external-tool-malfunction-report.md

```
1: ---
2: name: External tool malfunction report
3: about: Report an external tool that fails to work with KiCad
4: title: "[External tool malfunction]"
5: labels: ''
6: assignees: ''
7: ---
8: **Describe the malfunction**
9: A clear and concise description how the external tool failed.
10: **To Reproduce**
11: Steps to reproduce the behavior:
12: 1. Go to '...'
13: 2. Click on '....'
14: 3. Scroll down to '....'
15: 4. See error
16: **Expected behavior**
17: A clear and concise description of what you expected to happen.
18: **Screenshots**
19: If applicable, add screenshots to help explain your problem.
20: **Desktop (please complete the following information):**
21: - OS: [e.g. Windows 10, Ubuntu 20.04, ...]
22: - KiCad version (use **Help => About KiCad => Show Version Info**)
23: - External tool name and version (if applicable)
24: **Additional context**
25: Add any other context about the problem here.
```

README.md

```
26: # KiCad Third-Party Tools
27: > A curated list of third-party tools to be used in conjunction with
28: the
29: [KiCad](http://kicad.org/) open-source electronics design
30: automation suite.
31: * Please read the [contribution guidelines](contributing.md) before
32: adding a tool to the list.
33: * **If you find a tool that no longer works with KiCad, please enter
34: an [issue
35: report](https://github.com/devbisme/kicad-3rd-party-tools/issues/new?ass
36: * Depending upon the version of KiCad your tool supports, please add
37: the appropriate badges at the end of your entry:
38: ![](https://img.shields.io/badge/V5-%20KiCad-red)
39: ![](https://img.shields.io/badge/V6-%20KiCad-blue)
40: ![](https://img.shields.io/badge/V7-%20KiCad-green)
41: **NOTE:** The entries on this page were contributed by the
42: originators/maintainers of these tools. As such, the list is going to
43: be incomplete.
44: Go
45: [here](https://devbisme.github.io/RepoRecon/?topic=kicad&filter=(d%3Auti
46: for a larger table
47: of KiCad utilities that is automatically extracted from the
48: repositories on Github. Tools will be automatically included in this
49: table if they
50: include the terms "KiCad" and "utility" or "plugin" in their Github
51: repository description.
52: ## Table of Contents
53: <!-- TOC depthFrom:2 depthTo:6 withLinks:1 updateOnSave:1
54: orderedList:0 -->
55: - [KiCad Third-Party Tools](#kicad-third-party-tools)
56: - [Table of Contents](#table-of-contents)
57: - [Schematic Tools](#schematic-tools)
58: - [Schematic Entry Tools](#schematic-entry-tools)
59: - [Symbol Library Tools](#symbol-library-tools)
60: - [BOM Tools](#bom-tools)
61: - [PCB Layout Tools](#pcb-layout-tools)
62: - [Footprint Library Tools](#footprint-library-tools)
63: - [Layout Tools](#layout-tools)
64: - [3d Model tools](#3d-model-tools)
65: - [Manufacturing Output Tools](#manufacturing-output-tools)
66: - [Version Control Tools](#version-control-tools)
67: - [Half-Baked Tools](#half-baked-tools)
68: - [Plumbing](#plumbing)
69: - [Cheatsheets](#cheatsheets)
70: - [License](#license)
71: <!-- /TOC -->
72: ## Schematic Tools
73: ### Schematic Entry Tools
74: - [Skidl](http://xesscorp.github.io/skidl) - A  module that allows
```

you
to compactly describe the interconnection of electronic circuits and components
using Python. The resulting Python program performs electrical rules checking
for common mistakes and outputs a netlist that serves as input to
a PCB layout tool such as KiCad's PCBNEW.

- [KiField](https://devbisme.github.io/KiField) - A utility for manipulating
part fields in KiCad schematics. KiField can extract all the component fields
and place them into a spreadsheet for bulk editing, after
which you can insert the edited values from the spreadsheet back into the schematic.

- [KiCad Partslist Editor](https://github.com/BPJWES/KiCAD_Partslist_editor) - KiCad
Partslist Editor (PLE) allows you to export/import customizable
fields from a hierarchical KiCad schematic file to and from a CSV
- [qucs2kicad](https://github.com/Valber/qucs2kicad) - Convert [Quite Universal Circuit Simulator](http://qucs.sourceforge.net/) schematics
to KiCad schematics.
- [KiCad Sheet Rearranger](https://github.com/KarlZeilhofer/KiCadSheetRearranger) -
Simple tool for defining a certain order of multiple sub-sheets in a
schematic project
- [KiCad2sycira](https://github.com/danselmi/sycira) - Symbolic circuit analyzer for the Maxima computer algebra system. Using KiCad
Eeschema for circuit enrtry.
-
[KiCadEditBusAliases](https://github.com/HoTschir/KiCadEditBusAliases)
- Simple tool for easy edit of Bus Aliases across hierarchical
sheets.

Symbol Library Tools
-
[Kicad-tools/libgen](https://github.com/boseji/Kicad-tools/tree/master/
- A Python script to generate schematic symbols from XML input.
- [Quick Library Generator](http://kicad.rohrbacher.net/quicklib.php)
- A web service to generate common "box type" symbols for ICs from
pin descriptions.
- [KiPart](https://devbisme.github.io/KiPart) - A utility that generates single
and multi-unit symbols from a CSV file containing all the pin information for
one or more parts.
- [KiField](https://devbisme.github.io/KiField) - A utility for manipulating
part fields in KiCad symbol libraries. KiField can extract all the
component fields

and place them into a spreadsheet for bulk editing, after which you can insert the edited values from the spreadsheet back into the library.

- [QEDA](https://github.com/qeda/qeda)
QEDA is a Node.js library aimed to simplify the creation of Kicad libraries of electronic components. Qeda creates both symbols for Eeschema libraries and IPC7351 Compliant footprints for PcbNew placement.
- [kicadLibCreator](https://github.com/pioupus/kicadLibCreator)
KicadLibCreator is a tool which will generate 'atomic' parts from an Octopart query. By setting up some simple rules, entering a part number in the Octopart search and selecting the appropriate model will add a fully defined component with a consistant style to any library.
- [KiCAD Part Manager](http://mikecrash.com/index.php?name=Content&pa=showpage&pid=10
Part manager for KiCAD electronic design suite. Based on a MySQL database of components with ability to categorize parts, store part name, part label and part number, description, parameters and also stock count and price. Parts can be assigned to components in KiCAD schematic based on component name, type and value. (**Built for KiCad 4, this may not work with KiCad 5!**)
- [KiCad Librarian](http://www.compuphase.com/electronics/kicadlibrarian_en.htm)
The KiCad Librarian is a utility to manage and maintain libraries with schematic symbols and footprints. It supports the KiCad EDA suite.
Allows components to be moved between libraries, footprints adjusted etc. Can connect to a server based repository of components to facilitate sharing between workstations.
- [Kicad Multiedit](http://www.xonelectronics.it/download/kicad-medit/)
Simple way of editing large number of components in spreadsheet type view. Will parse the values and footprints of components found in a KiCAD schematic.
- [KiLibMan](http://www.xonelectronics.it/download/kicad/)
A utility to examine library components and move them between libraries.
- [uConfig](https://github.com/Robotips/uConfig) - A tool to extract pinout from PDF datasheet and create kicad schematics.
- [KiCad-Db-Library](https://github.com/Projektanker/kicad-db-lib) - Inspired by Altium, KiCad-Db-Lib creates one or more KiCad Symbol Libraries with atomic parts based on your database. Create and maintain a database for your electric components, symbol reference, footprint reference, value, reference (R, L, C, etc.), description, datasheet, keywords and your custom fields (manufacturer, order codes etc.) inside of KiCadDbLib. Created with Angular and Electron KiCad-Db-Lib can be used on Windows, Linux and MacOS.
- [KiCad CSV Symbol Libraries](https://github.com/eeintech/kicad-database-utils-csv) -

