ELEC-A7151 Object-oriented programming with C++

Circuit Simulator

Project Plan

Team Members:

Henrik Toikka

Kwasi A. Boateng

Patrick Linnanen

Unna Arpiainen

Contents

Introduction

Circuit simulation is a technique of modelling a circuit using mathematical expressions to check and verify the design of electrical and electronic circuits. Our main goal is to develop simulator software with an impressive GUI to aid researchers, students, and engineers understand the basic concepts of electronics.

Scope of the work

Our circuit simulator will include, but will not be limited to, the features listed below.

- Basic components: resistor, capacitor, inductor and a constant voltage source
- A graphical user interface for building the circuit and showing results (must be able to add, connect, move and remove components and wires)
- Saving and loading circuits from files
- RMS voltage and current calculations
- Basic ready-made circuits for showing the program's functionalities
- Capability to read circuits from LTSpice files.
- Plot the time domain for voltage and current
- Sinusoidal AC voltage source and sinusoidal AC and constant DC current sources

How does it work?

The simulation will use Modified Nodal Analysis (MNA) to approximate system behavior. The method is a modification of the traditional Nodal Analysis methodology, which relies on known Laplace transforms and simplification of a circuit into nodes and their interconnections. MNA uses the time-domain forms of components to form a differential-algebraic system of equations (DAEs), I.e. a system that contains either algebraic, differential or both kinds of equations.

To solve DAEs, we need an ODE solver library. Currently, the plan is to use "odeint", but we also found "GNU Scientific Library", which is implemented in C but should be easy to integrate with CMake.

The program will enable the user to create or import electrical networks with a GUI or CLI, which are then stored in the program as a collection of nodes that contain information about

their connections. This node container can then be given as a parameter to a simulator object, which will perform the following steps:

- parse the node container and determine the complexity needed if it is not explicitly stated
- perform the four MNA steps to formulate the DAEs
- and then call the necessary ODE solvers to output the desired data.

The file I/O of the program will – apart from graphs – be implemented with basic text files; the raw data about the electrical network can be stored as a netlist, and the graphical circuit diagram can be stored as a netlist with additional coordinate information. Both will be modeled after SPICE and LTSpice, respectively, to enable the use of 3rd-party software.

High Level Structure of the software

The GUI will be separated from the simulator. In addition, the program will be modular to allow for simple testing and simulation of complex circuits from the command line without graphical rasterization.

The program will consist of two main components, implemented as classes:

- Simulator
- GUI

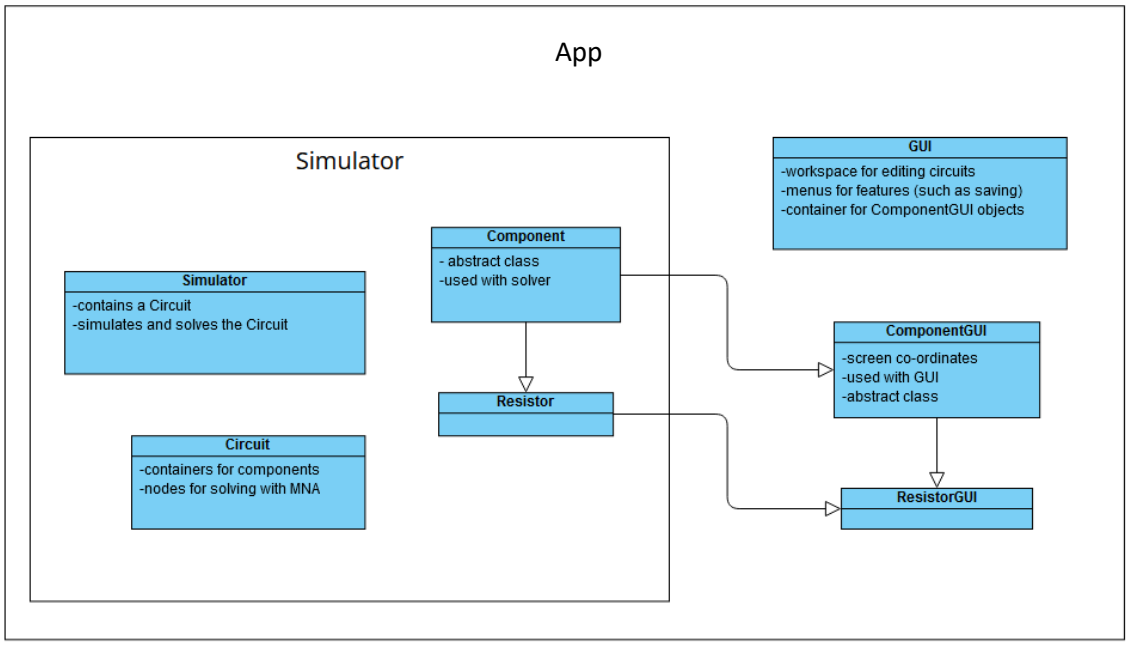

Figure 1. Application

Circuit

The Circuit class structure is intended to make it simple to use with the MNA solver. It includes a list of all the components in a linked circuit with details of their linking property (parallel, series).

Component

The Component class is an Abstract class that each component will inherit from. The derived components will include but are not limited to a resistor, capacitor, inductor, and DC voltage

source. ComponentGUI inherits from the base Component class and will only be used by the GUI.

The components and their connections will be kept in a list container, which the MNA solver will utilize after creating a list of nodes.

