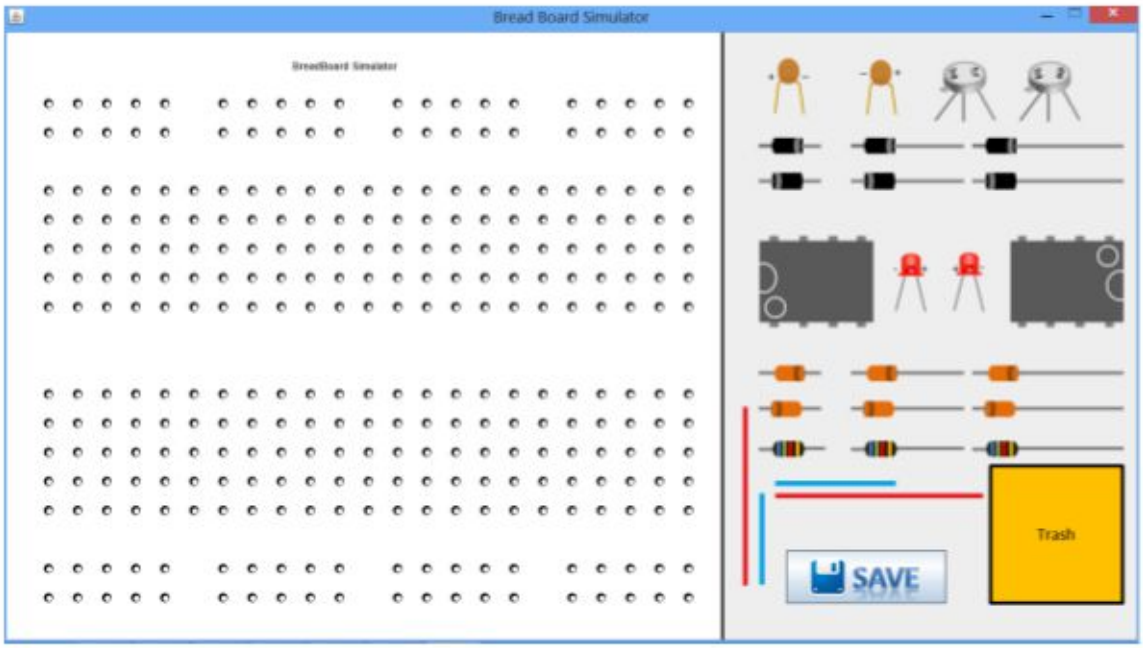
VIRTUAL BREADBOARD SIMULATOR

DOCUMENTATION

RAHUL HUILGOL - 11010156
RACHURI ANIRUDH - 11010155
HARSHITH REDDY - 11010149



UNDER DR.SANTOSH BISWAS AND DR.ARIJIT SUR
DEPT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

CONTENTS

FEATURES

- Drag and Drop
- Automatic Positioning of Components depending on their size
- Moving components on the breadboard
- Deletion of components
- Error Handling and Minimization through dialog boxes which notify the users if he/she makes a mistake
- Password-protected login system
- Registration of new users for this database
- Download facility for Program and Documentation
- On Windows, the program needs to command-line commands for starting the program, just simple double click on jar file
- Matrix generation
- Seven fields for every node in the matrix,i.e.lot of data passed through
- Printing matrix into text file

1 USER INTERFACE

This is a virtual breadboard simulator, designed to ease the process of setting up circuits and checking for the output. Here we allow the user to build the circuit virtually and simulate it.

1.1 LAYOUT

The user is presented with **two panels** : one consisting of the breadboard, and the other with all the essential components of an analog circuit.
The Breadboard provided is a standard breadboard with 10+4 rows and 23 columns.
The circuit components provided on the right panel can be added to the breadboard by **drag and drop** method. The user selects the element which he wants to add, drags it to the location where he/she intends to place it, and drops it there. When elements like the resistor and capacitor are added to the breadboard, a **dialog box** pops up asking the user to enter the details, i.e. the **value** of the resistor or the capacitor. This dialog box pops up only the **first time** a component is added from the Component Panel.
The user can **move** the position of an already inserted circuit component. In addition to this, the user can **remove an element** by just dragging it and dropping it in the **trash** area in the bottom right of the screen.

