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.

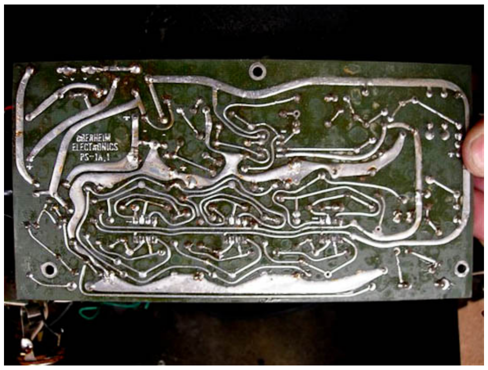


Figure 1: A vintage PCB laid out by hand.[4]

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.

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

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]

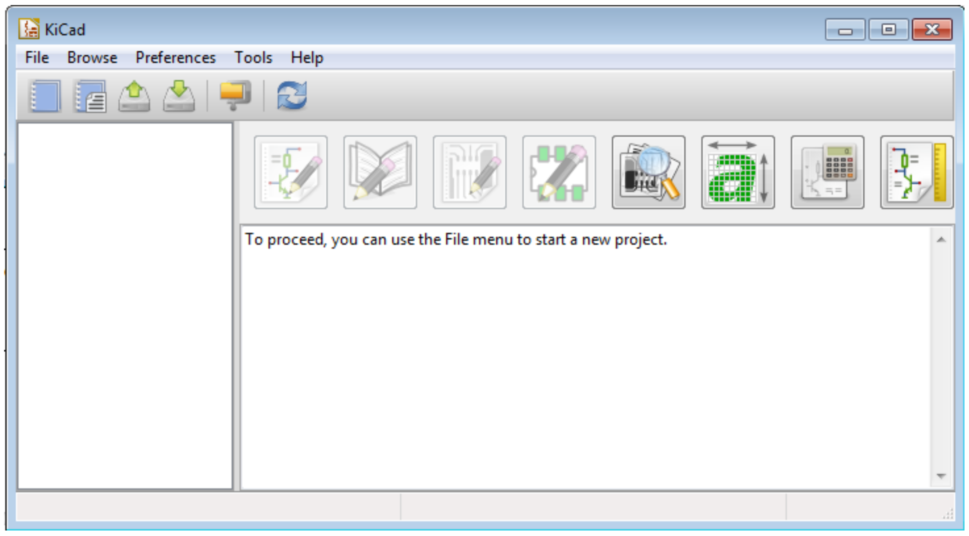
Table. 1 lists the components that are needed for this example.

Table 1: 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

2.1 Starting KiCad

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.



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

Figure 2: The main window of KiCad.

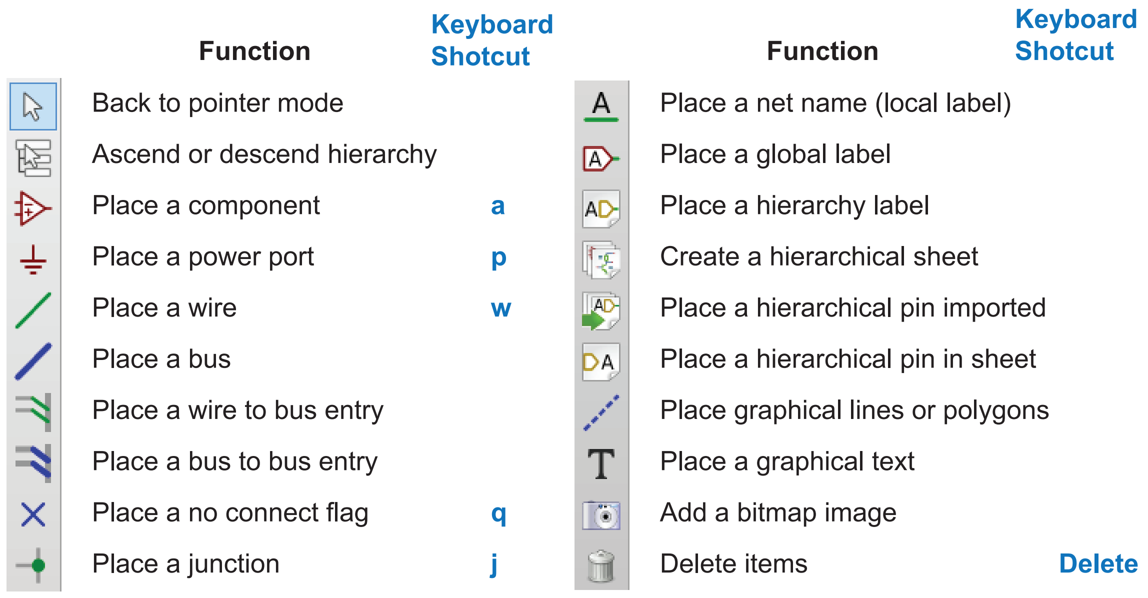
[2]The heart of the Arduino UNO platform.

Table 2: 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

2.2 Schematic Capture

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

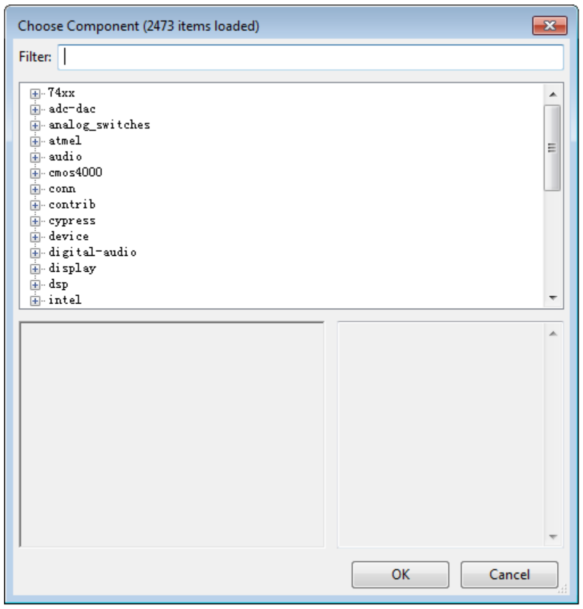


Figure 4: Place component dialog box.

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.

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.

