KiCad

December 21, 2020

Contents

Reference manual

Contributors

Jean-Pierre Charras, Fabrizio Tappero.

Feedback

Please direct any bug reports, suggestions or new versions to here:

- About KiCad document: https://gitlab.com/kicad/services/kicad-doc/issues
- About KiCad software: https://gitlab.com/kicad/code/kicad/issues
- About KiCad translation: https://gitlab.com/kicad/code/kicad-i18n/issues

Software version

KiCad 6.0

Chapter 1

Introduction

KiCad is an open-source software tool for the creation of electronic schematic diagrams and PCB artwork. Beneath its singular surface, KiCad incorporates an elegant ensemble of the following software tools:

- **KiCad**: Project manager
- **Eeschema**: Schematic editor and component editor
- **Pcbnew**: Circuit board layout editor and footprint editor
- **GerbView**: Gerber viewer

3 utility tools are also included:

- **Bitmap2Component**: Component maker for logos. It creates a schematic component or a footprint from a bitmap picture.
- **PcbCalculator**: A calculator that is helpful to calculate components for regulators, track width versus current, transmission lines, etc.
- **Page Layout Editor**: A tool to create drawing templates (worksheets) to be used on schematics and PCBs.

These tools are usually run from the project manager, but can be also run as stand-alone tools.

KiCad does not present any board-size limitation and it can handle up to 32 copper layers, 14 technical layers and 13 auxiliary layers.

KiCad can create all the files necessary for building printed circuit boards, including:

- Gerber files for photo-plotters
- drilling files
- component location files

Being open source (GPL licensed), KiCad represents the ideal tool for projects oriented towards the creation of electronic hardware with an open-source flavour.

KiCad is available for Linux, Windows and Apple macOS.

1.1 System Requirements

KiCad is capable of running on a wide variety of hardware and operating systems, but some tasks may be slower or more difficult on lower-end hardware. For the best experience, a dedicated graphics card and display with 1920x1080 or higher resolution is recommended.

Please check the KiCad website for the latest system requirements: https://kicad-pcb.org/help/system-requirements/

1.2 KiCad files and folders

KiCad creates and uses files with the following specific file extensions (and folders) for schematic and board editing.

Project manager file:

*.kicad_pro	Project file, containing settings that are shared between the schematic and PCB
*.pro	Legacy (KiCad 5.x and earlier) project file. Can be read and will be converted to a *.kicad_pro* file by the project manager.

Schematic editor files:

*.kicad_sch	Schematic files containing all info and the components themselves.
*.kicad_sym	Schematic symbol library file, containing the component descriptions: graphic shape, pins, fields.
*.sch	Legacy (KiCad 5.x and earlier) schematic file. Can be read and will be converted to a *.kicad_sch* file on write.
*.lib	Legacy (KiCad 5.x and earlier) schematic library file. Can be read but not written.
*.dcm	Legacy (KiCad 5.x and earlier) schematic library documentation. Can be read but not written.
*-cache.lib	Legacy (KiCad 5.x and earlier) schematic component library cache file. Required for proper loading of a legacy schematic (*.sch*) file.
sym-lib-table	Symbol library list (*symbol library table*): list of symbol libraries available in the schematic editor.

Board editor files and folders:

*.kicad_pcb	Board file containing all info but the page layout.
*.pretty	Footprint library folders. The folder itself is the library.
*.kicad_mod	Footprint files, containing one footprint description each.
*.kicad_dru	Design rules file, containing custom design rules for a certain *.kicad_pcb* file.
*.brd	Legacy (KiCad 4.x and earlier) board file. Can be read, but not written, by the current board editor.
*.mod	Legacy (KiCad 4.x and earlier) footprint library file. Can be read by the footprint or the board editor, but not written.
fp-lib-table	Footprint library list (*footprint library table*): list of footprint libraries available in the board editor.

fp-info-cache	Cache to speed up loading of footprint libraries.

Common files:

*.kicad_wks	Page layout (drawing border and title block) description file
*.net	Netlist file created by the schematic, and read by the board editor. This file is associated to the .cmp file, for users who prefer a separate file for the component/footprint association.
*.kicad_prl	Local settings for the current project, helps Kicad remember the last used settings such as layer visibility or selection filter. May not need to be distributed with the project or put under version control.

Other files:

*.cmp	Association between components used in the schematic and their footprints. It can be created by Pcbnew and imported by Eeschema. Its purpose is to import changes from Pcbnew to Eeschema, for users who change footprints inside Pcbnew (for instance using *Exchange Footprints* command) and want to import these changes in schematic.

Other files:

They are generated by KiCad for fabrication or documentation.

*.gbr	Gerber files, for fabrication.
*.drl	Drill files (Excellon format), for fabrication.
*.pos	Position files (ASCII format), for automatic insertion machines.
*.rpt	Report files (ASCII format), for documentation.
*.ps	Plot files (Postscript), for documentation.
*.pdf	Plot files (PDF format), for documentation.
*.svg	Plot files (SVG format), for documentation.
*.dxf	Plot files (DXF format), for documentation.
*.plt	Plot files (HPGL format), for documentation.

Storing and and sending KiCad files

KiCad schematic and board files contain all the schematic symbols and footprints used in the design, so you can back up or send these files by themselves with no issue. Some important design information is stored in the project file (*.kicad_pro*), so if you are sending a complete design, make sure to include it.

