

A Tutorial on PCB Design

— using KiCad

Xiaoguang “Leo” Liu
University of California Davis
lxgliu@ucdavis.edu

Aug. 9th, 2015

Contents

1	PCB Basics	2
1.1	KiCad	3
2	Example 1: Arduino dice	4
2.1	Schematic Capture	4
2.2	Creating or Editing Schematic Symbols	10
2.3	PCB Layout	11
2.4	Associating schematic symbols with footprint	11
2.5	Creating or Editing Footprint	11
2.6	Generating Fabrication Files	11
3	Advanced Topics	11
3.1	Controlled Impedance Lines	11

1 PCB Basics

1.1 KiCad

In the early days, PCBs are designed and laid out literally by hand. See Fig. 1 for an example board from that era. As technologies developed, it became more common to do the job with the help of a computer. Today, there are numerous software tools for PCB design. On the high end, industry-grade packages, such as Cadence Allegro¹, Mentor Graphics Xpedition, and Altium Designer, offer extensive features and capabilities with a high price tag and often a very steep learning curve. On the lower end, popular choices include CadSoft EAGLE, ExpressPCB, and DesignSpark, all of which offer a reasonable set of features at an affordable price.

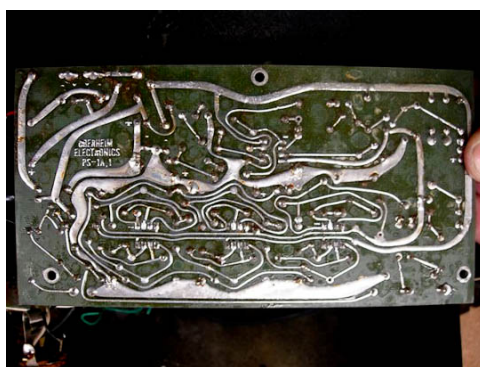


Figure 1: A vintage PCB laid out by hand.³

In recent years, KiCad has emerged as a popular open-source software package for designing and laying out PCBs.² KiCad is available on all three major personal computer operating systems, Windows, Linux, and Mac OS. Compared with the above mentioned software packages, KiCad is completely free of charge or any other limitation. Although KiCad is not as sophisticated as industry-level tools, it is capable of dealing with fairly complicated designs, and the active developer community is working hard to improve its capabilities. In fact, as of this writing, KiCad has not had an official stable release for the last two years because of the constant development progress being made. In this tutorial, we will use a recent build #6055, dated Aug. 8th, 2015.

Fig. 2 shows the main window of KiCad. The main window serves as a project management panel where you can launch the individual PCB tools.

¹UC Davis students have access to the full suite of Allegro PCB design tools through a donation from Cadence.

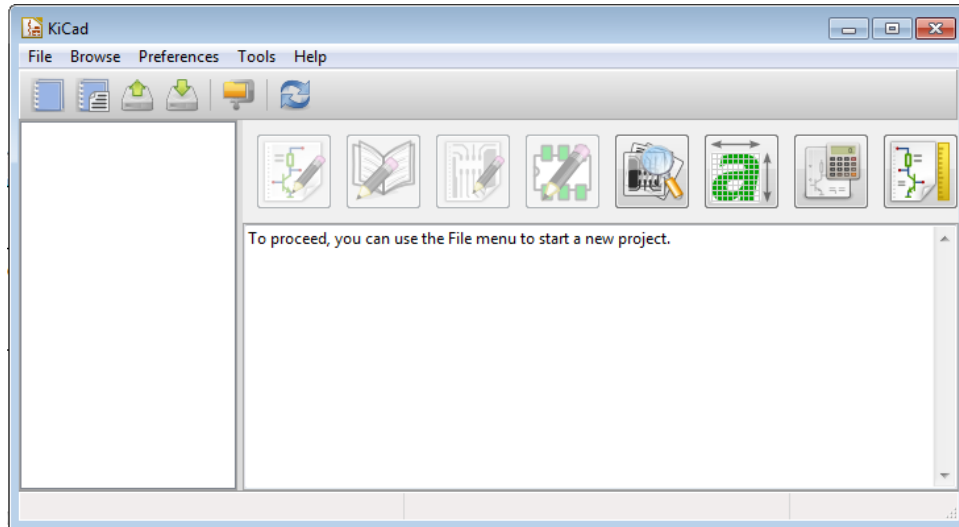


Figure 2: The main window of KiCad.

