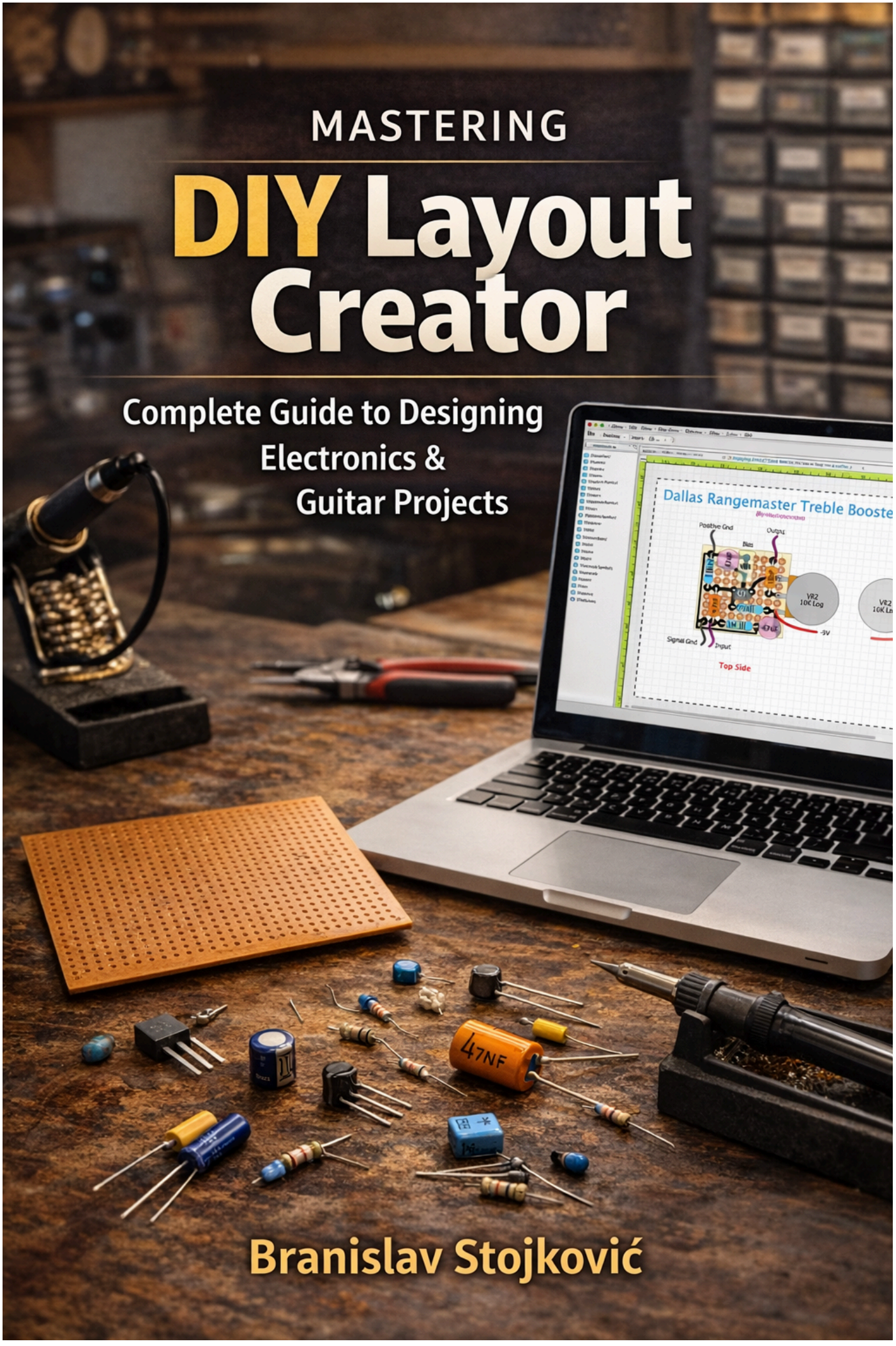


eBook v1.0 - August 2026 - DIYLC v6.0.0

Table of Contents

Introduction

DIY Layout Creator was built for people who love to build. Whether you are designing your first guitar pedal, refining a tube amplifier layout, drafting a PCB, or experimenting with a new wiring idea, this book is written for you. It is intended for electronics enthusiasts, hobbyists, pedal and amp builders, and anyone who enjoys turning ideas into working hardware.

This is not a textbook on electronics theory, nor is it a generic software manual. Instead, it is a practical guide to using DIY Layout Creator as a design tool — from your first component placement to exporting fabrication-ready files.

This book is especially useful if you are:

- A guitarist building pedals or modifying instruments
- An amp builder designing point-to-point or turret layouts
- A DIY enthusiast working with perfboard or vero board
- A maker exploring PCB design for the first time
- A HAM radio or vintage electronics enthusiast
- A curious learner wanting to visualize circuits clearly
- A Java developer who wants to understand how DIYLC works internally and how to extend it

By the end of this book, you will know how to navigate the DIYLC interface efficiently, configure projects properly from the start, place and edit components, design PCB and vero board layouts, create guitar wiring diagrams, generate BOMs and fabrication-ready files, and use AI-powered tools to validate and improve your designs.

The chapters follow a logical progression — beginning with installation and interface fundamentals, moving into core workflow concepts, then expanding into practical real-world design examples, documentation techniques, cloud collaboration features, and AI-assisted analysis tools. You can read the book sequentially or jump directly to the sections most relevant to your project.

DIYLC is intentionally focused and approachable. It does not attempt to replace professional electronic design automation suites or simulation environments. Instead, it prioritizes clarity, speed, and practical usability for real builders working on real projects.

DIY electronics is as much about learning and sharing as it is about soldering and schematics. This book reflects that mindset. Its purpose is to make the design process clearer, faster, and more enjoyable — so you can focus on building.

Let’s get started.

Chapter 1 – Getting Started with DIY Layout Creator

1.1 Welcome to DIY Layout Creator

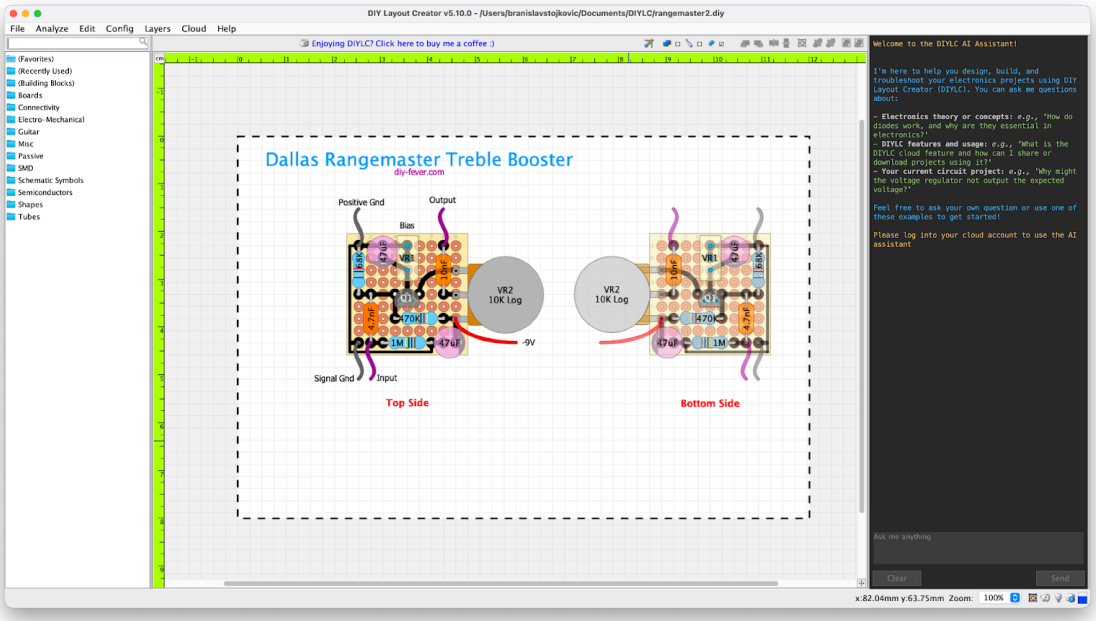
DIY Layout Creator, commonly known as DIYLC, is a lightweight and intuitive layout tool created specifically for DIY electronics enthusiasts.