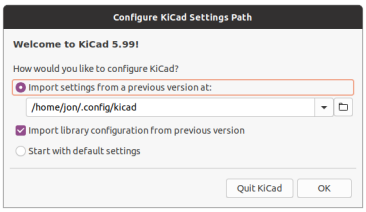
Some files, such as the project local settings file (*.kicad_prl*) and the *fp-info-cache* file, are not necessary to send with your project. If you use a version control system such as Git to keep track of your KiCad projects, you may want to add these files to the list of ignored files so that they are not tracked.

Chapter 2

Installing and Upgrading KiCad

2.1 Importing settings

Each major release of KiCad has its own configuration, so that you may run multiple KiCad versions on the same computer without the configurations interfering. The first time you run a new version of KiCad, you will be asked how to initialize the settings:



If a previous version of KiCad is detected, you will have the option to import the settings from that version. The location of the previous configuration files is detected automatically, but you may override it to choose another location if desired.

By default, the schematic symbol and footprint library tables from the previous version of KiCad will also be imported. If you would like to start with no library configuration, uncheck the **Import library configuration from previous version** checkbox.

You may also choose to start with default settings if you do not want to import settings from a previous version.

KiCad stores the settings files in a folder inside your user directory. Each KiCad version will store its settings in a subfolder of that folder (except for KiCad 5.1 and earlier, which did not use subfolders). Those folders are:

Windows	`%APPDATA%\kicad`
Linux	`~/.config/kicad`
Mac OS	`/Users/<username>/Library/Preferences/kicad`

2.2 Migrating files from previous versions

Modern versions of KiCad can open files created in earlier versions, but can only write files in the latest formats. This means that in general, there are no special steps to migrate files from a previous version besides opening the files. In some cases, the file extension for a file has changed from one KiCad version to the next. After opening these files, they will be saved in the new format with the new file extension. The old files will not be deleted automatically.

In general, files created or modified by one version of KiCad **cannot** be opened by older versions of KiCad. For this reason, it is important to keep backup copies of your projects when testing a new KiCad release, until you are confident that you will not need to use the older KiCad version anymore.

Chapter 3

Using KiCad project manager

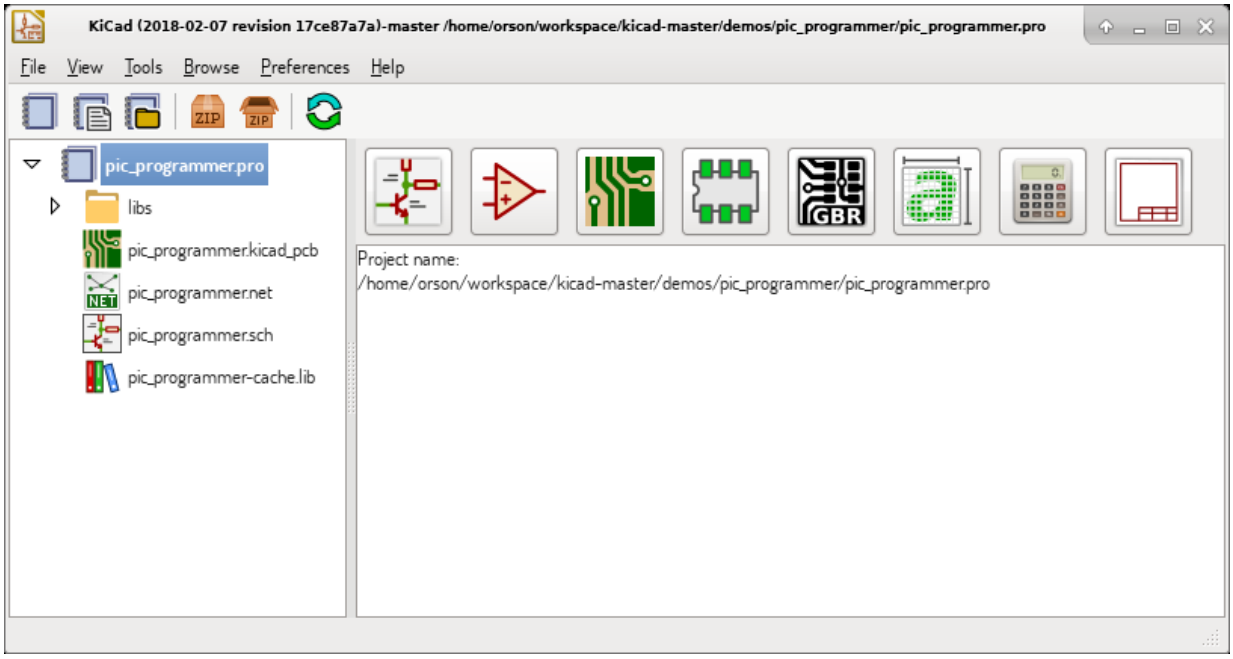
KiCad project manager (kicad or kicad.exe) is a tool which can easily run the other tools (schematic and board editors, Gerber viewer and utility tools) when creating a design.

Running the other tools from KiCad manager has some advantages:

- cross probing between schematic editor and board editor.
- synchronization of the design between the schematic editor and board editor (without creating netlist files)

KiCad currently only supports having one project open at a time. When running the schematic and board editors from the KiCad project manager, you can only edit the schematics and board associated with the open project. When these tools are run in *stand alone* mode, you can open any file in any project, but cross probing between tools can give strange results.

