A Tutorial on PCB Design

— using KiCad

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

Aug. 9th, 2015

Contents

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 become 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 Allegro [1], 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.

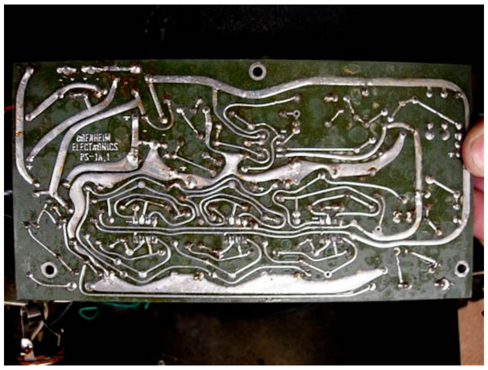
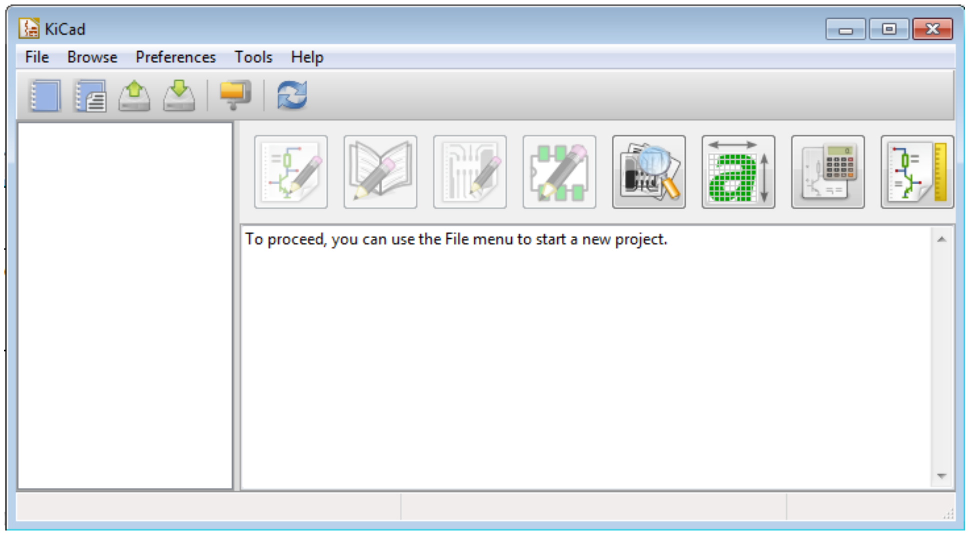


Figure 1: A vintage PCB laid out by hand.[4]

In recent years, KiCad has emerged as a popular open-source software package for designing and laying out PCBs.[3] 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 using 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.

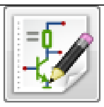
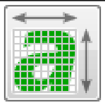
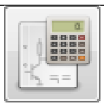
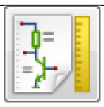
[1]UC Davis students have access to the full suite of Allegro PCB design tools through a donation from Cadence.



To proceed, you can use the File menu to start a new project.

Figure 2: The main window of KiCad.

Table 1: Individual tools within KiCad.

	Eeschema: schematic editor/capture tool		Gerbview: 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 eletronic dice consisting of a ATmega328P microcontroller[2], 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.[2]

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.

[2]The heart of the Arduino UNO platform.