Table 1: Individual tools within KiCad.

	Eeschema: schematic editor/capture tool		Gerbrview: Gerber file viewer
	Schematic symbol editor		Bitmap2Component: A tool for creating component symbol from a picture
	Pcbnew: PCB layout tool		A calculator for common PCB design related calculations
	Component footprint editor		Schematic sheet layout editor

2 Example 1: Arduino dice

In the first example we will make a small electronic dice consisting of a ATmega328P microcontroller², a switch, a 7-segment LED display, and some misc resistors and capacitors. Every time you press and release the switch, the microcontroller will generate a random number (1–6) for the dice value and display it on the 7-segment LED. This example is a stripped down version of a project from PrinceTronics.¹

Table. 2 lists the components that are needed for this example.

2.1 Schematic Capture

1. Click on the Eeschema icon. A new schematic window should appear.

²The heart of the Arduino UNO platform.

Table 2: List of components for Example 1.

Item Description	Quantity	Digikey Part #
Arduino Uno board, DIP version	1	1050-1024-ND
16-MHz crystal oscillator	1	300-6034-ND
22-pF ceramic capacitor, SMD, 0603, 5%	2	445-1273-1-ND
1-uF ceramic capacitor, SMD, 0603	1	1276-1041-1-ND
10k-Ohm resistor, SMD, 0603, 1/10W, 5%	2	P10KGCT-ND
Push button switch, 0.05 A, 24 V	1	SW400-ND
7-segment 1-digit display, common cathode	1	516-2734-ND

2. Save your schematic design with the file name “arduino_dice.sch”.
3. The default library that comes with KiCad installation has schematic symbols for many ATmel micro-controllers, including the ATmega328P that is used on the Arduino platform. You can place the symbol on your schematic by the following steps.
 - a) Click on the “Place a component” button from the toolbar on the right side.

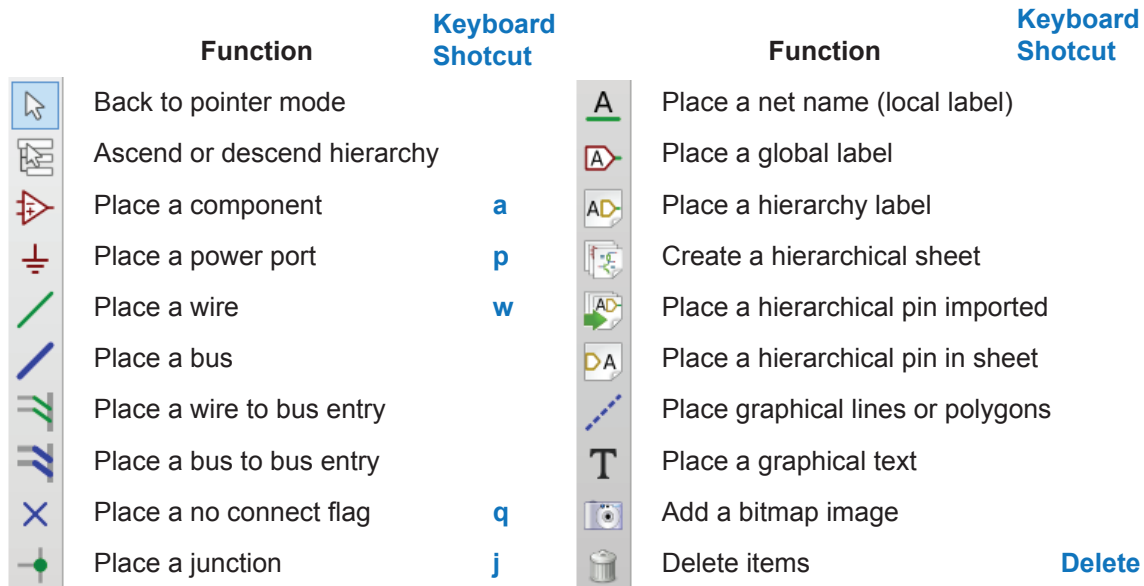


Figure 3: Eeschema toolbar icons.