Whether you are building your first guitar pedal, experimenting with circuit designs, or refining complex gear projects, DIYLC provides a focused environment for planning, visualizing, and documenting electronic layouts.

DIYLC was built by a self-taught electronics enthusiast — for fellow makers, musicians, and tinkerers. Its goal is to make layout design accessible, fast, and practical for real-world DIY use.

Unlike heavy professional engineering software, DIYLC focuses on the kinds of layouts and workflows commonly used in home workshops:

- Stripboard and perfboard circuits
- Printed circuit boards (PCBs)
- Various pre-fabricated proto boards
- Point-to-point wiring
- Guitar wiring diagrams
- Schematics and documentation



Throughout this ebook, you will learn how to:

- Set up and configure DIYLC
- Place and edit components

- Use layers, building blocks, and variants to speed up your workflow
- Draw clean schematics and wiring diagrams
- Design various board types, including PCBs and point-to-point layouts
- Use advanced features such as DIYLC Cloud and built-in AI tools
- Debug and analyze layouts to catch problems before soldering

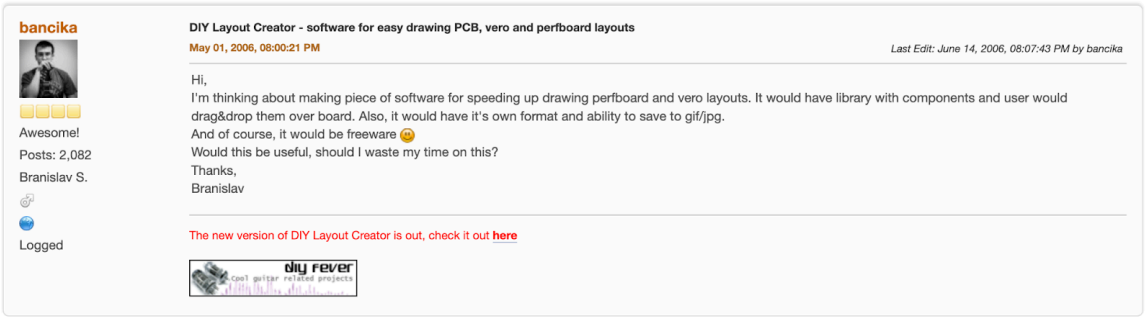
DIYLC is designed to give you confidence — whether you are documenting a simple fuzz pedal or mapping out an entire amplifier.

1.2 The History of DIY Layout Creator

Understanding where DIYLC came from helps explain why it focuses so strongly on practical DIY workflows.

The Origins

DIY Layout Creator began in 2006 with a simple idea shared on the DIY Stompboxes forum:



At the time, most builders relied on generic tools such as Paint or Visio — none of which were designed for electronics layout work. Many of the pedal layout diagrams were tediously drawn by copy-pasting individual components from a hand-drawn template that was floating around. Drawing a visually compelling perfboard or stripboard layout was a painful process and modifying it beyond that point was even more painful.

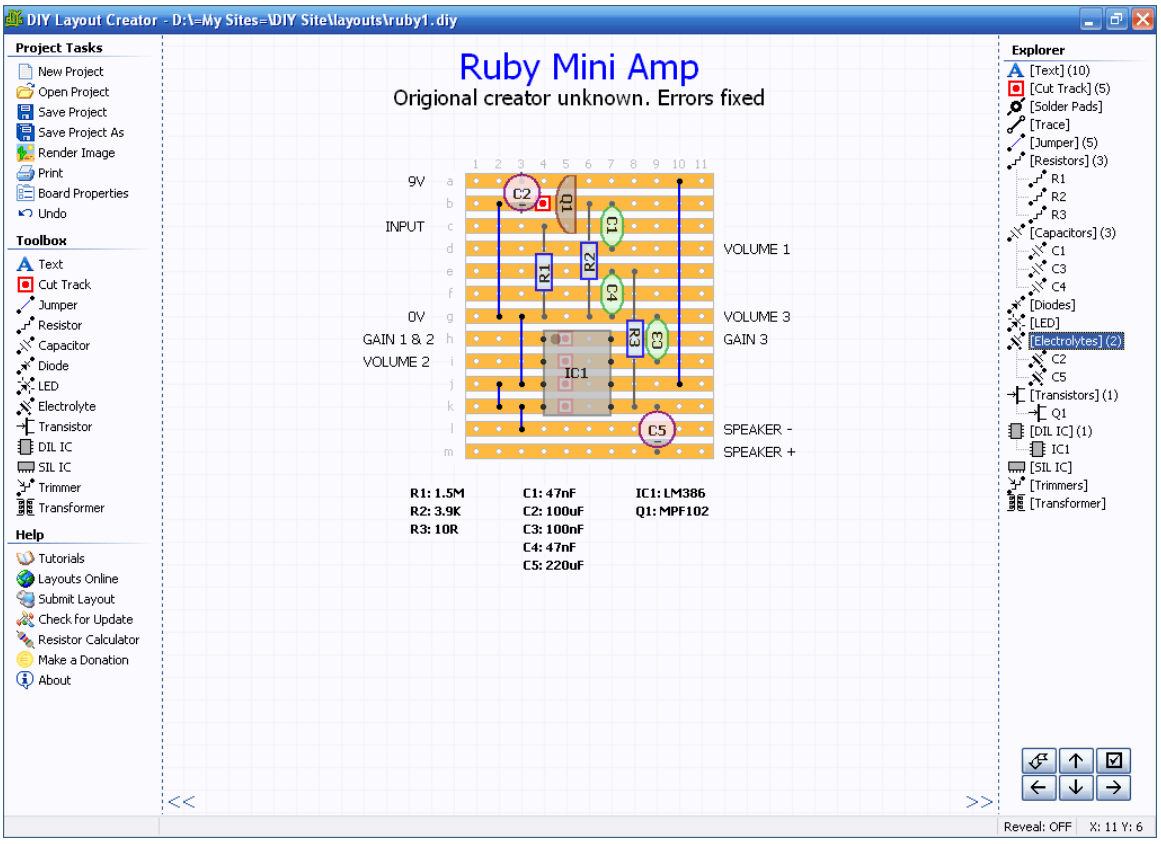
There were plenty of EDA options available but none were really geared towards DIY enthusiasts and types of projects typically built at home.