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

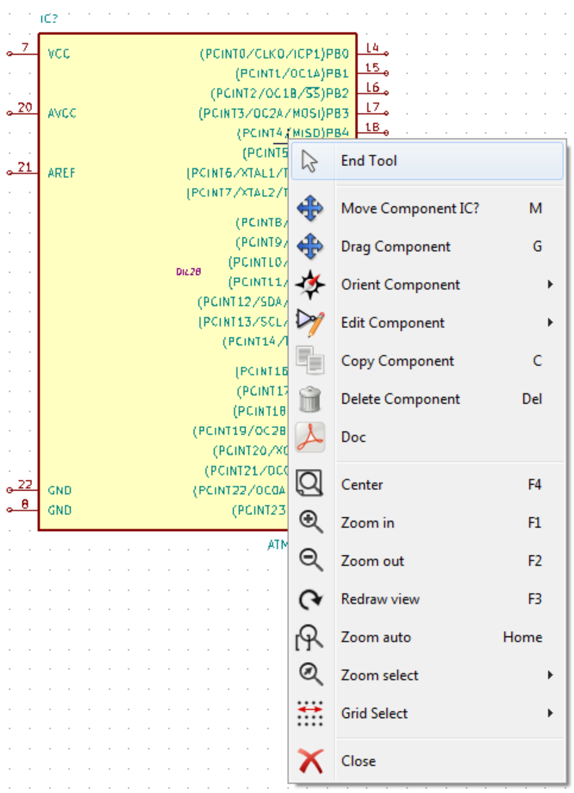


Figure 5: Edit component pop-up menu.

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.

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	

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.

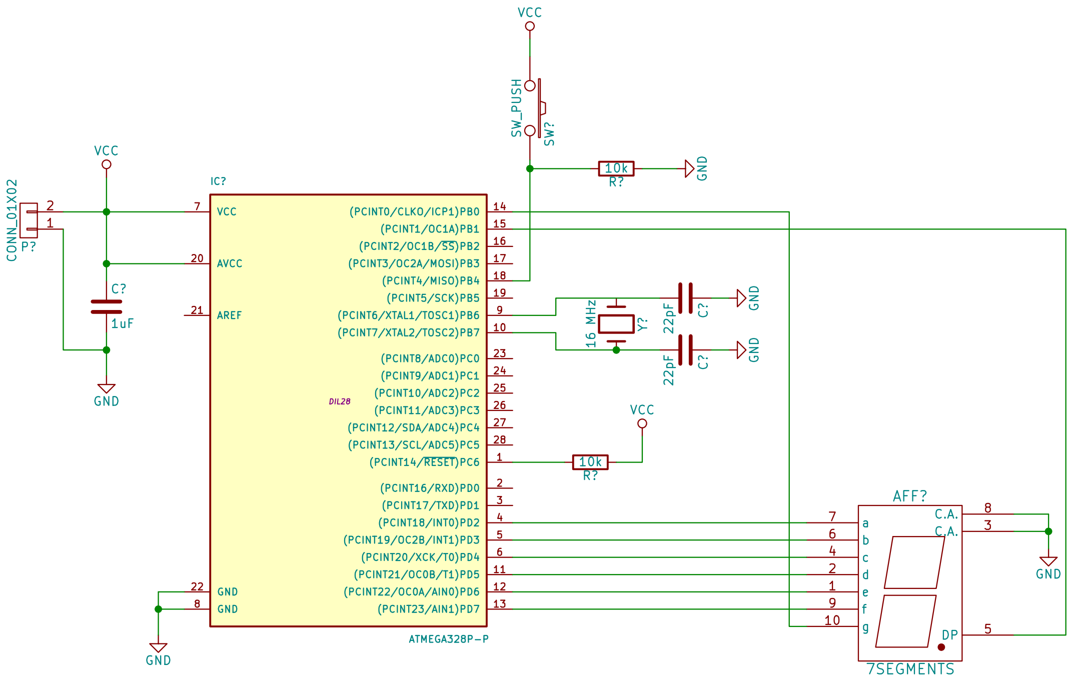


Figure 6: Initial schematic drawing of Example 1 circuit.

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.

   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.

   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.

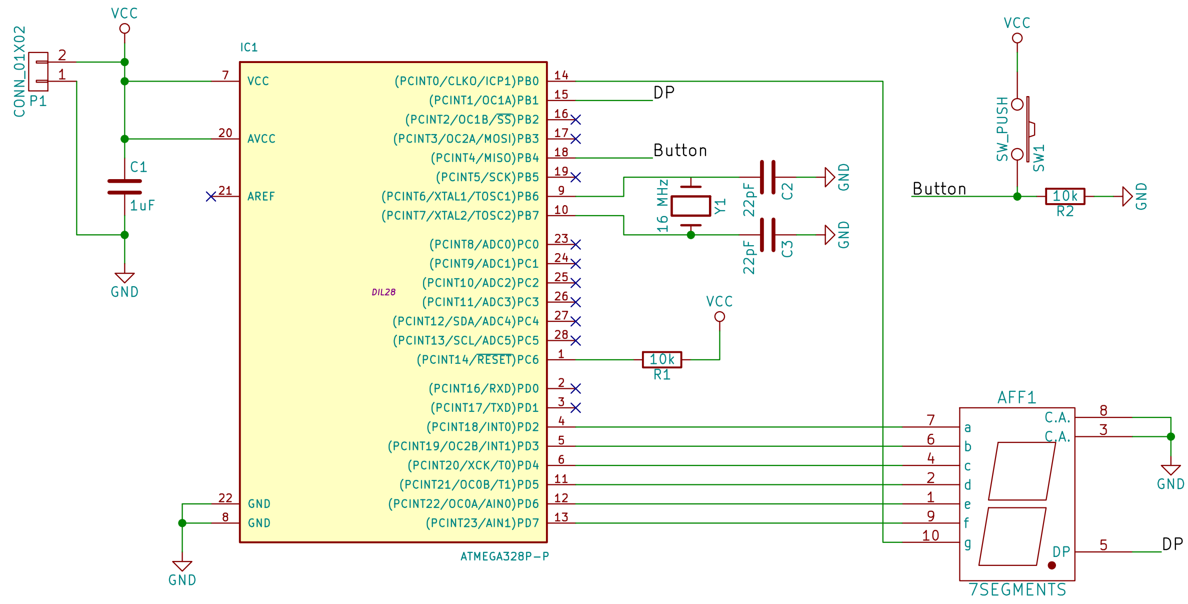


Figure 7: Completed schematic drawing of Example 1 circuit.

