HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION
COMMUNICATION TECHNOLOGY

OBJECTIVE-ORIENTED PROGRAMMING
IT3100E
INSTRUCTOR: PROF. TRAN THE HUNG
Class ID: 147839

Electrical Circuit Simulation

Group Members:

Đoàn Tiến Dũng - 20220072

Phạm Trường Sang - 20225996

Đinh Xuân Toàn - 20226067

Nguyễn Trọng Bách - 20226014

1. Introduction

This project is an electrical circuit simulation project built with Java, aimed at helping users gain a better understanding of the operation and characteristics of electrical circuits. With Electrical Circuit Simulation, users can design, simulate, and analyze electronic circuits with basic components such as resistors, capacitors, and inductors such as variable frequency drives, microcontrollers, or ICs.

2. Technical Requirements

- Programming Language: Java

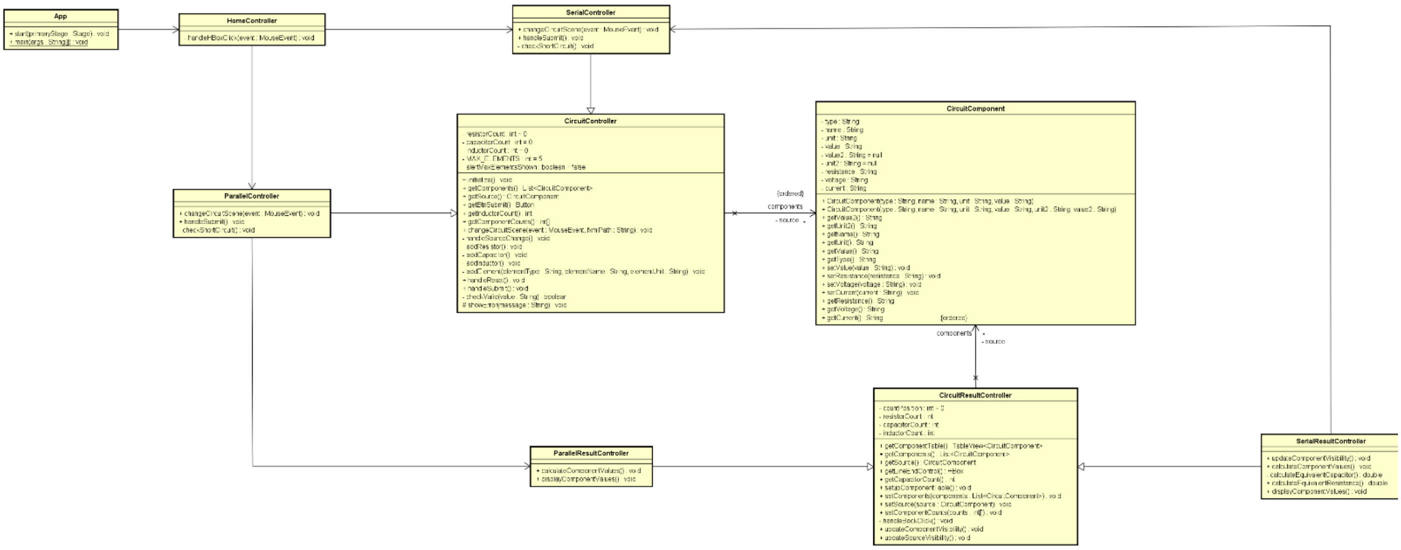
- User Interface: JavaFX - Use JavaFX to build a graphical user interface for the application.

- User Interaction: FX to create interactive user interfaces for circuit design and simulation.

- Operating System: The application can run on various operating systems such as Windows, macOS, or Linux, provided that the correct JDK/JRE is installed.