(:warning: Support limited to KiCad V5 and older versions) Manage KiCad symbol library files using CSV format. The purpose of this tool is to translate back and forth symbol library data (stored in .lib and .dcm files) into the CSV format.
- [Ki-nTree](https://github.com/sparkmicro/Ki-nTree) - Fast and automated part creation tool for KiCad and InvenTree. From a manufacturer or Digi-Key part number, Ki-nTree uses Digi-Key's API to fetch the part specifications and automatically generates a KiCad symbol and/or InvenTree part.

- [easyeda2kicad.py](https://github.com/uPesy/easyeda2kicad.py) - Convert any LCSC components (including EasyEDA components) to a KiCad library (symbol and footprint)

BOM Tools
- [KiCost](https://github.com/hildogjr/KiCost) - A utility that generates a
spreadsheet from a schematic filled with the part pricing information scraped
from distributors like Digi-Key, Mouser, etc. For each distributor and part,
the spreadsheet contains the quantity-dependent prices, available quantities,
link to the part page, and ordering codes.
-
[KiCad_BOM_Wizard](https://github.com/HashDefineElectronics/KiCad_BOM_W
This KiCad plugin can be used to create custom BOM files based on easy configurable templates files. The plugin is writing in JavaScript and has been designed to integrate into KiCads BOM plugin manager. Exports CSV, HTML and PDF files. The template files permit customisation of output to include (for example) certification docs, logo etc. KiCad_BOM_Wizard will group and sort all components together that have same parts value, the same starting designator reference prefix and the same fields value.
- [BOMs Away](https://github.com/Jeff-Ciesielski/Boms-Away)
A Component/BOM Management tool for KiCad. Maintains a local database of components and facilitates associating components on schmatic with identified parts. Simply enter a part's manufacturer, supplier, manufacturer PN, and supplier PN then click 'save to datastore'. Information is keyed off of component value and footprint, so future uses can simply use the part lookup button to retrieve the information. Multiple suppliers, manufacturers, and part numbers are supported. (wxPython)
- [KiBoM](https://github.com/SchrodingersGat/KiBoM)
KiBoM is a configurable BOM (Bill of Materials) generation tool for KiCad EDA. Written in Python, it can be used directly with KiCad software without the need for any external libraries or plugins. KiBoM intelligently groups components based on multiple factors, and can generate BoM files in multiple output formats.
BoM options are user-configurable in a per-project configuration file.

- [KC2PK](https://github.com/Gasman2014/KC2PK)
KiCad to PartKeepr BOM Tool. This tool, written in Python3, aims to
integrate component management using BOMs produced from KiCad and
inventory and stock management using PartKeepr. It also includes an
Octopart lookup function to check on current pricing, availability
and price breaks of components.
- [Interactive Html
Bom](https://github.com/openscopeproject/InteractiveHtmlBom)

([demo](https://openscopeproject.org/InteractiveHtmlBomDemo/OSPx201/ibo
tool designed to assist with hand assembling pcbs. Output is viewable
in any modern browser and allows user to easily
highlight a specific reference or all components in a group on a
visual rendering of pcb. Script works both as Pcbnew action
plugin and as a command line tool.

- [KiCad JLCPCB BOM
Plugin](https://github.com/urish/kicad-jlcpcb-bom-plugin) Eschema
plugin to produce BOM compatible with JLCPCB SMT Assembly BOM format.
Also includes a script to convert the Footprint Position file into
the format required by JLCPCB.
PCB Layout Tools
Footprint Library Tools
-
[Kicad-tools/modgen](https://github.com/boseji/Kicad-tools/tree/master/
- A Python Tkinter GUI for creating footprints.
-
[monostable/kicad_footprints](https://github.com/monostable/kicad_footp
- A collection of all the KiCad footprints available on the internet
and some scripts to manage them.
- [svg2mod](https://github.com/svg2mod/svg2mod) - A tool to convert
multi-layer Inkscape SVGs into footprints.
- [xess_fp_wizard.py](https://github.com/xesscorp/xess_fp_wizard) - A
utility
to make footprints for chips having pins around the periphery (SOICs,
QFP, etc.)
and ball grid arrays (BGAs).
-
[KicadModTree](https://github.com/pointhi/kicad-footprint-generator)
- Python library for generating footprints. The scripts subdirectory
contains the footprints that are already scripted with this tool.
-
[SpiralInductorFootprintGenerator](https://github.com/erichVK5/SpiralIn
- A java utility for generating helical or polygonal inductor
footprints in either gEDA footprint or Kicad legacy module format.
-
[fped](http://downloads.qi-hardware.com/people/werner/fped/gui.html)
- A parametric constraint based editor for footprints with a GUI and
KiCad and postscript outputs. Still quite rough around the edges but
available on Debian based systems through an `apt install fped`.
- [KiCad Footprint

Rotator](https://gitlab.com/salfter/kicad-footprint-rotator) - A sed script that takes a footprint and rotates it 90 counterclockwise. Run it twice to turn a footprint upside-down, or three times to turn it 90 clockwise. If you're designing a board for automated assembly, you'll need this tool to line up your footprints to match the alignment of components in your tapes and trays.
- [KiBuzzard](https://github.com/gregdavill/KiBuzzard) - A tool to create inverted labels on silk screen layer.
- [easyeda2kicad.py](https://github.com/uPesy/easyeda2kicad.py) - Convert any LCSC components (including EasyEDA components) to a KiCad library (symbol and footprint)

Layout Tools
- [Laksen/kicad-bga-tools](https://github.com/Laksen/kicad-bga-tools) - A script to generate via fanouts for BGA components on a board.
-
[panelize.py](http://projects.borg.ch/electronics/kicad/panelize.html) - A script to create panels. It can copy, rotate and flip rectangular areas from one or more PCB files into a new PCB file.
-
[RenumKicadPCB](https://documenteddesigns.com/2017/03/27/renumkicadpcb- - RenumKiCadPCB processes a KiCad PCB file and renumbers all the component reference designators ending in numbers based on where they are located on the PCB. It then processes the schematic hierarchy and updates the component reference designators to match. This makes working on a board much easier since you can locate all the components. The download includes a user manual, Windows executable and instructions for compiling to run on Linux.
- [KiPadCheck](https://github.com/HiGregSmith/KiPadCheck) - KiPadCheck provides additional basic DRC checks to KiCad with lists to make tweaking pads for stencil creation easier. Functions include pad list, drill list, drill to drill spacing check, drill to track spacing check, stencil aperture check vs. stencil thicknesses, stencil aperture width vs. paste type, silk to pad spacing check.
- [Layer View Set](https://github.com/HiGregSmith/LayerViewSet) - A gui for saving and loading ViewSets and interacting with a stack of ViewSets for quickly changing the currently visible layers and renders within KiCad.
-
[Teardrop](https://github.com/NilujePerchut/kicad_scripts/tree/master/t - A gui to teardrop the vias, pads and "T" tracks connections in the Pcbnew.
- [Replicate layout](https://github.com/MitjaNemec/Kicad_action_plugins) - This Kicad Action plugin replicates layout section. The replication is based upon hierarchical sheets. Basic requirement for replication is that the section for replication is completely contained within one hierarchical sheet, and replicated sections are just a copy of the same sheet.
- [svg2shenzhen](https://github.com/badgeek/svg2shenzhen-next) - An