2.3 Creating 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.

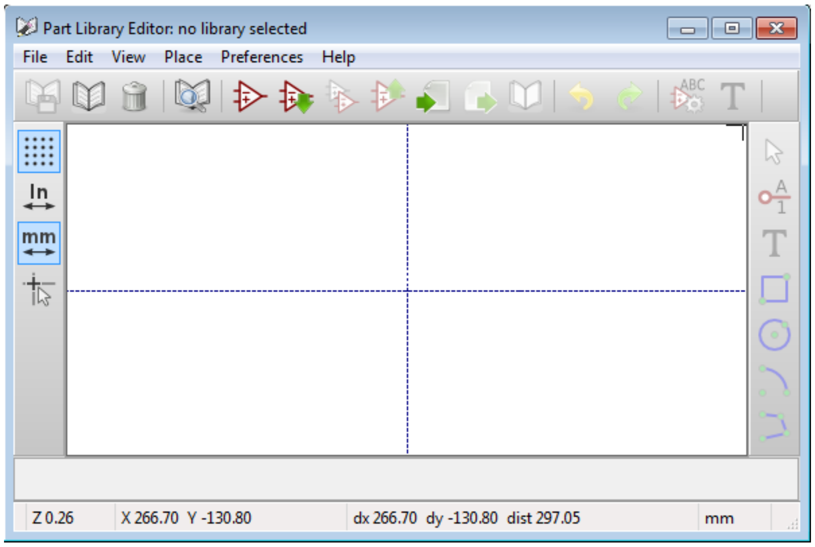

Part Library Editor: no library selected
File
Edit
View
Place
Preferences
Help
In
mm
Z 0.26
X 266.70 Y -130.80
dx 266.70 dy -130.80 dist 297.05
mm

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.[1]
4. Click the "Create a new component" button. A dialog box will appear. Put "ATmega328p" as the component name. Leave everything else as default. Worth mentioning is the "Number of parts per package" setting. Sometimes a component may have many pins and it becomes difficult to route the schematic (or to be more precise, it becomes difficult to read the schematic when you have lots of wire connections). It may be easier to split a component into several schematic symbols so that routing can be easier. In such a scenario, you can have multiple parts per package.

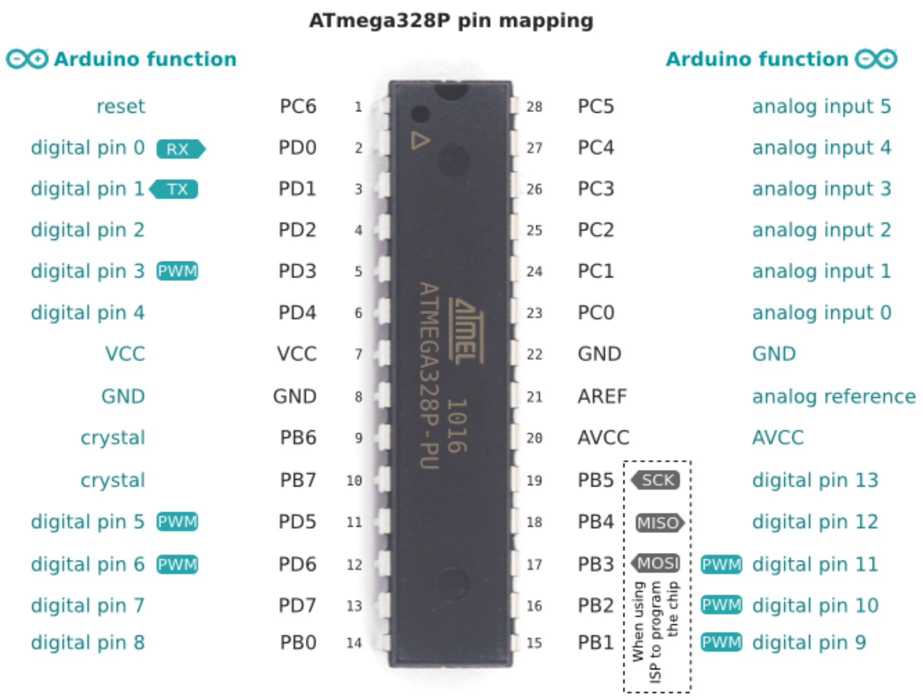
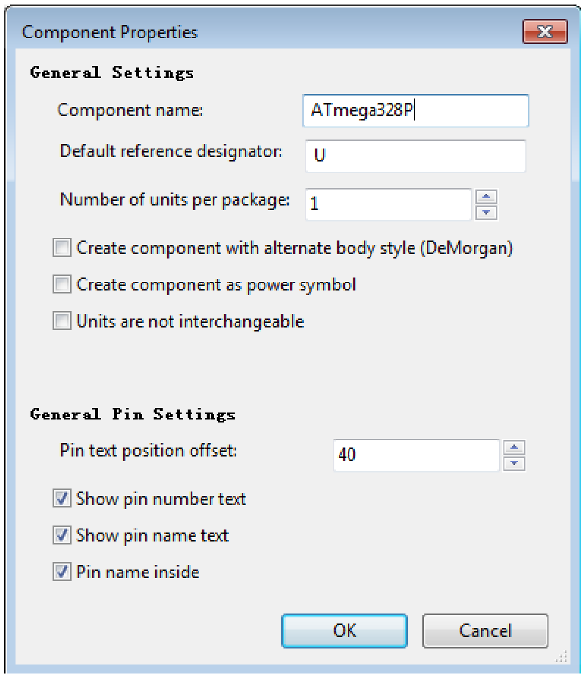

ATmega328P pin mapping
Arduino function
Arduino function
reset PC6 1 | 28 PC5 analog input 5
digital pin 0 RX PD0 2 | 27 PC4 analog input 4
digital pin 1 TX PD1 3 | 26 PC3 analog input 3
digital pin 2 PD2 4 | 25 PC2 analog input 2
digital pin 3 PWM PD3 5 | 24 PC1 analog input 1
digital pin 4 PD4 6 | 23 PC0 analog input 0
VCC VCC 7 | 22 GND GND
GND GND 8 | 21 AREF analog reference
crystal PB6 9 | 20 AVCC AVCC
crystal PB7 10 | 19 PB5 SCK digital pin 13
digital pin 5 PWM PD5 11 | 18 PB4 MISO digital pin 12
digital pin 6 PWM PD6 12 | 17 PB3 MOSI PWM digital pin 11
digital pin 7 PD7 13 | 16 PB2 PWM digital pin 10
digital pin 8 PB0 14 | 15 PB1 PWM digital pin 9
When using ISP to program the chip
ATMEL ATMEGA328P-PU 1016

Figure 9: Pin mapping for the ATmega328P microcontroller.



