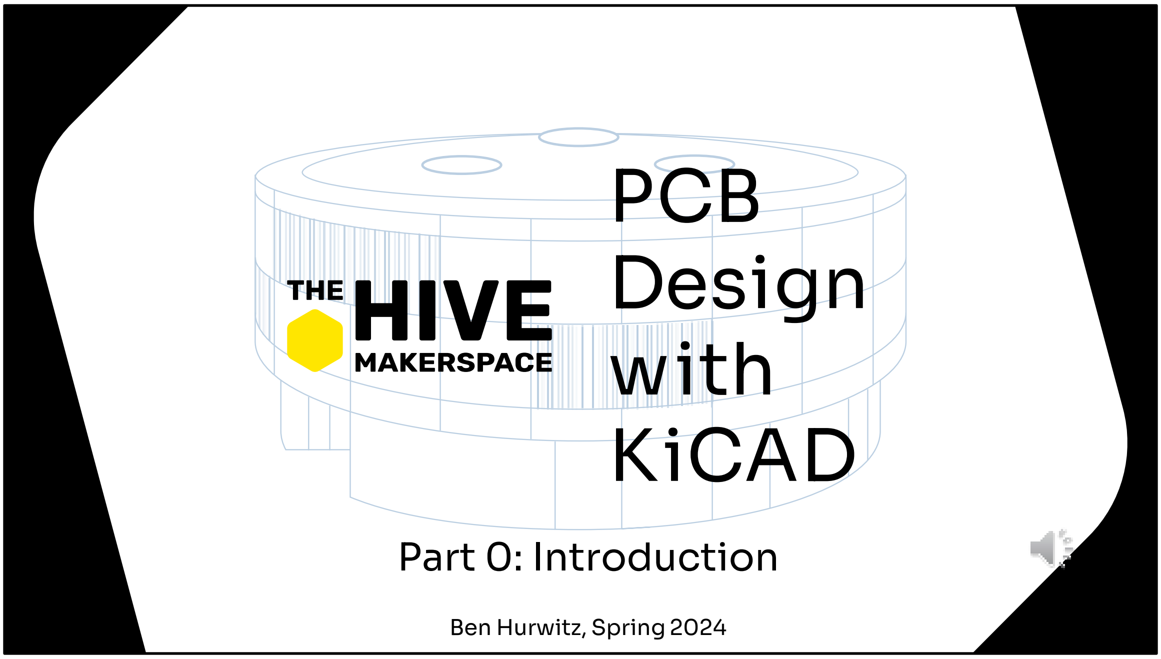


Hello, and welcome to The Hive's PCB Design with KiCAD video series. My name is Ben, and in this fifteen part walkthrough, I will guide you through the basic workflow of how to design a PCB, from conceptualization, through part selection, schematic capture, and layout, ending with a final board design that could be fabricated. The final four videos are not design-focused per-se, but discuss good library management practices and model creation.

In this video, we'll start with an overview of the series, offer some goals for what I hope you'll take away from this, and a short biographical sketch about me, your host.

Let's get started.

Series Overview

1. PCB basics and terminology
2. Electronics design software concepts
3. Circuit conceptualization
4. Schematic (5 parts)
   a) Adding parts
   b) Symbol creation
   c) Wiring
   d) Footprints
   e) ERC
5. Layout (3 parts)
   a) Setup
   b) Placement and routing
   c) Final DFM checks and DRC
6. Symbol libraries
7. Footprints (3 parts)
   a) Footprint libraries
   b) Custom footprints from scratch
   c) Custom footprints with the wizard