3. System Building

Class Diagram


App
+ start(primaryStage : Stage) : void
+ main(args : String[]) : void
HomeController
- handleHBoxClick(event : MouseEvent) : void
SerialController
+ changeCircuitScene(event : MouseEvent) : void
+ handleSubmit() : void
- checkShortCircuit() : void
CircuitController
- resistorCount : int = 0
- capacitorCount : int = 0
- inductorCount : int = 0
- MAX_ELEMENTS : int = 5
- alertMaxElementsShown : boolean = false
- initialize() : void
+ getComponents() : List<CircuitComponent>
+ getSource() : CircuitComponent
+ getBtnSubmit() : Button
+ getInductorCount() : int
+ getComponentCounts() : int[]
+ changeCircuitScene(event : MouseEvent, fxmlPath : String) : void
- handleSourceChange() : void
- addResistor() : void
- addCapacitor() : void
- addInductor() : void
- addElement(elementType : String, elementName : String, elementUnit : String) : void
+ handleReset() : void
+ handleSubmit() : void
- checkValid(value : String) : boolean
# showError(message : String) : void
ParallelController
+ changeCircuitScene(event : MouseEvent) : void
+ handleSubmit() : void
- checkShortCircuit() : void
{ordered}
components
- source
CircuitComponent
- type : String
- name : String
- unit : String
- value : String
- value2 : String = null
- unit2 : String = null
- resistance : String
- voltage : String
- current : String
+ CircuitComponent(type : String, name : String, unit : String, value : String)
+ CircuitComponent(type : String, name : String, unit : String, value : String, unit2 : String, value2 : String)
+ getValue2() : String
+ getUnit2() : String
+ getName() : String
+ getUnit() : String
+ getValue() : String
+ getType() : String
+ setValue(value : String) : void
+ setResistance(resistance : String) : void
+ setVoltage(voltage : String) : void
+ setCurrent(current : String) : void
+ getResistance() : String
+ getVoltage() : String
+ getCurrent() : String
{ordered}
components
- source
CircuitResultController
- countPosition : int = 0
- resistorCount : int
- capacitorCount : int
- inductorCount : int
+ getComponentTable() : TableView<CircuitComponent>
+ getComponents() : List<CircuitComponent>
+ getSource() : CircuitComponent
+ getLineEndControl() : HBox
+ getCapacitorCount() : int
+ setupComponentTable() : void
+ setComponents(components : List<CircuitComponent>) : void
+ setSource(source : CircuitComponent) : void
+ setComponentCounts(counts : int[]) : void
- handleBackClick() : void
+ updateComponentVisibility() : void
+ updateSourceVisibility() : void
ParallelResultController
+ calculateComponentValues() : void
+ displayComponentValues() : void
SerialResultController
+ updateComponentVisibility() : void
+ calculateComponentValues() : void
- calculateEquivalentCapacitor() : double
+ calculateEquivalentResistance() : double
+ displayComponentValues() : void

- **App**:

Methods:

- ○ start(primaryStage: Stage): void
- ○ main(args: String[]): void

This class can serve as the starting point of the JavaFX application, responsible for initializing the user interface.

- **Home Controller**:

Methods:

- ○ handleHBoxClick(event: MouseEvent): void

This class handles events from the main interface, for example when the user presses a button.

- **Circuit Controller**:

Properties:

- ○ resistorCount, capacitorCount, inductorCount, MAX_ELEMENTS
- ○ alertMaxElementsShown

Methods:

- ○ initialize()
- ○ getComponents()
- ○ getSource()
- ○ getBtnSubmit()
- ○ getInductorCount()
- ○ getComponentCount()
- ○ changeCircuitScene(event, fxmlPath)
- – handleSourceChange()
- – addResistor()
- – addCapacitor()
- – addInductor()
- – addElement(elementType, elementName, elementUnit)
- ○ handleReset()
- ○ handleSubmit()
- – checkValid(value)
- – showError(message)

• This is the main class that handles the logic of the circuit, including adding components and checking validity.

- **Circuit Components**:

Properties:

- ○ type, name, unit, value, unit2, value2, resistance, voltage, current

Methods:

- getType()
- getName()
- getValue()
- getUnit()

This class represents a circuit component (such as a resistor, capacitor, inductor), containing properties and methods related to that component.

- **Parallel Controller** and **Serial Controller**:

Methods:

- changeCircuit(event: MouseEvent): void
- handleSubmit(): void
- checkShortCircuit(): void

These classes manage the modification and testing of parallel and serial circuits.

