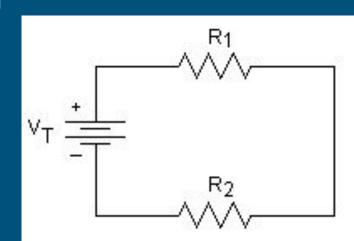
Circuit Board Designer - Final Presentation

Kenneth Shipley, Joseph Spear, and Jason Rivas

Introduction and Purpose

- Today's modern world is founded on the principles of electrical circuits and electronics. Without them, we wouldn't have light bulbs, heating and cooling, computers, and many other luxuries that we have today.
- Developing functional circuits and clean looking diagrams can be a hassle.

 In many cases, there is a huge difference between circuit diagrams and physical PCB layouts, which can cause big problems in converting from one to another.



Software Overview

- This program provides the user with the ability to produce basic circuit diagrams quickly and easily.
- It also provides the user the ability to convert their diagram to a physical PCB layout which is optimized.
 - The program outputs an image of the optimized PCB layout for use in any further electrical application.

Technologies

The technologies included in this program are:

Programs:

- Visual Studio Code Editor
- QT Designer
- Python 3

Libraries:

- PyQT Library
- PySide2 (PyQT fork) Library
- Numpy Library
- XML Library
- JSON Library
- PIL (Python Imaging Library)/Pillow (PIL fork)

Software breakdown

This software is broken up into four main sections:

- 1. Graphical User Interface
- 2. PCB Optimization
- 3. Image Construction
- 4. File Management

Graphical User Interface

Printable Circuit Board

Image Conversion

File Management

Graphical User Interface

- One of the key features of the Circuit Board Designer is the simple yet comprehensive Graphical User Interface.
- The GUI is broken up into three main pages:
 - a. Start Menu / File Menu
 - b. Circuit Building Workspace
 - c. Convert Menu
- All GUI implementation was done using PyQT.

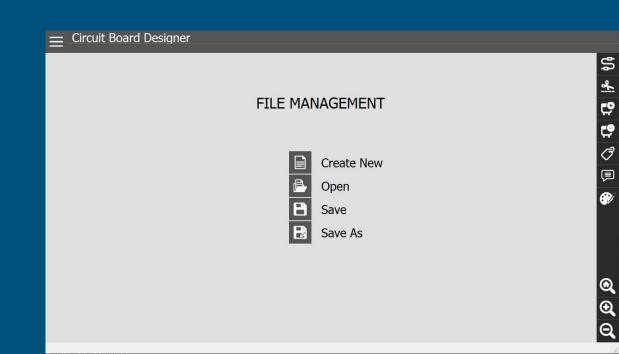


GUI - File Menu

The File Menu page is the initial page which opens when you start the program.

The File Menu includes the following functions:

- Create New
- Open
- Save
- Save As

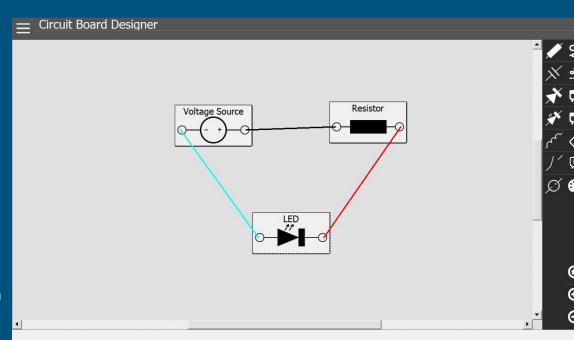


GUI - Circuit Building Workspace

The Circuit Building Workspace is the main GUI page. This is where the actual circuit building takes place.

The functions on the right tab are:

- Wire
- Wire Snipper
- Add Component
 - Subsequent components are shown on a second tab
- Delete Component
- Add Label
- Comment (Not Implemented)
- Color
 - Subsequent Colors are shown on a second tab

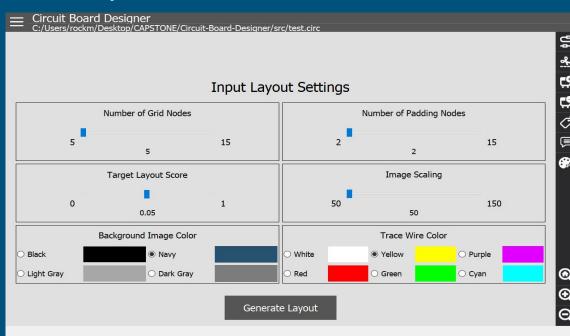


GUI - Convert Menu

 The Convert Menu is where the users input will be able to be converted from a circuit diagram into a physical PCB layout.