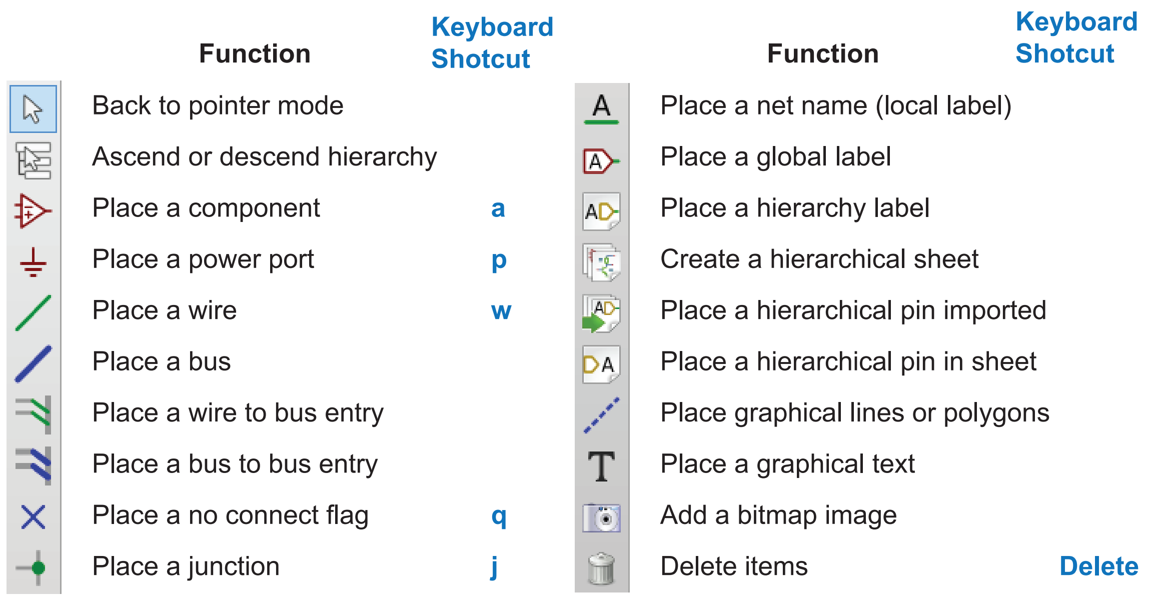
Table 2: List of components for Example 1.

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.



Function	Keyboard Shotcut	Function	Keyboard Shotcut
Back to pointer mode		Place a net name (local label)	
Ascend or descend hierarchy		Place a global label	
Place a component	a	Place a hierarchy label	
Place a power port	p	Create a hierarchical sheet	
Place a wire	w	Place a hierarchical pin imported	
Place a bus		Place a hierarchical pin in sheet	
Place a wire to bus entry		Place graphical lines or polygons	
Place a bus to bus entry		Place a graphical text	
Place a no connect flag	q	Add a bitmap image	
Place a junction	j	Delete items	Delete

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.

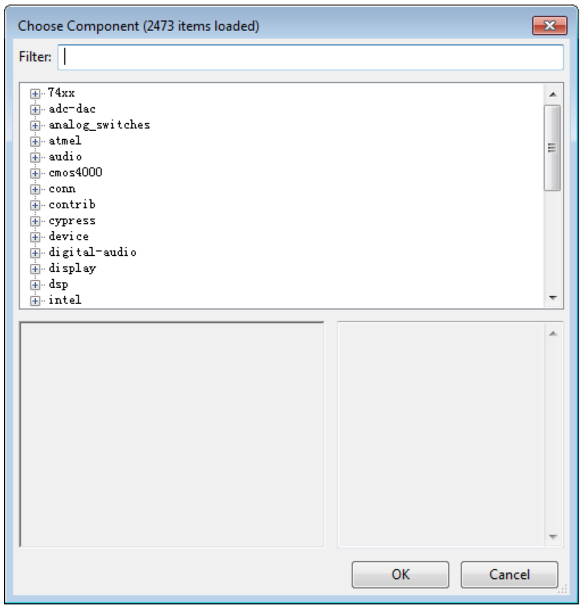


Figure 4: Place component dialog box.

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.