Component Properties

General Settings

Component name: ATmega328P

Default reference designator: U

Number of units per package: 1

☐ Create component with alternate body style (DeMorgan)

☐ Create component as power symbol

☐ Units are not interchangeable

General Pin Settings

Pin text position offset: 40

☑ Show pin number text

☑ Show pin name text

☑ Pin name inside

Figure 10: Creat component dialog box.

5. Click “OK” and you should see the component “Reference” and “Value” strings appearing near the center of the library editor main window.

6. Click the “Add graphic rectangle to the component body” button , and use the mouse cursor to draw a rectangle from ($x = -0.400''$, $y = 0.700''$) to ($0.400''$, $-0.700''$) (Fig. 11). The current position of the cursor is displayed at the bottom of the main window. Notice that the positive direction of the $y-$axis points downward.

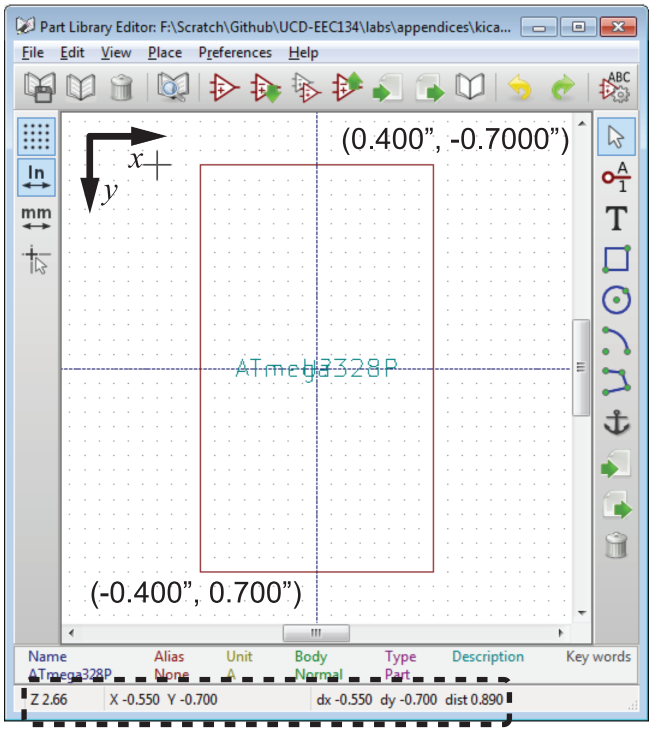
7. Add component pins.
   a) Click the “Add pins to the component” button
   b) Click anywhere in the main drawing window. A dialog box should appear. Put “PC6” for the “Pin name”, “1” for the “Pin number”, “Right” for the “Orientation”, “Input” for the “Electrical type”, and leave everything else as default.
   c) Click “OK”, and then click at (-0.600, -0.650) to add the pin.

8. Repeat the above process and add the rest of the pins as follows.

Pin Name	**Pin number**	**Orientation**	**Electrical Type**	**Pin Name**	**Pin number**	**Orientation**	**Electrical Type**
PD0	2	Right	Input	PC5	28	Left	Input
PD1	3	Right	Input	PC4	27	Left	Input
PD2	4	Right	Input	PC3	26	Left	Input
PD3	5	Right	Input	PC2	25	Left	Input
PD4	6	Right	Input	PC1	24	Left	Input
VCC	7	Right	Power Input	PC0	23	Left	Input
GND	8	Right	Power Input	GND	22	Left	Power Input
PB6	9	Right	Input	AREF	21	Left	Input
PB7	10	Right	Input	AVCC	20	Left	Power Input
PD5	11	Right	Input	PB5	19	Left	Input
PD6	12	Right	Input	PB4	18	Left	Input
PD7	13	Right	Input	PB3	17	Left	Input
PB0	14	Right	Input	PB2	16	Left	Input
				PB1	15	Left	Input

9. Now move the “Reference” string above the rectangle (only for aesthetic reasons) and the “Value” string below the rectangle. You can also add a half circle (arc) to the top of the rectangle to indicate the orientation of the component. The completed component looks that in Fig. 12.

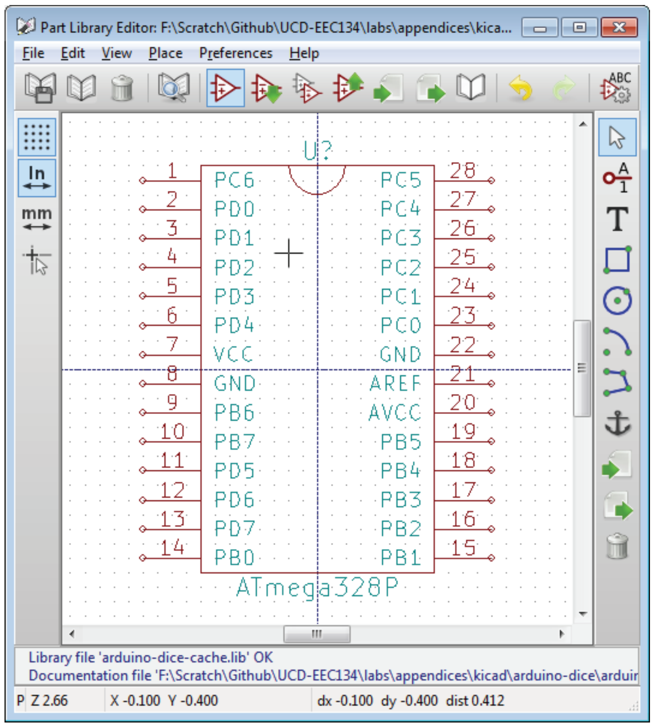
10. Click the Save current library to disk to save your work. This should complete the creation of the schematic symbol. To verify that you have successfully created the symbol, go back to the Eeschema window and check whether you can find the ATmega328P symbol you just built in the “arduino-dice-cache” library.


