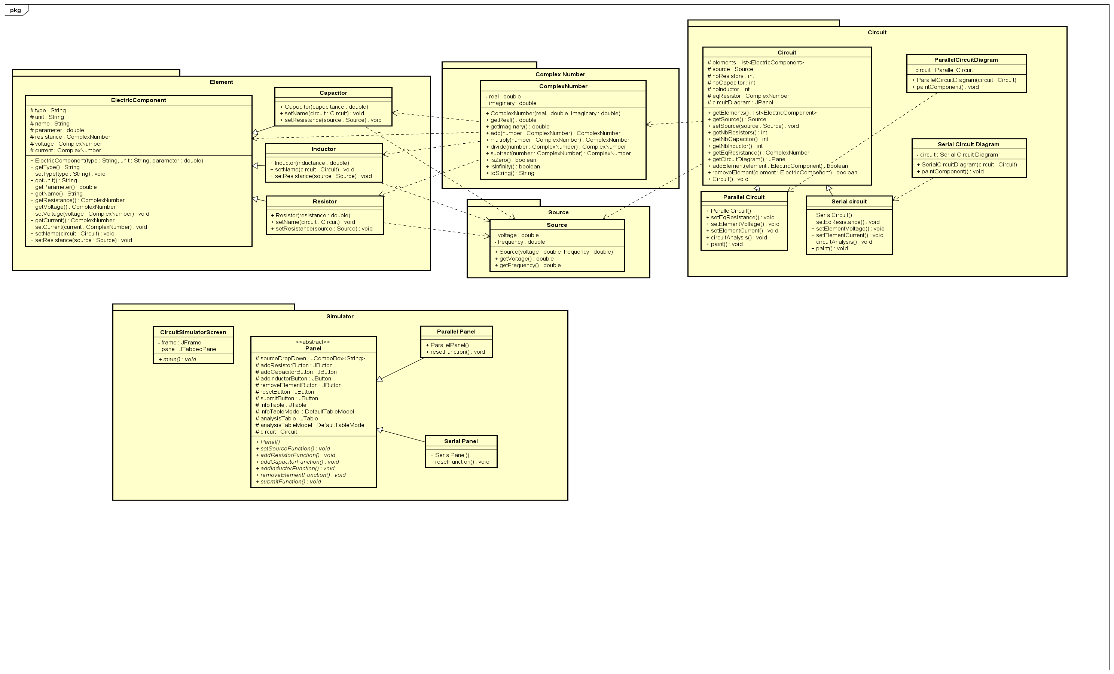


Figure 2: General class diagram.



### element package

* This package includes the core components of the circuit:
  + **ElectricComponent**: An abstract class representing generic electrical components with common attributes like type, name, parameter, resistance, voltage, and current. It also includes methods for setting these attributes.
  + **Resistor, Capacitor, Inductor**: Subclasses of ElectricComponent class, each with specific methods to set their values and calculate their respective parameters within the circuit.

### complex\_number package:

* This package includes:
  + **ComplexNumber**: A class to handle complex number operations necessary for AC circuit calculations, including methods for addition, subtraction, multiplication, and division of complex numbers. Also, it checks if a complex number is zero or infinity and displays it under a string form.

### voltage\_source package

* This package includes:
  + **Source**: A class representing the power source for the circuit, with attributes for voltage and frequency. It includes getters/setters methods.

### circuit package

* This package manages the overall circuit, containing:
  + **Circuit**: An abstract class that includes methods to manage the circuit components and the power source, such as adding and removing components, and calculating the overall circuit parameters.
  + **SerialCircuit & ParallelCircui**t: Subclasses of Circuit class to handle specific calculations and configurations for serial and parallel circuits.
  + **SerialCircuitDiagram** & **ParallelCircuitDiagram:** responsible for drawing the circuit.

### simulator package

* This package handles the user interactions and the graphical user interface, including:
  + **CircuitSimulatorScreen**: Class to handle the main interface, allowing users to interact with the circuit, add or remove components, set their values, and display the circuit.
  + **Panel**: Abstract class representing a panel in the interface, with subclasses for specific panel types.
  + **ParallelPanel, SerialPanel**: Subclasses of Panel to manage user inputs for parallel and serial circuits respectively.

# CHAPTER 3: IMPLEMENTATION DETAILS



## Core functionality

* The core functionality of the application revolves around user interaction for creating and managing circuit designs. The application validates user input, performs necessary calculations, and provides visual feedback. Key features include:
  + **Input Validation**: Ensures users provide complete and correct information before simulation.
  + **Time Calculations**: Computes voltage, current, and resistance as users modify the circuit.
  + **Visualization**: Displays a graphical representation of the circuit.