The Parameters here are:

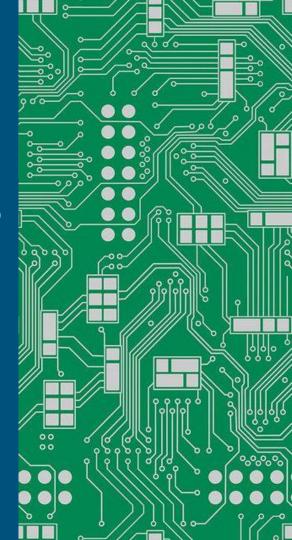
- Number of Grid Nodes
- Number of Padding Nodes
- Target Layout Source
- Image Scaling
- Background Image Color
- Trace Wire Color



PCB Optimization

 The second feature of Circuit Board Designer is converting the circuit diagram input from the user into an optimized physical layout for a PCB.

- This is done by using two common computational techniques:
 - Monte Carlo Algorithm
 - A-Star (A*) Algorithm



PCB Optimization - Monte Carlo

```
paths = []
curr runs score = 0
self.initialize connections list()
best layout = [-1, None]
while (best layout[0] < self.target score) and (i < max iters):</pre>
    self.paths.clear()
    self.pin placement dict.clear()
    self.randomize layout()
    self.initialize pin placement dict()
    paths = self.run a star()
```

PCB Optimization - Monte Carlo

```
i = 0
while paths == [] and j < 4*len(self.connections list):
    np.random.shuffle(self.connections list)
    paths = self.run a star()
    j += 1
if paths == []:
    i += 1
    continue
curr runs score = self.calculate score(paths)
if curr runs score > best layout[0]:
    best layout[0] = curr runs score
    best layout[1] = paths
i += 1
self.paths = best layout[1]
```

PCB Optimization - A*

```
CLOSED
add the start node to OPEN
loop
  current = node in OPEN with the lowest f cost
  remove current from OPEN
  add current to CLOSED
  if (current is the target node)
  return
 for each neighbour of the current node
 if (neighbour is not traversable) OR (neighbour is in CLOSED)
   skip to the next neighbour
   if (new path to neighbour is shorter) OR (neighbour is not in OPEN)
    set f_cost of neighbour
     set parent of neighbour to current
    if (neighbour is not in OPEN)
      add neighbour to OPEN
```

PCB Optimization - A*

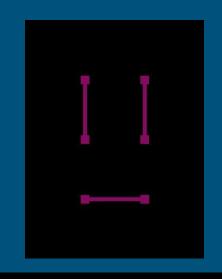
```
while (best_layout[0] < self.target_score) and (i < max_iters):
    self.paths.clear()
    self.pin_placement_dict.clear()

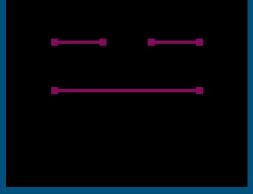
self.randomize_layout()
    self.initialize_pin_placement_dict()
    paths = self.run_a_star()</pre>
```

Image Construction

The final output of the Circuit Board Diagram is a set PCB Layout color coded however the user decides based on the Convert Menu options.

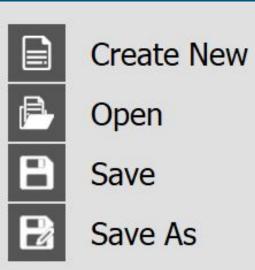
To the right are two different outputs from two circuits where there are three components each.





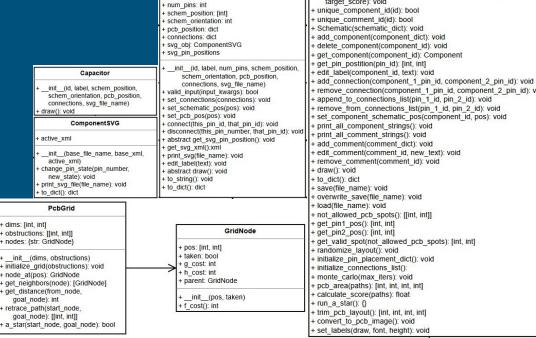
File Management

- PyQt offers its own way to save and open files.
- Save, Save As, and Open all call the backend's save and load methods as well.
- Create New deletes the schematic instance and creates a new one.
- .circ



