Circuit Board Designer Update

Kenneth Shipley, Joseph Spear, Jason Rivas

1. Changes

a. GUI

One of the biggest changes made to the GUI is that the Pallet Wheel has become a feature that will be completed if time permits. This is because the addition of the toolbar simplified the access to the tools and is much easier to implement.

Another change made to the GUI was the addition of the main workspace being turned into three different pages. These pages consist of the Circuit Designer Canvas, Circuit Board Converter, and File Management. This could have been done in two pages, however, since we are omitting the menu bar (File, Edit, View, etc.) adding a File Management page would keep the GUI more coherent. The Circuit Designer Canvas page is where the user will design the circuit and place all the components. The Circuit Board Converter page is where the user's made circuit is converted to the end product PCB image. The File Management page is self-explanatory as this is where all the file management buttons exist such as Save, Save As, Create, and Load.

b. Classes

The overall class structure has remained the same (see Figure 1). There are still ten component classes that are child classes of a Component class and there is a Schematic class that has a list of Component and Comment objects. The difference is the redistribution of functions from Component's child classes to the Component class itself. This was done because the same thing was being done in each of them. What's left in the child classes, is the number of pins and a draw function as of now. In the future they will also have a reference to an SVG file for drawing. The Schematic class has many new functions that are auxiliary to main functions within it. The monte_carlo method has several of these added functions to help make a randomized pin layout, to create paths between components, and to score the layout.

2. What has been accomplished

a. Completed

i. GUI

The only part of the GUI that is completed is the GUI animations as this was one of the simplest things to animate.

ii. Classes

The structure of the Component and Schematic classes is completed. We can create a schematic class and add any of the component types to it. For these components, we can give them all of the information they need to be drawn, moved, connected, and deleted; its label can also be edited.

b. Partially Completed

i. GUI

A majority of the GUI is in a state where all components are partially completed. All of the different pages have been implemented but none of them have been completed yet.
The page that has been completed the most is the File Management page as the buttons have been placed, they just do not have any functionality to them yet. There are built-in functions with PyQT that should allow this to be easily implemented as saving, opening, and creating files are standard Windows functions. As for the other pages, they exist but only labels exist on the page in order to differentiate the pages from each other since they would be blank otherwise.
The toolbar on the right side of the program is mostly finished as well as the buttons exist and any animations that are a part of it have been completed. The buttons do not have any other functions otherwise as the Circuit Designer Canvas function has yet to be implemented.

ii. Classes

_____Currently, the save function works if the filename given doesn't exist. Likewise, the load function works if the filename given does exist. For the Monte Carlo method, we can currently randomly place all components pins onto a grid but cannot yet generate paths to connect them. The placement method can also get stuck and loop for a very long time, so we have to implement some form of cap to it. We also have a few of the components' SVGs finished but need to finish the rest of them.

3. What still needs to be done

a. GUI

The next steps to the GUI are to implement the Circuit Designer Canvas and the menu for the Circuit Board Converter. The Circuit Designer Canvas is the major hurdle as it implements aspects of drawing on a canvas and drag-and-drop features. A test project is being made to

learn how PyQT uses these features and then this test project will be implemented into the Circuit Designer page. There are three primary test projects that are being worked on, each of which are providing a solution to a problem found in the GUI. The first solution is the Drag and Drop implementation, which is allowing for an image (such as diode.svg) to be dragged from the side, representing the area where a "Diode" button would be located, onto the main workspace. The second test project is a click coordinate project, where clicking on the workspace area causes the coordinates to be output to the command line. Both the screen coordinates, and the local window coordinates are displayed. The final project is a canvas workspace which would be used to allow all of the widgets to work together, and would take the place of the main window for the program.

