EFFECTIVE ASPECTS OF SETTING UP CIRCUITS IN MATPOTLIB IN PYTHON

Zarif Nishonov
Student of Tashkent University of Information Technologies Karshi branch

Aslbek Utkirov
Student of Tashkent state university economics

Abstract: *In this article, the opinions of domestic and foreign scientists about the effective aspects of installing schemes in Python in matpotlib are mentioned.*

Keywords: *CircuitPython, SchemDraw, Skidl, 2D plots, electronic design automation (EDA), circuit design and representation, matplotlib plotting.*

Introduction.

It appears there may have been a typo in your query. If you meant to ask about installing libraries for plotting circuits in Matplotlib in Python, it's important to note that Matplotlib is primarily focused on creating 2D plots, such as line plots, scatter plots, bar plots, etc., and may not be the most suitable library for plotting circuit diagrams.

For plotting circuit diagrams in Python, you may want to consider using specialized libraries or tools specifically designed for this purpose, such as:

1. `CircuitPython`: CircuitPython is a programming language based on Python developed for microcontrollers and circuitry projects. It provides a simple way to interact with electronic components and create circuits.[1]
2. `SchemDraw`: SchemDraw is a Python library for drawing electronic circuit schematics. It allows you to create circuit diagrams with customizable components and connections.
3. `Skidl`: Skidl is a Python-based library for creating electronic schematics and circuit board layouts. It can be used to define circuits in code and export them to popular electronic design automation (EDA) tools.[2]

If you still wish to use Matplotlib for plotting circuit diagrams, you may need to create custom solutions to represent the circuit components and connections as graphical elements in the plot. This may involve drawing lines, shapes, and annotations to simulate circuit elements. However, specialized libraries like the ones mentioned above are specifically designed for this purpose and may provide more efficient and convenient tools for plotting circuits.

You can't directly "install circuits" in matplotlib. Matplotlib is a plotting library, not a circuit simulation tool. However, you can create visual representations of circuits using matplotlib's plotting capabilities.[3]

[1] "Python for Scientists and Engineers" by Matt Harrison: A book that covers Python for scientific computing, including circuit simulations.
[2] https://schemdraw.readthedocs.io/en/latest/
[3] "Analog and Digital Electronics with Python" by Steven T. Karris: A useful book for understanding how to simulate electronics with Python.

Here's how to approach this:

1. Circuit Design and Representation

Data Structure: Define your circuit using a data structure. This could be:

Dictionary: Keys for components (e.g., "resistor1", "capacitor2"), values for component data (e.g., resistance, capacitance).

Object-Oriented: Create classes for circuit components (resistor, capacitor, etc.) and store them in a list or other container.

Visual Mapping: Decide how you want to map circuit elements to matplotlib plot elements. Common choices are:

Lines: Represent wires as lines.

Shapes: Use shapes (rectangles, circles) for components (e.g., resistor shape for a resistor).

Text: Add text labels for component names and values.[4]

2. Matplotlib Plotting

Basic Structure:

Figure and Axes: Create a figure and axes using plt.figure() and plt.axes().

Coordinate System: Decide on a coordinate system for your circuit (e.g., x-y plane, grid).

Drawing Components:

Lines: Use plt.plot() to draw wires.

Shapes: Use plt.plot() with markers or plt.fill() to draw components.

Text: Use plt.text() to add labels.

Customization:

Colors: Use color parameter in plotting functions.

Styles: Use linestyle for wires, marker for shapes.

Annotations: Add arrows, text boxes for clarity.

Example (Basic Circuit)

```
import matplotlib.pyplot as plt

# Circuit components

components = {

 "resistor1": {"type": "resistor", "value": 100, "location": (1, 1)},

 "capacitor1": {"type": "capacitor", "value": 10, "location": (2, 1)}

}

# Plotting

fig, ax = plt.subplots()

# Draw wires

ax.plot([0, 1], [1, 1], color='black')
```

[4] https://pyspice.fabrice-salvaire.fr/pages/documentation.html

```
ax.plot([1, 2], [1, 1], color='black')
# Draw resistor
ax.plot([1, 1], [0.8, 1.2], color='brown', linewidth=3)
ax.text(1, 1.3, "R1 (100 ohms)", ha='center')
# Draw capacitor
ax.plot([2, 2], [0.8, 1.2], color='red', linewidth=3)
ax.plot([1.8, 2.2], [0.8, 0.8], color='black', linewidth=2)
ax.plot([1.8, 2.2], [1.2, 1.2], color='black', linewidth=2)
ax.text(2, 1.3, "C1 (10 F)", ha='center')
plt.show()
```
[5]

Advanced Techniques:

Object-Oriented Design: Create classes for circuit components that handle their drawing and behavior.

Interactive Features: Use event handling to allow users to modify the circuit (move components, change values).

Circuit Simulation: Use external libraries like NumPy or SciPy to simulate circuit behavior and update the plot based on the simulation results.

List of used literatures:

1. https://matplotlib.org/stable/index.html
2. https://schemdraw.readthedocs.io/en/latest/
3. https://pyspice.fabrice-salvaire.fr/pages/documentation.html
4. "Python for Scientists and Engineers" by Matt Harrison: A book that covers Python for scientific computing, including circuit simulations.
5. "Analog and Digital Electronics with Python" by Steven T. Karris: A useful book for understanding how to simulate electronics with Python.

[5] https://matplotlib.org/stable/index.html