- b) Click anywhere on the schematic, a dialog box should appear.
- c) We'll add our first item, the ATmega328P microcontroller, from the “atmel” library. Select the “atmel” entry, and click “OK”. A new dialog box appears for you select the particular device, the ATMEGA328P-P. Click “OK”. Note that if you already know the name of the component, you can simply start typing the name and Eeschema will filter out the components with the same initial characters. It wouldn't take long before you arrive at your desired component.

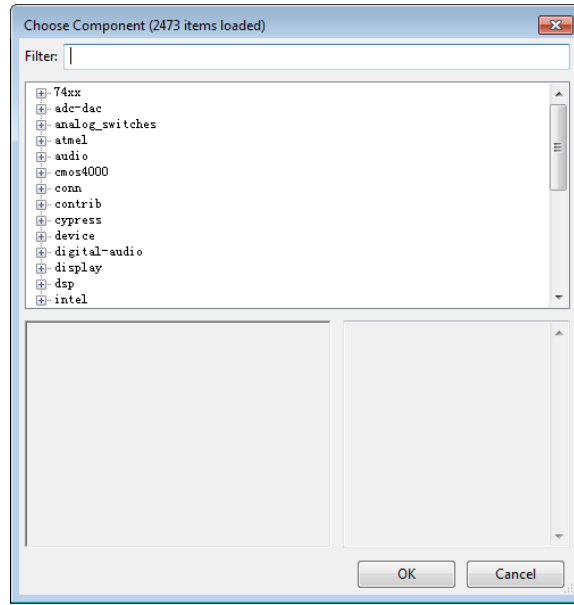


Figure 4: Place component dialog box.

- d) The ATmega328P symbol should now cling to your mouse cursor. Click on the schematic to place it at a location you like.
- e) A number of component editing operations are available by right clicking on the component. Some of the operations have keyboard shortcut. The general way to use the shortcut is to place the cursor on the component and press the corresponding shortcut key. Pressing the “ESC” key will cancel the current operation.
 - i. “Move component” will move the component and break all circuit connections to it. To retain the connections, use “Drag component”.
 - ii. “Orient component” has further options to rotate and mirror the component. Experiment the shortcuts by pressing “r” or “y” while placing the cursor on the component.
 - iii. “Edit component → Edit” will bring up a dialog box that allows you to edit all of the component properties.
 - A. The “Reference” and “Value” are the two properties that you are most likely to edit in this dialog box. “Reference” is the annotation of the component. In this case, it should read “IC1”. You may also write it as “IC?” where the “?” is a placeholder for a numeric value. KiCad can auto annotate the components and assign a unique value for each component; we will look at how to do this shortly.
 - B. The “Value” entry is usually used to mark the component value. For resistors, capacitors, and inductors for examples, the “Value” can simply be their corresponding resistance, capacitance, and inductance values. For this ATmega microcontroller we will simply use the “Value” to mark the components name.