3.1 Project manager window



The KiCad project manager window is composed of a project tree view, a launch pane containing buttons used to run the various software tools, and a message window. The menu and the toolbar can be used to create, read and save project files.

3.2 Tool launcher

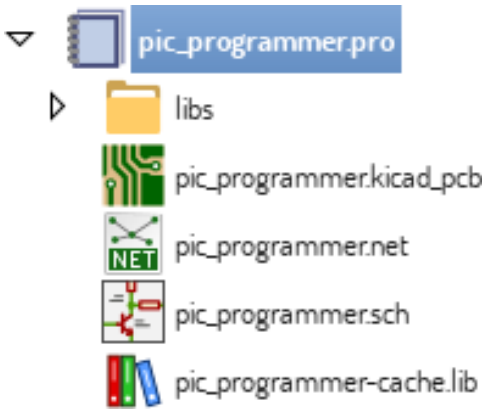
The tool launcher opens the different KiCad tools from the project manager. When opening the schematic or board editor, the project files will automatically be loaded.

The 8 buttons in the launcher correspond to the following commands (1 to 8, from left to right):

1	**Eeschema**	Schematic editor.
2	**LibEdit**	Schematic symbol editor and symbol library manager.
3	**Pcbnew**	Board layout editor.
4	**FootprintEditor**	Footprint editor and footprint library manager.
5	**Gerbview**	Gerber file viewer. It can also display drill files.
6	**Bitmap2component**	Tool to build a footprint or a component from a B&W bitmap image to create logos.
7	**Pcb Calculator**	Tool to calculate track widths, and many other things.
8	**Page Layout Editor**	Page layout (drawing border and title block) editor

3.3 Project tree view

The tree view shows a list of files inside the project folder.



Double-clicking on the schematic file runs the schematic editor, in this case opening the file **pic_programmer.kicad_sch**.

Double-clicking on the board file runs the layout editor, in this case opening the file **pic_programmer.kicad_pcb**.

Right clicking on any of the files in the project tree allows generic file manipulation.

Note

Only files that KiCad understands how to open are displayed in the project tree view.

3.4 Top toolbar

KiCad top toolbar allows for some basic file operations:

	Create a new project. If the default template file (kicad.kicad_pro) is found in **kicad/template**, it is copied into the working directory.
	Create a new project from an existing template.
	Open an existing project.
	Create a zip archive of the whole project. This includes schematic files, libraries, PCB, etc.
	Extract a project zip archive into a directory. Files in the destination directory will be overwritten.
	Refresh the tree view, sometimes needed after a tree change.
	Open the project working directory in a file explorer.

3.5 Creating a new project

Most KiCad designs start with the creation of a project. There are two ways to create a project from the KiCad project manager: you may create an empty project, or create a project based on an existing template. This section will cover the creation of a new, empty project. Creating projects from templates is covered in the Project Templates section.

To create a new project, use the **New Project⋯** command in the **File** menu, click the **New Project** button in the top toolbar, or use the keyboard shortcut (`Ctrl+N` by default).

You will be prompted for a name to give your project. By default, a directory will be created for your project with the same name. For example, if you enter the name `MyProject`, KiCad will create the directory `MyProject` and the project file `MyProject/MyProject.kicad_pro` inside it.

If you already have a directory to store your project files in, you can uncheck the *Create a new directory for the project* checkbox in the **New Project** dialog.

Note

It is strongly recommended that you store each KiCad project inside its own directory.

Once you select the name of your project, KiCad will create the following files inside the project directory:

example.kicad_pro	KiCad project file.
example.kicad_sch	Main schematic file.
example.kicad_pcb	Printed circuit board file.

3.6 Importing a project from another EDA tool

KiCad is able to import files created by some other software packages. Currently the following types of project are supported:

*.sch, *.brd	Eagle 6.x or newer (XML format)
*.csa, *.cpa	CADSTAR archive format

To import a project from one of these tools, choose the appropriate option from the **Import Non-KiCad Project** submenu of the **File** menu.

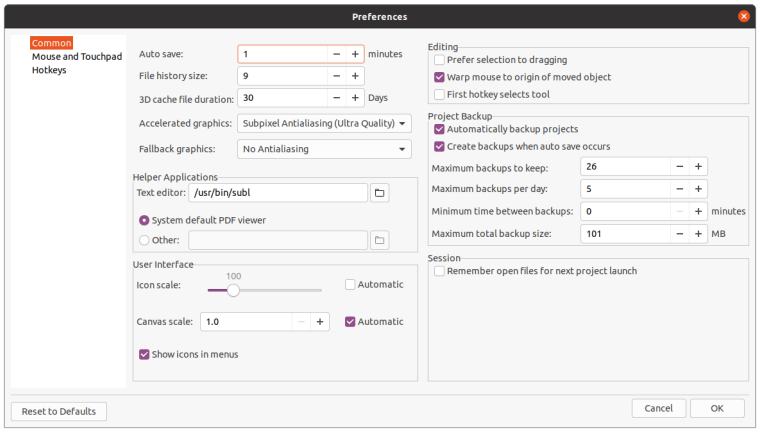
You will be prompted to select either a schematic or a board file in the import file browser dialog. The imported schematic and board files should have the same base file name (e.g. project.sch and project.brd). Once the requested files are selected, you will be asked to select a directory to store the resulting KiCad project.

Chapter 4

KiCad configuration

The KiCad preferences can always be accessed from the **Preferences** menu, or by using the hotkey (default `Ctrl+,`). The Preferences dialog is shared between the running KiCad tools. Some preferences apply to all tools, and some are specific to a certain tool (such as the schematic or board editor).