Inkscape plugin to export SVG layers to KiCad PCB layers. You name your layers what they would be called in KiCad (F.Cu, B.Cu etc.), draw things on them and can then turn them into a kicad_pcb or a kicad_mod. Accepts arbitrary shapes on most layers (unlike svg2mod) by using PNGs as an intermediate step and automatically converting them with a fork of KiCad's own bitmap2component.
- [HierPlace](https://github.com/devbisme/HierPlace) - A PCBNEW plugin that creates an initial arrangement of parts into groups that reflect the hierarchy of the design.
- [PadPainter](https://github.com/devbisme/PadPainter) - This PCBNEW plugin identifies pins that meet specified criteria and highlights the associated pads on the PCB. This is helpful for identifying sets of related pins when physically planning the layout of high pin-count packages such as FPGAs.
- [WireIt](https://github.com/devbisme/WireIt) - This PCBNEW plugin lets you add wires between pads on a PCB, delete them, and swap wires between pads. This is helpful for physically connecting sets of related pins when doing the layout of high pin-count packages such as FPGAs.
- [flexRoundingSuite](https://github.com/jcloiacon/flexRoundingSuite) - Python script to round the corners of Kicad Pcbnew traces for RF / FlexPCB applications
- [DRMgr](https://github.com/devbisme/DRMgr) - A plugin that allows you to extract the design rules from a KiCad board and store them into a file, and then load the file into other boards to replicate the design rule settings.
- [RF-Tools for KiCAD](https://github.com/easyw/RF-tools-KiCAD) - A Kicad Action plugin suite to help in RF and Flex pcb design. Footprint wizards for designing mitred bends, tapered track connectors, and arc tracks (radius bends) for RF layout included. Round track corners routing, track length measurement and a mask expansion tool for direct pcb routing. Via fencing tool for RF via shielding. [Live demo](https://youtu.be/LDblUeaB7_s) available on line.
- [Qucs-RFlayout](https://github.com/thomaslepoix/Qucs-RFlayout) - A tool to export Qucs RF schematics (microstrip) to PcbNew board layout or footprint.
- [dren.dk/kicad-util](https://gitlab.com/dren.dk/kicad-util) - Java utility for PCB layout cloning and panelization.
- [KiKit: Automation & panelizaton for KiCAD](https://github.com/yaqwsx/KiKit) - Tool to automatically produce panels, export gerbers and board presentation pages.
- [SchematicPositionsToLayout](https://github.com/jenschr/KiCad-parts#sch - A script that takes a Kicad 5 schematic (.sch) and a PCB Layout (.kicad_pcb) file and arranges all the components on the PCB to mimic their positions in the schematic.
- [Stretch](https://github.com/JarrettR/Stretch) - Provides a *bidirectional path* so functional layout in PCBNEW can be iteratively combined with artistic design in [Inkscape](https://inkscape.org/).

- [FilletEdge](https://github.com/tywtyw2002/FilletEdge) - Fillet the Edge inside KiCAD.
3d Model tools
- [KiCad StepUp](https://github.com/easyw/kicadStepUpMod/) - A FreeCAD Workbench for collaborative electrical + mechanical design which allows:
+ Export of KiCad board and parts as STEP and WRL models.
+ Precise alignment of `kicad_mod` footprints with their mechanical models.
+ Editing of KiCad PCB outlines in FreeCAD Sketcher.
+ Adjustment of PCB part positions between FreeCAD and KiCad.
- [cadquery 3d model generator](https://github.com/easyw/kicad-3d-models-in-freecad/tree/mas
- 3d model generators using freecad and the cadquery plugin. The scripts generate step and scaled wrl files similar to kicad stepup.
- [pcbmodelgen](https://github.com/jcyrax/pcbmodelgen) - Convert KiCAD PCB files to models for import in openEMS
- [fcad_pcb](https://github.com/realthunder/fcad_pcb) - The original purpose of these tools was to do PCB milling in FreeCAD. It can do much more now. It can generate gcode from kicad_pcb directly without going though gerber stage. It can let you modify the PCB directly inside FC (done already), and potentially export back to kicad_pcb (partially done). And finally it can generate solid tracks, pads and plated drills to enable FEM and thermal analysis on KiCad pcb boards.
Manufacturing Output Tools
- [kiplot](https://github.com/johnbeard/kiplot) - A python module and program that lets you run and script KiCad's various manufacturing outputs such as Gerbers and other plots in a configurable way.
- [kibot](https://github.com/INTI-CMNB/kibot) - A much more advanced fork of kiplot.
- [kicad-exports](https://github.com/nerdyscout/kicad-exports) - Auto generate files (schematics, gerbers, BoM, plots, 3D model) for any KiCAD project. You could run it locally or on every git push with Github Actions.
- [obra/kicad-tools](https://github.com/obra/kicad-tools) - Productivity-enhancing tools primarily focused on automating generation of fabrication outputs and commandline productivity for projects tracked in git.
- [GerberZipper](https://github.com/g200kg/kicad-gerberzipper) - A plugin that plots Gerber-files and Zip it for a specified PCB manufacturer.
- [PcbDraw](https://github.com/yaqwsx/PcbDraw) - Script that converts PCB to nice looking 2D drawing.
- [Board2Pdf](https://gitlab.com/dennevi/Board2Pdf/) - A plugin which creates customized high resolution searchable pdf files from the PCB.
- [gerber_to_order](https://github.com/asukiaaa/gerber_to_order) - A plugin that creates zip compressed gerber files for PCB manufacturers.
- [PCBWay Plug-in for Kicad](https://github.com/pcbway/PCBWay-Plug-in-for-Kicad) Plugin to

automate generation of Gerber files, IPC-Netlist file, BOM file and Pick-n-Place file in the appropriate format for PCB manufacturing and assembly at [PCBWay](https://www.pcbway.com/).

- [KiCAD JLCPCB tools](https://github.com/Bouni/kicad-jlcpcb-tools) Plugin to automate generation of Gerber files, Excellon files, BOM file, CPL file in the appropriate format for PCB manufacturing and assembly at [JLCPCB](https://jlcpcb.com/).

- [KiCAD Test Points](https://github.com/snhobbs/kicad-testpoints) CLI to create test-point reports for making bed-of-nails jigs. Allows any pad to be set as a testpoint. Reports are generated in the [JigsApp](https://www.thejigsapp.com) format. Also available as a [PCM plugin](https://github.com/TheJigsApp/kicad-testpoints-pcm).