Goals for this tutorial

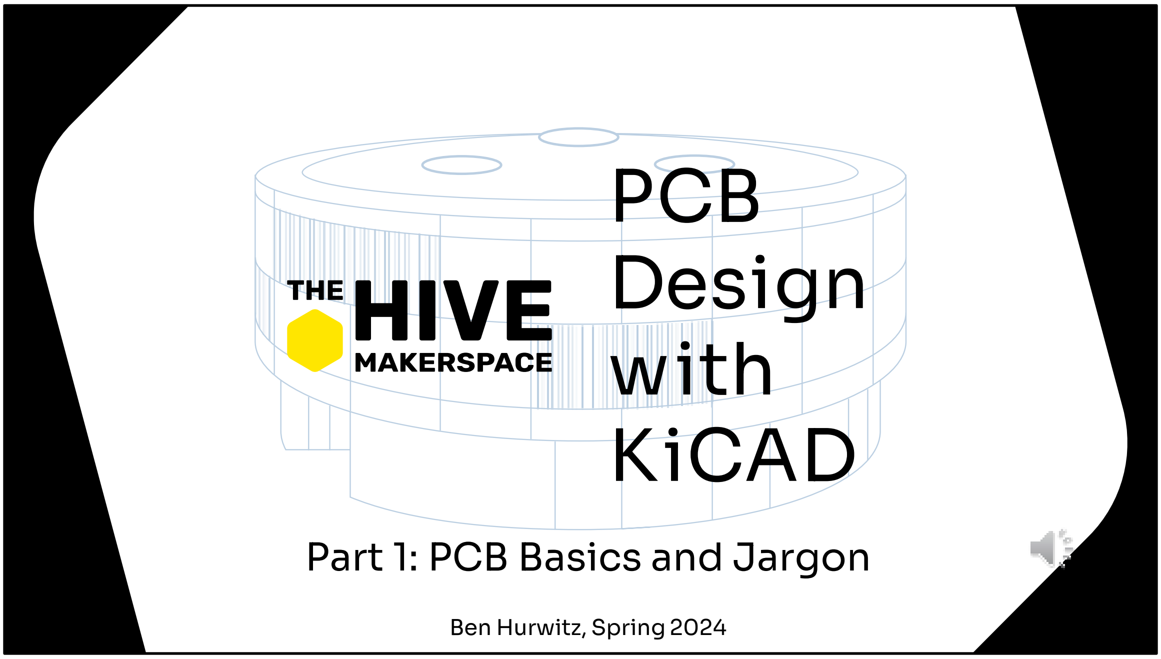
1. Introduce you to PCB design terminology
   - Understand the jargon, be able to have a conversation, and transfer knowledge to other PCB design software
2. Demonstrate the PCB design workflow
   - Learn how to actually design a circuit board
3. Make custom models, or locate pre-designed ones online.
   - Making your own is the very last resort.
4. Provide you with additional resources
   - Where to find answers to questions and additional tools/techniques

Who am I?

- As of this writing, I'm a 7th year PhD candidate in the Ocean Science and Engineering program trying desperately to graduate.
- My work was in developing a novel ocean sensor and the accompanying instrument, which turned into a lot of KiCAD development.
- I am not an expert. I just have experience with this software.

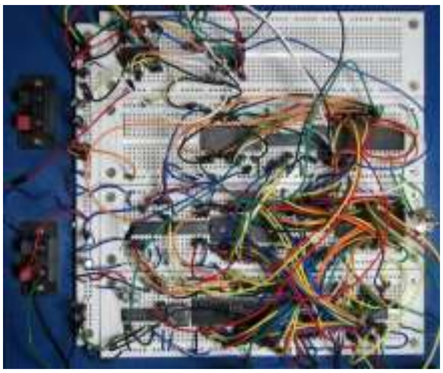
And with that, we'll end the introduction to the series. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In the next video, I'll introduce PCBs and some of the jargon you'll encounter on your journey through PCB design and fabrication.



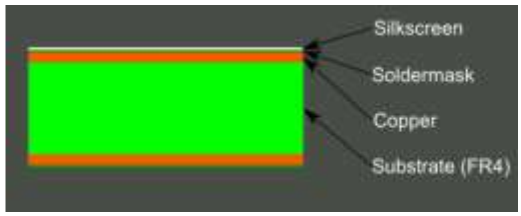
Hi, and welcome to part 1 in The Hive's PCB Design with KiCAD series. My name is Ben, and in this video, I'll be covering some PCB basics and terminology. Let's jump into it.

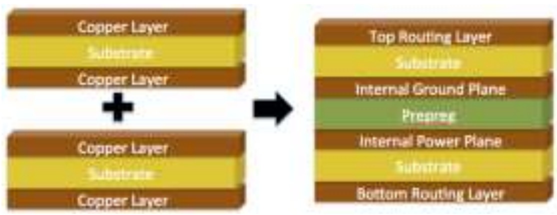
What is a PCB?



The traditional ways of connecting chips and components together has been through wire wrapping and breadboarding, but those come with a whole host of non-idealities and parasitics, are dreadful to debug, and are difficult to safely integrate into large systems. PCBs, short for “printed circuit boards” eliminate the wires and plug-in components in favor of metaphorically printing the wiring and connection points onto a flat, typically rigid, surface, and then soldering the components onto points for a mechanical and electrical connection. This helps to both alleviate many of the parasitics (though not eliminating them), as well as making the board much cleaner for visual acuity and mechanical safety and integration. PCBs are not truly printed these days, but the name remains.