4.1 Common preferences



Auto save: When editing schematics and board files, KiCad can automatically save your work periodically. Set to `0` to disable this feature.

File history size: Configure the number of entries in the list of recently-opened files

3D cache file duration: KiCad creates a cache of 3D models in order to speed up the 3D viewer. You can configure how long to keep this cache before deleting old files.

Accelerated graphics antialising: KiCad can use different methods to prevent aliasing (jagged lines) when rendering using a graphics card. Different methods may look better on different hardware, so you may want to experiment to find the one that looks best to you.

Fallback graphics antialiasing: KiCad can also apply antialiasing when using the fallback graphics mode. Enabling this feature may result in poor performance on some hardware.

Text editor: Choose a text editor to use when opening text files from the project tree view.

PDF viewer: Choose a program to use when opening PDF files

Icon scale: Sets the size of the icons used in menus and buttons throughout KiCad. Choose *Automatic* to pick an appropriate icon scale automatically based on your operating system settings.

Canvas scale: Sets the scale of the drawing canvas used in the KiCad editors. Choose *Automatic* to pick an appropriate canvas scale automatically based on your operating system settings.

Show icons in menus: Enables icons in drop-down menus throughout the KiCad user interface.

Note

Icons in menus are not displayed on some operating systems.

Prefer selection to dragging: When enabled, clicking and dragging the mouse will always perform a selection operation, even when you start dragging on top of an object. When disabled, clicking and dragging on top of an object that can be moved will perform a move operation.

Warp mouse to origin of moved object: When enabled, the mouse cursor will be repositioned (warped) to the origin of an object when you start a move command on that object.

First hotkey selects tool: When disabled, pressing the hotkey for a command such as *Add Wire* will immediately start the command at the current cursor location. When enabled, pressing the hotkey the first time will just select the *Add Wire* tool but will not immediately begin a wire.

Automatically backup projects: When enabled, KiCad projects will be archived to ZIP files automatically according to the settings below. The archives will be stored in a subfolder of the project folder. Backups are created when saving files in the project.

Create backups when auto save occurs: When enabled, a backup will be created every time an automatic file save occurs (if the backup is permitted by the settings below). This setting has no effect if the auto save interval is set to 0 (disabled).

Maximum backups to keep: When creating a new backup, the oldest backup file will be deleted to keep the total number of backup files below this limit.

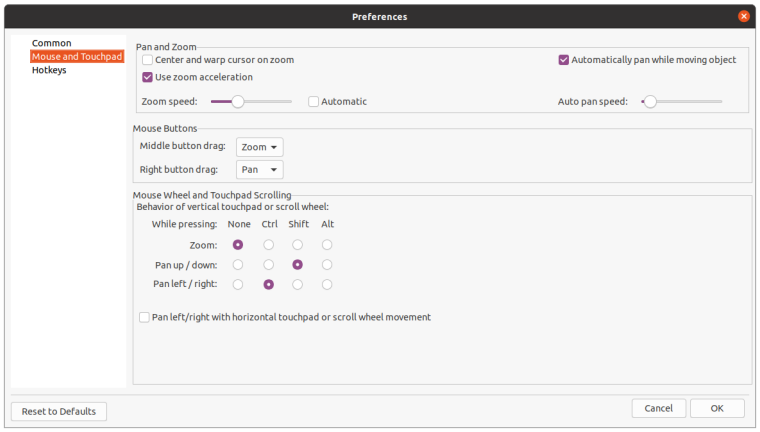
Maximum backups per day: When creating a new backup, the oldest backup file created on the current day will be deleted to stay below this limit.

Minimum time between backups: If backup is triggered (for example, by saving a board file), the backup will not be created if an existing backup file is newer than this limit.

Maximum total backup size: When creating a new backup file, the oldest backup files will be deleted to keep the total size of the backup files directory below this limit.

Remember open files for next project launch: When checked, KiCad will re-open the schematic and board editor if they were open the last time you closed the project manager.

4.2 Mouse and touchpad preferences



Center and warp cursor on zoom: When enabled, zooming using the hotkeys or mouse wheel will cause the view to be centered on the cursor location.

Use zoom acceleration: When enabled, scrolling the mouse wheel or touchpad faster will cause the zoom to change faster.

Zoom speed: Controls how much the zoom changes for a given amount of scrolling the mouse wheel or touchpad. Use *Automatic* to set a default value depending on your operating system.

Automatically pan while moving object: When enabled, the view can be panned while moving an object by moving close to the edge of the canvas.

Auto pan speed: Controls how fast the canvas pans while moving an object.

Mouse buttons: You can set the behavior of dragging the middle and right mouse buttons to zoom the view, pan the view, or have no effect.

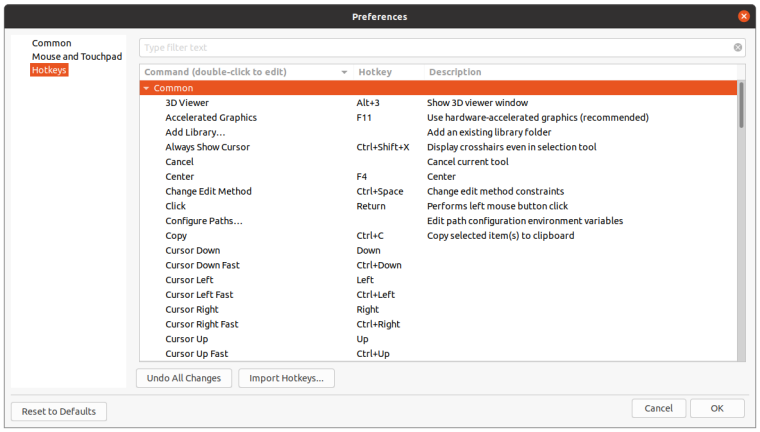
Note

The left mouse button is always used for selecting and manipulating objects.