Version Control Tools
- [KiCad-Diff](https://github.com/Gasman2014/KiCad-Diff) - Python3 script for performing image diffs between pcbnew layout revisions in Git, SVN and Fossil VCS. Recent SVG based diff for significant speed improvements.
- [Massaging your git for kicad](https://jnavila.github.io/plotkicadsch/) - Instructions how to integrate KiCad with Git VCS
- [PlotKicadsch](https://github.com/jnavila/plotkicadsch) - PlotKicadsch is a small tool to export Kicad Sch files to SVG pictures.
- [Scripting KiCad Pcbnew exports](https://scottbezek.blogspot.si/2016/04/scripting-kicad-pcbnew-
- How to plot properly formated SVG files from pcbnew for VCS diff-ing
- [Improving open source hardware: Visual diffs](https://www.evilmadscientist.com/2011/improving-open-source-hard
- Using ImageMagick for visual diff file generation
- [KiCAD to SVG Converter](https://github.com/jmwright/oshw-code/tree/master/kicad_to_s
- Scripts for headless SVG generation of schematic files using eeschema.
- [kiri](https://github.com/leoheck/kiri) - Kicad Revision Inspector (KiRI) that combines PlotKicadSch and KiCad-Diff into a single website, for Git repositories, having a visual and interactive revision system for schematics and layouts.
Half-Baked Tools
If you have an interesting tool that's not quite ready for prime-time, post it here!
Maybe someone else can move it forward or use it as a starting point for their own tool.
- [footwork](https://github.com/monostable/footwork) - Unfinished footprint editor that tries to mix the s-expression footprint format with Racket (Scheme) to programmatically create footprints.
- [fpmagic](http://fpmagic.engineerjs.com/) - Web based, Chrome only, SMD only experimental footprint editor. Draw footprints using the

constraints from a datasheet drawing.
Plumbing
These are libraries/packages/modules that can help when creating tools like the ones listed above.
- [KinJector](https://github.com/devbisme/kinjector) - Inject/eject JSON/YAML data to/from KiCad Board files. Primarily used to read/write design rules, net classes, net class assignments, and part (X,Y)/orientation/top-bottom positions.
- [kinparse](https://github.com/devbisme/kinparse) - A parser for KiCad schematic netlist files that are output by EESCHEMA. Just pass a file containing a netlist to the `parse_netlist()` function and it will deliver a [pyparsing](https://pypi.python.org/pypi/pyparsing) object containing all the netlist's information.

- [pykicad](https://github.com/dvc94ch/pykicad) - The aim of this project is to provide high quality and well tested support for reading and writing KiCad file formats.
- [kicad-python](https://github.com/pointhi/kicad-python) - An abstraction layer for the KiCad python interface. (Be aware this is in initial development and the interface can change anytime!)
- [pykicadlib](https://code.fueldner.net/opensource/pykicadlib) - A Python library to read and write KiCAD footprints and schematic files.
- [kicad-utils](https://github.com/cho45/kicad-utils) - KiCAD library / schematic / pcb parser and plotter written in TypeScript (JavaScript)
- [KiCad-RW](https://github.com/FabriceSalvaire/kicad-rw) - A Python library to read/write KiCad 6 Sexpr file format. In addition this library can compute the netlist of a circuit. (It actually only implements a `.kicad_sch` reader)
Cheatsheets
These are handy guides for using KiCad and related tools.
- [KiCad Cheatsheet](https://github.com/KiCad/kicad-doc/blob/master/src/cheatshe
- Lists common operations and keyboard shortcuts for KiCad.
- [KiCad StepUp Cheatsheet](https://github.com/easyw/kicadStepUpMod/raw/master/demo/kic
- A set of concise descriptions on how to do mechanical CAD tasks on KiCad PCBs with the StepUp tool.
- [Git Cheatsheet](https://www.git-tower.com/learn/cheat-sheets/git)
- Summarizes common operations on Git repositories that are often used to store KiCad libraries and projects.
License

[![CC0](http://mirrors.creativecommons.org/presskit/buttons/88x31/svg/c

acknowledgement.md

```
530: # Acknowledgements
531: I created this repo of KiCad 3rd-party tools by mercilessly copying
532: everything from
533: [this repo of embedded system
534: resources](https://github.com/embedded-boston/awesome-embedded-systems)
535: and making a few small changes.
536: Thanks [cwoodall](https://github.com/cwoodall)!
```

contributing.md

```
537: # Contribution Guidelines
538: First of all, thank you for making a suggestion or addition to this
539: list!
540: To make the process as easy as possible (for you and me), please read
541: the [guidelines](#guidelines) and [submission
542: procedure](#how-to-add-something-to-this-list) below.
543: ## Table of Contents
544: - [Contribution Guidelines](#contribution-guidelines)
545: - [Table of Contents](#table-of-contents)
546: - [Guidelines](#guidelines)
547: - [How to Add Something to This
548: List](#how-to-add-something-to-this-list)
549: - [Updating Your Pull Request](#updating-your-pull-request)
550: ## Guidelines
551: Please ensure your pull request (PR) adheres to the following
552: guidelines:
553: - Search previous suggestions/additions before making a new one, as
554: yours may be a duplicate.
555: - Make sure your addition is useful before submitting. That implies
556: it has enough content and a good, succinct description.
557: - Make an individual PR for each addition.
558: - Use [title-casing](http://titlecapitalization.com) (AP style).
559: - Use the following format: `[List Name](link)`
560: - Link additions should be added to the **bottom** of the relevant
561: category. **The newest tools are always closer to the botttom!**
562: - If you have added **new features** to your tool, you may move its
563: entry to the bottom of the list, or you may leave it in its current
564: position.
565: - If your tool addition supports the new KiCad V6, please add this
566: badge at the beginning of your entry:
567: ![](https://img.shields.io/badge/V6-%20KiCad-blue)
568: - New categories or improvements to the existing categorization are
569: welcome.
570: - Check your spelling and grammar.
571: - Make sure your text editor is set to remove trailing whitespace.
572: - **The PR and commit should have a useful title.** PRs with `Update
573: readme.md` as their title will be closed right away.
574: - The body of your commit message should contain a link to the
575: repository.
576: ## How to Add Something to This List
577: If you have something to add to this list, here's how to do it.
578: You'll need a [GitHub account](https://github.com/join)!
579: 1. Access the [this list's GitHub
580: page](https://github.com/devbisme/kicad-3rd-party-tools).
581: 2. Click on the `README.md` file: ![Step 2 Click on
582: README.md](view-readme.png)
583: 3. Now click on the edit icon. ![Step 3 - Click on
584: Edit](start-editor.png)
585: 4. You can start editing the text of the file in the in-browser
```

editor. Make sure you follow the guidelines above. You can use [GitHub Flavored Markdown](https://help.github.com/articles/github-flavored-markdown/).

5. Say why you're proposing the changes, and then click on "Propose file change".

6. Submit the [pull request](https://help.github.com/articles/using-pull-requests/)!

Updating Your Pull Request

Sometimes, a maintainer of this list will ask you to edit your Pull Request before it is included. This is normally due to spelling errors or because your PR didn't match the list's guidelines.