1.2 LIST OF COMPONENTS AVAILABLE FOR USE

- **Diode** - Three sizes and reversed orientation
- **Zener Diode** - Three sizes and reversed orientation
- **Operational Amplifier (Op-Amp)** - Normal and reversed
- **Light emitting diode (LED)** - Two orientations with positive and negative terminals inverted
- **Resistor** - Three sizes, Any Value
- **Capacitor** - Any Value, Two orientations with positive and negative terminals inverted
- **Connecting Wires** - Horizontal two sizes, Vertical two sizes
- **Bipolar Junction Transistors** - Two orientations EBC or ECB

2

DETECTING AND RESOLVING ERRORS

- When the user places an element in a place where the element can not go into by the rules of electronics, then the element doesn't get placed there. It **automatically goes** to the **nearest correct position** which it can take, even if the user tries to place a component outside the breadboard.
- Other than the connecting wires, **no component can be placed** in the **top two and bottom two rows** . This is because these rows are meant only for grounding or power supply.
- For example, the **Op-amp** can only be placed in the middle part of the circuit. If the op-amp is dropped in either the upper or lower parts of the breadboard, it gets **short-circuited**. So the op-amp automatically goes to the region in between the two parts of the breadboard.
- The connecting wire doesn't get placed in regions where it is **not necessary**, i.e. when the points are already at the same potential due to the construction of a breadboard. For example, a vertical connecting wire won't get placed within the five rows of a part of the breadboard as both the ends of the wire are already at the same potential. This is because in a breadboard, every point in a column in a set of five rows is at the same potential.
- The user also gets an error message when he/she tries to place **one op-amp over another** op-amp.
- Later when giving the power supply, if the user enters points whose **x and y co-ordinates** are **not in range** of the breadboard, the user gets an error message.
- Also, the user can not give a **non integer value** of the **voltage** or **frequency** of the power source. If he/she does so, an error message pops up.

3

THE MATRIX

3.1 SAVING

- When the user is finished building the circuit, he/she can just click the **save** button present towards the bottom right of the screen.
- On clicking Save, a dialog box pops up which takes the details of **power supply** and the points across which the power supply is to be applied.
- The user can choose whether to apply AC/DC and also whether a sine wave or square wave, if AC.
- The user has been given the option of entering peak voltage and frequency of the signal of AC, or the steady voltage of DC.
- The circuit is then converted into a **simple matrix representation** which is then put in a text file for further processing.

3.2 MATRIX REPRESENTATION

- The final matrix to which the circuit is converted holds the following data
  - The **Row** number of a point
  - The **Column** number of a point
  - The **Potential** of a point. If two points have same Potential value, it signifies that the points have the same potential.
  - The **Component** associated with a point followed by its **ID**. Here, ID tells us the serial number of all components of that type on the breadboard.
  - The **terminal** of the component associated with a point. It can be 1 or -1 for positive and negative terminals respectively. For components without any terminals, the default value is 0.
  - The **value** of the resistor or capacitor associated with a point, if present

    Example: 5 10 4 cap2 -1 1000.00

- In the final matrix representation output to the text file, the representation of each node is separated by **two tab spaces**. When reading this output.txt, these two tab spaces should be considered.
- Initially, the different points of the matrix are **set to default values**, such that all the points with same potential have same default values. For example, the 3rd point of the 2nd row and the 3rd point on the 4th row are assigned the same default values.
- When a component is dropped onto the breadboard,the **component, id and value** fields of the points are changed.
- When a **horizontal connecting wire** is added to the breadboard, all the points with the potential value equal to that of the right end are set to the potential of the left end.
- When a **vertical connecting wire** is added to the breadboard, all the points with the potential value equal to that of the bottom end are set to the potential of the top end.
- When a component is **removed** from a position the default values of a point are reset.

A

APPENDIX

The following section will help in reading the code.

A.1 COMPONENT IDS

These component ids were used repeatedly to refer to the components in the code.

- 0 and 1 represent capacitor
- 2 and 3 represent transistors
- 4 and 5 represent the small sized diode
- 6 horizontal blue connecting wire
- 7 vertical blue connecting wire
- 8 and 9 op-amps
- 10,11 and 12 resistors
- 13 and 14 medium sized and large sized normal diodes
- 15 and 16 LEDs
- 17 and 18 medium sized and large sized reverse diodes
- 19 normal zener diode medium sized
- 20 normal zener diode large sized
- 21 normal zener diode small sized
- 22 reverse zener diode small sized
- 23 reverse zener diode medium sized
- 24 reverse zener diode large sized
- 25 horizontal red connecting wire
- 26 vertical red connecting wire