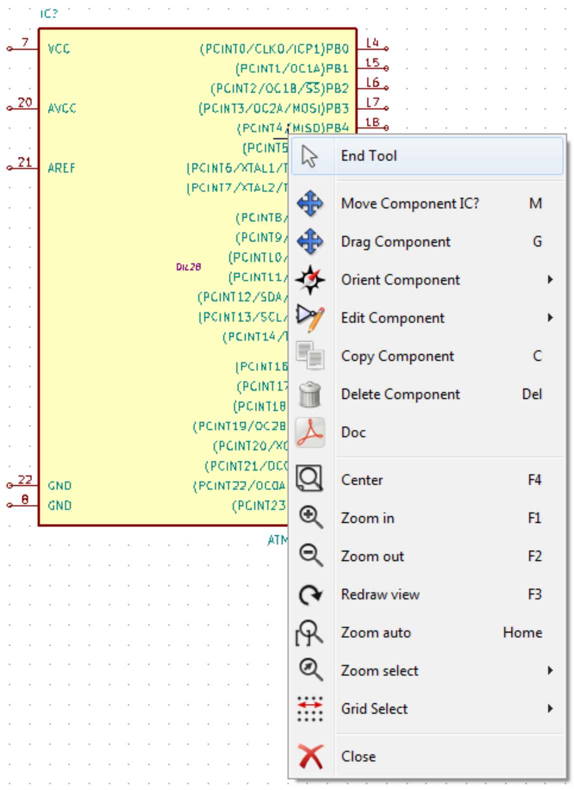


Figure 5: Edit component pop-up menu.

C. For now, we will delete the value for the "Footprint" field.

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.

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	

a) Delete the wire between the ATmega328P's *PB1* pin and the 7-segment LED's *DP* pin.

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.

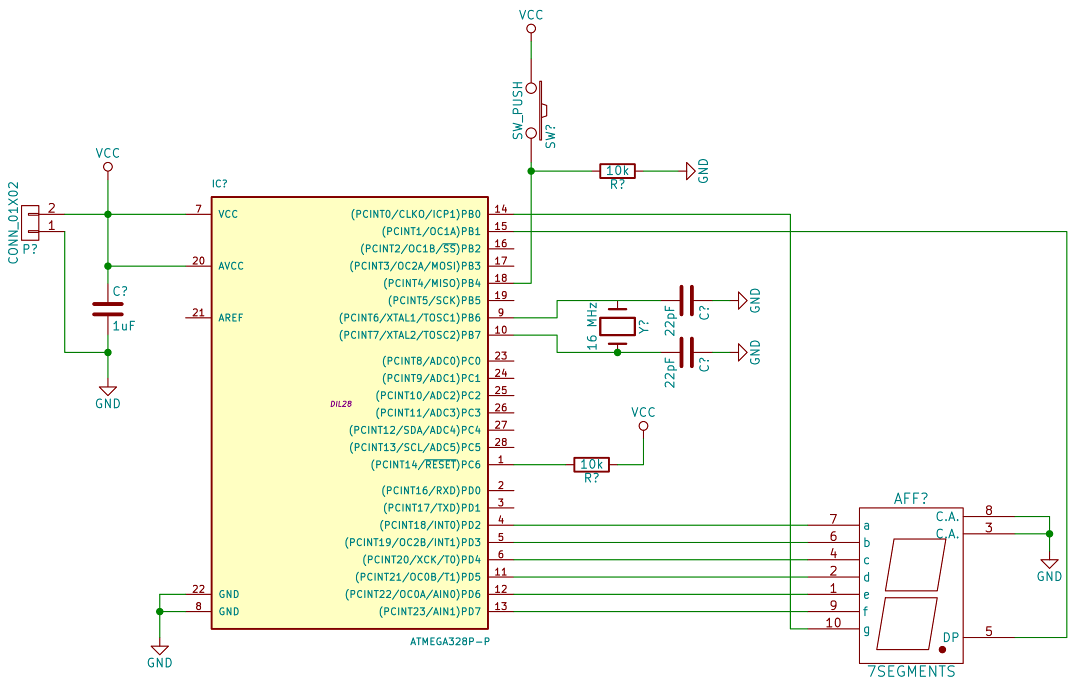


Figure 6: Initial schematic drawing of Example 1 circuit.

c) Click the “Annotate” button to finish. You must have noticed that you can “un-annotate” the schematic by clicking the “Clear annotation” button.

d) After annotation, you should see that all the components are numbered.

9. It is always a good idea to run an ERC before proceeding. ERC checks the electrical connections between components and try to detect potential errors in the schematic.