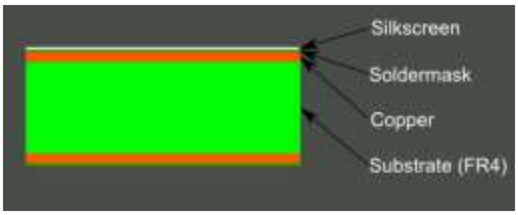
PCB Structure

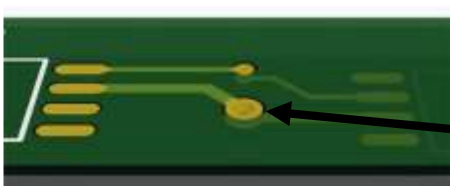




PCBs are physically composed of different layers that each fulfill different purposes. The solid non-conductive portion that forms the backing of the circuit is known as the substrate, and is typically made of a polymer-resin called FR4 (though there are many other options these days). PCB stackups (meaning the ordering and structure of the compositional layers) are generally described by the number of metal layers built in. It's most common that this metal is copper, though others are sometimes used. The thickness of the metal can vary; in the US, it's generally described by ounces per square foot, and you'll most commonly find quarter-ounce, half-ounce, one-ounce, and two-ounce weights, though others are possible, whereas internationally these would be described in microns, with half-ounce copper equating to 18 microns and full-ounce as 35 microns. Over the external copper is a layer of protective paint called soldermask, which protects the copper from oxidation, physical damage, and electrical impropriety. Finally, over the soldermask, an ink known as silkscreen may be applied to add text or graphics to either surface of the board. Additional internal copper layers may be built into the stack as well to create 4, 8, or even more layers for additional wiring area and heat conduction.

PCB Structure - Electroplating





Must be plated!

Electroplating (or just plating):

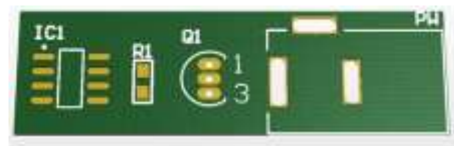
The process by which a hole between two copper layers will be internally metalized to create an electrical connection between those layers.

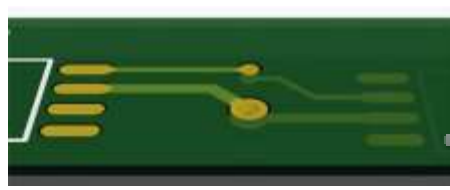
- “Internally metalized” means to add a layer of metal to the interal sidewalls of the hole.

I won’t go into much detail about the fabrication process during these videos, but there is one process that needs to be understood, and that is electroplating. Electroplating described the electrochemical process by which copper can be grown onto a variety of surfaces, and is the most common way to add copper onto a substrate in the PCB world. Electroplating, or just “plating”, is most commonly heard when referring to vias. Consider a component sitting on the top side of the board. If it needs to cconnect to something that is also on the top side of the board, no problem – we just connect with a single trace (which is what we call the printed copped wire) along the surface. But what if we need to connect it to an object on the bottom side? A hole can be drilled through the substrate to make a path, but it is still non-conductive. Vias are electroplated holes that connect traces across different copper layers. These plated holes are usually quite small. Through-hole components also typically use plated holes. Some holes, such as mounting holes, may be left as non-plated through-holes.

If you’d like to know more about the actual fabrication process itself, The Hive has a basic set of tools with which you can learn a basic method of hobbyist-level fabrication, or Google to learn the nitty-gritty of production-grade manufacturing processes.

PCB Terminology

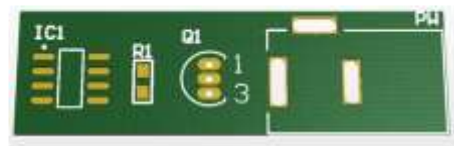


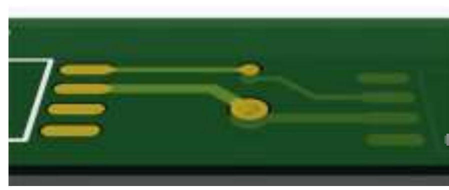


Let's go through some PCB terminology. All of this will be in relation to the graphics on the right.

PCB Terminology

- Green is soldermask
  - Electrically isolates copper
  - Mitigates corrosion and other physical damage

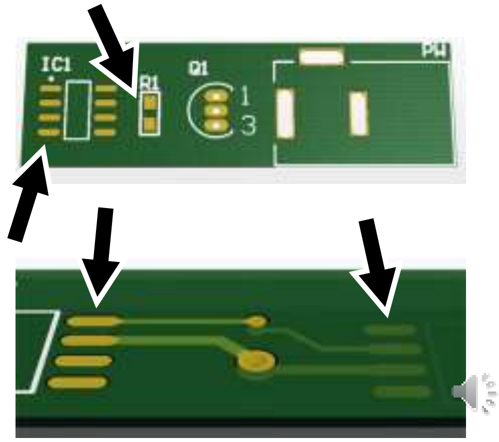




The green surface coating is the soldermask that, as I mentioned before, acts as a protective layer over any copper underneath. These days, soldermask comes in a wide variety of colors, including black, white, red, purple, and yellow.