Part Library Editor: F:\Scratch\Github\UCD-EEC134\labs\appendices\kica...
File Edit View Place Preferences Help
In
mm
x
y
(0.400”, -0.7000”)
ATmega328P
(-0.400”, 0.700”)

Name	Alias	Unit	Body	Type	Description	Key words
ATmega328P	None	A	Normal	Part		

```
Z 2.66    X -0.550  Y -0.700    dx -0.550  dy -0.700  dist 0.890
```

Figure 11: Drawing a rectangle in the library editor main window with the help of the built-in cordinate system.



```
Library file 'arduino-dice-cache.lib' OK
Documentation file 'F:\Scratch\Github\UCD-EEC134\labs\appendices\kicad\arduino-dice\arduir
P  Z 2.66    X -0.100  Y -0.400    dx -0.100  dy -0.400  dist 0.412
```

Figure 12: Completed schematic symbol for the ATmega328P microcontroller.

2.4 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.

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.

2.5 Creating or Editing Footprint

2.5.1 SMD capacitor

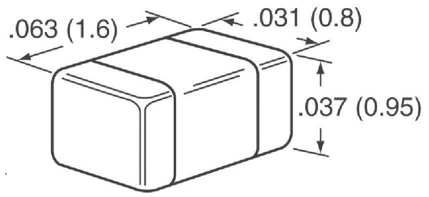
We will start with the simplest component in the circuit, the 22 pF capacitor. This capacitor is a 0603 SMD capacitor produced by TDK Corporation. "0603" refers to its lateral dimensions in thousandth of an inch (normally referred to as a **mil** or a **thou**), that is, the capacitor is roughly 60 mil long and 30 mil wide. If we look at the dimension drawing provided by the vendor (Fig. 13a), 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. 13b, shows the recommended land pattern (footprint) for the capacitors in this serie of products. The recommended dimensions for the 0603 (1608 metric) capacitor is highlighted; here we will use the median value of 0.7 mm for A, B, and C.

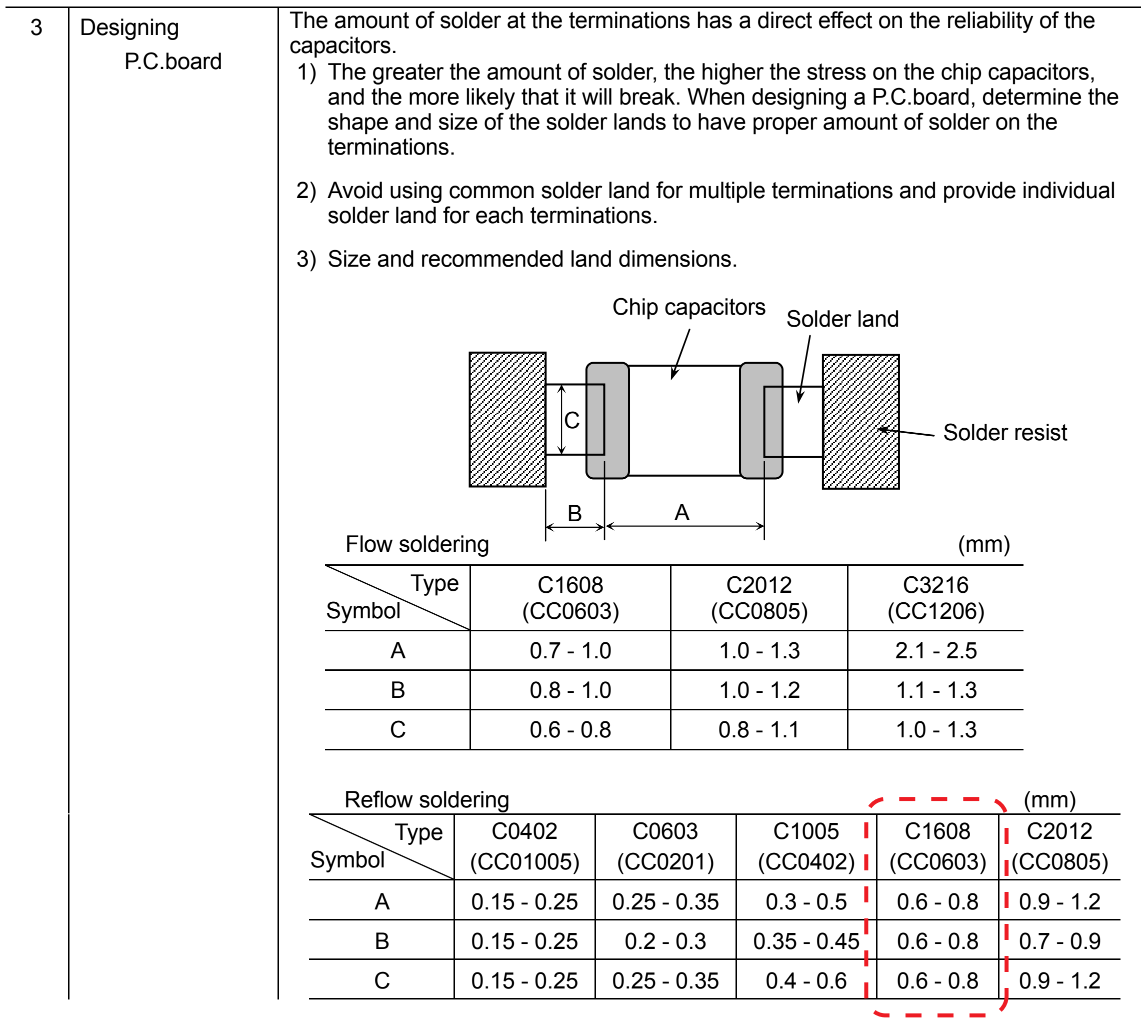
Now let's start drawing the footprint for the capacitor.

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".



