Circuit Board Designer

Authors: Kenneth Shipley, Joseph Spear, and Jason Rivas

Abstract:

This program provides a simple UI that allows the user to create a circuit schematic using 10 provided components. The user will be able to edit the schematic via drag/drop, delete, connect mechanisms, and, once satisfied with the schematic, the user can export the schematic to an optimal PCB layout image.

Description:

Nearly everyone has seen some type of electronic device in their home, at work, or out in town. From a TV remote, to a fridge, someone only needs to take apart something to find some electronic bits in there. In most cases, these electronic 'bits', or components, are put onto a plastic board and connected in such a way that allows the device to work. The alternative is a rats nest of wires going every which way that is nearly impossible to fix if anything wrong happens. To make such a board, engineers in industry typically use some design software to layout every component that is needed to produce a working TV remote, for example. They design it in the form of what is called an electrical diagram/schematic that is easy to read by anyone that knows what the magic symbols mean. Once they are done with the design, they need to convert the schematic into something that can be put onto plastic. This includes: the pads or spots the components sit on, the traces or connections between components, and text for a technician to be able to read. One way for this to be accomplished is for the engineer to design the layout with every pad, trace, and bit of text in the place the engineer deems best. A better way is to get rid of the human in the layout process.

This software will have three primary functions: an interactive interface that the user will be able to easily create circuit diagrams on, a pathfinding algorithm to determine the optimal PCB layout, and an exporter that will turn the PCB layout file into an image. These three functions will make up the entirety of the Circuit Board Designer, and they are discussed in detail below.

Circuit Builder:

Open-source circuit builders exist for download, but tend to have many features that are beyond the scope of the basic needs of the user. For those in need of a basic circuit design, the extensive GUI's of these programs can be very daunting to those wanting a quick circuit builder without the hassle of having to learn the ins-and-outs of the more advanced programs. We want to develop an intuitive, easy to understand circuit builder that does not complicate the building process with too many features. Simplicity is the driving factor for this software. Figures 1 and 2 show the difference between a typical Design Software GUI, and our proposed GUI.