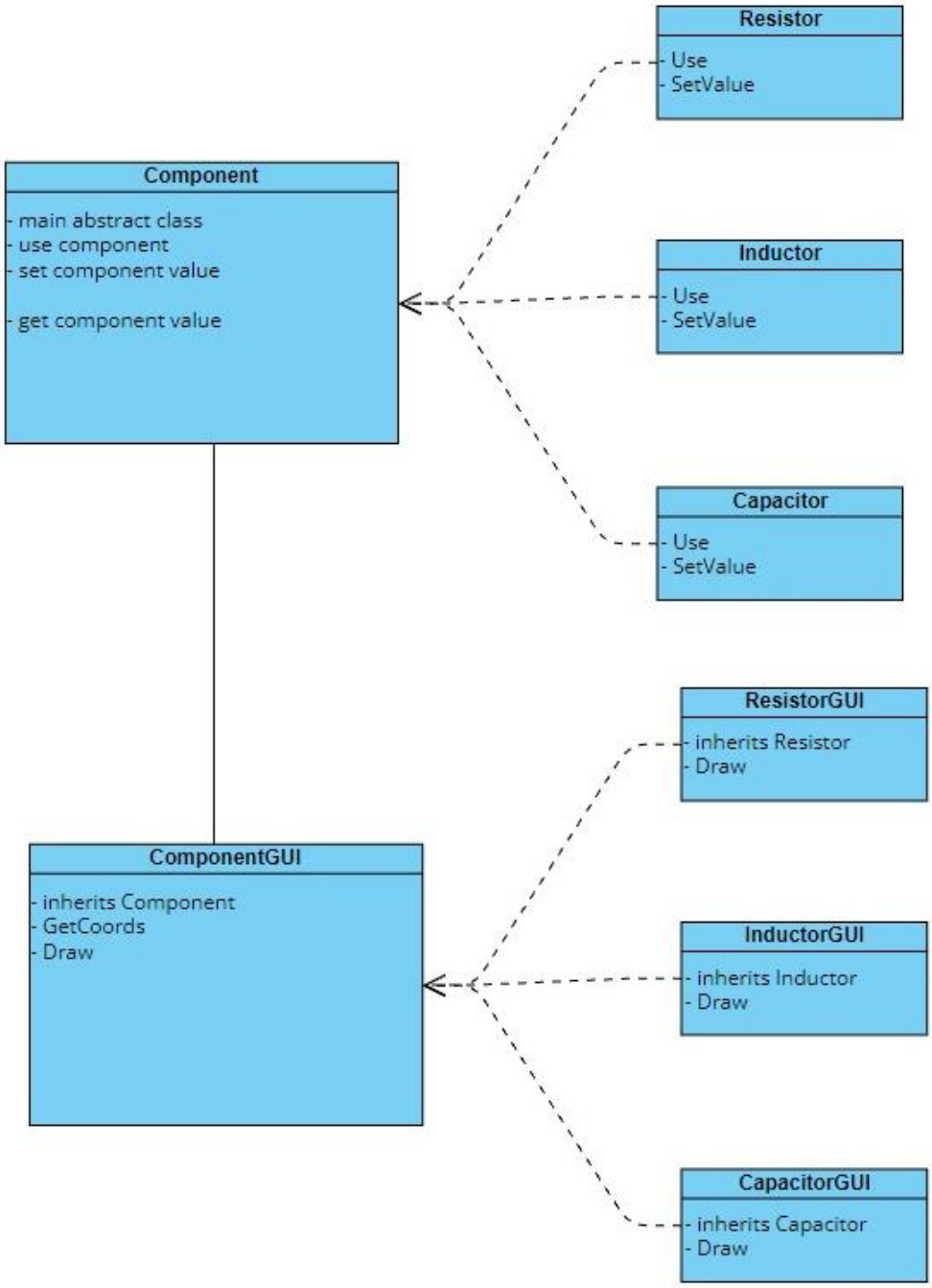

Figure 2. Component and its GUI counterpart classes

Details about the GUI

The user should be able to select and add components from a selection menu as well as move and connect them on the workspace. The circuit will be visualized with electronic symbols with additional information as text near the components. Other features like saving and loading should be accessible through a menu bar. The interface will be in a similar style to the popular simulator software LTSpice.

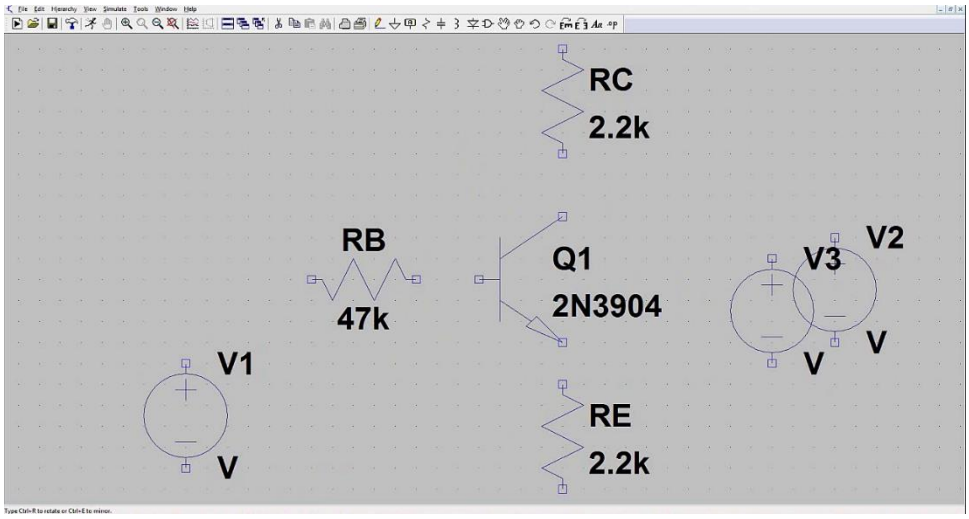

Figure 3. Snapshot of LTSpice simulator software

Planned use of external libraries

GUI

- SFML/ImGui

Solving circuits

- Odeint

Division of work

Primary GUI:

- Henrik (Front/Back end)
- Kwasi (Front/Back end)
- Patrick (Back end)

Primary solver:

- Patrick
- Unna

NB: Work responsibilities may change depending on the difficulty, and assistance will be given where needed.

Planned schedule

Week 1, Nov 1 - Nov 6

- ❑ Project plan

Week 2, Nov 7 - Nov 13

- ❑ Finalizing project plan
- ❑ Creation of main classes
- ❑ Initial GUI interface

Week 3, Nov 14 - Nov 20

- ❑ Read/Write circuits to files
- ❑ Initial solver
- ❑ Data structure for circuits done
- ❑ Component manipulation and all basic features done in GUI

Week 4, Nov 21 - Nov 27

- ❑ Solving basic circuits
- ❑ Displaying recorded data in a plot

Week 5, Nov 28 – Dec 4

- ❑ Optimizing solver
- ❑ Finalizing graphical design
- ❑ Refactoring

Week 6, Dec 5 – Dec 9

- ❑ Project documentation
- ❑ Project submission