**ELECTRICAL LABORATORY SIMULATOR**

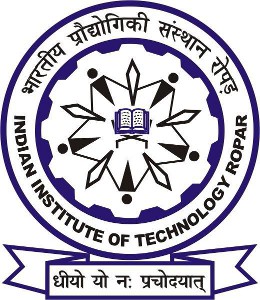
Submitted in fulfillment of term project

By

Electrical Lab simulator Team

Under the Guidance of

Prof. Balwinder Sodhi



Department of Computer Science and Engineering

Indian Institute of Technology Ropar

**ACKNOWLEDGMENT**

We the team of Electrical lab simulator are grateful to all those who have helped us to put our ideas and assigned work, well above the level of simplicity and into something concrete.

We all thank whole heartedly Prof. Balwinder Sodhi for constantly motivating for doing better, for his valuable suggestions, guidance and showing complete confidence in our work.

We would also like to thank Prof. Ranjana Sodhi (Department of Electrical Engineering) for providing us the necessary details regarding the basic functionality required in electrical laboratory simulation systems.

**TEAM MEMBERS**

|  |  |  |
| --- | --- | --- |
| **Name** | **Entry No.** | **e-mail** |
| Abhinav Puri | 2011CS1001 | [pabhinav@iitrpr.ac.in](mailto:pabhinav@iitrpr.ac.in) |
| Arbaaz Singh Sidhu | 2011CS1004 | [arbaazs@iitrpr.ac.in](mailto:arbaazs@iitrpr.ac.in) |
| Gurasis Singh | 2011CS1008 | [sgurasis@iitrpr.ac.in](mailto:sgurasis@iitrpr.ac.in) |
| Harsimran Singh | 2011CS1009 | [sharsimran@iitrpr.ac.in](mailto:sharsimran@iitrpr.ac.in) |
| Sahil Dabra | 2011CS1033 | [sahildb@iitrpr.ac.in](mailto:sahildb@iitrpr.ac.in) |
| Gourav Bansal | 2011CS1057 | [gouravbn@iitrpr.ac.in](mailto:gouravbn@iitrpr.ac.in) |

**DECLARATION**

We declare that this written submission represents our ideas in our own words. We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Electrical Lab Simulator team

Date: 11-11-2013

**ABSTRACT**

Electrical circuits are the building blocks of all electronic devices whether they are simple devices such as digital watches, timer circuits or sophisticated devices such as mobile phones, Laptops, etc. While designing new circuits, errors may occur that can have adverse effects sometimes leading to monetary loss. Therefore, a need for virtual electrical laboratory arises. Electrical circuits can be tested in a virtual environment before the actual physical implementation to prevent any kind of loss. This “Electrical Lab Simulator” project aims at providing a user-friendly GUI to easily and quickly assemble circuits and get the required output.

**CONTENTS**

1 Introduction

* 1. Purpose 7
  2. Scope 7

2 Design Objectives and Functionality

2.1 Software Architecture pattern (MVC) 8

2.2 Design & implementation of GUI 10

2.3 Circuit Solving using Octave 11

2.4 Saving/Loading the Circuit 12

3 Diagrammatic Description

3.1 Logical View

3.2 Usecase Diagram

3.2.1 Usecase diagram for solving a circuit

3.2.2 Usecase diagram for saving/loading a circuit

3.3 Sequence Diagram

3.3.1 Sequence diagram for solving a circuit

3.3.2 Sequence diagram for saving a circuit

4 Conclusion and Future Scope

4.1 How is “ELS” useful?

4.2 Conclusion

Bibliography

**1.1 PURPOSE**

Technology is taking great leaps especially in the silicon industry. New gadgets, electronic devices are under constant development. Building new electronic device means designing new electrical circuits. Once a circuit is designed on the paper, it has to tested and verified for the required results before putting it into actual physical implementation. While designing a new circuit, errors may occur. Circuits may malfunction that can sometimes lead to serious consequences such as component loss leading to financial losses. Also it is difficult to rectify the circuit changing the wires between terminals of different electrical components especially when we are dealing with very complex circuits.