10. Once youve passed the ERC, generate a netlist by clicking the “Generate netlist” button.

11. This will create a netlist file that describes the circuit connections. The netlist file will be used to guide the PCB layout process.

12. The final schematic should look like that in Fig. 7.

2.2 Associating schematic symbols with footprint

In KiCad, the schematic symbol and the footprint of a device are stored in separate files (.lib for schematic symbols and .mod for footprint). In order for the PCB layout tool (Pcbnew) to put the correct footprints in the layout, we need to assign a relationship between the schematic symbols and the footprints in a circuit. The program CvPcb is used to do this.

1. Open CvPcb from the Eeschema window. By default, CvPcb should automatically loaded the netlist of the project. If not, you can go to File-¿Open to open the desired netlist. Ignore any error messages at this stage.

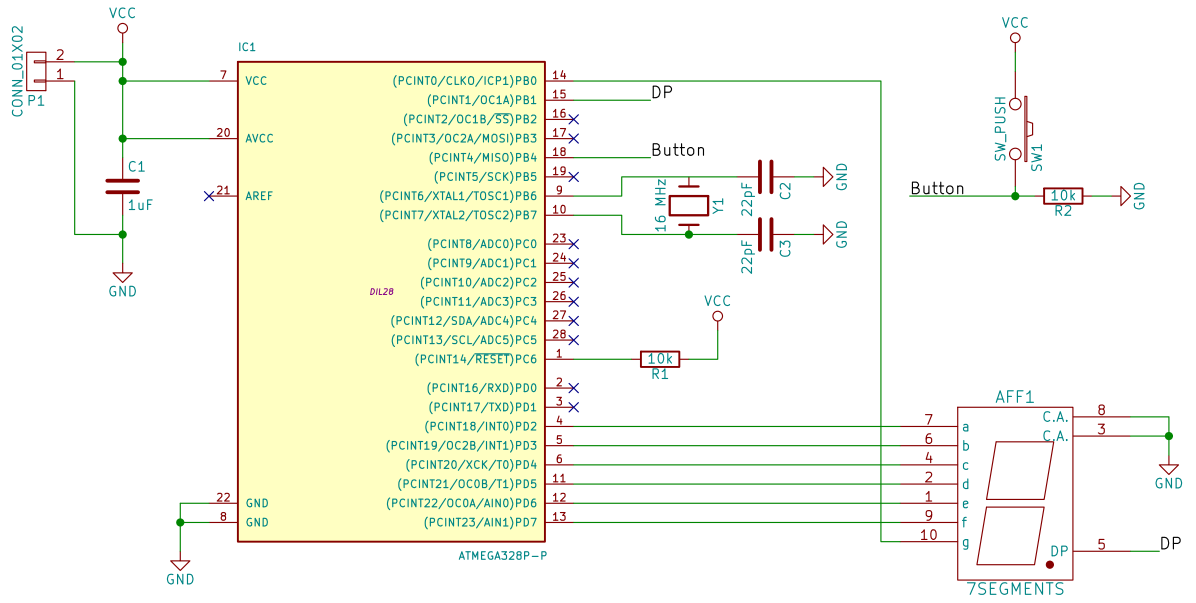


Figure 7: Completed schematic drawing of Example 1 circuit.

2. The CvPcb window consists of three panes. The leftmost one lists the footprint libraries. The center one lists all the schematic components used in the circuit. The right one presents a list of available footprints. These footprints are installed with the KiCad program. By default, CvPcb presents only footprints that it thinks that are relevant to the components. To see a full list of available components, uncheck the “Filter footprint list by keywords” button.

3. Our job in CvPcb is to associate the schematic components with their corresponding footprint. This is done by first selecting the component in the left pane, and then double clicking on the right footprint in the right pane. You can preview the drawing of the footprint by clicking the “View selected footprint” button .

4. **A very important rule in PCB design is to never trust footprints provided by others unless 1) they are directly from the component vendors; 2) you have verified them yourself.** The default footprint library that comes KiCad is particularly error prone. For this reason, we will be creating all the footprints by ourselves in this example. The next section outlines this process.