Backend Class Diagram

Backend Class Diagram Link



+ id: int

+ label: str



Schematic + COMPONENT CLASSES: {str: ComponentClass} + components: {str: Component} + comments: {str: Comments}

Component

+ converted image color mode: str + converted image scaling; int + converted image trace color; (int, int, int) + converted image: PIL.Image

+ paths: [{}] + curr runs score: int + pin placement dict: {str:[int,int]}

+ connections list = [str, str] + area weight: float + path length weight: float + n grid spaces: int + a star grid padding; int + target score: float

+ set monte carlo parameters(n grid spaces, a star grid padding, target score); void + unique component id(id); bool + unique comment id(id); bool

+ converted image bg color; (int.int.int)

+ Schematic(schematic dict): void + add component(component dict): void + delete component(component id): void + get component(component id): Component + get pin postition(pin id): [int, int]

+ remove connection(component 1 pin id, component 2 pin id); void + append to connections list(pin 1 id, pin 2 id): void + remove from connections list(pin 1 id. pin 2 id); void + set component schematic pos(component id, pos): void + print all component strings(): void

+ print all comment strings(): void + add comment(comment dict): void + edit comment(comment id, new text): void + remove comment(comment id); void + draw(): void

+ overwrite save(file name): void + load(file_name): void + not allowed pcb spots(): [[int, int]]

+ get pin1 pos(); [int. int] + get pin2 pos(): [int, int] + get valid spot(not allowed pcb spots); [int. int] + randomize layout(): void

+ monte carlo(max iters): void + pcb_area(paths): [int. int. int. int]

+ calculate score(paths); float + run a star(): {}

+ trim pcb layout(); [int. int. int. int] + convert to pcb image(); void

GUI Class Diagram

Class Diagram

Component(QGraphicsRectItem)

- + scene: QGraphicsScene
- + name: str
- + compType: str
- + id: int
- + boundingBox: QGraphicsRectItem
- + widget: Widget
- + sceneWidget: QGraphicsProxyWidget
- + schematicArgs: {str:int}
- + pin0Connection: [QGraphicsLine]
- + pin1Connection: [QGraphicsLine
- + __init__(scene, pen, compType, name, id)

Widget(QWidget)

- + ui: QForm
- + id: int
- + boundingBox: QGraphicsRectItem
- + scene: QGraphicsScene
- + __init__(compType, name, scene, boundingBox, id, parent = None)
- + addLeftButtonPos()
- + addRightButtonPos()
- + setImage(compType)

MainWindow (QMainWindow)

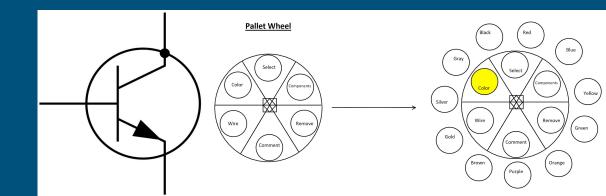
- + ui: QMainWindow
- + zoom: int
- + scene: QGraphicsScene
- + ids
- + outlineColor: QPen
- + penColor: QPen
- + penColors: QColor
- + file: str
- + components: [Component]
- + connections: [QGraphicsLine]
- + schematic: Schematic
- + backgroundGroup: QButtonGroup
- + traceGroup: QButtonGroup + grid: int
- + padding: int
- + target: int
- + scaling:
- + background: RGB(int, int, int)
- + trace: RGB(int, int, int)
- + __init__()
- + updatePositions(): void+ printSchematic(): void
- + addComponent(component, name, posX, posY, compld, loading): void
- + loadComponent(compType, compld, name, posX, posY): void
- + deleteComponent(): void
- + addConnection(comp0ld, comp1ld): void
- + removeConnection(): void
- + changeLabel(): void
- + zoomln(): void
- + zoomOut(): void
- + zoomHome(): void
- + changePenColor(color): void
- + clearSchematic(): void
- + fileOpen(): void
- + fileCreate(): void
- + fileSave(): void
- + fileSaveAs(): void
- + toggleManu(maxWidth, enable): void + toggleTools(maxWidth, enable): void
- + updateSliderValue(label, slider, target): void
- + setupMonteCarlo(): void
- + savelmage(): void

Demonstration

Future Works

Due to time constraints, some of the more complex features had to be omitted from version 1.0. In the event that version 2.0 was to be created, the following features would be implemented:

- Pallet Wheel
- Transistors
- Comments
- Follower Wires



Questions?