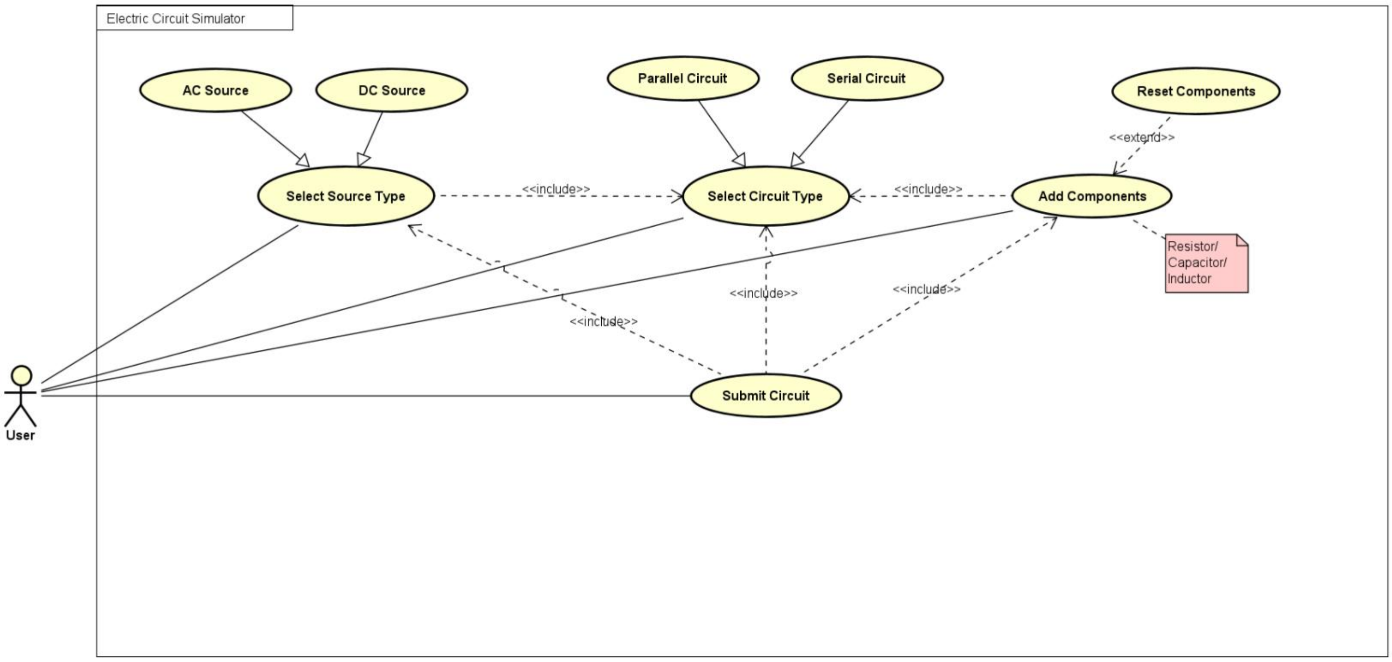
- **ParallelResultController** and **SerialResultController**:

Methods:

- calculateComponent()
- displayComponent()

• These classes process and display the analysis results of parallel and serial circuits.

UseCase Diagram



User:

Represents the system user, capable of performing the following actions:

- Choosing the circuit type (parallel or serial).
- Selecting the type of power source (AC or DC).
- Adding components to the circuit (resistors, capacitors, inductors).
- Submitting the circuit for analysis.
- Resetting components within the circuit.

Use Cases:

- Select Circuit Type: The user chooses the circuit type (including parallel and serial).
- Select Source Type: The user selects the type of power source (including AC and DC).
- Add Components: The user adds components such as resistors, capacitors, and inductors to the circuit.
- Submit Circuit: The user submits the constructed circuit for system analysis.
- Reset Components: The user resets components within the circuit.

4. Design and Using instruction

- This simulator will demonstrate two types of circuits: parallel circuit and serial circuit. To pick a type of circuit, the user can choose one or two tabs in the navigation bar, like in the reference above. Then, the user can start to construct a circuit and press submit when they are done.
- After the user press submit, the application will draw the circuit diagram, output the circuit analysis results in a table, and calculate the equivalent resistance.
- Circuit construction: Each circuit will allow the user to pick a single voltage source and several electrical elements. There are three types of electrical element: resistor, capacitor, and inductor. Likewise, there are two types of sources, AC and DC:
  - To add an element, the user can pick one of three buttons: add resistors, add capacitor, or add inductor. After a button is picked, a row is added after the last row in the panel for the user to specify the parameter of the new element, like in the reference. The parameters corresponding to each element:

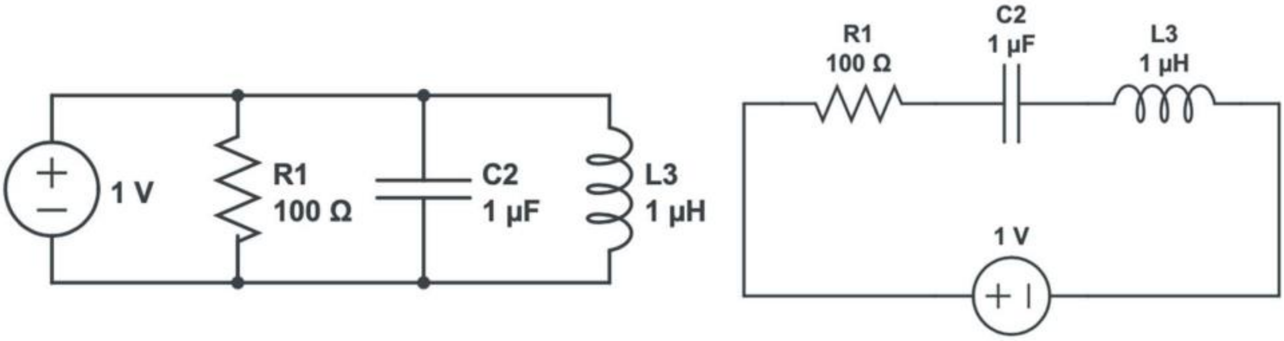
Element type	Parameter	Unit	Name prefix
Resistor	Resistance	Ω	R
Capacitor	Capacitance	F	C
Inductor	Inductance	H	L

Note that each new element must be named according to a specific convention: The prefix is a letter depending on the type of the element (as specified in the table above), and the remaining part of the name is a number that is incremented for each new element. For example, if the user adds two resistors in a row and then an inductor, their names will be R1, R2, L3, respectively. A maximum of 5 elements can be added.

- To pick the source, the user first chooses the type of source (AC or DC) from a dropdown button and based on the option, some text boxes appear next to the dropdown for the user to provide the associated parameters for the type of source chosen:

Type of source	Parameter (unit)
DC	Voltage (V)
AC	Voltage (V), Frequency (Hz)

- Circuit diagram: After the user has constructed the circuit and press the submit button, the simulator will show the corresponding circuit diagram. You are free to choose any form you're your diagram, but here is a suggestion:



You will need to calculate values in the circuit to fill in the empty cells in the above table.

- Short circuit: If a short circuit is detected, you must inform the user of the element causing the short circuit and ask them to change it.

5. Knowledge summary

The solution and formulae to calculate the circuit analysis table and the equivalent circuit will be presented below.

- For the R (Resistance) row, the values are calculated as in the table below:

Element type	R (resistance)	
	AC ($f \neq \infty$)	DC ($f = \infty$)
Resistor	R	
Capacitor	$R_c = -\,2\pi f L\hat{\imath}$	$R_c = \infty$
Inductor	$R_l = \dfrac{1}{2\pi f C}\hat{\imath}$	$R_l = 0$

- For the U (Voltage) and I (Current intensity) rows and the equivalent resistance:

- For the serial circuit:

Equivalent resistance: $R_{eq} = \sum \quad R_i$

I (Current intensity) row: $I_i = \frac{V_{source}}{R_{eq}}$

U (Voltage) row: $U_i = I_i \times R_i$

- For parallel circuit:

If the user uses an inductor element in a parallel DC circuit, notify them that it will cause a short circuit and ask them to change that circuit element.

U (Voltage) row: $U_i = V_{source}$

I (Current intensity) row: $I_i = \frac{V_{source}}{R_i}$

Equivalent resistance: $R_{eq} = \frac{1}{\sum \quad \frac{1}{R_i}}$

In the above formulae, in any case there is a divide by zero (because a resistance value is equal to zero), it means that there is a short circuit. Another case to consider is divided by infinity, in such cases the result is assumed to be 0.

6. Video Demo

Here is the video demo of project: Electrical Circuit Simulation.

7. Contribution of Members

Member Name - ID	**Work**
Đoàn Tiến Dũng – 20220072 (Leader)	Circuit fxml and Source Code. Logic processing.
Phạm Trường Sang – 20225996	UML diagram. Circuit Result fxml and Source Code.
Đinh Xuân Toàn – 20226067	Support design Interface (Circuit fxml). Tester. Report Writing. Video demo. Presentation.
Nguyễn Trọng Bách – 20226014	Support design Interface (Circuit Result fxml). Tester. Report Writing. Slide and Presentation.