2.3 Creating or Editing Footprint

2.3.1 SMD Capacitor

We will start with the simplest component in the circuit, the 22 pF capacitor. This capacitor is a 0603 SMD capacitor produced by TDK Corporation. “0603” refers to its lateral dimensions in thousandth of an inch (normally referred to as a **mil** or a **thou**), that is, the capacitor is roughly 60 mil long and 30 mil wide. If we look at the dimension drawing provided by the vendor (Fig. 8a), we see that its actual dimensions are 63 mil×31 mil. In fact, the capacitor is realy sized in the metric system, with a nominal dimension of 1.6 mm×0.8 mm. So this capacitor is a "0603" in imperial units and "1608" in metric units. **The difference between the imperial and metric units is a common source of confusion when choosing components. Alwasy double check!**

On the Digikey product page, we can find the datasheet to the capacitor "C Series, Gen Appl & Mid-Voltage Spec". Page 15 of the datasheet, copied here in Fig. 8b, shows the recommended land pattern (footprint) for the capacitors in this serie of products. The recommended dimensions for the 0603 (1608) capacitor is highlighted; here we will use the median value of 0.7 mm for A, B, and C.

Now let's start drawing the footprint for the capacitor.

1. Start the PCB footprint editor by clicking on the "PCB footprint editor" button.

2. Click the "New footprint" button to create a new footprint. Put "smd_0603" as the name in the pop-up dialog box. Click "OK".

3. By default, the footprint name "smd_0603" and the Reference string "REF**" will appear in the center of the drawing window.

4. Set the working unit to "mm" (millimeters). Then set the grid size to "User grid". The user grid size can then by set by going to menu "Dimensions→ User Grid Size". Set the unit to "Millimeters", "Size X" to "0.05", and "Size Y" to "0.05".

5. Click the "Add pads" button. Move the cursor to position (-0.7,0) and click. A donut-shape patch should appear. This is because by default KiCad assumes a through-hole type pad.

6. To change the pad properties, move the cursor on top of the pad and type keyboard short "e" for editing. The *Pad Properties* dialog box should appear. Change the properties as follows:

Pad number	1
Pad type	SMD
Shape	Rectangular shape
Position X	-0.7
Position Y	0
Size X	0.7
Size Y	0.7
Layers	F.Cu
Technical Layers	Check "F.Paste", "F.SilkS", and "F.Mask"

7. Repeat Step 5 and 6 to add the second pad at the correct location. You will notice that KiCad assumes the properties of the previous pads as the default properties of the new pad.

8. Draw the courtyard outline of the capacitor using "Add graphic line or polygon" tool. The courtyard

9. The finished footprint drawing should look like that shown in Fig. 9.

10. We wish to create a new library to contain our newly created footprint. To do this, click the “Create new library and save current module” icon. Set the location and name of the library and click “OK”. The “Library Path” field should contain the full path to your desired library location. In this example, we will create a footprint library with the name “arduino-dice-footprints” (KiCad will automatically add the “.pretty” suffix) under the project folder “...\arduino-dice”. Fig. 10 shows the proper settings. At this point, a new folder named “arduino-dice-footprints.pretty” should appear in the project tree in the KiCad main window (Fig. 11). Click on the “+” sign will expand the library and show all the footprints in this library; in this case, we only have the “smd_0603” footprint.

11. Although we now have the new footprint library in our file system and the project tree, KiCad still does not recognize it as part of the available libraries that we can pull footprints from. We will need to manually add the library to KiCad’s library tables.
    a) Open CvPcb and click the “Edit footprint library table” icon. The “PCB Library Tables” dialog box should appear.
    b) Click the “Append with Wizard” button.
    c) In the pop-up dialog box, select “Files on my computer” and click “Next”.
    d) Browse to and select the footprint library folder we just created, click “Next”. Click “Next” again in the next window to confirm.
    e) Select “To the current project only” and click “Finish” to finish adding the footprint library to this project. If you have created a generic library that you want to use across multiple projects, you could selecte “To global library configuration” option. Fig. 12 shows how the “PCB Library Tables” looks like now. Notice how KiCad replaces the absolute path with the “KIPRJMOD” variable.