Mouse wheel and touchpad scrolling: You can set the behavior of scrolling the mouse wheel or vertical motion of the touchpad while pressing certain modifier keys.

Pan lef/tright with horizontal touchpad or scroll wheel movement: When enabled, you can pan the view using the touchpad or horizontal scroll wheel (if present on your mouse).

4.3 Hotkey preferences



You can use this dialog to customize the hotkeys used to control KiCad. The hotkeys in the *Common* section are shared between every KiCad program. Hotkeys for each specific KiCad program are shown when that program is running. You can assign the same hotkey to a different action in different KiCad programs (for example, the schematic editor and the board editor), but you cannot assign a hotkey to more than one action in the same program.

There are many available commands, and so not all of them have a hotkey assigned by default. You can add a hotkey to any command by double-clicking on the command in the list. If you choose a hotkey that is already assigned to a different command, you can choose to use that hotkey on your chosen command, which will remove the hotkey assignment from the conflicting command.

Changes that you have made to hotkey assignments are shown with a `*` character at the end of the command name. You can undo changes to a specific command by right-clicking that command and selecting *Undo Changes*, or you can undo all changes with the button below the command list.

Importing hotkeys

Hotkey preferences are stored in `.hotkeys` files in the KiCad settings directory (see the Settings section for information about where the settings directory is on your operating system). If you have configured KiCad hotkeys the way you like on one computer, you can transfer that configuration to another computer by importing the appropriate `.hotkeys` file(s).

4.4 Paths configuration

In KiCad, one can define paths using an *environment variable*. A few environment variables are internally defined by KiCad, and can be used to define paths for libraries, 3D shapes, etc.

This is useful when absolute paths are not known or are subject to change (e.g. when you transfer a project to a different computer), and also when one base path is shared by many similar items. Consider the following which may be installed in varying locations:

- Schematic symbol libraries
- Footprint libraries

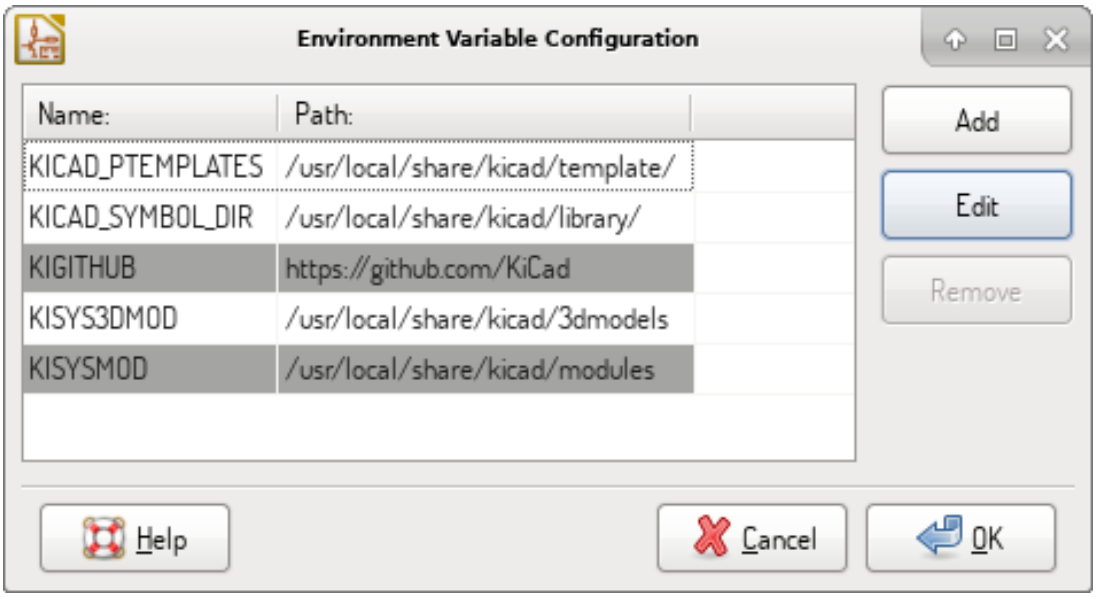
- 3D shape files used in footprint definitions

For instance, the path to the ***connect.pretty*** footprint library, when using the **KISYSMOD** environment variable, would be defined as ***${KISYSMOD}/connect.pretty***

The **Preferences** → **Configure Paths···** menu allows you to define paths for some built-in KiCad environment variables, and add your own environment variables to define personal paths, if needed.

KiCad environment variables:

KICAD_PTEMPLATES	DEPRECATED since version 5.0, use KICAD_TEMPLATE_DIR or KICAD_USER_TEMPLATE_DIR instead.
KICAD_SYMBOL_DIR	Base path of symbol library files.
KICAD_TEMPLATE_DIR	Location of project templates installed with KiCad.
KICAD_USER_TEMPLATE_DIR	Location of personal project templates.
KIGITHUB	Frequently used in example footprint library tables. If you are using this variable, it must be defined.
KISYS3DMOD	Base path of 3D shapes files, and must be defined because an absolute path is not usually used.
KISYSMOD	Base path of footprint library folders, and must be defined if an absolute path is not used in footprint library names.