The idea to develop a “from-builder-to-builders” piece of software quickly gained interest, the community provided enough material to draft a roadmap and the first version of DIYLC was created soon after.

Version 1 – Built for Builders (2006)

The first release was written in Delphi and ran only on Windows. The initial scope of the app was strictly limited to circuit boards typically used in home-made guitar pedals.

As such, it had a limited component library and basic support for vero board, perfboard, and simple PCBs with no off-board components or chassis wiring support.



Despite its simplicity, it was still revolutionary for the DIY community. For the first time, builders had a tool tailored to their actual needs. It quickly gained traction and pushed out generic graphics software for pedal circuit board design.

At the same time it was very limiting both in terms of the platform it ran on but also in terms of circuits it could visualize. Also, the functionality was quite limited and the UX was not as comfortable as the modern iteration of the app, but still miles better than drawing everything manually.

Version 2 – Cross-Platform and Expandable (2008)

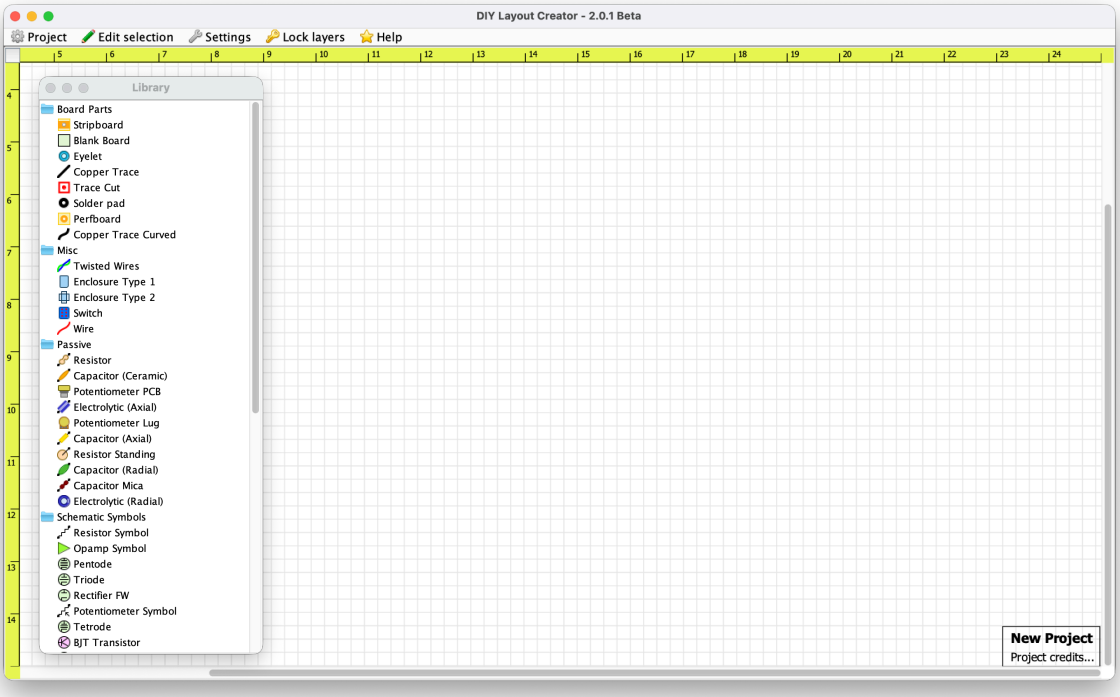
Version 2 introduced major improvements:

- Written in Java from scratch
- A new XML-based component format

That meant that DIYLC could run on any machine capable of running Java. In addition to this, the flexible component format meant that the library could be virtually endless and users could add their own components combining basic shapes.

Given that flexibility, the app began expanding into:

- Schematic symbols
- Off-board and chassis diagram design



The basic premise of the new version was fine, but my Java skills at the time were not great, so the code was architected in a messy way that resulted in not-so-great performance and limited extendability to support feature extensions that were to follow. Furthermore, the choice to define components as collections of primitive drawing elements also limited the usability of components to purely visual elements without a lot of room for customization of their behavior or logic.

Another major limitation was that projects created in Version 1 could not be opened in Version 2, so people who had already gotten used to Version 1 and had a library of projects were not incentivised to upgrade.

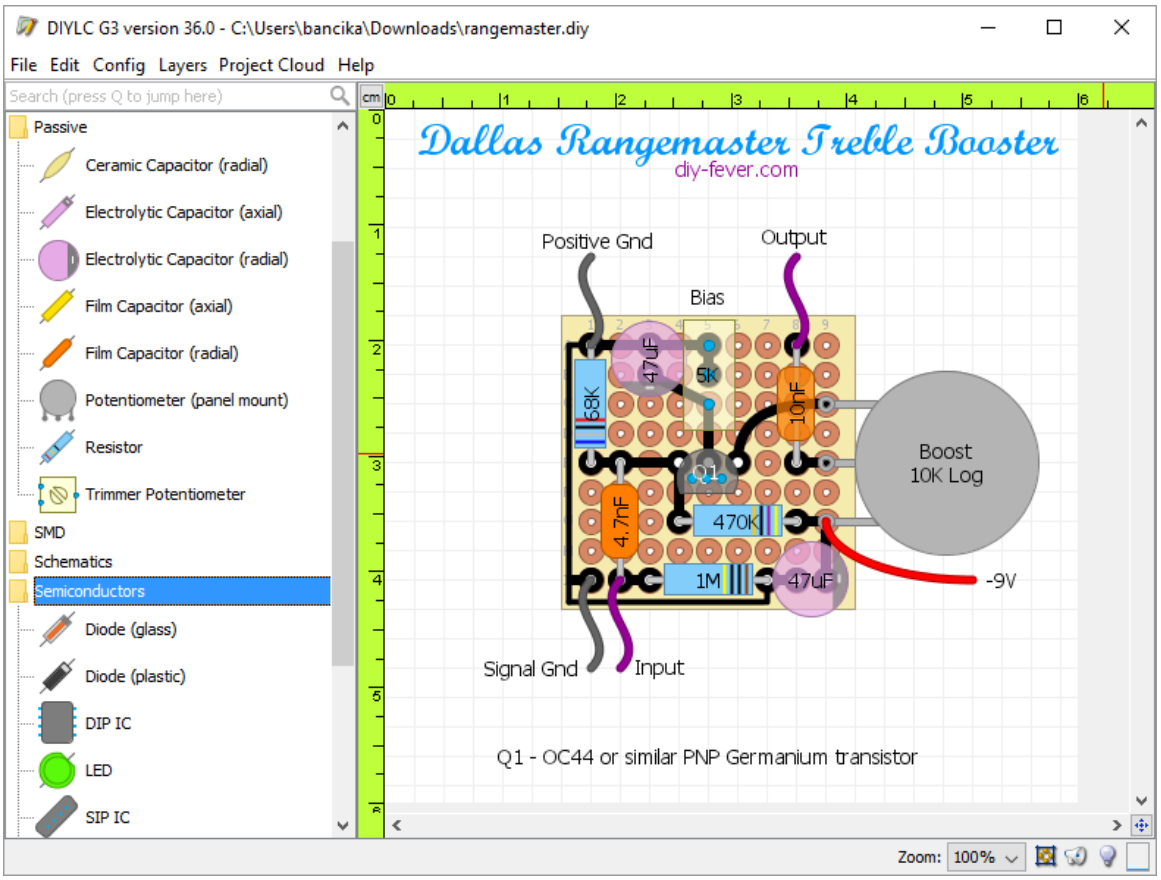
Ultimately, version 2 was abandoned and DIYLC was completely rewritten from scratch with all the lessons learned from versions 1 and 2 and with fresh ideas from the growing community.