## User interface (UI)

* The user interface is designed to be intuitive and easy to navigate, allowing users to add, modify, and submit circuit components with ease. Visual representations of circuits are generated to help users understand their designs better. Key UI components:
  + **Circuit Diagram**: Shows the layout of the circuit.
  + **Component List**: Displays all added components and their properties.
  + **Control Buttons**: Allows users to manage the circuit design process.

A screenshot of a computer

Description automatically generated

Figure 3: Program's main window.

## Calculation and analysis

* The application performs real-time calculations to determine voltage, current, and resistance for the user-designed circuits. Both AC and DC circuits are supported, with special consideration for complex impedance calculations in AC circuits. Key calculation methods:
  + **Impedance Calculation**: Computes impedance for resistors, inductors, and capacitors.
  + **Voltage and Current Calculation**: Uses Ohm's Law to determine values.
  + **Short Circuit Detection**: Identifies and notifies users of potential short circuits.

A screenshot of a computer program

Description automatically generated

Figure 4: Calculation results.

# CHAPTER 4: OOP ANALYSIS

## Encapsulation

* We hide the implementation details of a class by:
  + Declaring the attributes of a class as private/protected (for parent classes).
  + Defining getters and setters methods.
  + Packing the same-purpose classes in a package.



Figure 5: Encapsulation.

## Inheritance

* Mono inheritance is mostly applied in this project:
  + Use “extend” keyword.
  + Subclasses inherits all parent’s attributes and methods

A screen shot of a computer

Description automatically generated

Figure 6: Inheritance.

## Abstraction

* We use abstract classes in this project:
  + Circuit class.
  + ElectricComponent class.
  + Panel class.

## Polymorphism

* Subclass overrides the method of its parent:
  + For example, in circuit package, ParallelCircuit and SerialCircuit overrides SetEqResistance method in Circuit class.
  + This is because each type of circuit has different formula to achieve the equivalent resistance.

A screenshot of a computer screen

Description automatically generated

Figure 7: Polymorphism & abstraction.

# CHAPTER 5: TESTING AND VALIDATION



## Testing strategies

* The testing strategy involves unit testing, integration testing, and user acceptance testing to ensure the application functions correctly and meets user requirements. Key testing phases:
  + **Unit Testing**: Tests individual classes and methods for correctness.
  + **Integration Testing**: Ensures different parts of the application work together.
  + **User Acceptance Testing**: Validates the application with real users to ensure it meets their needs.

## Test cases

* Test cases are designed to cover various scenarios, including normal operation, edge cases, and error handling. Example test cases:
  + **Adding Components**: Tests adding different types of components and verifying their properties.
  + **Removing Components**: Ensures components can be removed correctly.
  + **Calculation Accuracy**: Verifies the correctness of calculated voltage, current, and resistance.
  + **User Interface**: Tests the functionality and usability of the user interface.

# CONCLUSION

Our electrical circuit simulator project has successfully met the requirements to offer a user-friendly interface for designing and analysing electrical circuits. The application is equipped to handle both series and parallel circuit configurations, providing a versatile tool for users of various expertise levels. One of the standout features of our simulator is its ability to perform real-time calculations, delivering immediate feedback on key electrical parameters such as voltage, current, and resistance. This real-time analysis helps users quickly understand the behaviour of their circuits and make necessary adjustments.

In addition to its analytical capabilities, the application offers comprehensive visual feedback, allowing users to see a detailed representation of their circuit designs. This visual aspect is crucial for both educational and practical applications, as it helps users to better understand the spatial and functional relationships between different components within the circuit. The intuitive drag-and-drop interface makes it easy for users to place and connect components, streamlining the circuit design process.

Looking ahead, there are several areas for potential improvement that could further enhance the functionality and usability of our simulator. One future enhancement could involve expanding the range of available components, introducing more specialized and advanced elements such as transistors, diodes, and integrated circuits. This would allow users to design more complex and sophisticated circuits, broadening the application’s appeal to advanced users and professionals.

Another area for improvement is the enhancement of the visualization features. This could include more detailed and dynamic graphical representations of circuit behaviour, such as animated current flows, voltage gradients, and interactive simulation controls. Improved visualization would make it easier for users to interpret the results of their analyses and gain deeper insights into circuit operation.

Furthermore, adding more advanced analysis tools would significantly boost the simulator’s capabilities. These could include frequency response analysis, transient analysis, and fault simulation features. Advanced tools would enable users to perform more comprehensive and nuanced analyses of their circuits, catering to more complex and high-level engineering and research tasks.

In summary, our electrical circuit simulator project has established a solid foundation by providing an intuitive interface, robust real-time calculations, and clear visual feedback. By expanding the range of components, enhancing visualization features, and incorporating advanced analysis tools, we can further develop this application into an even more powerful and versatile tool for both educational and professional use.

# REFERENCES

1. Our GitHub repository: <https://github.com/NguyenHoangMinh1312/Group31_OOP_147839.git>