PCB Terminology

- Green is soldermask
- In copper (gold):
  - Surface-mounted (SMD) pads
    - Connection points for surface-mounted legs/pins

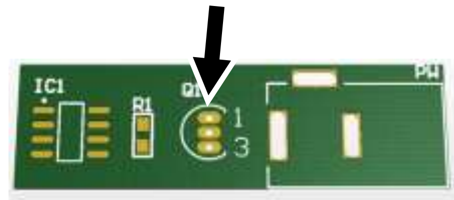


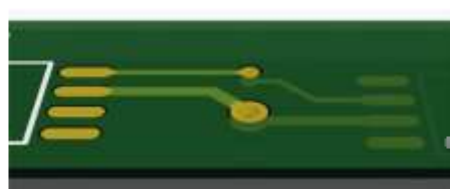
The gold is copper. All the places at which a component needs to be connected to the board will be free of soldermask to allow for soldering, but because copper oxidizes so readily in ambient atmosphere, the copper is nearly always coated in another less-oxidizable metal, a process known as finishing. Gold is a typical finish, which is why often the so-called “copper” actually is golden in color. Tin is a less-expensive common alternative.

Areas of exposed metal include surface mounted pads, where surface mounted devices connect to…

PCB Terminology

- Green is soldermask
- In copper (gold):
  - Surface-mounted (SMD) pads
  - Through-holes (TH), with rings
    - Connection points for through-hole leads, with copper rings for soldering
    - Plated by default (unless no plating available)

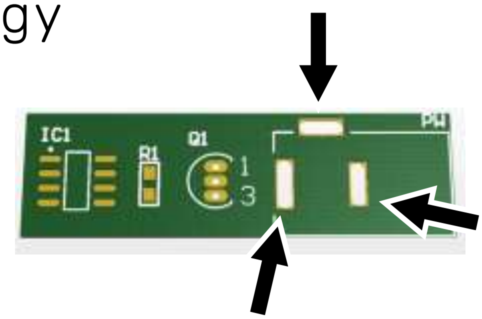


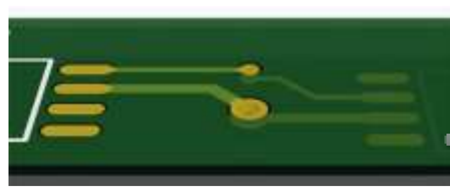


.... Annular rings around through-holes...

PCB Terminology

- Green is soldermask
- In copper (gold):
  - Surface-mounted (SMD) pads
  - Through-holes (TH), with rings
  - Slots, with rings
    - Non-circular THs
    - May or may not be plated, depending on design

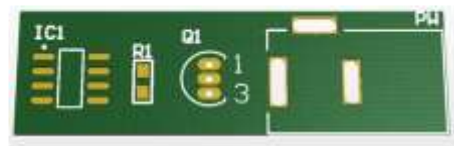


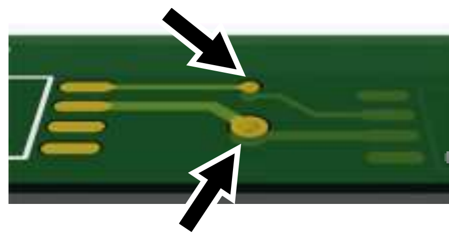


… slots….

PCB Terminology

- Green is soldermask
- In copper (gold):
  - Surface-mounted (SMD) pads
  - Through-holes (TH), with rings
  - Slots, with rings
  - Via, with annular ring
    - Connects traces across copper layers through the substrate
    - Also used for heat dissipation

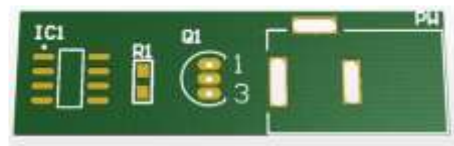


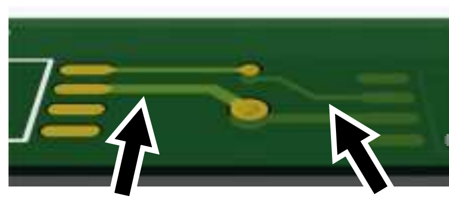


… and vias. These rings of copper allow for solder joints to be made between components and the covered traces; without them, soldering would be much more challenging. Through-holes describe any circular hole through which a component, electrical or not, is slotted. As mentioned before, these are typically plated by default. Slots are non-circular through-holes, and may or may not be plated. Vias are the inter-layer connection holes that must be plated, and can also be used for heat conduction and dissipation.