Version 3 – Open Source and Rebuilt from Scratch (2011)

Version 3 marked a major milestone.

- Completely rewritten in Java
- Made open-source
- Able to load projects from both earlier versions
- Totally redesigned user experience which is still being used
- Implements component-as-code approach

This was when DIYLC truly became a community-driven project.



Being given a freedom to rewrite the application from scratch allowed for a much better internal architecture which is now much more extensible and better performing. DIYLC Version 3 was envisioned as a shell providing some basic functionality and having no components. Each of the components and each functionality has its own codebase that communicates with the shell to provide a unified user experience. This architecture is still used to this day with some minor evolutionary changes.

The component library expanded significantly to include:

- Guitar wiring parts such as pickups, blade switches, and jacks
- Vacuum tubes and sockets
- Early schematic representation support

Components-as-code approach also made it possible to break the paradigm of components being dumb visual elements and opened up possibilities of custom behaviors of components and ultimately making the application into something more.

This version also introduced DIYLC Cloud, enabling layout sharing and collaboration. DIYLC was no longer just a drawing tool — it became a creative hub for DIY builders.

There were some drawbacks though, mainly in the project setup itself. The project used ANT as a means to build the tool and Eclipse IDE and both of these were already at the sunset of their existence, making code contributions very rare due to the steep learning curve and initial project setup hassle.

Version 4 – Smarter and More Capable (2020)

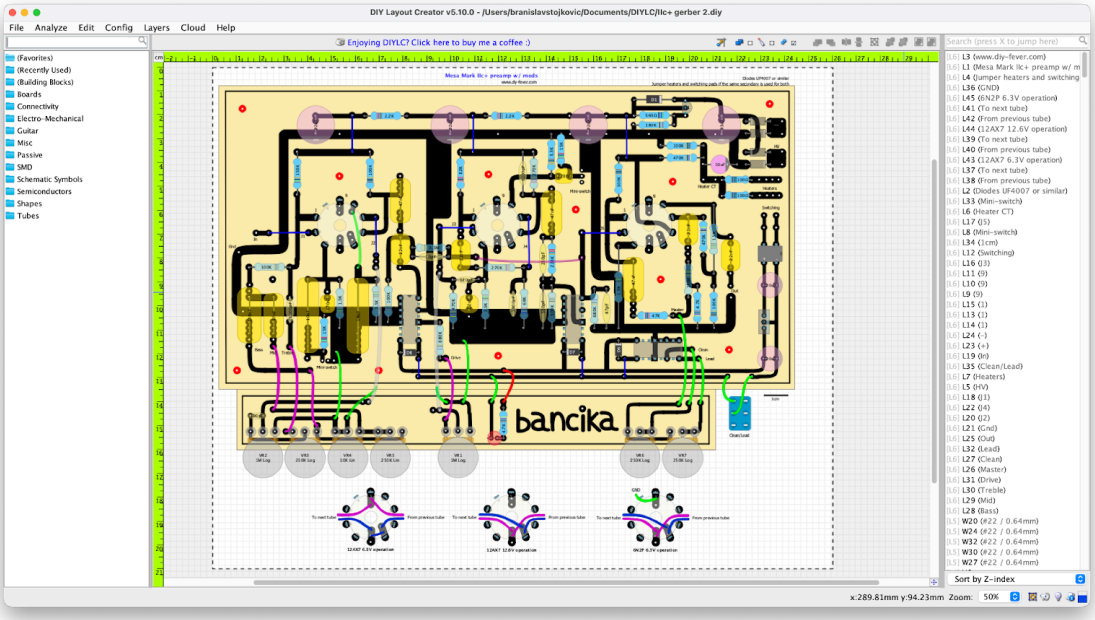
Version 4 builds on top of the same codebase and architecture, but modernizes the core system by upgrading to Java 8.

More importantly, it also introduced the first real circuit analysis and debugging tools, shifting DIYLC from a purely visual editor to a design-aware assistant that could help users catch problems before building.

Version 5 – AI-Powered (2025)

Version 5 brought the largest transformation so far:

- Upgrade to Java 17
- Adoption of Maven for easier development and maintenance
- Introduction of AI features: AI Assistant and AI Analyzer
- Notarized macOS builds for smoother installation



The component library continues to grow with nearly every release, supporting advanced schematics, complex tube designs, and detailed guitar wiring setups.

Version 6 – Modern and AI-Powered (2026)

Version 6 marks a significant milestone in the software's lifecycle, focusing on Java modernization to ensure long-term stability and performance. A core part of this update is the migration of the underlying Java engine from version 17 to version 21. By adopting Java 21, the latest Long-Term Support (LTS) release, DIYLC benefits from enhanced runtime efficiency, improved memory management, and access to modern language features that streamline further development. This upgrade not only keeps the application compatible with modern operating systems but also provides a more robust foundation for the AI-powered design tools introduced in recent versions.

A One-Man Project Powered by the Community

Although still developed and maintained primarily by one person, DIYLC has always been shaped by its community.

From shared layouts and feature suggestions to code contributions, DIYLC remains deeply rooted in the DIY spirit. We are particularly grateful for the many contributors who submit smaller code improvements—whether it's fixing a pesky bug or adding support for a new component—as these incremental additions are highly valued and essential for keeping DIYLC robust, up-to-date, and aligned with the evolving needs of the maker community.

It is open-source, constantly evolving, and driven by real-world builder needs.