[Here](https://github.com/RichardLitt/docs/blob/master/amending-a-commi is a write up on how to change a PR, and the different ways you can do that.

external-tool-malfunction-report.md

```
1: ---
2: name: External tool malfunction report
3: about: Report an external tool that fails to work with KiCad
4: title: "[External tool malfunction]"
5: labels: ''
6: assignees: ''
7: ---
8: **Describe the malfunction**
9: A clear and concise description how the external tool failed.
10: **To Reproduce**
11: Steps to reproduce the behavior:
12: 1. Go to '...'
13: 2. Click on '....'
14: 3. Scroll down to '....'
15: 4. See error
16: **Expected behavior**
17: A clear and concise description of what you expected to happen.
18: **Screenshots**
19: If applicable, add screenshots to help explain your problem.
20: **Desktop (please complete the following information):**
21: - OS: [e.g. Windows 10, Ubuntu 20.04, ...]
22: - KiCad version (use **Help => About KiCad => Show Version Info**)
23: - External tool name and version (if applicable)
24: **Additional context**
25: Add any other context about the problem here.
```

README.md

```
26: # KiCad Third-Party Tools
27: > A curated list of third-party tools to be used in conjunction with
28: the
29: [KiCad](http://kicad.org/) open-source electronics design
30: automation suite.
31: * Please read the [contribution guidelines](contributing.md) before
32: adding a tool to the list.
33: * **If you find a tool that no longer works with KiCad, please enter
34: an [issue
35: report](https://github.com/devbisme/kicad-3rd-party-tools/issues/new?ass
36: * Depending upon the version of KiCad your tool supports, please add
37: the appropriate badges at the end of your entry:
38: ![](https://img.shields.io/badge/V5-%20KiCad-red)
39: ![](https://img.shields.io/badge/V6-%20KiCad-blue)
40: ![](https://img.shields.io/badge/V7-%20KiCad-green)
41: **NOTE:** The entries on this page were contributed by the
42: originators/maintainers of these tools. As such, the list is going to
43: be incomplete.
44: Go
45: [here](https://devbisme.github.io/RepoRecon/?topic=kicad&filter=(d%3Auti
46: for a larger table
47: of KiCad utilities that is automatically extracted from the
48: repositories on Github. Tools will be automatically included in this
49: table if they
50: include the terms "KiCad" and "utility" or "plugin" in their Github
51: repository description.
52: ## Table of Contents
53: <!-- TOC depthFrom:2 depthTo:6 withLinks:1 updateOnSave:1
54: orderedList:0 -->
55: - [KiCad Third-Party Tools](#kicad-third-party-tools)
56: - [Table of Contents](#table-of-contents)
57: - [Schematic Tools](#schematic-tools)
58: - [Schematic Entry Tools](#schematic-entry-tools)
59: - [Symbol Library Tools](#symbol-library-tools)
60: - [BOM Tools](#bom-tools)
61: - [PCB Layout Tools](#pcb-layout-tools)
62: - [Footprint Library Tools](#footprint-library-tools)
63: - [Layout Tools](#layout-tools)
64: - [3d Model tools](#3d-model-tools)
65: - [Manufacturing Output Tools](#manufacturing-output-tools)
66: - [Version Control Tools](#version-control-tools)
67: - [Half-Baked Tools](#half-baked-tools)
68: - [Plumbing](#plumbing)
69: - [Cheatsheets](#cheatsheets)
70: - [License](#license)
71: <!-- /TOC -->
72: ## Schematic Tools
73: ### Schematic Entry Tools
74: - [Skidl](http://xesscorp.github.io/skidl) - A  module that allows
```

you
to compactly describe the interconnection of electronic circuits and components
using Python. The resulting Python program performs electrical rules checking
for common mistakes and outputs a netlist that serves as input to
a PCB layout tool such as KiCad's PCBNEW.

- [KiField](https://devbisme.github.io/KiField) - A utility for manipulating
part fields in KiCad schematics. KiField can extract all the component fields
and place them into a spreadsheet for bulk editing, after
which you can insert the edited values from the spreadsheet back into the schematic.

- [KiCad Partslist Editor](https://github.com/BPJWES/KiCAD_Partslist_editor) - KiCad
Partslist Editor (PLE) allows you to export/import customizable
fields from a hierarchical KiCad schematic file to and from a CSV
- [qucs2kicad](https://github.com/Valber/qucs2kicad) - Convert [Quite Universal Circuit Simulator](http://qucs.sourceforge.net/) schematics
to KiCad schematics.
- [KiCad Sheet Rearranger](https://github.com/KarlZeilhofer/KiCadSheetRearranger) -
Simple tool for defining a certain order of multiple sub-sheets in a
schematic project
- [KiCad2sycira](https://github.com/danselmi/sycira) - Symbolic circuit analyzer for the Maxima computer algebra system. Using KiCad
Eeschema for circuit enrtry.
-
[KiCadEditBusAliases](https://github.com/HoTschir/KiCadEditBusAliases)
- Simple tool for easy edit of Bus Aliases across hierarchical
sheets.

Symbol Library Tools
-
[Kicad-tools/libgen](https://github.com/boseji/Kicad-tools/tree/master/
- A Python script to generate schematic symbols from XML input.
- [Quick Library Generator](http://kicad.rohrbacher.net/quicklib.php)
- A web service to generate common "box type" symbols for ICs from
pin descriptions.
- [KiPart](https://devbisme.github.io/KiPart) - A utility that generates single
and multi-unit symbols from a CSV file containing all the pin information for
one or more parts.
- [KiField](https://devbisme.github.io/KiField) - A utility for manipulating
part fields in KiCad symbol libraries. KiField can extract all the
component fields

and place them into a spreadsheet for bulk editing, after which you can insert the edited values from the spreadsheet back into the library.

- [QEDA](https://github.com/qeda/qeda)
QEDA is a Node.js library aimed to simplify the creation of Kicad libraries of electronic components. Qeda creates both symbols for Eeschema libraries and IPC7351 Compliant footprints for PcbNew placement.
- [kicadLibCreator](https://github.com/pioupus/kicadLibCreator)
KicadLibCreator is a tool which will generate 'atomic' parts from an Octopart query. By setting up some simple rules, entering a part number in the Octopart search and selecting the appropriate model will add a fully defined component with a consistant style to any library.
- [KiCAD Part Manager](http://mikecrash.com/index.php?name=Content&pa=showpage&pid=10
Part manager for KiCAD electronic design suite. Based on a MySQL database of components with ability to categorize parts, store part name, part label and part number, description, parameters and also stock count and price. Parts can be assigned to components in KiCAD schematic based on component name, type and value. (**Built for KiCad 4, this may not work with KiCad 5!**)
- [KiCad Librarian](http://www.compuphase.com/electronics/kicadlibrarian_en.htm)
The KiCad Librarian is a utility to manage and maintain libraries with schematic symbols and footprints. It supports the KiCad EDA suite.
Allows components to be moved between libraries, footprints adjusted etc. Can connect to a server based repository of components to facilitate sharing between workstations.
- [Kicad Multiedit](http://www.xonelectronics.it/download/kicad-medit/)
Simple way of editing large number of components in spreadsheet type view. Will parse the values and footprints of components found in a KiCAD schematic.
- [KiLibMan](http://www.xonelectronics.it/download/kicad/)
A utility to examine library components and move them between libraries.
- [uConfig](https://github.com/Robotips/uConfig) - A tool to extract pinout from PDF datasheet and create kicad schematics.
- [KiCad-Db-Library](https://github.com/Projektanker/kicad-db-lib) - Inspired by Altium, KiCad-Db-Lib creates one or more KiCad Symbol Libraries with atomic parts based on your database. Create and maintain a database for your electric components, symbol reference, footprint reference, value, reference (R, L, C, etc.), description, datasheet, keywords and your custom fields (manufacturer, order codes etc.) inside of KiCadDbLib. Created with Angular and Electron KiCad-Db-Lib can be used on Windows, Linux and MacOS.
- [KiCad CSV Symbol Libraries](https://github.com/eeintech/kicad-database-utils-csv) -