2.4 PCB Layout

2.5 Generating Fabrication Files

3 Advanced Topics

3.1 Controlled Impedance Lines

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

4 Further Reading

* KiCad footprint generator: http://kicad.rohrbacher.net/quickmod.php
 * KiCad library management: https://docs.google.com/document/d/1M38ByFyqnhwGo8b_jDDyBceyZtEGeaSAuQaP9REzWrU/edit

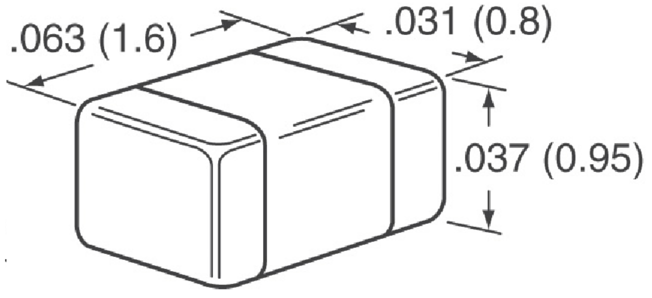
References

[1] Atmega328p pin mapping.

[2] Electronic dice w/ tilt sensor, 7 segment display and arduino.

[3] Kicad eda software suite.

[4] Maestro ps-1b teardown.



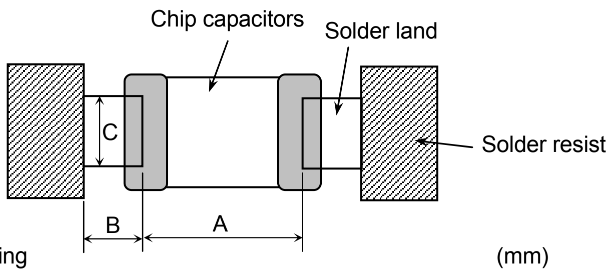
(a)

3	Designing P.C.board	The amount of solder at the terminations has a direct effect on the reliability of the capacitors.

1) The greater the amount of solder, the higher the stress on the chip capacitors, and the more likely that it will break. When designing a P.C.board, determine the shape and size of the solder lands to have proper amount of solder on the terminations.

2) Avoid using common solder land for multiple terminations and provide individual solder land for each terminations.

3) Size and recommended land dimensions.



Flow soldering (mm)

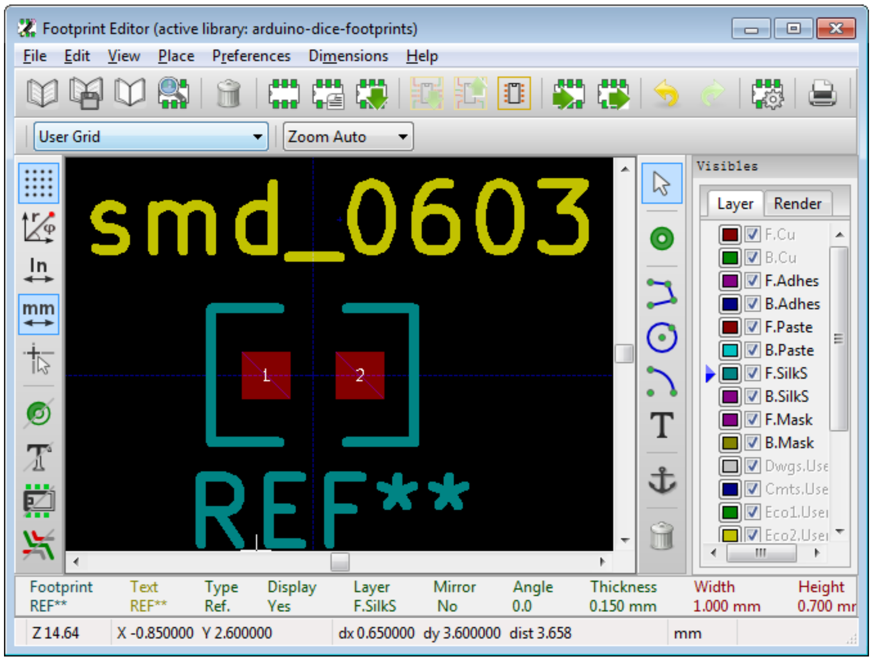
Symbol \ Type	C1608 (CC0603)	C2012 (CC0805)	C3216 (CC1206)
A	0.7 - 1.0	1.0 - 1.3	2.1 - 2.5
B	0.8 - 1.0	1.0 - 1.2	1.1 - 1.3
C	0.6 - 0.8	0.8 - 1.1	1.0 - 1.3