1.3 What DIYLC Does — and What It Doesn't Do

DIYLC vs Professional EDA Tools

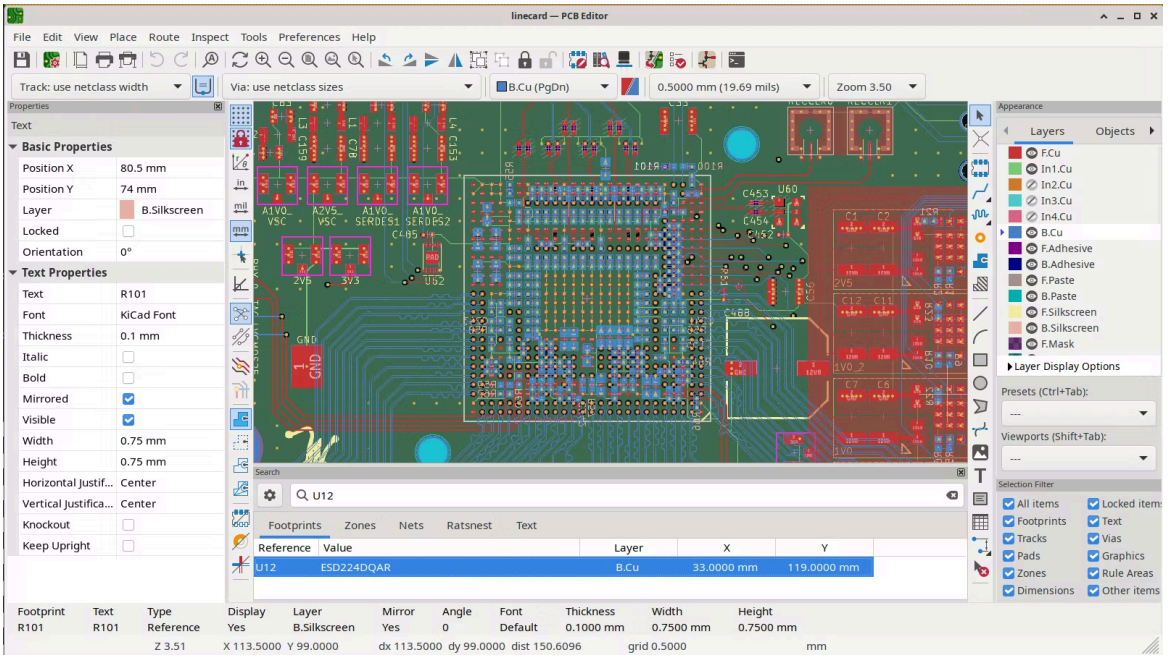
DIYLC is intentionally designed to serve a specific audience and workflow.

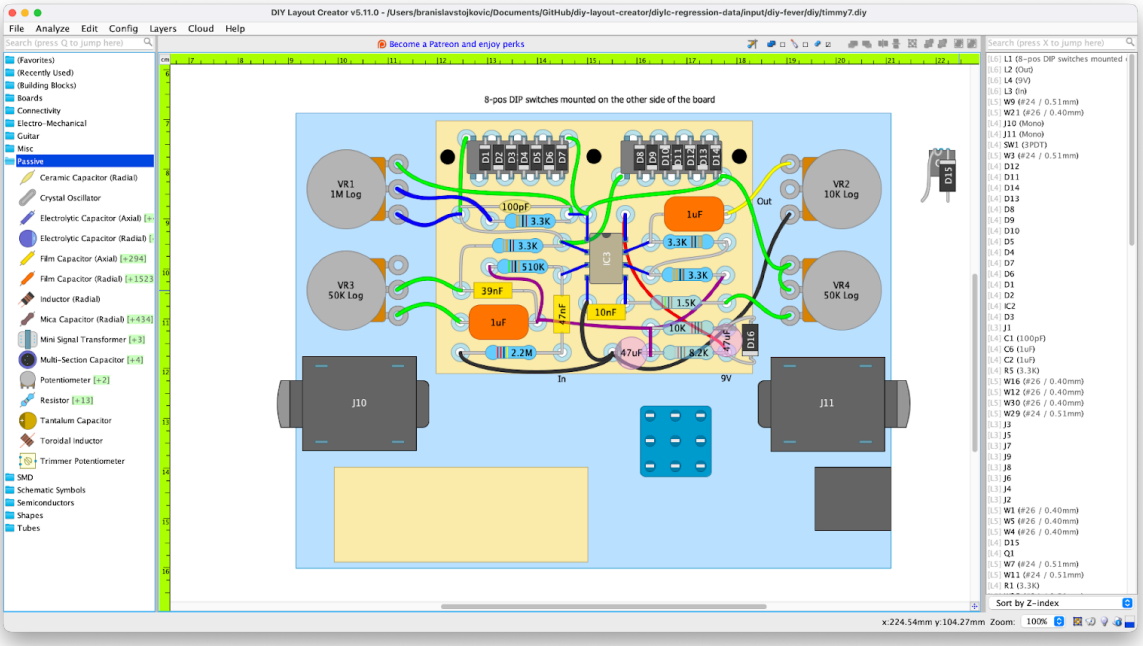
Unlike professional EDA tools such as KiCAD or Eagle, DIYLC focuses on a friendly learning curve, a clean interface, and fast layout creation.

It does not attempt to replicate complex engineering workflows or heavy simulation systems.

Instead, it is optimized for hobbyists and DIY builders.

DIYLC vs KiCAD - UI Comparison





What DIYLC Can Do

DIYLC supports a wide range of layout design and documentation tasks, including:

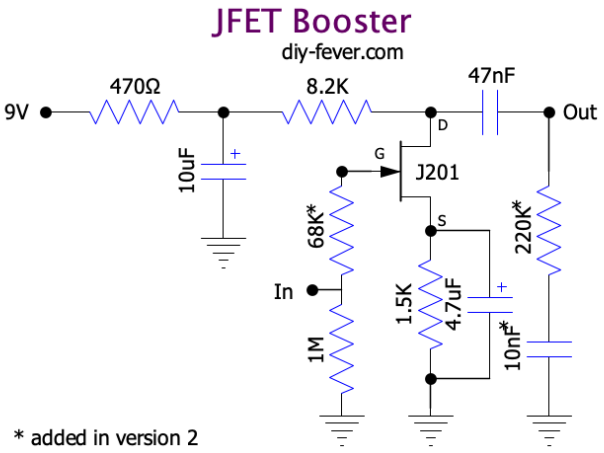
- Point-to-point layouts
- Veroboard and perfboard circuits
- PCB design
- Schematics
- Guitar wiring diagrams
- Once your design is complete, you can export it as:
- High-quality images (PNG)
- PDFs
- Gerber files ready for fabrication

DIYLC also supports netlist generation and a basic Bill of Materials (BOM).