Note also the environment variable **KIPRJMOD** is **always** internally defined by KiCad, and is the **current project absolute path**.

For instance, ***${KIPRJMOD}/connect.pretty*** is always the ***connect.pretty*** folder (the pretty footprint library) found inside **the current project folder**.

If you modify the configuration of paths, please quit and restart KiCad to avoid any issues in path handling.

4.5 Libraries configuration

The **Preferences** → **Manage Symbol Libraries···** menu let you manage the library list files called **symbol library table** (sym-lib-table).

Likewise, use the **Preferences** → **Manage Footprint Libraries···** menu to manage the library list files called **footprint library table** (fp-lib-table).

There are 2 library list files: the first (located in the user home directory) is global for all projects and the second (located in the project directory) is optional and specific to the project.

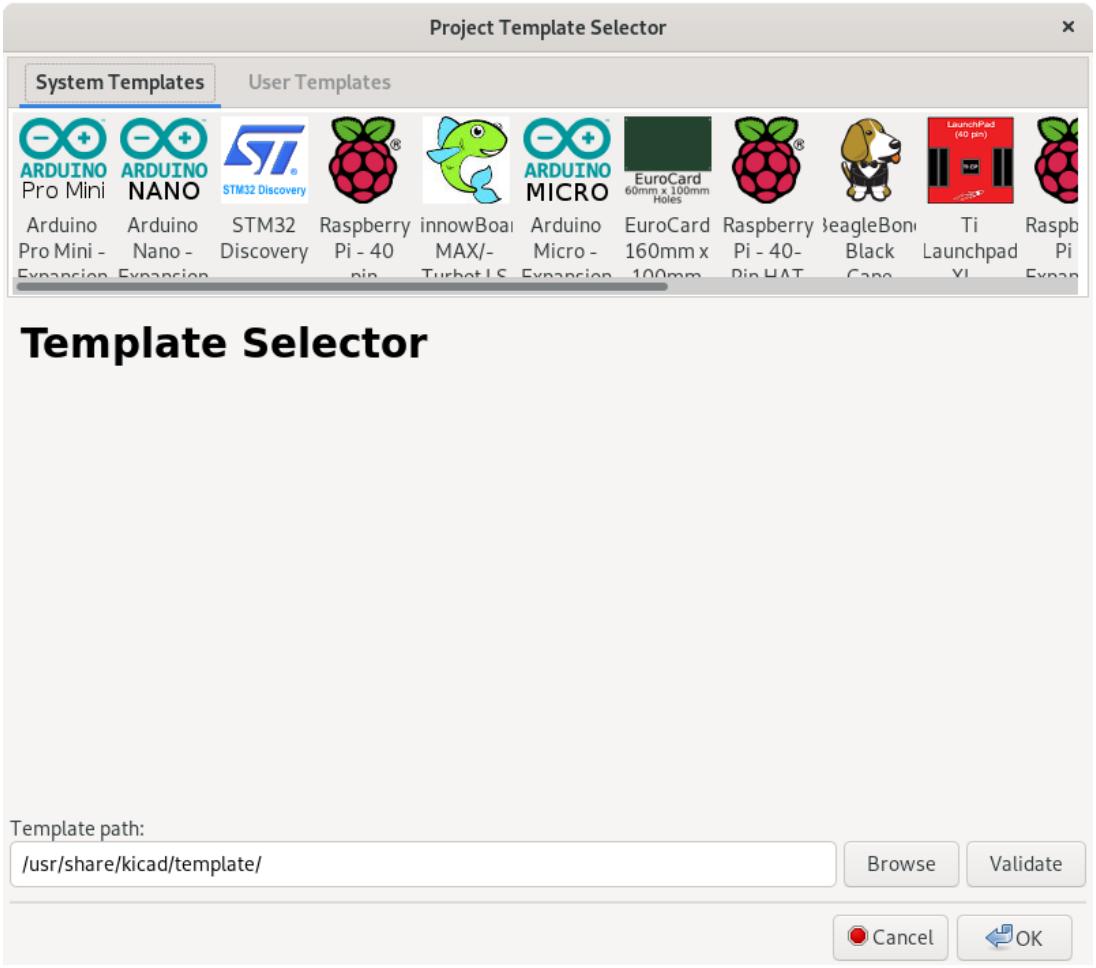
Chapter 5

Project templates

Using a project template facilitates setting up a new project with predefined settings. Templates may contain predefined board outlines, connector positions, schematic elements, design rules, etc. Complete schematics and/or PCBs used as seed files for the new project may even be included.