Reflow soldering (mm)

Symbol \ Type	C0402 (CC01005)	C0603 (CC0201)	C1005 (CC0402)	C1608 (CC0603)	C2012 (CC0805)
A	0.15 - 0.25	0.25 - 0.35	0.3 - 0.5	0.6 - 0.8	0.9 - 1.2
B	0.15 - 0.25	0.2 - 0.3	0.35 - 0.45	0.6 - 0.8	0.7 - 0.9
C	0.15 - 0.25	0.25 - 0.35	0.4 - 0.6	0.6 - 0.8	0.9 - 1.2

(b)

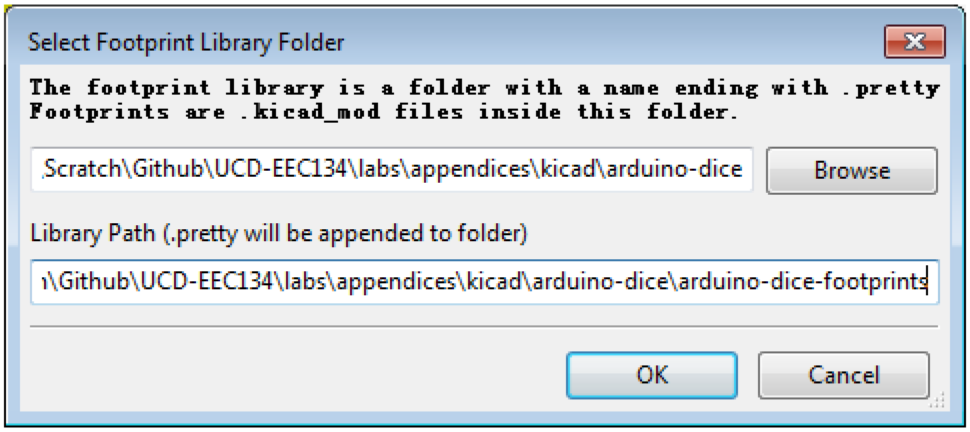
Figure 8: (a) Dimensions of the TDK C-series 22 pF SMD capacitors. (b) PCB design recommendation for TDK C-series SMD capacitors. Land pattern dimensions for the C1608 type is highlighted in red.



Footprint	Text	Type	Display	Layer	Mirror	Angle	Thickness	Width	Height
REF**	REF**	Ref.	Yes	F.SilkS	No	0.0	0.150 mm	1.000 mm	0.700 mr

Z 14.64   X -0.850000 Y 2.600000   dx 0.650000 dy 3.600000 dist 3.658   mm

Figure 9: Completed footprint of the SMD 0603 capacitor.


Select Footprint Library Folder

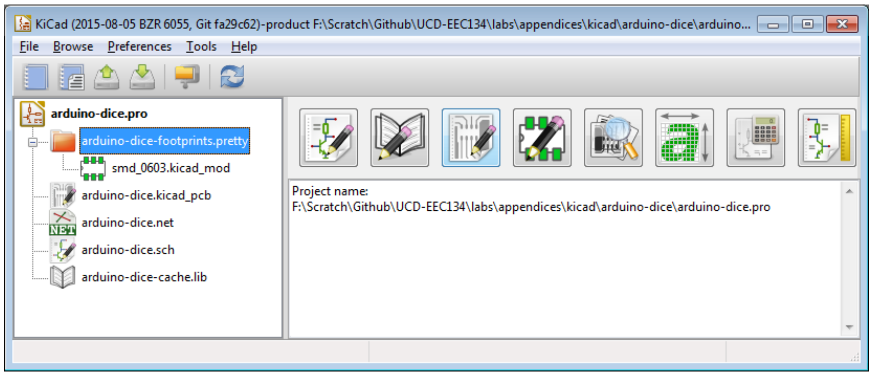
The footprint library is a folder with a name ending with .pretty
Footprints are .kicad_mod files inside this folder.