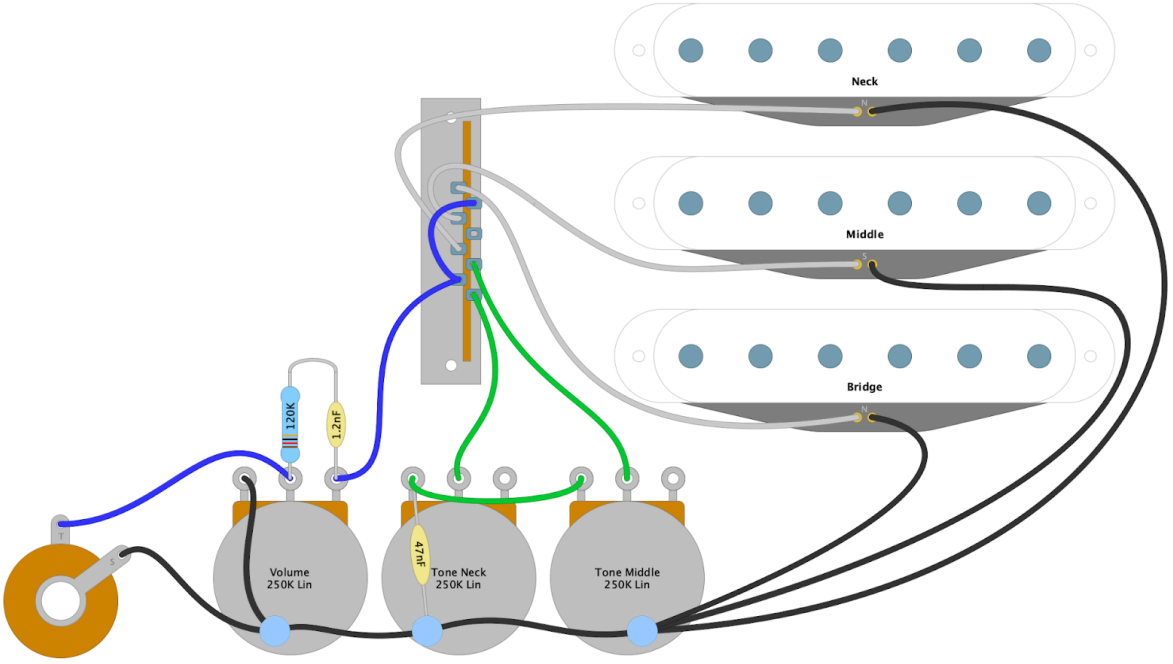
A unique feature is the ability to design custom control faceplates for amplifiers, pedals, and other gear. Using the same placement and labeling tools, you can create aligned, production-ready front panels.