(a)

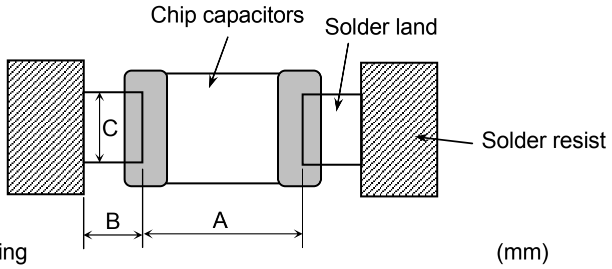


3	Designing P.C.board	The amount of solder at the terminations has a direct effect on the reliability of the capacitors.

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)

Type / Symbol	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)

Type / Symbol	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 13: (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.

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"

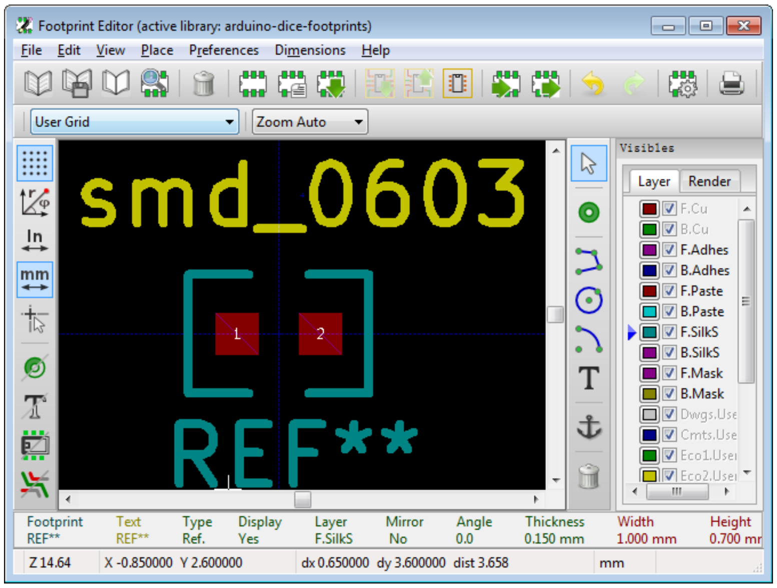
7. Repeat Step 5 and 6 to add the second pad at the correct location. You will notice that KiCad assumes the properties of the previous pads as the default properties of the new pad.

8. Draw the courtyard outline of the capacitor using "Add graphic line or polygon" tool. The courtyard is the actual dimension of the device plus some margin. It serves as a guide to prevent the PCB designer from placing components too close to each other. Here we have put a decent margin of 0.45 mm in the x-direction and 0.65 mm in the y-direction around the pads (or 0.7 mm in x and 0.6 mm in y around the component). This margin can be adjusted according to the assembly requirement and tolerances in the PCB assembly process.

9. The finished footprint drawing should look like that shown in Fig. 14.

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. 15 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. 16). 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.

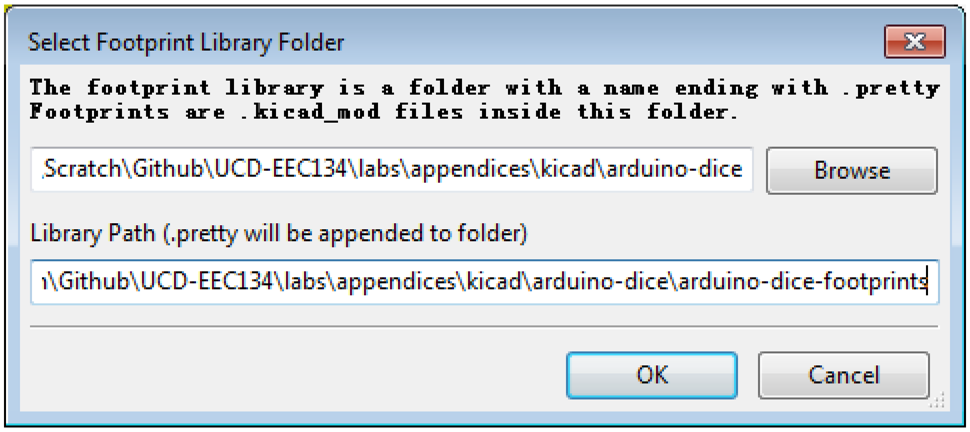
11. Although we now have the new footprint library in our file system and the project tree, KiCad still does not recognize it as part of the available libraries that we can pull footprints from. We will need to manually add the library to KiCad's *library tables*.



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 14: Completed footprint of the SMD 0603 capacitor.



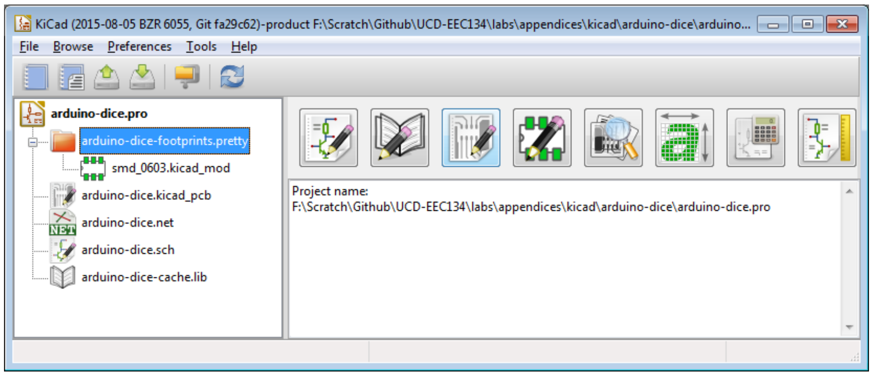
The footprint library is a folder with a name ending with .pretty
Footprints are .kicad_mod files inside this folder.

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

Library Path (.pretty will be appended to folder)

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

Figure 15: 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 16: The new footprint library is now visible in the project tree.

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 select the “To global library configuration” option. Fig. 17 shows how the “PCB Library Tables” looks like now. Notice how KiCad replaces the absolute path with the “KIPRJMOD” variable.