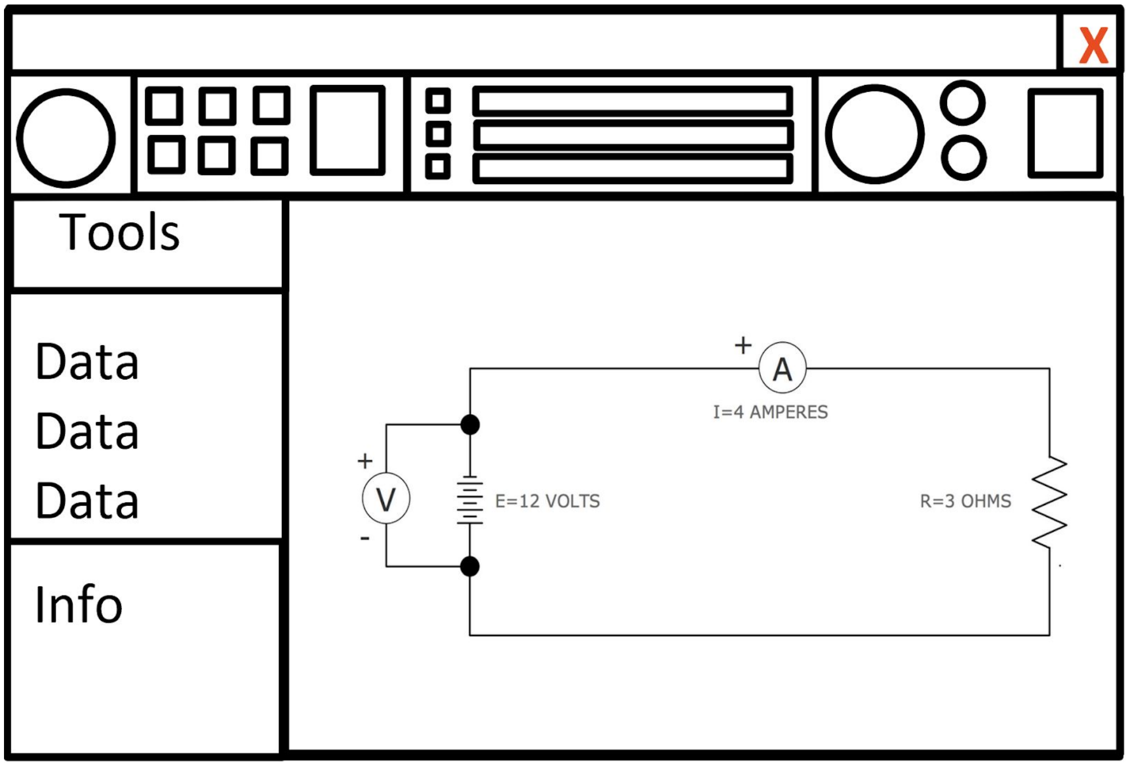

Figure 1

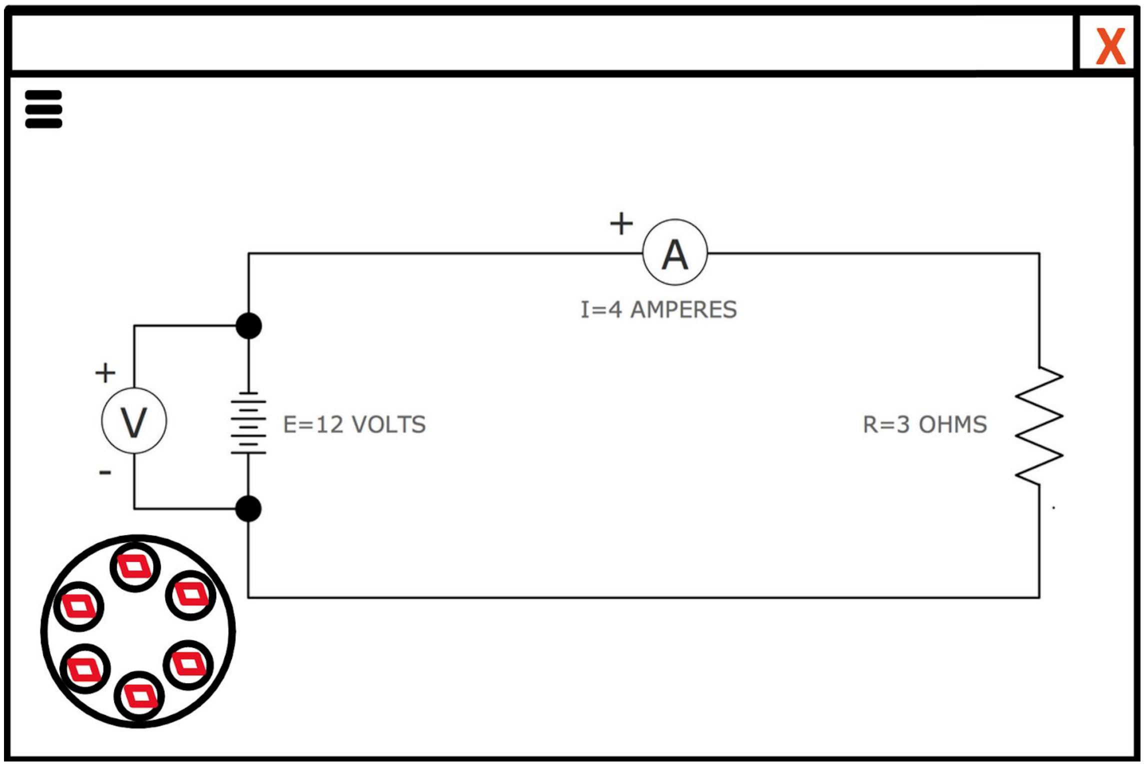

Figure 2

It is clear to see the difference. The software represented by Figure 1 contains a variety of useful features that allow for very fine tuning of the circuit. The tradeoff with this, is the extensive amount of GUI elements complicating the program. This can be a huge barrier for entry for a new user and will inevitably lead to a lot of time being consumed in just learning how to navigate the software, let alone using its plethora of features.

Figure 2 is a general idea of what our software may look like. Notice how the workspace occupies nearly all of the window, and how much less space is occupied by the GUI. Aside from removing many of the niche features in the advanced program, another key component of our simplicity-focused design is the pallet wheel widget. Figure 3 shows a more in depth look at our theorized pallet wheel.