(:warning: Support limited to KiCad V5 and older versions) Manage KiCad symbol library files using CSV format. The purpose of this tool is to translate back and forth symbol library data (stored in .lib and .dcm files) into the CSV format.
- [Ki-nTree](https://github.com/sparkmicro/Ki-nTree) - Fast and automated part creation tool for KiCad and InvenTree. From a manufacturer or Digi-Key part number, Ki-nTree uses Digi-Key's API to fetch the part specifications and automatically generates a KiCad symbol and/or InvenTree part.

- [easyeda2kicad.py](https://github.com/uPesy/easyeda2kicad.py) - Convert any LCSC components (including EasyEDA components) to a KiCad library (symbol and footprint)

BOM Tools
- [KiCost](https://github.com/hildogjr/KiCost) - A utility that generates a
spreadsheet from a schematic filled with the part pricing information scraped
from distributors like Digi-Key, Mouser, etc. For each distributor and part,
the spreadsheet contains the quantity-dependent prices, available quantities,
link to the part page, and ordering codes.
-
[KiCad_BOM_Wizard](https://github.com/HashDefineElectronics/KiCad_BOM_W
This KiCad plugin can be used to create custom BOM files based on easy configurable templates files. The plugin is writing in JavaScript and has been designed to integrate into KiCads BOM plugin manager. Exports CSV, HTML and PDF files. The template files permit customisation of output to include (for example) certification docs, logo etc. KiCad_BOM_Wizard will group and sort all components together that have same parts value, the same starting designator reference prefix and the same fields value.
- [BOMs Away](https://github.com/Jeff-Ciesielski/Boms-Away)
A Component/BOM Management tool for KiCad. Maintains a local database of components and facilitates associating components on schmatic with identified parts. Simply enter a part's manufacturer, supplier, manufacturer PN, and supplier PN then click 'save to datastore'. Information is keyed off of component value and footprint, so future uses can simply use the part lookup button to retrieve the information. Multiple suppliers, manufacturers, and part numbers are supported. (wxPython)
- [KiBoM](https://github.com/SchrodingersGat/KiBoM)
KiBoM is a configurable BOM (Bill of Materials) generation tool for KiCad EDA. Written in Python, it can be used directly with KiCad software without the need for any external libraries or plugins. KiBoM intelligently groups components based on multiple factors, and can generate BoM files in multiple output formats.
BoM options are user-configurable in a per-project configuration file.

- [KC2PK](https://github.com/Gasman2014/KC2PK)
KiCad to PartKeepr BOM Tool. This tool, written in Python3, aims to
integrate component management using BOMs produced from KiCad and
inventory and stock management using PartKeepr. It also includes an
Octopart lookup function to check on current pricing, availability
and price breaks of components.
- [Interactive Html
Bom](https://github.com/openscopeproject/InteractiveHtmlBom)

([demo](https://openscopeproject.org/InteractiveHtmlBomDemo/OSPx201/ibo
tool designed to assist with hand assembling pcbs. Output is viewable
in any modern browser and allows user to easily
highlight a specific reference or all components in a group on a
visual rendering of pcb. Script works both as Pcbnew action
plugin and as a command line tool.

- [KiCad JLCPCB BOM
Plugin](https://github.com/urish/kicad-jlcpcb-bom-plugin) Eschema
plugin to produce BOM compatible with JLCPCB SMT Assembly BOM format.
Also includes a script to convert the Footprint Position file into
the format required by JLCPCB.
PCB Layout Tools
Footprint Library Tools
-
[Kicad-tools/modgen](https://github.com/boseji/Kicad-tools/tree/master/
- A Python Tkinter GUI for creating footprints.
-
[monostable/kicad_footprints](https://github.com/monostable/kicad_footp
- A collection of all the KiCad footprints available on the internet
and some scripts to manage them.
- [svg2mod](https://github.com/svg2mod/svg2mod) - A tool to convert
multi-layer Inkscape SVGs into footprints.
- [xess_fp_wizard.py](https://github.com/xesscorp/xess_fp_wizard) - A
utility
to make footprints for chips having pins around the periphery (SOICs,
QFP, etc.)
and ball grid arrays (BGAs).
-
[KicadModTree](https://github.com/pointhi/kicad-footprint-generator)
- Python library for generating footprints. The scripts subdirectory
contains the footprints that are already scripted with this tool.
-
[SpiralInductorFootprintGenerator](https://github.com/erichVK5/SpiralIn
- A java utility for generating helical or polygonal inductor
footprints in either gEDA footprint or Kicad legacy module format.
-
[fped](http://downloads.qi-hardware.com/people/werner/fped/gui.html)
- A parametric constraint based editor for footprints with a GUI and
KiCad and postscript outputs. Still quite rough around the edges but
available on Debian based systems through an `apt install fped`.
- [KiCad Footprint

Rotator](https://gitlab.com/salfter/kicad-footprint-rotator) - A sed script that takes a footprint and rotates it 90 counterclockwise. Run it twice to turn a footprint upside-down, or three times to turn it 90 clockwise. If you're designing a board for automated assembly, you'll need this tool to line up your footprints to match the alignment of components in your tapes and trays.
- [KiBuzzard](https://github.com/gregdavill/KiBuzzard) - A tool to create inverted labels on silk screen layer.
- [easyeda2kicad.py](https://github.com/uPesy/easyeda2kicad.py) - Convert any LCSC components (including EasyEDA components) to a KiCad library (symbol and footprint)

Layout Tools
- [Laksen/kicad-bga-tools](https://github.com/Laksen/kicad-bga-tools) - A script to generate via fanouts for BGA components on a board.
-
[panelize.py](http://projects.borg.ch/electronics/kicad/panelize.html) - A script to create panels. It can copy, rotate and flip rectangular areas from one or more PCB files into a new PCB file.
-
[RenumKicadPCB](https://documenteddesigns.com/2017/03/27/renumkicadpcb- - RenumKiCadPCB processes a KiCad PCB file and renumbers all the component reference designators ending in numbers based on where they are located on the PCB. It then processes the schematic hierarchy and updates the component reference designators to match. This makes working on a board much easier since you can locate all the components. The download includes a user manual, Windows executable and instructions for compiling to run on Linux.
- [KiPadCheck](https://github.com/HiGregSmith/KiPadCheck) - KiPadCheck provides additional basic DRC checks to KiCad with lists to make tweaking pads for stencil creation easier. Functions include pad list, drill list, drill to drill spacing check, drill to track spacing check, stencil aperture check vs. stencil thicknesses, stencil aperture width vs. paste type, silk to pad spacing check.
- [Layer View Set](https://github.com/HiGregSmith/LayerViewSet) - A gui for saving and loading ViewSets and interacting with a stack of ViewSets for quickly changing the currently visible layers and renders within KiCad.
-
[Teardrop](https://github.com/NilujePerchut/kicad_scripts/tree/master/t - A gui to teardrop the vias, pads and "T" tracks connections in the Pcbnew.
- [Replicate layout](https://github.com/MitjaNemec/Kicad_action_plugins) - This Kicad Action plugin replicates layout section. The replication is based upon hierarchical sheets. Basic requirement for replication is that the section for replication is completely contained within one hierarchical sheet, and replicated sections are just a copy of the same sheet.
- [svg2shenzhen](https://github.com/badgeek/svg2shenzhen-next) - An