Sample Project Exports

Solid state circuit schematic diagram

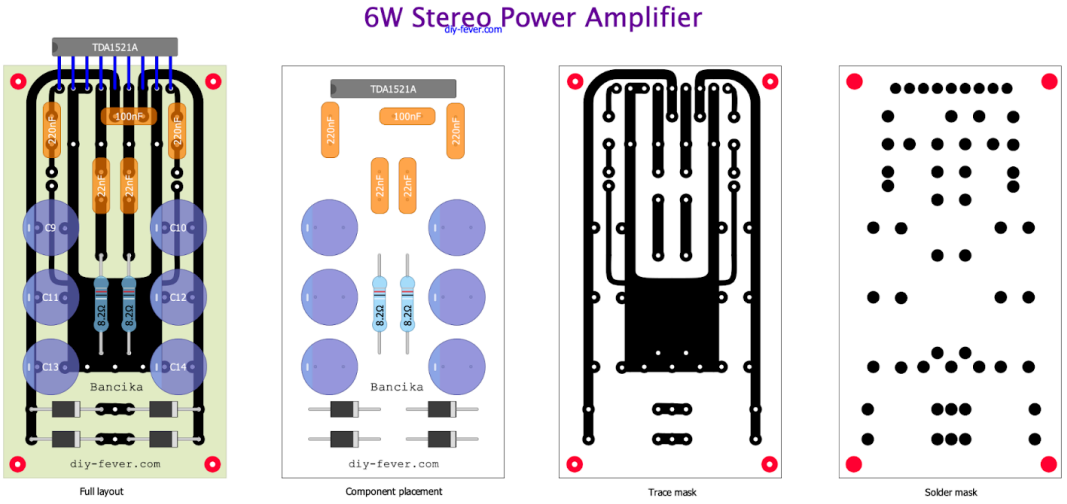


Guitar wiring diagram

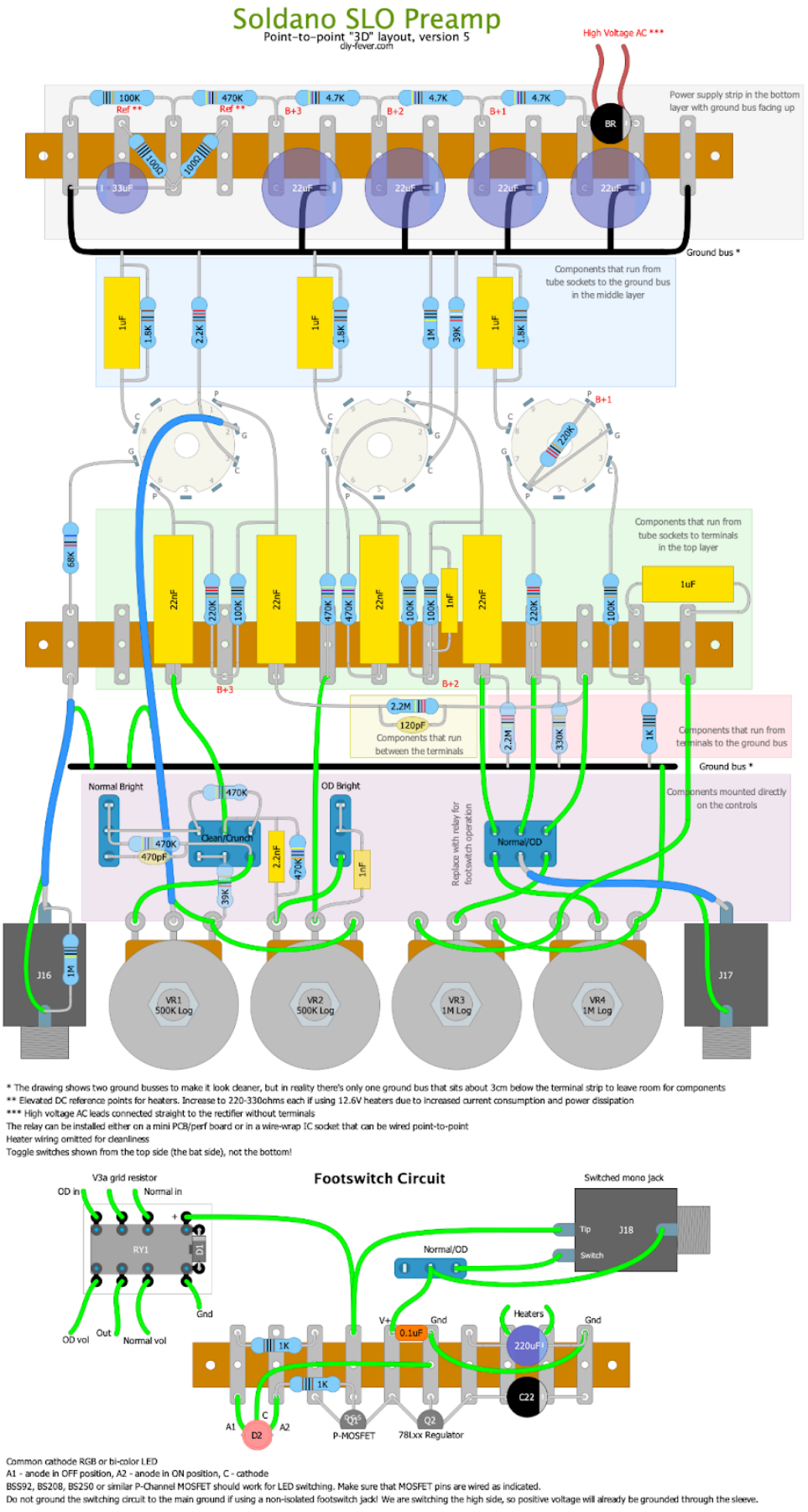


Amplifier faceplate

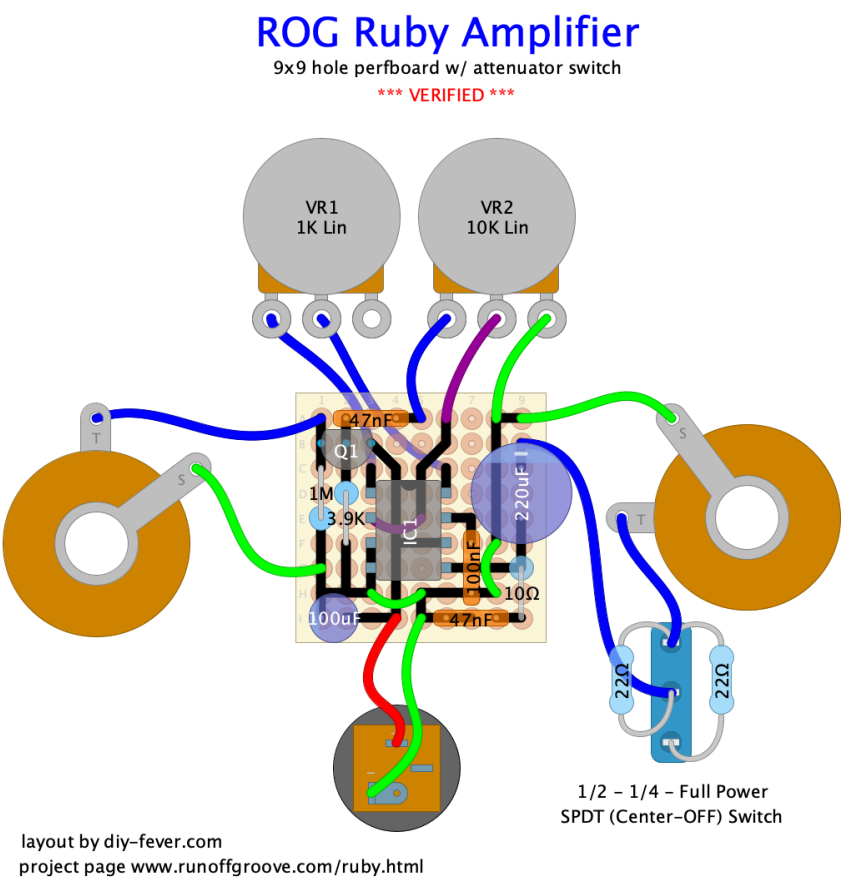
PCB circuit layout



Point-to-point tube circuit layout



Perfboard circuit layout with offboard components

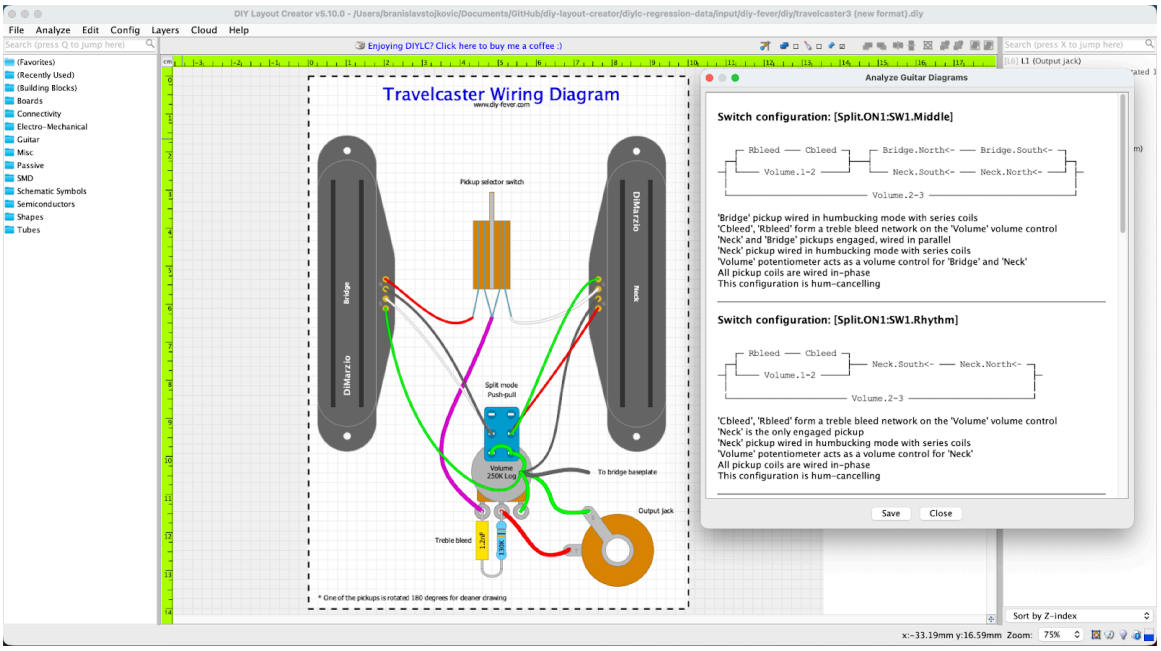


Exported PCB design to gerber format

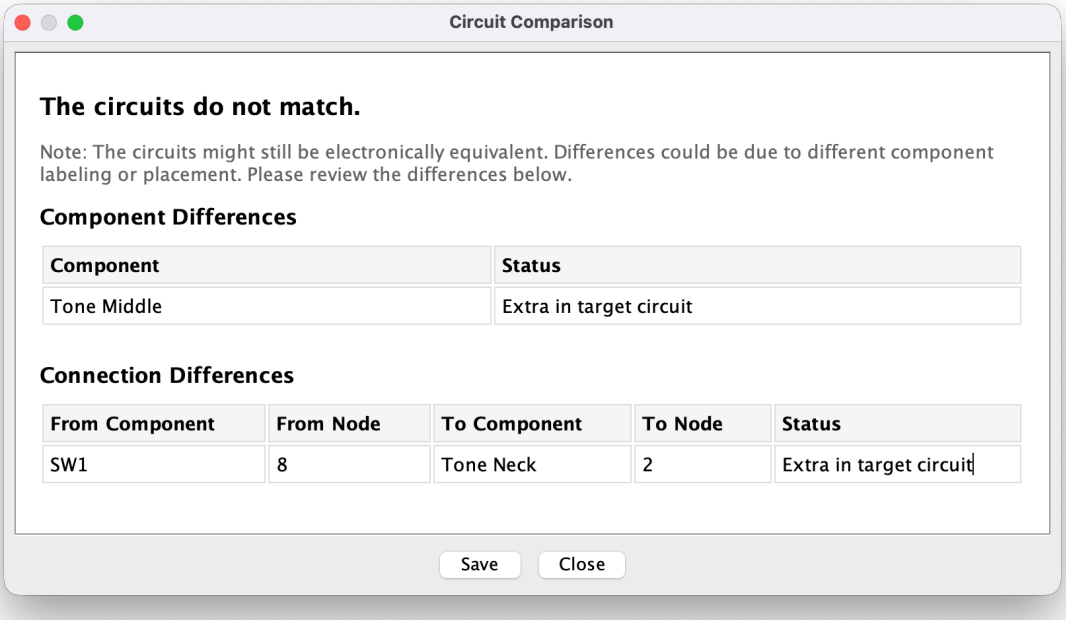
Powerful Built-In Tools

DIYLC includes several advanced tools to help analyze and debug designs:

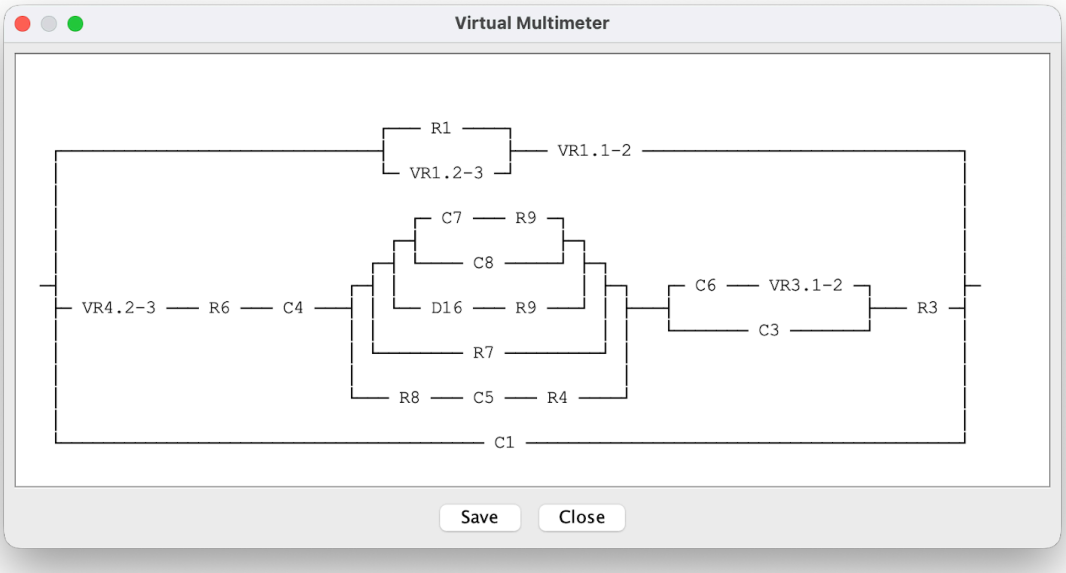
Guitar Wiring Analyzer: checks all switch combinations, coil split states, phase relationships, and hum-canceling configurations. Outputs a comprehensive report listing how the guitar will behave in each of the switch combinations.



File Comparison tool: allows you to verify that two diagrams are electronically equivalent — useful when comparing layouts and schematics or when comparing between two revisions of the same diagram.



Virtual Multimeter: traces all electrical paths between two nodes and visualizes them as series/parallel graphs.

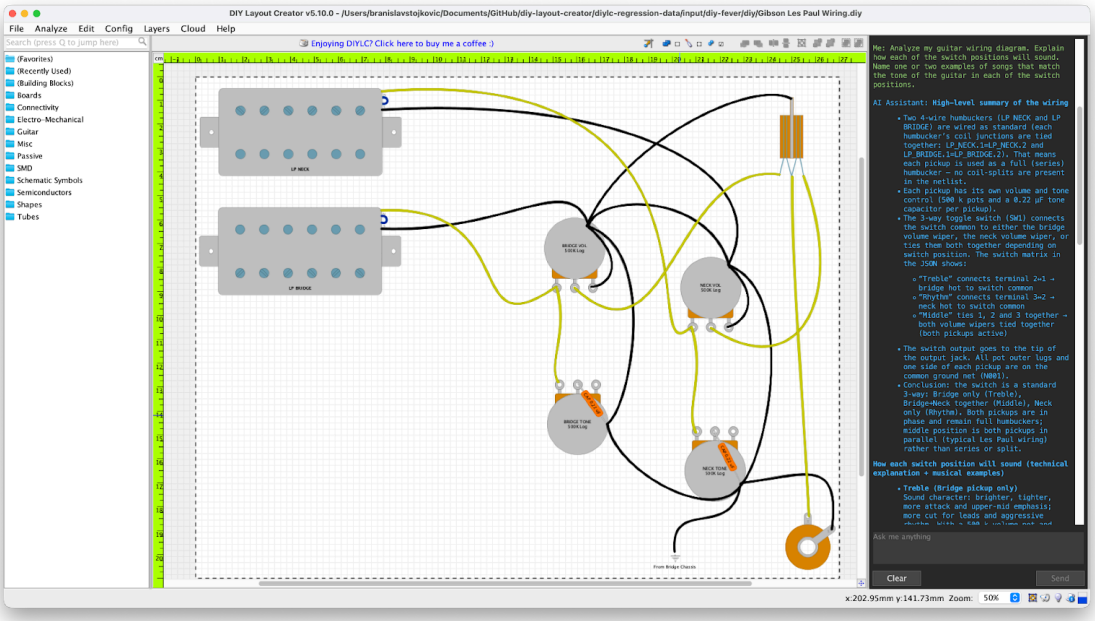


AI Assistant

DIYLC includes a built-in AI Assistant directly inside the application.

You can use it to ask electronics-related questions, get help with DIYLC features, or analyze the circuit you are currently working on.

Common applications include identifying errors in your layout, offering guidance during project assembly, troubleshooting circuit faults, and providing clarity on overall electrical functionality. Additionally, you can leverage the **AI Assistant** as a companion to the integrated **Guitar Wiring Diagram Analyzer**, creating a powerful duo for scrutinizing instrument electronics. This pairing enables a comprehensive exploration of intricate coil-tap logic and elaborate switching arrangements, providing clarity on even the most complex setups.



Customization and Flexibility

DIYLC also allows customization of the component library:

- Create your own components using primitive shapes
- Group components into reusable building blocks
- Save frequently used parts as favorites or variants

For advanced users and developers, DIYLC can also be extended using Java to create entirely new component types or features. DIYLC is built on top of an extendable architecture that allows anyone to create plugins and introduce new functionalities to the app.

What DIYLC Does Not (Yet) Support

DIYLC is intentionally lightweight and does not aim to replace professional EDA software.

Currently, it does not include:

- Real-time electrical simulation
- Automatic schematic-to-PCB conversion
- Autorouting of traces
- Real-time electrical rule validation
- Multi-sheet or hierarchical designs
- 3D PCB or enclosure previews

These omissions keep DIYLC fast, focused, and accessible.

Platform Compatibility

Designed for maximum flexibility, DIYLC is compatible with Windows, macOS, and Linux, as well as any other operating system that provides a Java runtime. This cross-platform capability ensures that your design environment remains consistent across different machines.

The majority of supported environments utilize convenient, pre-bundled packages that require no manual configuration or additional dependencies to operate.

In instances where a platform-specific build is unavailable, the system merely requires the manual installation of a Java JRE, provided it is version 21 or later.

Chapter Summary

In this chapter, you learned what DIYLC is and who it is designed for, how the project evolved from a simple forum idea into a modern open-source tool, the core strengths of DIYLC, its advanced analysis and AI tools, and the limitations that differentiate it from professional EDA software.