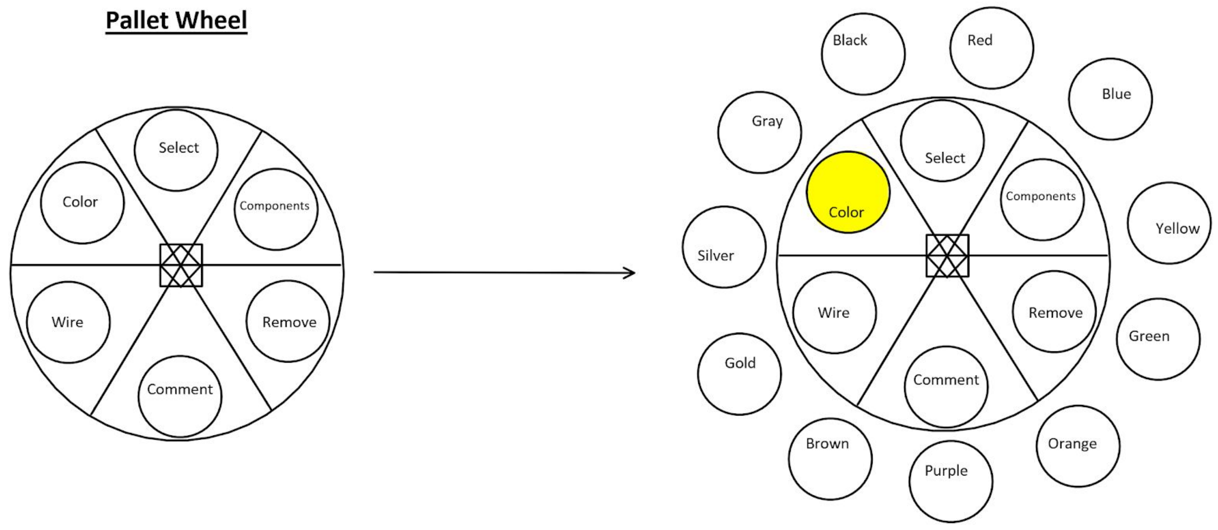

Figure 3

This is the general idea behind the pallet wheel, and what happens when one of the options are selected. A useful feature of the pallet wheel is the ability to move it around the workspace to wherever is most convenient for the user. With this feature, we are able to keep the workspace clean.

In Figure 2, there are three little lines (known as a hamburger menu) in the top left corner. When clicked, these will open up a slide from the left side with file settings, such as save, load, undo, redo, etc.

Optimal PCB Pathfinding:

In order for the program to build an optimal PCB layout, it must be able to place components and connect them optimally. This will be done in two steps, the program will utilize Monte Carlo to place all of the components, and then the program will use a three dimensional pathfinding algorithm to find the optimal paths between components that are connected without intersections. This three dimensional path finding algorithm will be a modified version of a popular algorithm such as Dijkstra's or A*.

The circuit designer will start by creating a PCB layout for a circuit using Monte Carlo to randomly place the components around the board. It will then use the path finding algorithm to connect the components. Each layout will be given a weight to keep track of the best layout. After a layout has been created and analyzed, a new PCB layout will be created and then analyzed. Whichever PCB layout has the lowest weight, or most optimal layout, the program will use that layout. There will be a feature to allow the program to stop searching for optimal layouts, in the event that one cannot be found.

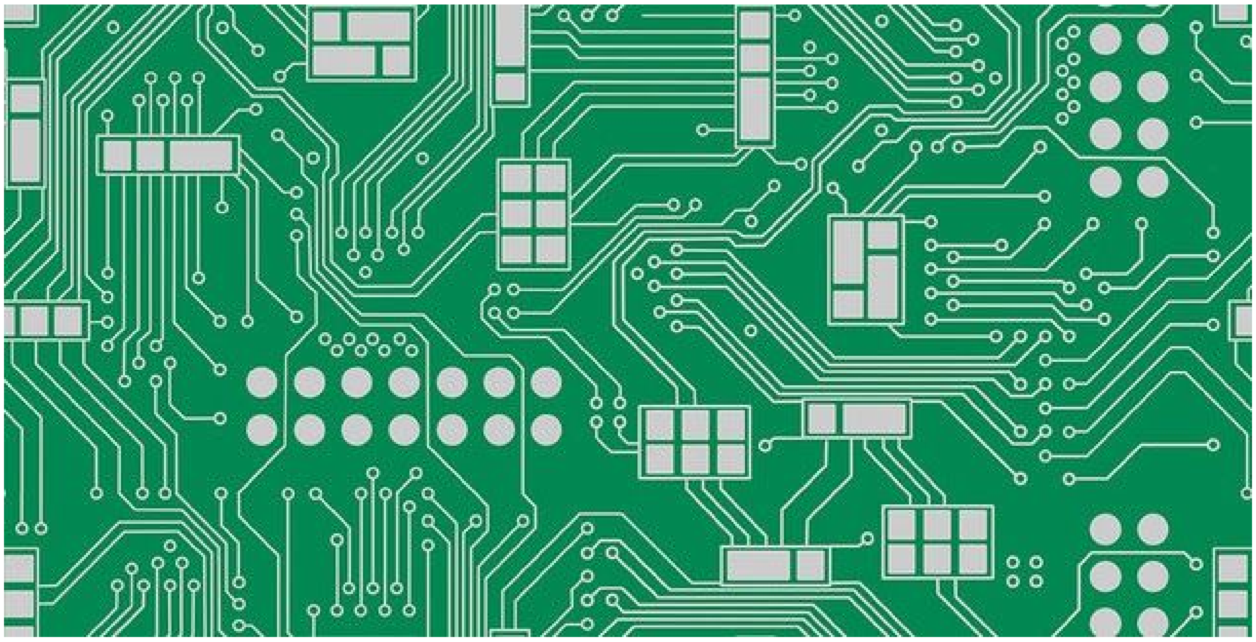
Figure 4
Source: https://www.flickr.com/photos/stallio/3149911976