PCB Terminology

- Green is soldermask
- In copper (gold):
  - Surface-mounted (SMD) pads
  - Through-holes (TH), with rings
  - Slots, with rings
  - Via, with rings
  - Traces/tracks/routes
    - Connects pads or rings (i.e. components)



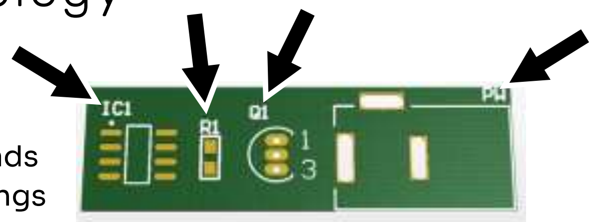


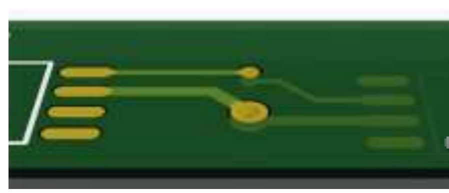
Finally, in copper, we have traces, also known as tracks or routes, that run between pads and plated holes to electrically connect components and devices together. These are usually hidden by the soldermask, but can often be seen as slight ridges pushing up from underneath.

PCB Terminology

- Green is soldermask
- In copper (gold):
  - Surface-mounted (SMD) pads
  - Through-holes (TH), with rings
  - Slots, with rings
  - Via, with rings
  - Traces/tracks/routes

In silkscreen (white):
- Reference designators
  - Part identification for assembly





In these graphics, the silkscreen is in white, though modern fabrication houses can (like with soldermask) print silkscreen in a wide range of colors. Silkscreen is generally used to print informational text and graphics onto the board with a non-oxidizing ink.

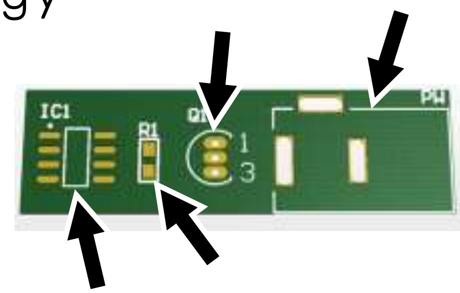
This informational text can include reference designators, which are part identifiers for assembly…

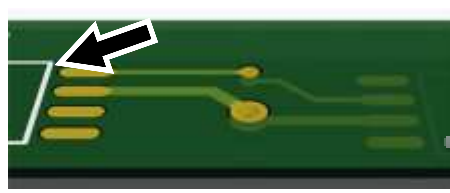
PCB Terminology

- Green is soldermask
- In copper (gold):
  - Surface-mounted (SMD) pads
  - Through-holes (TH), with rings
  - Slots, with rings
  - Via, with rings
  - Traces/tracks/routes

In silkscreen (white):

- Reference designators
- Part outlines
  - Shape and orientation; “keep-out” area for avoiding part overlap

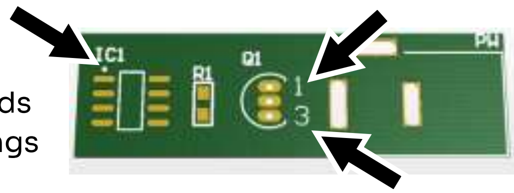




…. Part outlines used for orientation and avoiding part overlap….

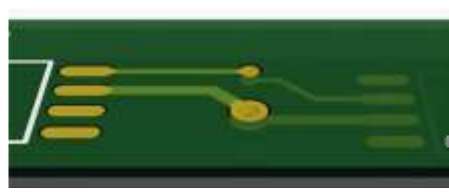
PCB Terminology

- Green is soldermask
- In copper (gold):
    - Surface-mounted (SMD) pads
    - Through-holes (TH), with rings
    - Slots, with rings
    - Via, with rings
    - Traces/tracks/routes