Throughout the test project, we are being mindful of figuring out how to keep track of the different elements that are being added such as their position. The Circuit Board Converter page should not take much to implement as it will involve a menu system for the inputs of the Monte Carlo and A* algorithm for component placement. The page will also have a space for the finished PCB image to be placed.

b. Classes

All A* methods still need to be implemented. When they are done, we can connect the components and then finish out the Monte Carlo method. The next thing to do is then tying all of the tools from the GUI to the backend functions within the schematic class. This includes how we will draw the components as well as manipulating each component's SVG image.

c. PCB Converter

The PCB Converter function has yet to be implemented. Thankfully the test project that is being made to understand how to build the Circuit Designer page has several similarities to the PCB converter. The Circuit Designer page allows for drawing using a mouse whereas the PCB converter will do a similar thing but instead of being controlled by the user, all the elements are drawn by the program automatically using the same functions. The only thing will be figuring out how to use the outputs of the Monte Carlo/A* function to draw the PCB image.

4. Issues

a. GUI

One of the greatest issues currently is the ability to have both drag/drop functionality, as well as sending a coordinate of the location to storage. This is mainly an issue with understanding how PyQt is implemented and being able to effectively utilize it's plethora of features. Currently, we have been able to generate a solution to drag and drop, as well as a solution to locate the coordinates of a click. The current issue is combining the two to have them occur simultaneously. Implementing this in a canvas is also necessary as the canvas will be the object that the user interacts with, as well as where all of the data for the PCB will derive from.

Another issue with the GUI is the inability to snap the terminals together and make the terminals have the same coordinate (this is important to create the corresponding array needed for the optimization of the PCB). This issue piggy-backs off of the first stated issue, but it is integral in the functionality of the primary goal.

b. Classes and PCB Layout Creation

The only issues for the classes are that they don't talk to the GUI and we will have to work together to figure out how to tie it all together. After we figure that out, the last step is to make the PCB layout, but we don't quite understand how to map the output from Monte Carlo to a physical PCB.

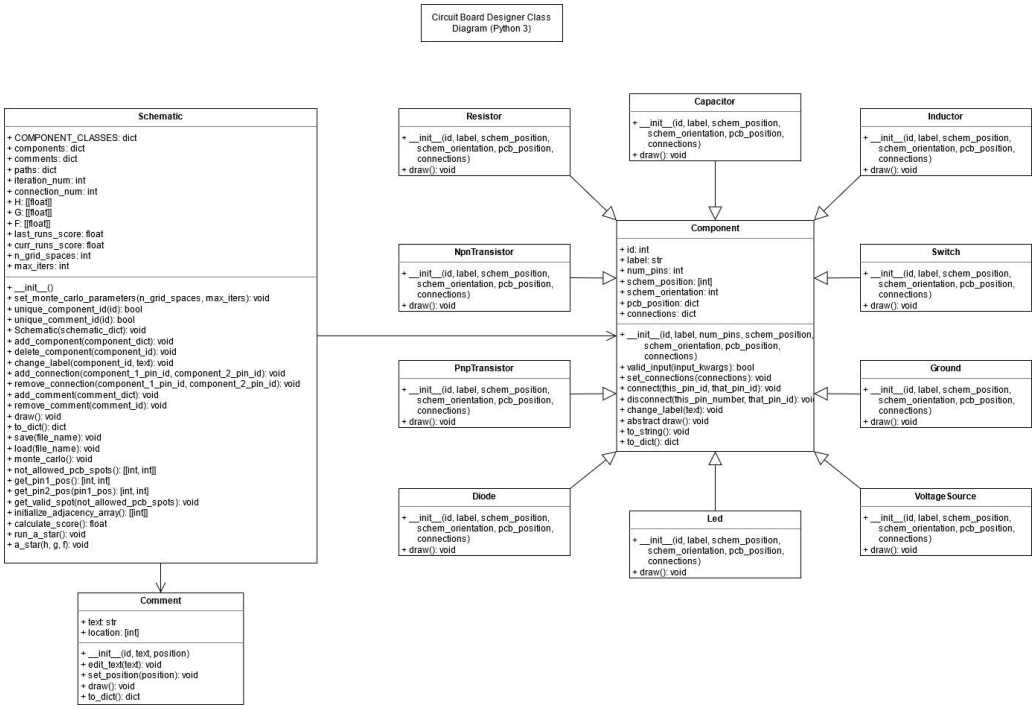

Figure 1: UML diagram for the classes used.

Class Diagram Link