Better manage KiCAD projects using Git

Usama Abid Follow
Feb 11 · 5 min read

Git version control with KiCAD — InventHub

Hardware development can get messy. If we're not paying attention, at the end of the day, we are left with a project folder containing everything from schematics & layouts to gerbers, BoM, 3D models, firmware and what not! This folder is then either passed around in a USB or constantly uploaded to and downloaded from

google drive just to keep working as a team.

In an effort to end this painful cycle, let's take a page from the software development world and start using git. With git, you can track the entire development history of your project automatically and effectively collaborate with everyone on your team in real time. You can use any git platform you prefer but keep in mind that most git platforms are tailored to software. Except InventHub, which lets you examine your designs in any web browser, visually track the project as your team makes modifications and much more!

The purpose of this article, however, is to devise a practical strategy using KiCAD with git (in general) for rapid development. So, to begin:

Project Repository Structure

We need to create general folders for each category of our design and name them descriptively as *"Hardware", "Firmware", "Docs", "Production", "Simulation"* and *"CAD*". But they don't need to be *"Libraries_v1_001"* as git will take care of the *"_v1_001"* part by keeping an exact record of each iteration.

- Hardware — contains the KiCAD Project, Schematics & Layout and Project Libraries
- Firmware — contains any software developed for the Hardware
- Docs — contains any data sheets or documentations
- Production — contains the gerber files, BOM or anything required by the fabrication houses

- Simulation — contains any simulation files and generated results
- CAD — contains the 3D models and mechanical designs for enclosures or support

All these folders are kept in the root folder which must have the same name as the project repository. It is best practice to name it after the hardware under development in that repository. For example, *"A64-OlinuXino"*.

KiCAD Project

The KiCAD project would be initialized in the subfolder named "Hardware". The name of the project must be named after the hardware you need to develop. Upon creating a new project, KiCAD creates several different files throughout the development of the board, many of which are temporary files and unimportant. You can refer to the official KiCAD documentation to more detail. But the files that your KiCAD project really cares about are:

Project Manager File:

```
*.pro — a small file containing the component library lists for
the current project along with a few more parameters.
```

Schematic Files (eeschema):

```
*.sch — the schematic files of the project

*_cache.lib — this is a local copy of all the symbols used in
the project. It contains each and every component used in the
schematic and it is very important to view schematic properly.
```

PCB Layout (pcbnew):

```
*.kicad_pcb — the pcb layout for the board design. It used to
be a .brd file for the older versions of KiCAD which has been
deprecated since version 4
```

Special File:

```
*.cpm — used for component symbol association in schematic file
with the relevant footprints in the pcb_layout. It is used to
track changes from eeschema (the schematics tool) to pcbnew
(PCB Layout tool) or vice versa.
```

We'll discuss project libraries shortly, but any other files in the project folder are temporary files which should be ignored by the version control system. To do that, we need to create an empty file in the root folder named ".gitignore" and list the extensions for all the files we want git to ignore. Generally, the .gitignore file should look like this:

```
# Ignore list for KiCAD Projects

# Temporary files
*.000
*.bak
*.bck
*.kicad_pcb-bak
*.sch-bak
*~

_autosave-*
*.tmp
*-save.pro
*-save.kicad_pcb
fp-info-cache

# Netlist files (exported from Eeschema)
*.net

# Autorouter files (exported from Pcbnew)
*.dsn
*.ses
```

Project Libraries:

There are as many strategies for managing libraries as there are hardware developers. While each engineer is entitled to his own strategy that works best for him, a very convenient way of managing libraries is to gather all of them in a subfolder named “Libraries” located in the KiCAD project folder. If you are more comfortable with using git version control, you can go ahead and initiate a submodule for global library management. A strategy we previously discussed in this article. However, simply copying and pasting libraries in the “Libraries” folder also works well. The library files are:

For Schematics:

```
*.lib — containing the symbol description, pins and graphical
information i.e. shape, size or filled color.

*.dcm — these the library documentations for each symbol
```

For PCB Layouts:

```
*.pretty — Footprint library folders. The folder itself is the
library.

*.kicad_mod — Footprint files, containing one footprint
description each.
```

To configure the Libraries, you can go to *Manage Symbol Libraries* under *Preferences* and add the following path in the *Project Specific Libraries.* Replace the *symbol_name.lib* with the filename of the symbol you want to use in your project.

```
${KIPRJMOD}/Libraries/symbol_name.lib
```

You can go to *Manage Footprint Libraries* also under *Preferences* and add the following path in the *Project Specific Libraries.* Just replace the *footprint.pretty* with the filename of the footprint library you want to use in your project.

```
${KIPRJMOD}/Libraries/symbol_name.lib
```

The *${KIPRJMOD}* is an environment variable defined by KiCAD which always represents the current location of your project. So, no matter how many engineers clone the project repository and where they keep it on their local system, the libraries would remain associated with the designs. If you ever need to update any symbols or footprints, just get the latest versions and paste them in our Libraries folder. Make sure the filenames are the same and voila! The project still works perfectly.

Conclusion:

Keeping track of each iteration manually, can be confusing and prone to errors. Git helps you by making the project history automatically as you push designs and making the latest files instantly available for your team.

By utilizing git you better manage your project history:

1. Accurate history for development process
2. Real Time collaboration with team
3. Improvise rapidly or develop multiple product variations in parallel using branches
4. Central Repositories for reusable components i.e. libraries or template boards.

With this approach, you can better manage your KiCAD projects,

while avoiding the hassle to save tens of iterations along with documentation. With InventHub it gets even better, visually review changes, discuss updates on the designs and keep the process more streamlined. 😎

I hope you try this. If you do, or if you have any questions, please do get in touch: @usamaabidj, usama@inventhub.io

Kicad Git Version Control Inventhub