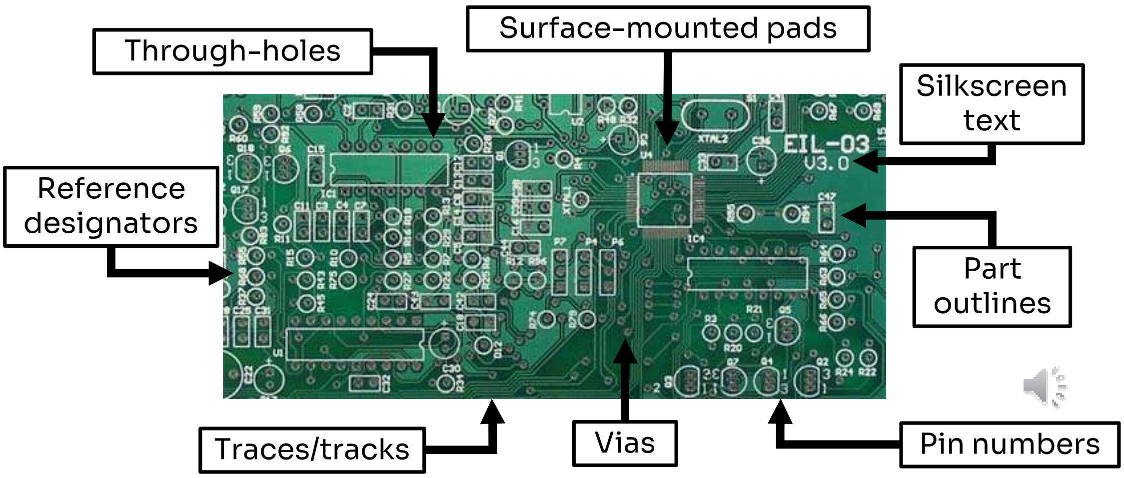
In silkscreen (white):

- Reference designators
- Part outlines
- Pin numbers/indicators
    - Pin function ID and part orientation



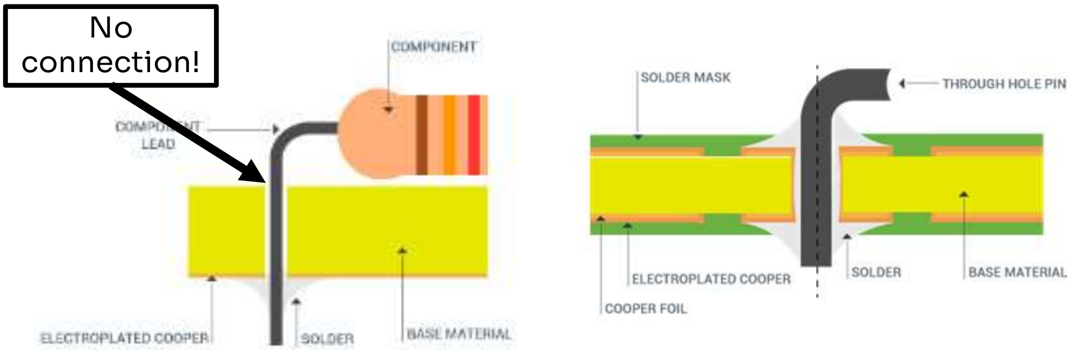
…. And pin numbers or indicators for functional referencing and part orientation. A common use-case is a small dot indicating pin one of an IC. Silkscreen is also used to identify the designer, projects, revision, year, company, warnings, and important information such as input range or mechanical restrictions.