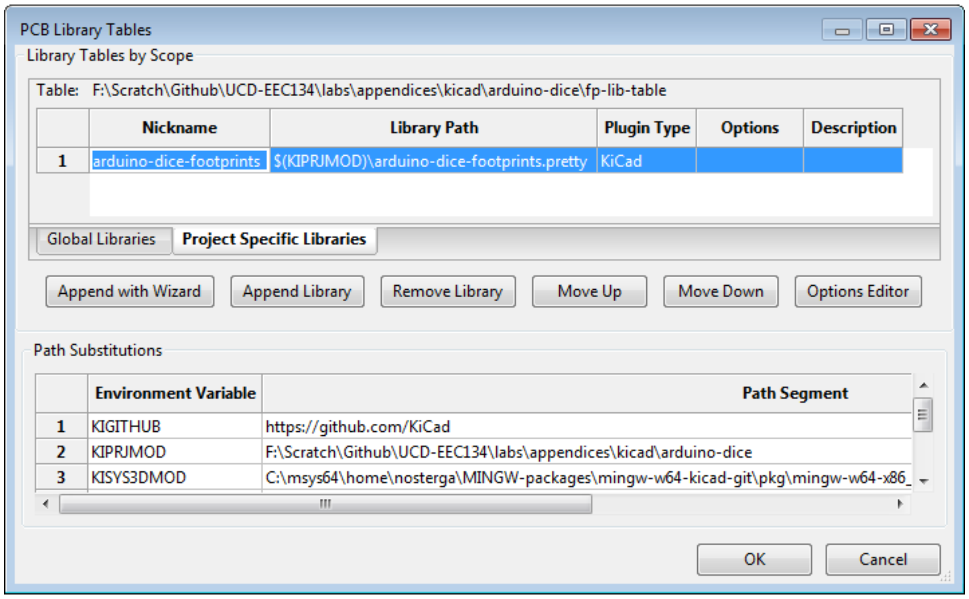


Table: F:\Scratch\Github\UCD-EEC134\labs\appendices\kicad\arduino-dice\fp-lib-table

	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 17: Add a project footprint library.

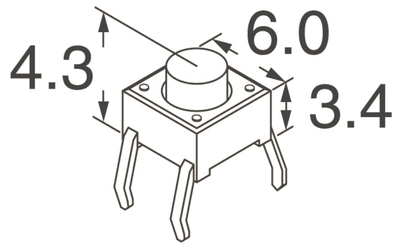
f) Finally, if you plan to create more footprints, it’s a good idea to set the active library in the Footprint Editor. In the Footprint Editor window, click the “Set active library” icon, select the “arduino-dice-footprints” library which should be at the bottom of the list, and click “OK”.

2.5.2 Push button switch

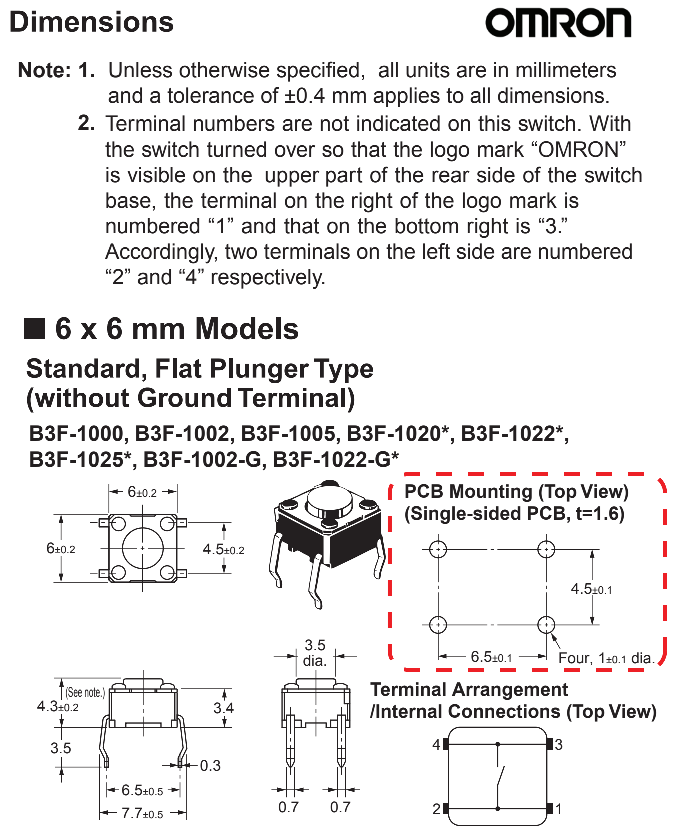
The Omron B3F-1000 push botton switch is a through-hole component. Page 225 of its datasheet shows the detailed dimensions of the device (Fig. 18) and can be used to guide the footprint design.

The footprint for the switch consists of 4 1 mm diameter holes spaced 6.5 mm apart in the x-direction and 4.5 mm apart in the y-direction. Use similar steps as in Sec. 2.5.1 to create the switch footprint. Use the following parameters for the through hole pads.

Fig. 19 shows the completed footprint drawing for the switch.



(a)



Dimensions

Note: 1. Unless otherwise specified, all units are in millimeters and a tolerance of ±0.4 mm applies to all dimensions.

2. Terminal numbers are not indicated on this switch. With the switch turned over so that the logo mark “OMRON” is visible on the upper part of the rear side of the switch base, the terminal on the right of the logo mark is numbered “1” and that on the bottom right is “3.” Accordingly, two terminals on the left side are numbered “2” and “4” respectively.

■ 6 x 6 mm Models

Standard, Flat Plunger Type (without Ground Terminal)

B3F-1000, B3F-1002, B3F-1005, B3F-1020*, B3F-1022*, B3F-1025*, B3F-1002-G, B3F-1022-G*

PCB Mounting (Top View) (Single-sided PCB, t=1.6)

Terminal Arrangement /Internal Connections (Top View)

(b)

Figure 18: (a) Dimension drawing and (b) suggested footprint of the B3F-1000 switch.

Table 4: Parameters for through-hole pads for the B3F switch.

Pad type	Through-hole
Shape	Circular shape
Position X	± 3.25
Position Y	± 2.25
Size X	2.032 (80 mil)
Drill: Size X	1
Layers	All copper layers
Technical Layers	Check “F.SilkS”, “F.Mask”, and “B.Mask”

2.5.3 ATmega328P microcontroller

The ATmega328P microcontroller powers the popular Arduino UNO platform. There are two variants of the UNO, one with the ATmega328P-PU which is a through hole package and the other with an SMD ATmega328P. In this example, we will be using the through-hole version (ATmega328P-PU) because it allows us to program the microcontroller on the UNO, unplug the microcontroller, and put it on our own PCB.