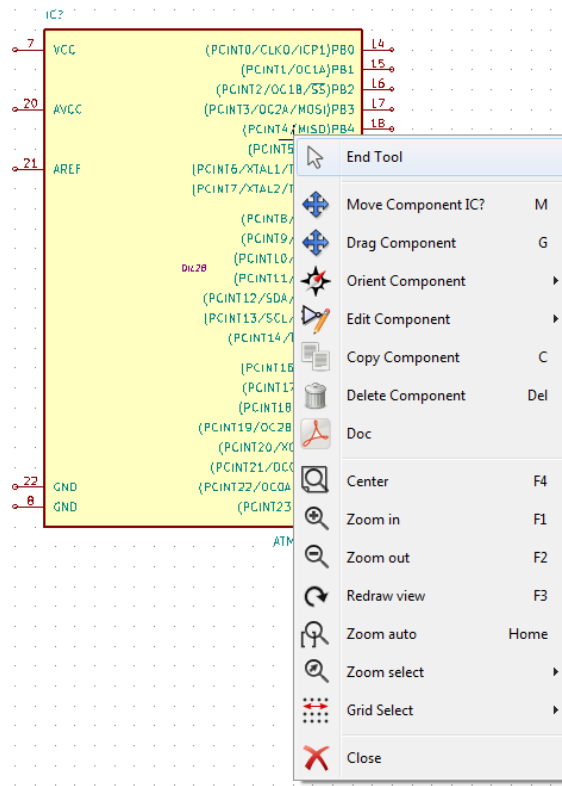


Figure 5: Edit component pop-up menu.

4. Follow a similar procedure to place the other circuit components. Table. 3 lists which library they belong to and their names in the library.

Note that the *Vcc* and *ground* symbols, and all other symbols related to providing power to the circuits, are organized into the *power* library. They can be accessed directly by clicking the *Place a power port* button from the toolbar on the right. Place a *Vcc* and several *GND* pins where necessary.

5. Connect the components together according to Fig. 6 by placing wires between corresponding pins. The components should be arranged to make connecting them together easier with less wire clutter. To start placing a wire, click on the “Place a wire” button, then click on the starting point, and finish by clicking on the endpoint of the wire. It is often easier to use the keyboard shortcut. First move your cursor to the starting point and press the “w” key. A wire is started at the cursor location. Move the cursor to the endpoint and click to finish the connection.
6. The schematic drawing can become difficult to read if there are too many wire crossovers. “Named netlist” can be used to alleviate the issue. Although our example circuit is quite simple and easy to read, we will still use it to illustrate how to “clean up” the schematic with named net.
 - a) Delete the wire between the ATmega328P’s *PB1* pin and the 7-segment LED’s *DP* pin.

Table 3: Schematic components for Example 1.

Component	Library	Name in Library	Value
ATmega328P	atmel	ATMEGA328P-P	
Two 22-pF capacitors	device	C	22 pF
One 1-uF capacitor	device	C	1 uF
One 16-MHz crystal oscillator	device	CRYSTAL	16 MHz
One 7-segment LED display	display	7SEGMENTS	
One push button	device	SW_PUSH	
Two 10-k resistors	device	R	10 k
2-pin header	conn	CONN_01X02	
Vcc	power	vcc	
Ground	power	GND	

- b) Draw a short section of wire on the ATmega328P's *PB1* pin; one of the ends of the wire is now floating.
 - c) Click the "Place a net name – local label" button and then click on the schematic. A dialog box will appear, input "DP" in the "Text" field, and click "OK". The text "DP" can now be seen to cling on the cursor.
 - d) Click on the floating terminal of the short wire on the *PB1* pin to finish naming a net. You should now see the text DP attached to the wire on the *PB1* pin; the little hollow square at the floating end of the wire has also disappeared.
 - e) Repeat steps b–d for the *DP* pin of the 7-segment LED. The ATmega328P's *PB1* pin and the 7-segment LED's *DP* pin are now connected by the net name "DP" even though there is no direct wire connection on the schematic view.
 - f) Repeat steps a–e for the connection between ATmega328P's *PB4* pin and the push button. Refer to the final schematic (Fig. 7) for how it looks.
7. In most circuits, the unused pins can be left floating. In this example, however, we will terminate all the unused pins by placing a no connect label on them; this tells KiCad to ignore these pins during the electrical rule check (ERC).
 8. The schematic capture is now almost done. Notice that the references to some of the components still have question marks. For example, the two capacitors connected to the crystal oscillator look identical to each other; we need to differentiate them. In KiCad, we do this by annotating the schematic, i.e. giving each component a unique identifier (reference). Annotation can be done manually by changing the "Reference" property of a component and making sure that each reference is unique, but it is much easier to let KiCad do the annotation automatically.
 - a) Click the "Annotate" button.
 - b) A dialog box appears. The options are all self-explanatory.
 - c) Click the "Annotate" button to finish. You must have noticed that you can "un-annotate" the schematic by clicking the "Clear annotation" button.