Inkscape plugin to export SVG layers to KiCad PCB layers. You name your layers what they would be called in KiCad (F.Cu, B.Cu etc.), draw things on them and can then turn them into a kicad_pcb or a kicad_mod. Accepts arbitrary shapes on most layers (unlike svg2mod) by using PNGs as an intermediate step and automatically converting them with a fork of KiCad's own bitmap2component.
- [HierPlace](https://github.com/devbisme/HierPlace) - A PCBNEW plugin that creates an initial arrangement of parts into groups that reflect the hierarchy of the design.
- [PadPainter](https://github.com/devbisme/PadPainter) - This PCBNEW plugin identifies pins that meet specified criteria and highlights the associated pads on the PCB. This is helpful for identifying sets of related pins when physically planning the layout of high pin-count packages such as FPGAs.
- [WireIt](https://github.com/devbisme/WireIt) - This PCBNEW plugin lets you add wires between pads on a PCB, delete them, and swap wires between pads. This is helpful for physically connecting sets of related pins when doing the layout of high pin-count packages such as FPGAs.
- [flexRoundingSuite](https://github.com/jcloiacon/flexRoundingSuite) - Python script to round the corners of Kicad Pcbnew traces for RF / FlexPCB applications
- [DRMgr](https://github.com/devbisme/DRMgr) - A plugin that allows you to extract the design rules from a KiCad board and store them into a file, and then load the file into other boards to replicate the design rule settings.
- [RF-Tools for KiCAD](https://github.com/easyw/RF-tools-KiCAD) - A Kicad Action plugin suite to help in RF and Flex pcb design. Footprint wizards for designing mitred bends, tapered track connectors, and arc tracks (radius bends) for RF layout included. Round track corners routing, track length measurement and a mask expansion tool for direct pcb routing. Via fencing tool for RF via shielding. [Live demo](https://youtu.be/LDblUeaB7_s) available on line.
- [Qucs-RFlayout](https://github.com/thomaslepoix/Qucs-RFlayout) - A tool to export Qucs RF schematics (microstrip) to PcbNew board layout or footprint.
- [dren.dk/kicad-util](https://gitlab.com/dren.dk/kicad-util) - Java utility for PCB layout cloning and panelization.
- [KiKit: Automation & panelizaton for KiCAD](https://github.com/yaqwsx/KiKit) - Tool to automatically produce panels, export gerbers and board presentation pages.
- [SchematicPositionsToLayout](https://github.com/jenschr/KiCad-parts#sch - A script that takes a Kicad 5 schematic (.sch) and a PCB Layout (.kicad_pcb) file and arranges all the components on the PCB to mimic their positions in the schematic.
- [Stretch](https://github.com/JarrettR/Stretch) - Provides a *bidirectional path* so functional layout in PCBNEW can be iteratively combined with artistic design in [Inkscape](https://inkscape.org/).

- [FilletEdge](https://github.com/tywtyw2002/FilletEdge) - Fillet the Edge inside KiCAD.
3d Model tools
- [KiCad StepUp](https://github.com/easyw/kicadStepUpMod/) - A FreeCAD Workbench for collaborative electrical + mechanical design which allows:
+ Export of KiCad board and parts as STEP and WRL models.
+ Precise alignment of `kicad_mod` footprints with their mechanical models.
+ Editing of KiCad PCB outlines in FreeCAD Sketcher.
+ Adjustment of PCB part positions between FreeCAD and KiCad.
- [cadquery 3d model generator](https://github.com/easyw/kicad-3d-models-in-freecad/tree/mas
- 3d model generators using freecad and the cadquery plugin. The scripts generate step and scaled wrl files similar to kicad stepup.
- [pcbmodelgen](https://github.com/jcyrax/pcbmodelgen) - Convert KiCAD PCB files to models for import in openEMS
- [fcad_pcb](https://github.com/realthunder/fcad_pcb) - The original purpose of these tools was to do PCB milling in FreeCAD. It can do much more now. It can generate gcode from kicad_pcb directly without going though gerber stage. It can let you modify the PCB directly inside FC (done already), and potentially export back to kicad_pcb (partially done). And finally it can generate solid tracks, pads and plated drills to enable FEM and thermal analysis on KiCad pcb boards.
Manufacturing Output Tools
- [kiplot](https://github.com/johnbeard/kiplot) - A python module and program that lets you run and script KiCad's various manufacturing outputs such as Gerbers and other plots in a configurable way.
- [kibot](https://github.com/INTI-CMNB/kibot) - A much more advanced fork of kiplot.
- [kicad-exports](https://github.com/nerdyscout/kicad-exports) - Auto generate files (schematics, gerbers, BoM, plots, 3D model) for any KiCAD project. You could run it locally or on every git push with Github Actions.
- [obra/kicad-tools](https://github.com/obra/kicad-tools) - Productivity-enhancing tools primarily focused on automating generation of fabrication outputs and commandline productivity for projects tracked in git.
- [GerberZipper](https://github.com/g200kg/kicad-gerberzipper) - A plugin that plots Gerber-files and Zip it for a specified PCB manufacturer.
- [PcbDraw](https://github.com/yaqwsx/PcbDraw) - Script that converts PCB to nice looking 2D drawing.
- [Board2Pdf](https://gitlab.com/dennevi/Board2Pdf/) - A plugin which creates customized high resolution searchable pdf files from the PCB.
- [gerber_to_order](https://github.com/asukiaaa/gerber_to_order) - A plugin that creates zip compressed gerber files for PCB manufacturers.
- [PCBWay Plug-in for Kicad](https://github.com/pcbway/PCBWay-Plug-in-for-Kicad) Plugin to

automate generation of Gerber files, IPC-Netlist file, BOM file and Pick-n-Place file in the appropriate format for PCB manufacturing and assembly at [PCBWay](https://www.pcbway.com/).

- [KiCAD JLCPCB tools](https://github.com/Bouni/kicad-jlcpcb-tools) Plugin to automate generation of Gerber files, Excellon files, BOM file, CPL file in the appropriate format for PCB manufacturing and assembly at [JLCPCB](https://jlcpcb.com/).

- [KiCAD Test Points](https://github.com/snhobbs/kicad-testpoints) CLI to create test-point reports for making bed-of-nails jigs. Allows any pad to be set as a testpoint. Reports are generated in the [JigsApp](https://www.thejigsapp.com) format. Also available as a [PCM plugin](https://github.com/TheJigsApp/kicad-testpoints-pcm).