Scratch\Github\UCD-EEC134\labs\appendices\kicad\arduino-dice

Library Path (.pretty will be appended to folder)

ı\Github\UCD-EEC134\labs\appendices\kicad\arduino-dice\arduino-dice-footprints

Figure 10: The "Select Footprint Library Folder" dialog box.


KiCad (2015-08-05 BZR 6055, Git fa29c62)-product F:\Scratch\Github\UCD-EEC134\labs\appendices\kicad\arduino-dice\arduino...
File Browse Preferences Tools Help

arduino-dice.pro
arduino-dice-footprints.pretty
smd_0603.kicad_mod
arduino-dice.kicad_pcb
arduino-dice.net
arduino-dice.sch
arduino-dice-cache.lib

Project name:
F:\Scratch\Github\UCD-EEC134\labs\appendices\kicad\arduino-dice\arduino-dice.pro

Figure 11: The new footprint library is now visible in the project tree.

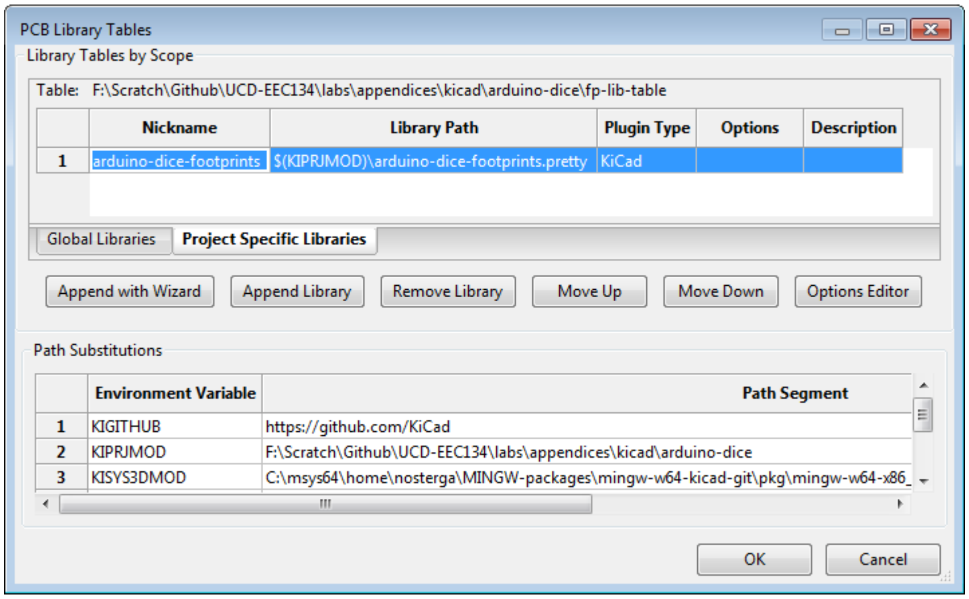


Table: F:\Scratch\Github\UCD-EEC134\labs\appendices\kicad\arduino-dice\fp-lib-table

	Nickname	Library Path	Plugin Type	Options	Description
1	arduino-dice-footprints	$(KIPRJMOD)\arduino-dice-footprints.pretty	KiCad		

	Environment Variable	Path Segment
1	KIGITHUB	https://github.com/KiCad
2	KIPRJMOD	F:\Scratch\Github\UCD-EEC134\labs\appendices\kicad\arduino-dice
3	KISYS3DMOD	C:\msys64\home\nosterga\MINGW-packages\mingw-w64-kicad-git\pkg\mingw-w64-x86_

Figure 12: Add a project footprint library.