Taking into account the above mentioned facts, the need for some virtual environment simulating the basic electrical laboratory arises. This project aims at providing a user friendly Graphical User Interface (GUI) tool that can be used to implement the circuits in the virtual environment and check for the behavior of circuits. The tool provides the basic components such as resistors, capacitors, inductors, power source, voltmeter, ammeter, etc. These components can be used to implement a simple circuit and check if the circuit is behaving in a way it should have behaved in a real world. Also with the software, we can easily modify the circuit, changing the connections/components with only few clicks of mouse. The circuits can be rectified rather easily using this tool opposite to changing the circuit in reality on breadboard.

**1.2 SCOPE**

“Electrical Lab Simulator” is a java-based project, so it is compatible with all operating systems supporting java. The GUI part is implemented using java swing components (Jpanels and Jframes). The calculations are handled by GNU Octave (free software under GNU General Public License).Octave is high-level programming language intended for numerical computations. Octave is linked with java using JavaOctave library.

**2 DESIGN OBJECTIVES AND FUNCTIONALITY**

**2.1 MVC (***Software Architecture pattern used***)**

Model-View-Controller (MVC) is a software architecture pattern which is used to separate application’s concerns. MVC pattern separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes:

* **Model**: The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).
* **View**: The view manages the display of information.
* **Controller**: The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate.

**Model** Component objects (resistors, power sources)

Updates Manipulates

**View** - draw component routines - drawPanels

**Controller -** Saving Routines - Event Handlers

Sees Uses

A typical collaboration of the MVC components is shown in the figure above.

Our project uses the MVC pattern. Various electrical components such as resistors, power sources are created as objects acting as model. Under the View part, we have various draw functions which draw the circuit on screen with the help of information from Controller which contains certain event handlers. Controller also updates the model objects accordingly.

When the project is launched, *main.java* within the controller is run and it executes the default draw routines which draw panels and frames and the workspace area. Panels contain the various electrical components using which the circuit can be designed in the workspace area.

Event Handlers in the controller respond to the inputs provided using mouse or keyboard. When user clicks on the resistor to insert it in the circuit, mouseEvent and keyboard handlers in *Resistorhandler.java* get activated and get the input values of resistor, give it to the resistor object in model and then update the view to draw the resistor on screen across the selected endpoints.

In the project, controller manipulates the model exclusively so we can say that the passive model is employed here. The controller modifies the model and then informs the view that the model has changed and should be refreshed.

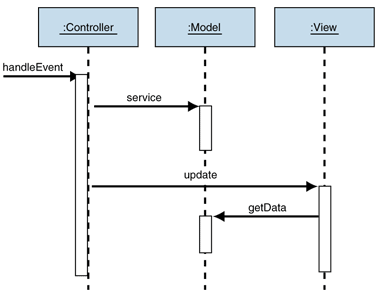
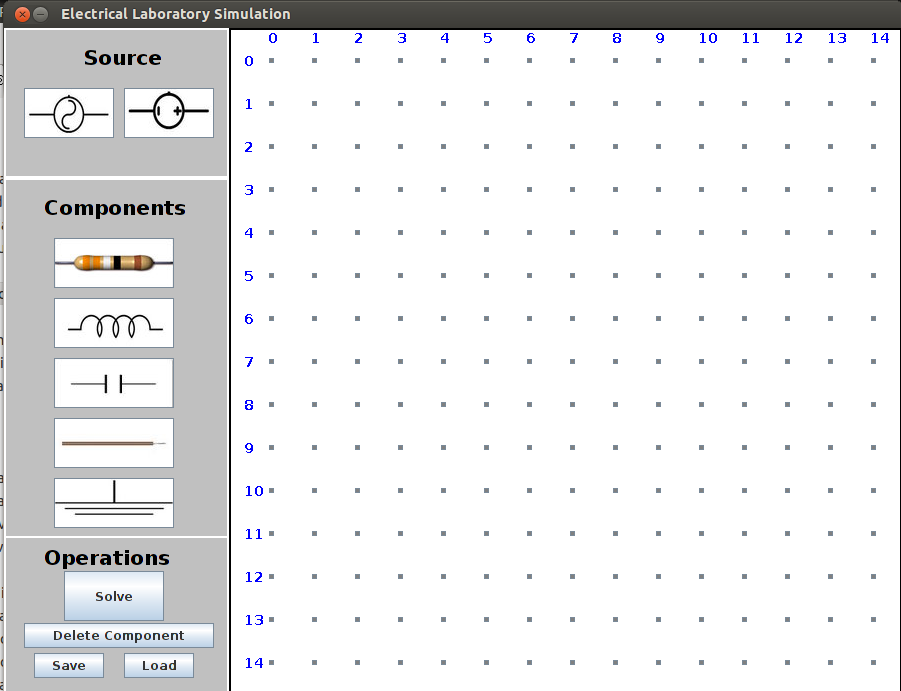