The ATmega328P-PU follows a standard 0.3" 28-pin dual inline package (28-DIP), i.e. there are two rows of pins with a 0.1" pitch (distance between pins) and 0.3" row spacing. We can follow the same procedures as in Sec. 2.5.1 and 2.5.2 to create its footprint, but this time we'll show you how to do it by modifying an existing footprint from the default KiCad library.

1. In Footprint Editor, click the "Select active library" button. In the pop-up dialog box, select the "Sockets_DIP" library and click "OK".
2. Click the "Load footprint from library". In the pop-up dialog box, click "List all". In the component listing, click "DIP-28_300", and then click "OK". The default 28-DIP package should appear in the Library Editor window.
3. Verify that all the pads have the right dimensions (drill size, pitch and row spacing).
4. Mouse over the string "DIP-28_300" and change it to "atmega328p-pu" so that it's more recognizable when we do the layout later on. Fig. 21 shows completed
5. Now click the "Select active library" icon and select the "arduino-dic-footprints" library. Click the "Save footprint in active library" icon to save the footprint in our project footprint library.

2.5.4 The rest of the components

Similar to the ATmega328P-PU microcontroller, the Avago HDSP-313E 7-segment LED display follows a standard 0.3" 10-DIP pin arrangement. Therefore we can modify a standard 10-DIP footprint (Library: "Sockets_DIP→DIP-10_300") for the 7-segment. Again, always double check all the dimensions if you are basing your work on someone else's footprint. Also notice that the 7-segment display has a much wider body than a typical 10-DIP IC. You will need to modify the courtyard and the silkscreen markings according to the datasheet.

The footprint of the 16 MHz crystal can be created manually from its datasheet. Given the practice we have done so far, it should be relatively straightforward to do.

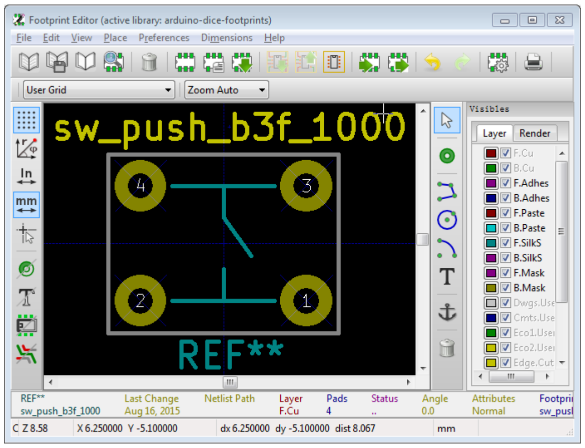
2.6 PCB Layout

2.7 Generating Fabrication Files

3 Advanced Topics

3.1 Controlled Impedance Lines

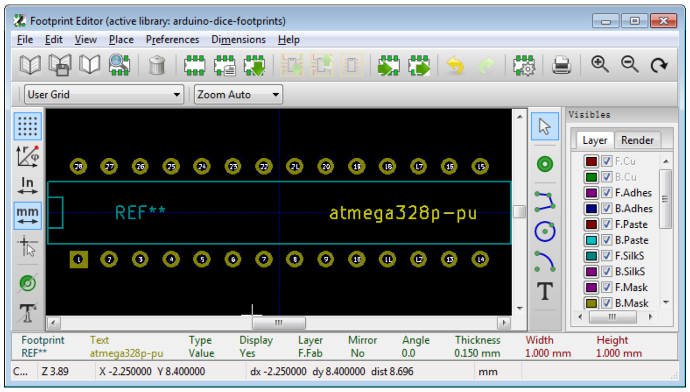
* Transmission line parameter calculator: http://wcalc.sourceforge.net/



REF**	Last Change	Netlist Path	Layer	Pads	Status	Angle	Attributes	Footpri
sw_push_b3f_1000	Aug 16, 2015		F.Cu	4	..	0.0	Normal	sw_pusl

```
C Z 8.58    X 6.250000  Y -5.100000    dx 6.250000  dy -5.100000  dist 8.067    mm
```

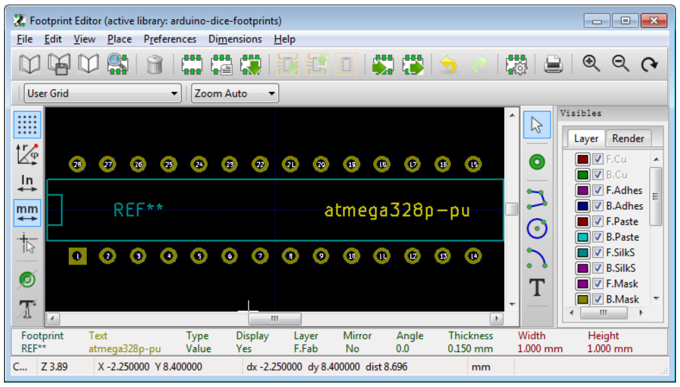
Figure 19: Completed footprint for the Omron B3F-1000 push button switch.



Footprint	Text	Type	Display	Layer	Mirror	Angle	Thickness	Width	Height
REF**	atmega328p-pu	Value	Yes	F.Fab	No	0.0	0.150 mm	1.000 mm	1.000 mm

```
C...  Z 3.89    X -2.250000  Y 8.400000    dx -2.250000  dy 8.400000  dist 8.696    mm
```

Figure 20: Completed footprint for the ATmega328P-PU microcontroller.



Footprint	Text	Type	Display	Layer	Mirror	Angle	Thickness	Width	Height
REF**	atmega328p-pu	Value	Yes	F.Fab	No	0.0	0.150 mm	1.000 mm	1.000 mm

```
C...  Z 3.89    X -2.250000  Y 8.400000    dx -2.250000  dy 8.400000  dist 8.696    mm
```

Figure 21: Completed footprint for the ATmega328P-PU microcontroller.

4 PCB design checklist

5 Further Reading

* KiCad footprint generator: http://kicad.rohrbacher.net/quickmod.php
  * KiCad library management: https://docs.google.com/document/d/1M38ByFyqnhwGo8b_jDDyBceyZtEG
edit
  * Easier way to via stitching: https://www.youtube.com/watch?v=Hp5ngKtl7S4

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.