Diagram-to-Image Exporter:

Based on the solution developed from the pathfinding algorithm, the coordinates which are used in the optimal path can be drawn to an image file such as jpeg or png.

Saving a Circuit Project:

In order to save the users current work, the utilization of a .json file will be used. By default, executable files will be stored in a local directory under the system's

Program Files, while the .json will be located in the system's Document files. Recurring saves will overwrite existing save data.

When exporting the circuit diagram to an image, it will save the image in the same directory that the project's .json files are located.

Feature List:

Completed by May 5th	**Possible by May 5th**	**Not Possible by May 5th**
Circuit schematic builder	Wire/Circuit Checking	Circuit simulator
Convert to PCB diagram	Further optimization of Single-layer PCB Layout	Circuit auto-generation
Full GUI	Pull from external database of parts (Possibly LCSC Electronics database)	Optimizations to signal integrity.
Save state mechanism	PCB layout to .PCB file	Triple-layer PCB
Database of parts (10)	Variable sized traces	
Drag/Drop	Double-layer PCB	
Zoom		
Remove, Add, Wire, New, Load, Save		
Space-optimized Single-layer PCB Layout		

Technology:

- VS Code editor.
- GNU compiler (g++)
- Python 3
- PythonGUI/Tkinter library
- Python/C++ Binding/Integration (Maybe)

Backgrounds:

Kenneth Shipley -

- Basic circuits knowledge including design and analysis.
- Experienced in C++ and Python programming.

- Active user of Unix/Linux, bash command line interfacing, and VS Code.
- Little experience with GUI programming. Previous work done with Java FX and C# .net framework.
- No experience having Python access and use C++ functions.

Joseph Spear -

- Basic circuits knowledge including design and analysis.
- Moderate C/C++ programming in VS and VS Code. Can utilize GNUcompiler as well.
- Moderate usage of Ubuntu command line/ bash programming.
- Moderate Python 2/3 programming.
- Basic understanding of GUI programming.
- No experience with linking two separate languages together.

Jason Rivas -

- Advanced circuit knowledge including design, analysis, and optimization.
- Adept C++ programming in VS Code as well as using the GNUcompiler.
- Adept usage of Ubuntu command line/bash programming.
- Novice Python 3 programming with no experience with GUI design/programming.
- No experience linking Python to/embedding it in a C++ program.

Dependencies, Limitations, or Risks:

Risks:

- Problems with convergence in the Monte Carlo method.
- Lack of knowledge with GUI programming
- Lack of knowledge with interfacing Python with C++ functions

Timeline:

Task:		**Finish By:**
Proposal		Jan 29
GUI	Basic window with menus	Mar 1
	Item list	
	Drag/Drop	
	Wire tool	
	Delete element tool	
	Undo/Redo	
	Circuit element class	

Task:		Finish By:
Component Database	Create element sprites	Mar 15
	Create fake physical dimensions	
	Put database in a file or something	
Save State for Schematic Builder	.JSON prototype	Mar 15
	Python export/import JSON to dict	
	Dict to schematic interpreter	
Save State for PCB Builder	.JSON Prototype	Mar 22
	Dict to schematic interpreter	
Metropolis Monte Carlo	RNG	April 10
	Pathfinding Algorithm	
	Accept/Reject Algorithm	
	Image production for layout	
Integrate the above		May 5