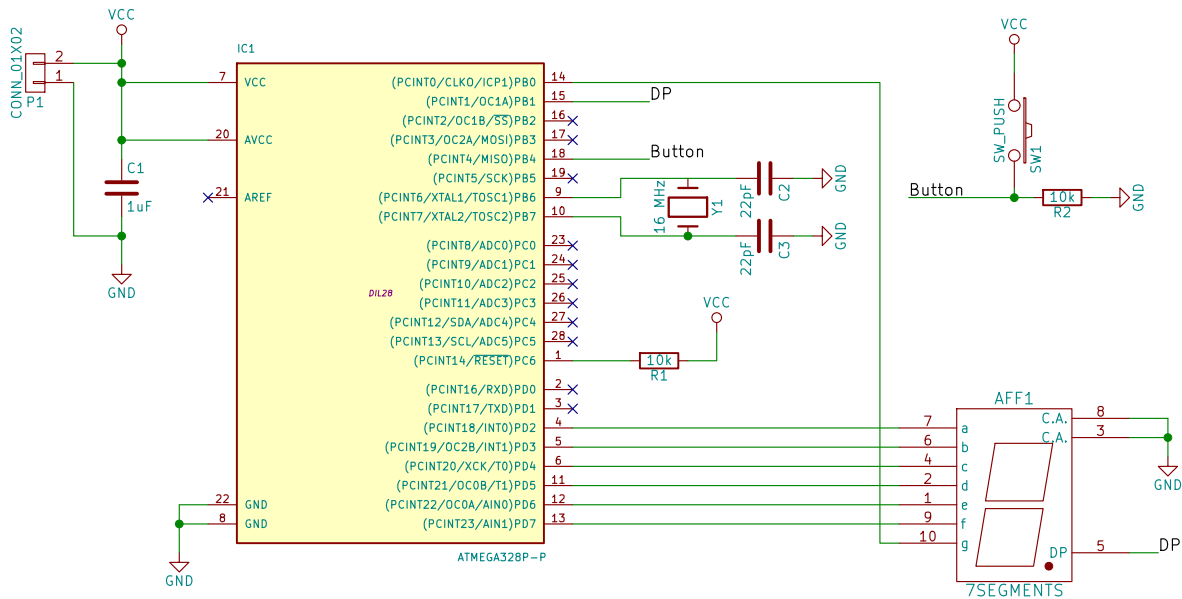



Figure 7: Completed schematic drawing of Example 1 circuit.

2.2 Creating or Editing Schematic Symbols

Sometimes you run into a situation where you cant find a proper symbol in the default KiCad library for the component that you want to use. The following steps will show you how to create a schematic symbol of your own.

1. From the KiCad project window, click the “Schematic library editor” button  to launch the schematic symbol editor and library manager. Alternatively, you can launch it by clicking the same button from Eeschema. The library editor window should appear.