Putting it all together…



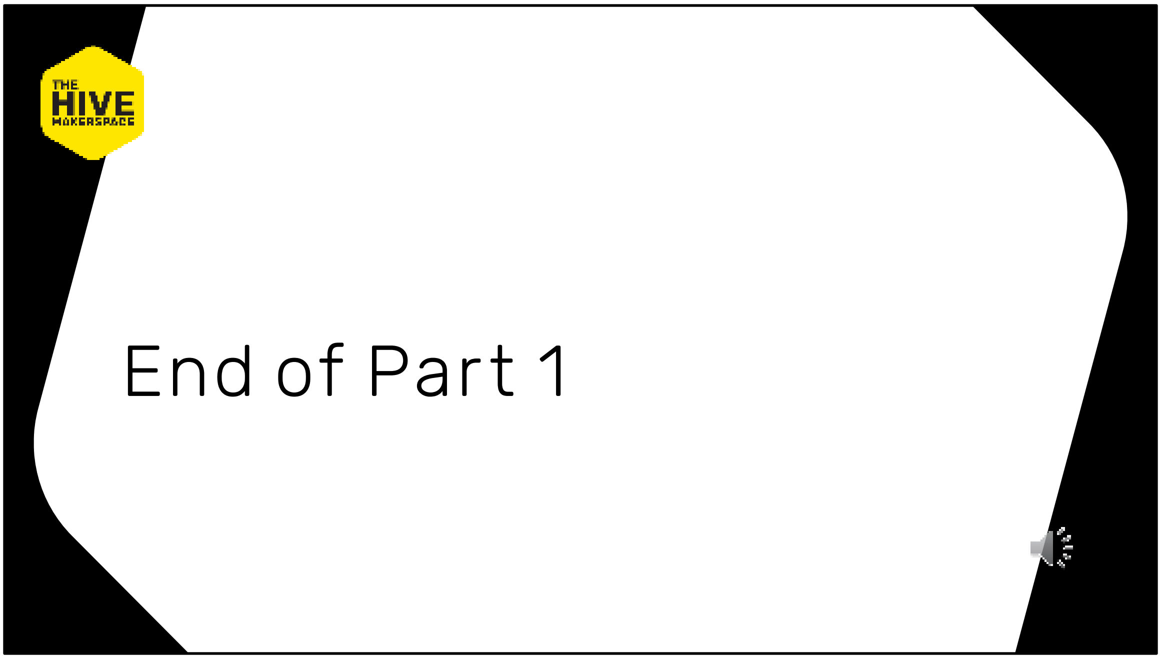
PCBs are commonly filled with these elements as identified here, with reference designators, through-holes, surface-mounting pads, text, part outlines, pin numbers, vias, and traces, all being present to enable assembly and use of the board.

Important fabrication note!



Single-sided solder   Double-sided solder

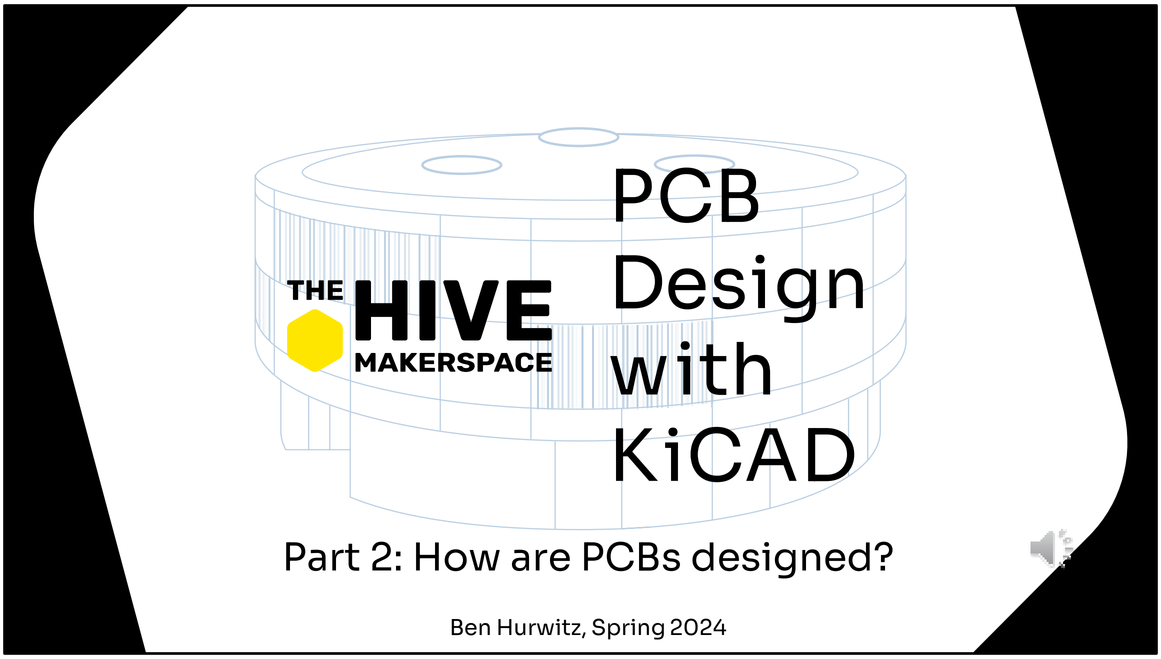
One last important note about fabrication that you should keep in mind while designing is the idea of where the component sits relative to where the traces are, and how the component will connect to the trace. Surface-mounted components, for example, require vias to connect between layers. Through-hole component have built-in holes, but if those holes are not plated, you must be aware of where the trace comes from. Consider the case on the left, a through-hole component with a non-plated hole. If the trace is on the bottom side of the board, there's not problem because the solder joint connects to lead to the annular ring of the hole (and therefore the trace). However, if the track is on the top side of the board, the component lead will not be connected to the trace because the hole is non-conducting, and the lead is not soldered on the top side. This issue can be avoided by either: 1) placing traces and components properly with non-plated through-holes to avoid this; 2) using the double-sided solder method shown on the right, if possible; or 3) always plating your component through-holes. Consider your fabrication house, cost, and time when making these choices.