Version Control Tools
- [KiCad-Diff](https://github.com/Gasman2014/KiCad-Diff) - Python3 script for performing image diffs between pcbnew layout revisions in Git, SVN and Fossil VCS. Recent SVG based diff for significant speed improvements.
- [Massaging your git for kicad](https://jnavila.github.io/plotkicadsch/) - Instructions how to integrate KiCad with Git VCS
- [PlotKicadsch](https://github.com/jnavila/plotkicadsch) - PlotKicadsch is a small tool to export Kicad Sch files to SVG pictures.
- [Scripting KiCad Pcbnew exports](https://scottbezek.blogspot.si/2016/04/scripting-kicad-pcbnew-
- How to plot properly formated SVG files from pcbnew for VCS diff-ing
- [Improving open source hardware: Visual diffs](https://www.evilmadscientist.com/2011/improving-open-source-hard
- Using ImageMagick for visual diff file generation
- [KiCAD to SVG Converter](https://github.com/jmwright/oshw-code/tree/master/kicad_to_s
- Scripts for headless SVG generation of schematic files using eeschema.
- [kiri](https://github.com/leoheck/kiri) - Kicad Revision Inspector (KiRI) that combines PlotKicadSch and KiCad-Diff into a single website, for Git repositories, having a visual and interactive revision system for schematics and layouts.
Half-Baked Tools
If you have an interesting tool that's not quite ready for prime-time, post it here!
Maybe someone else can move it forward or use it as a starting point for their own tool.
- [footwork](https://github.com/monostable/footwork) - Unfinished footprint editor that tries to mix the s-expression footprint format with Racket (Scheme) to programmatically create footprints.
- [fpmagic](http://fpmagic.engineerjs.com/) - Web based, Chrome only, SMD only experimental footprint editor. Draw footprints using the

constraints from a datasheet drawing.
Plumbing
These are libraries/packages/modules that can help when creating tools like the ones listed above.
- [KinJector](https://github.com/devbisme/kinjector) - Inject/eject JSON/YAML data to/from KiCad Board files. Primarily used to read/write design rules, net classes, net class assignments, and part (X,Y)/orientation/top-bottom positions.
- [kinparse](https://github.com/devbisme/kinparse) - A parser for KiCad schematic netlist files that are output by EESCHEMA. Just pass a file containing a netlist to the `parse_netlist()` function and it will deliver a [pyparsing](https://pypi.python.org/pypi/pyparsing) object containing all the netlist's information.

- [pykicad](https://github.com/dvc94ch/pykicad) - The aim of this project is to provide high quality and well tested support for reading and writing KiCad file formats.
- [kicad-python](https://github.com/pointhi/kicad-python) - An abstraction layer for the KiCad python interface. (Be aware this is in initial development and the interface can change anytime!)
- [pykicadlib](https://code.fueldner.net/opensource/pykicadlib) - A Python library to read and write KiCAD footprints and schematic files.
- [kicad-utils](https://github.com/cho45/kicad-utils) - KiCAD library / schematic / pcb parser and plotter written in TypeScript (JavaScript)
- [KiCad-RW](https://github.com/FabriceSalvaire/kicad-rw) - A Python library to read/write KiCad 6 Sexpr file format. In addition this library can compute the netlist of a circuit. (It actually only implements a `.kicad_sch` reader)
Cheatsheets
These are handy guides for using KiCad and related tools.
- [KiCad Cheatsheet](https://github.com/KiCad/kicad-doc/blob/master/src/cheatshe
- Lists common operations and keyboard shortcuts for KiCad.
- [KiCad StepUp Cheatsheet](https://github.com/easyw/kicadStepUpMod/raw/master/demo/kic
- A set of concise descriptions on how to do mechanical CAD tasks on KiCad PCBs with the StepUp tool.
- [Git Cheatsheet](https://www.git-tower.com/learn/cheat-sheets/git)
- Summarizes common operations on Git repositories that are often used to store KiCad libraries and projects.
License

[![CC0](http://mirrors.creativecommons.org/presskit/buttons/88x31/svg/c

acknowledgement.md

```
530: # Acknowledgements
531: I created this repo of KiCad 3rd-party tools by mercilessly copying
532: everything from
533: [this repo of embedded system
534: resources](https://github.com/embedded-boston/awesome-embedded-systems)
535: and making a few small changes.
536: Thanks [cwoodall](https://github.com/cwoodall)!
```

contributing.md

```
537: # Contribution Guidelines
538: First of all, thank you for making a suggestion or addition to this
539: list!
540: To make the process as easy as possible (for you and me), please read
541: the [guidelines](#guidelines) and [submission
542: procedure](#how-to-add-something-to-this-list) below.
543: ## Table of Contents
544: - [Contribution Guidelines](#contribution-guidelines)
545: - [Table of Contents](#table-of-contents)
546: - [Guidelines](#guidelines)
547: - [How to Add Something to This
548: List](#how-to-add-something-to-this-list)
549: - [Updating Your Pull Request](#updating-your-pull-request)
550: ## Guidelines
551: Please ensure your pull request (PR) adheres to the following
552: guidelines:
553: - Search previous suggestions/additions before making a new one, as
554: yours may be a duplicate.
555: - Make sure your addition is useful before submitting. That implies
556: it has enough content and a good, succinct description.
557: - Make an individual PR for each addition.
558: - Use [title-casing](http://titlecapitalization.com) (AP style).
559: - Use the following format: `[List Name](link)`
560: - Link additions should be added to the **bottom** of the relevant
561: category. **The newest tools are always closer to the botttom!**
562: - If you have added **new features** to your tool, you may move its
563: entry to the bottom of the list, or you may leave it in its current
564: position.
565: - If your tool addition supports the new KiCad V6, please add this
566: badge at the beginning of your entry:
567: ![](https://img.shields.io/badge/V6-%20KiCad-blue)
568: - New categories or improvements to the existing categorization are
569: welcome.
570: - Check your spelling and grammar.
571: - Make sure your text editor is set to remove trailing whitespace.
572: - **The PR and commit should have a useful title.** PRs with `Update
573: readme.md` as their title will be closed right away.
574: - The body of your commit message should contain a link to the
575: repository.
576: ## How to Add Something to This List
577: If you have something to add to this list, here's how to do it.
578: You'll need a [GitHub account](https://github.com/join)!
579: 1. Access the [this list's GitHub
580: page](https://github.com/devbisme/kicad-3rd-party-tools).
581: 2. Click on the `README.md` file: ![Step 2 Click on
582: README.md](view-readme.png)
583: 3. Now click on the edit icon. ![Step 3 - Click on
584: Edit](start-editor.png)
585: 4. You can start editing the text of the file in the in-browser
```

editor. Make sure you follow the guidelines above. You can use [GitHub Flavored Markdown](https://help.github.com/articles/github-flavored-markdown/).

5. Say why you're proposing the changes, and then click on "Propose file change".

6. Submit the [pull request](https://help.github.com/articles/using-pull-requests/)!

Updating Your Pull Request

Sometimes, a maintainer of this list will ask you to edit your Pull Request before it is included. This is normally due to spelling errors or because your PR didn't match the list's guidelines.

[Here](https://github.com/RichardLitt/docs/blob/master/amending-a-commi is a write up on how to change a PR, and the different ways you can do that.