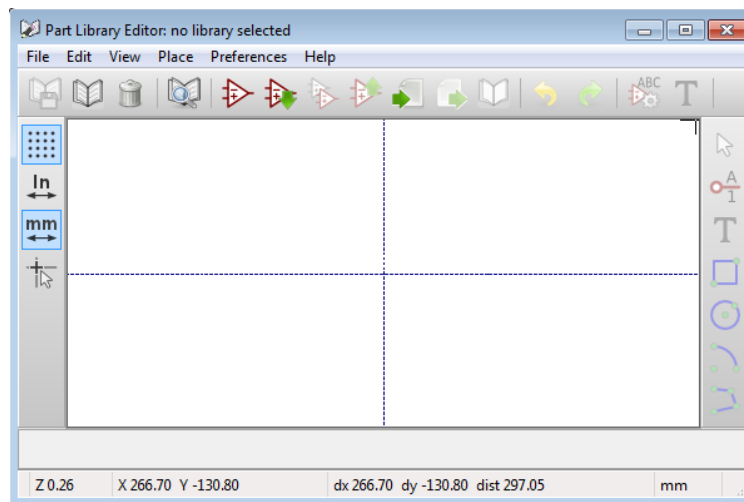


Figure 8: The schematic library editor program

2. Click the “Select working library” button to set the current working library. A dialog box should appear with the same list of component libraries that we saw when we placed components on the schematic.
 - a) If you are creating a new component, it is recommended that you select the project library, which is by default named “*project_name-cache*” and should be the last one in the list. In this example, select the library named “*arduino_dice-cache*”.
 - b) If you are trying to edit an existing component, click the corresponding library.
 - c) Click “OK” to finalize the selection.
3. In this example, we are actually not missing any schematic symbol. Just for the sake of demonstrating how to use the library editor, we will build our own version of the ATmega328P microcontroller IC. Fig. shows the pin diagram of the ATmega328P.[?]

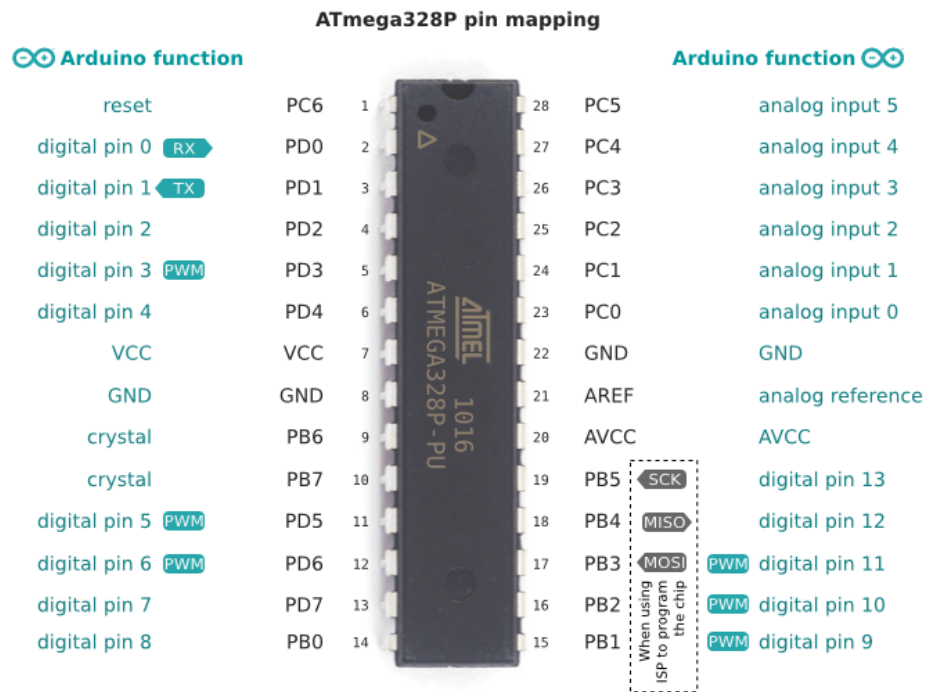


Figure 9: Pin mapping for the ATmega328P microcontroller.

2.3 PCB Layout

2.4 Associating schematic symbols with footprint

2.5 Creating or Editing Footprint

2.6 Generating Fabrication Files

3 Advanced Topics

3.1 Controlled Impedance Lines

* Transmission line parameter calculator: <http://wcalc.sourceforge.net/>

References

¹ Electronic dice w/ tilt sensor, 7 segment display and arduino.

² Kicad eda software suite.

³ Maestro ps-1b teardown.