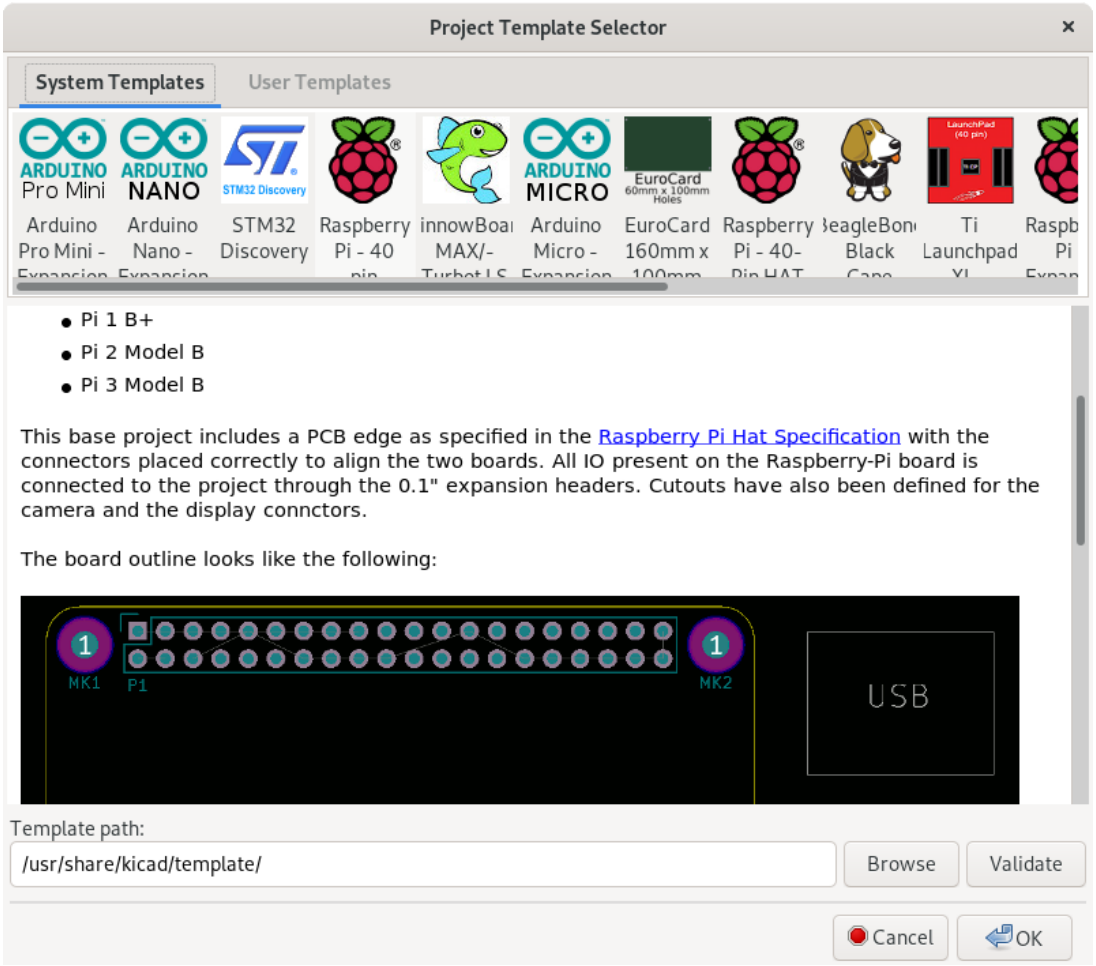
5.1 Using templates

The ***File → New Project → New Project from Template*** menu will open the Project Template Selector dialog:



A single click on a template' s icon will display the template information, and a further click on the OK button creates the new project. The template files will be copied to the new project location and renamed to reflect the new project' s name.

After selection of a template:



5.2 Template Locations:

KiCad looks for template files in the following paths:

- path defined in the environment variable KICAD_USER_TEMPLATE_DIR
- path defined in the environment variable KICAD_TEMPLATE_DIR
- System templates: <kicad bin dir>/../share/kicad/template/
- User templates:
  - Unix: ~/kicad/template/
  - Windows: C:\Documents and Settings\username\My Documents\kicad\template or C:\Users\username\Documents\k
  - Mac: ~/Documents/kicad/template/

5.3 Creating templates

The template name is the directory name where the template files are stored. The metadata directory is a subdirectory named **meta** containing files describing the template.

The metadata consists of one required file, and may contain optional files. All files must be created by the user using a text editor or previous KiCad project files, and placed into the required directory structure.

All files and directories in a template are copied to the new project path when a project is created using a template, except **meta**. Files and directories containing the template name will be renamed with the new project file name.

For example, creating a project called **newproject** from a template named **example**:

Files in template **example** directory	Files created in project **newproject** directory
example.kicad_pro	**newproject.kicad_pro**
example.kicad_sch	**newproject.kicad_sch**
example.kicad_pcb	**newproject.kicad_pcb**
example-first.kicad_sch	**newproject-first.kicad_sch**
second-example.kicad_sch	**second-newproject.kicad_sch**
third.kicad_sch	third.kicad_sch
third.kicad_pcb	third.kicad_pcb

A template does not need to contain a complete project, if a required project file is missing, KiCad will create it using its default create project behavior:

Files in template **example** directory	Files created in **newproject** directory
example.kicad_sch	**newproject.kicad_sch**
first-example.kicad_sch	**first-newproject.kicad_sch**
first-example.kicad_pcb	**first-newproject.kicad_pcb**
second-example.kicad_sch	**second-newproject.kicad_sch**
second-example.kicad_pcb	**second-newproject.kicad_pcb**
	newproject.kicad_pro (default)
	newproject.kicad_pcb (default)

As an exception to the template name renaming rule, if one project file (.kicad_pro) exists and its name doesn' t match the template name, KiCad will do the renaming based on that project file name instead:

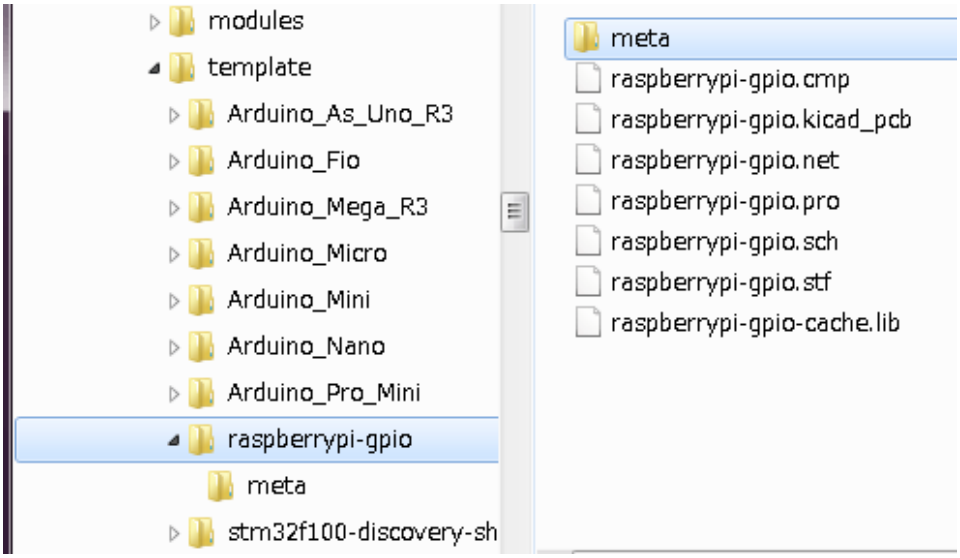
Files in template **example** directory	Files created in **newproject** directory
example.kicad_sch	example.kicad_sch
example.kicad_pcb	example.kicad_pcb
first-example.kicad_pro	**newproject.kicad_pro**
first-example.kicad_sch	**newproject.kicad_sch**
first-example.kicad_pcb	**newproject.kicad_pcb**
second-example.kicad_sch	second-example.kicad_sch
second-example.kicad_pcb	second-example.kicad_pcb

Note

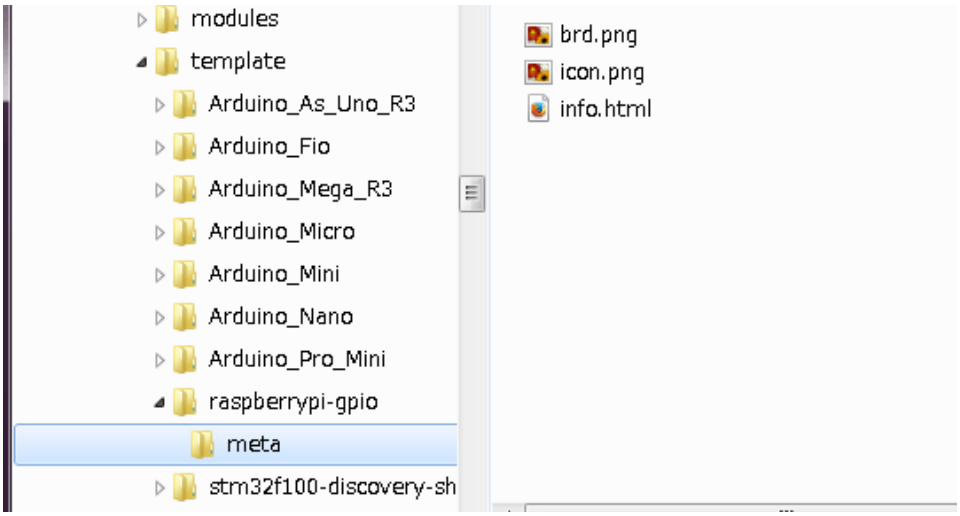
It is not recommended to create a template with multiple project files.

5.3.1 Template example

Here is an example showing project files for **raspberrypi-gpio** template:



And the metadata files:



5.3.2 Required File:

meta/info.html	HTML-formatted information describing the template.

The <title> tag determines the actual name of the template that is exposed to the user for template selection. Note that the project template name will be cut off if it' s too long.

Using HTML means that images can be easily in-lined without having to invent a new scheme. Only basic HTML tags can be used in this document.

Here is a sample **info.html** file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html;
charset=windows-1252">
<TITLE>Raspberry Pi - Expansion Board</TITLE>
</HEAD>
<BODY LANG="fr-FR" DIR="LTR">
<P>This project template is the basis of an expansion board for the
<A HREF="http://www.raspberrypi.org/" TARGET="blank">Raspberry Pi $25
ARM board.</A> <BR><BR>This base project includes a PCB edge defined
as the same size as the Raspberry-Pi PCB with the connectors placed
correctly to align the two boards. All IO present on the Raspberry-Pi
board is connected to the project through the 0.1&quot; expansion
headers. <BR><BR>The board outline looks like the following:
</P>
<P><IMG SRC="brd.png" NAME="brd" ALIGN=BOTTOM WIDTH=680 HEIGHT=378
BORDER=0><BR><BR><BR><BR>
</P>
<P>(c)2012 Brian Sidebotham<BR>(c)2012 KiCad Developers</P>
</BODY>
</HTML>
```

5.3.3 Optional Files:

meta/icon.png	A 64 x 64 pixel PNG icon file which is used as a clickable icon in the template selection dialog.

Any other image files used by **meta/info.html**, such as the image of the board file in the dialog above, are placed in this folder as well.