Figure: Behavior of the passive model

**2.2 DESIGN AND IMPLEMENTATION OF GUI**



* Workspace: The panel on the right is workspace consisting of 15 \* 15 grid of nodes across which components are added. These nodes are java buttons which when clicked become green in colour. To add a component, select two buttons as terminals and then select the required component from the components list in the left panel.
* Sources:
  + AC (alternating current)
  + DC (Direct Current)
* Components :
  + Resistance
  + Inductor
  + Capacitor
  + Wire
  + Ground (earthing)
* Operations
  + *Solve:* Click on the solve button to solve the circuit drawn in workspace.
  + *Delete Component:* Select the component to be deleted (terminals of the component) and then delete on delete component button to delete the same.
  + *Save:* Clicking on this button will save the circuit in *name.els* file.
  + *Load:* This button will load the existing valid .els file and draw the circuit

**Implementation**:

As our project is java-based, GUI part is implemented using java swing (GUI widget toolkit) and AWT (Abstract Window Toolkit). The panel on the left side containing all the components is a JPanel. On the right side, we have a grid of small buttons which become green when clicked. These buttons can be thought of as the holes in a breadboard used for making circuits. User has to select the two buttons across which he wants to put the electrical component. Once the buttons are selected, electrical component is chosen from the list of components provided in the left panel simply by clicking on the required component. After clicking on the component, a frame pops up asking for the details of the component selected. E.g. if user selects a resistor, it will ask for the resistor value or the colour code scheme followed in resistors. After getting all the required values, the component is inserted across the two buttons selected in the workspace on the right. Thus, the whole circuit can be assembled step by step by first using the two terminal points across which the component is to be inserted and then selecting the component with required details.

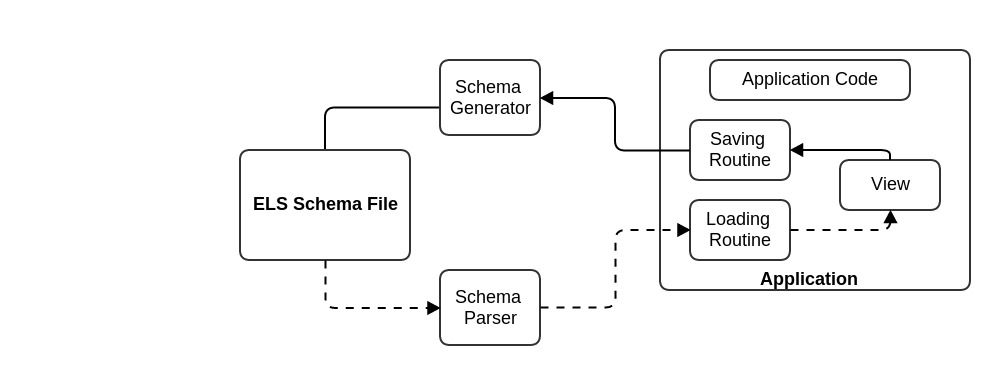
**2.3 Circuit solving**

First of all circuit is simplified. The simplifier removes wires i.e. it joins two end points of a wire to create a single junction. It removes one of the junctions from all the electrical components and assign second junction of wire to all the electrical components.

The circuit solver creates an array of junctions (nodes) involved in the circuit. Then it assigns zero potential to grounded terminal. If not present, then it assigns zero potential to negative terminal of dc power source. Then, it applies Kirchoff Current laws and voltages laws to create a matrix of coefficients of junctions with unknown potential. Then we solve the equations by finding inverse of matrix. After finding potential at each node we can easily find current flowing through any electrical component.

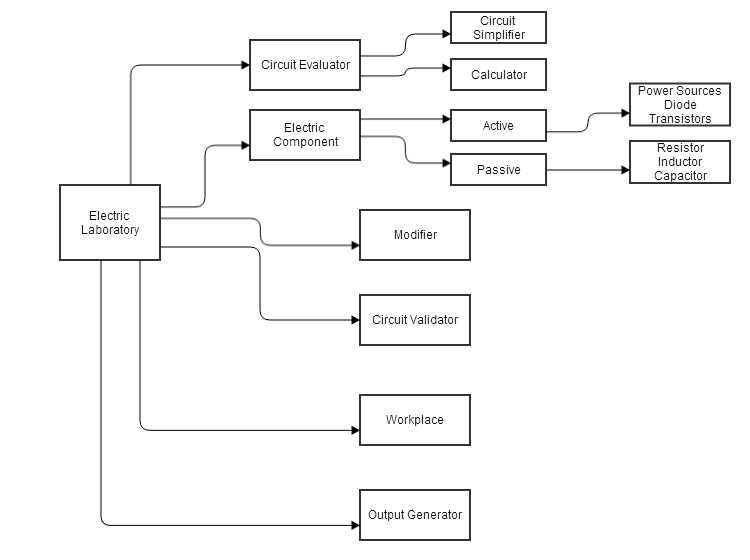
**2.4** **Saving the Circuit**

The circuit once drawn can also be saved for future references. All the parameters of the components used in the circuit are saved schematically in file having .els extension. The schema generator captures all the parameters of the circuit drawn and writes them in .els file which is saved at the destination specified by the user. In future reference, when load button is pressed, saved .els file is selected and the circuit corresponding to that particular .els file is rendered on the screen. This task is accomplished by parser. Parser takes in .els file and look for the tags defined for various components like <resistor> tag for resistances. Within <resistor> tag, all the attributes like value, its end point terminals are present. These values are extracted by parser and given to drawing routines to draw the circuit on the screen.

****

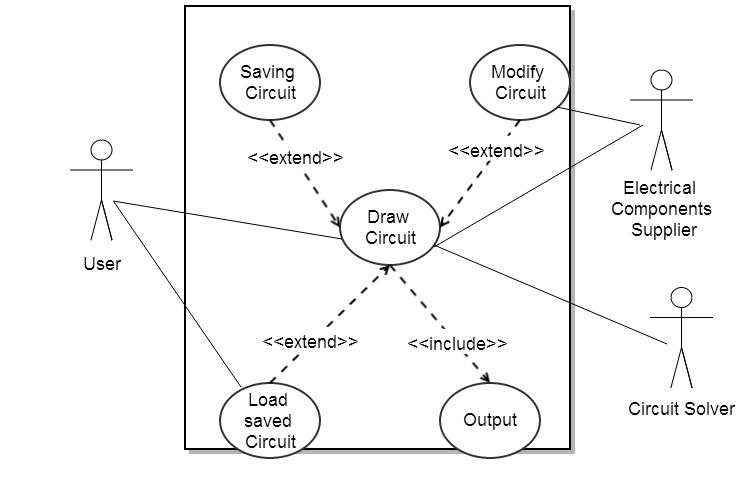
**3 DIAGRAMMATIC DESCRIPTION**

**3.1 Logical View**

****

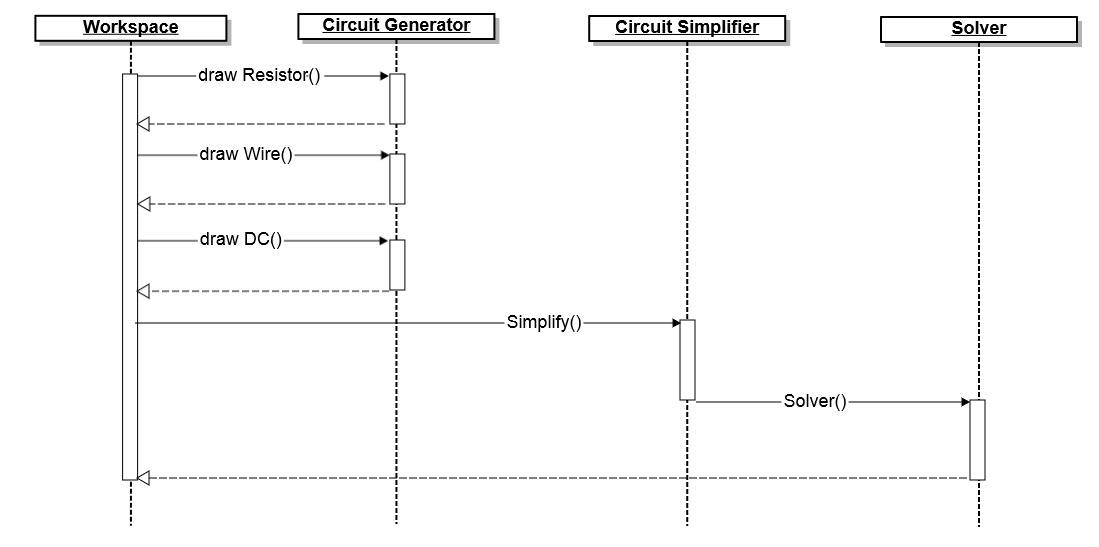
**3.2 Usecase Diagram**:

**3.2.1 Usecase diagram for solving a cicruit**:

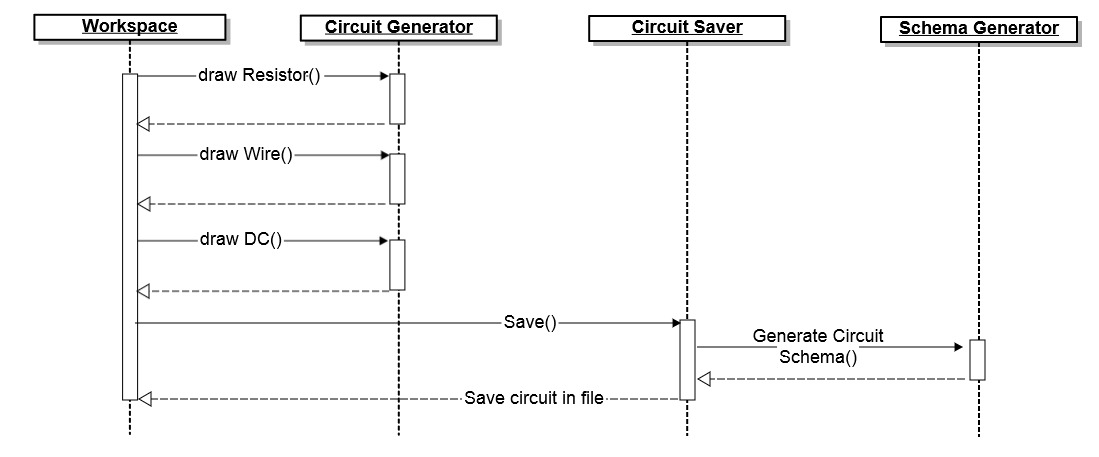


**3.3 Sequence Diagram:**

**3.3.1 Sequence diagram for solving a circuit:**



**3.3.2 Sequence diagram for saving a circuit:**



**4. CONCLUSION AND FUTURE SCOPE**

**4.1 How is “Electrical Lab Simulator” useful ?**

* Electrical Lab Simulator aims at providing the user-friendly GUI tool for making and editing the circuits and make the circuit behaves in a way it should have behaved in the real world.
* Circuit may malfunction due to errors. Erroneous circuits may have adverse effects causing serious consequences like costly component loss, or circuit may catch fire. Using a simulator tool prevents these losses.
* Circuits can be rectified rather easily in the virtual environment compared with real deployment on some breadboard. Especially when dealing with complex circuits, this simulator is of great help.

**4.2 CONCLUSION**

**Electrical Lab Simulator** offers the user a simple and easy way to design and implement the circuit virtually and study its behavior. Its interface is very user-friendly and welcoming. One can design a circuit in just few clicks of the mouse.

The world of electrical circuits is very vast and expanding. New circuits are being designed each day and the older ones are improved. So, the simulation of large variety of circuits is quite a difficult task. There are many types of electronic components involved in making these circuits. E.g. resistors, capacitors, inductors, diodes, transistors, integrated circuits (ICs), transducers, switches, power sources and many more. Further, there are many topologies of circuits (components can connect in various fashions). So, modeling each and every type of circuit is a challenging task.

In this project, we aim at simulating a basic circuit using resistors and power sources.

**BIBLIOGRAPHY**

* JAVA-Complete Reference
* [stackoverflow.com](http://stackoverflow.com/)
* Gliffy.com