And that ends part 1 of The Hive's PCB Design Tutorial with KiCAD. Today, we covered what PCBs are, and a lot of jargon and terminology surrounding their design and fabrication. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In part 2, I'll introduce PCB design software with a broad overview of how this family of software works, bring KiCAD into our lexicon, and end with a generic PCB design flow that we'll try to follow throughout the subsequent videos.

See you then!



Hi, and welcome to part 2 of The Hive's PCB Design With KiCAD tutorial series. My name is Ben, and in this video, I'll be going over how PCB design software is structure, and describe the basic design flow you might follow to go from an idea to a read-to-fab board. Let's get started.

PCB Design Software Overview

- PCBs are designed with specially-crafted software called, among others, “e-CAD”, “PCB CAD”, or “EDA”
- There are many different such applications with different strengths and weaknesses
- Understanding layers is key
  - Placing traces on a silkscreen layer will cause your circuit not to work (and you’ll be sad)

PCBs are designed with specially-developed computer-aided design software, known equivalently as e-CAD, PCB CAD, or EDA for “Electronics design automation”. Many different software applications and suites exists that can perform the required functions for this process, and all of them have strengths and weaknesses. However, the jargon and processes are generally very similar, meaning if you learn one, you have the tools to learn others with relative ease; it’s often just a matter of discovering the new locations of the necessary icons and settings.

The understanding of layers and how the layers are represented on screen versus their physical manifestation is critical. Place your components or object polygons onto the wrong layer and you’ll be very sad later (and probably a bit less wealthy).

PCB Design Views

- All such software involves making a *schematic*, and then arranging components on layers in the *layout/PCB*
- The *schematic* is the circuit diagram - what we might hand-draw on paper
- The *layout* (which KiCAD calls the "PCB Editor") is what the circuit board actually looks like, in *layers*
- These views are inextricably linked together so that changes in one are appropriately reflected in the other

Note! The layout will often look <u>*nothing*</u> like the schematic! And that's fine – the schematic is for people to read, whereas the layout is for electrons to read.

PCB CAD software uses two primary views, the schematic and the layout, which can also be called the PCB view. The schematic is for the circuit diagram, what might be drawn on paper with symbols and lines for connections. The layout is the physical design of the board in layers – the size and placement of components, the actual location of routes, and the mechanical constraints. These views are inextricably linked so that changes in one will be reflected in the other.

It's very important to recognize that the layout will often look nothing like the schematic. And that's fine, because schematics are for people to read and understand, whereas the layout is for electrons to travel. These two views do not have the same goals.

Layers

- PCBs are designed in a stack of metaphysical layers that use polygons to indicate which objects are filling the physical space
  - Similar to Photoshop or Illustrator or other graphics design software
- There are many available layers in EDA software, many of which you will not need or use
- Here are some important ones (using KiCAD names):
  - F.Cu (and B.Cu) – the front/back copper
  - F.Silkscreen (and B.Silkscreen) – the front/back silkscreen
  - Edge.Cuts – the board's physical outline/dimensions
  - Less used: Adhesive (for solder glue), Paste (solder paste), Mask (soldermask), and Courtyard (part body outlines)
- Software terminology will vary - Google is your friend!

As I mentioned before, layers are a crucial component to PCB design. Each layer in the software will either describe a physical layer within the actual manufactured stackup, such as copper or silkscreen, or an informational layer for the designer to add information for themselves or the fabrication house, such as dimensions, part names, project revisions, and more. These layers are similar to other graphic design tools, such as Photoshop or Illustrator, but they generally don't have priority; overlapping polygon on different layers is typically acceptable (though not always).

There are numerous layers that you will need to become familiar with, though there are often many more within a given software package that are either left unused or you will not interact with. The most commonly used ones include those for the copper layers, the silkscreen layers, and the board's outline layer. Less used layers include those for solder glue, paste, and mask, as well as part names and outlines.

The layer names given here are for KiCAD, whereas other software will likely use different naming conventions. Google is your friend here.

How do parts work?

- Virtually all components you add to your design will comprise two models: a *symbol* and a *footprint* (and maybe a 3D model)
- *Symbols* are the schematic representation of a part – e.g. the squiggle of a resistor, the coil of an inductor
  - Symbol connection points are called *pins.*
- *Footprints* are the physical form of the part – through-hole sizes, pitch (spacing), body shape, etc.
  - Footprint connection points are called *pads*. (Yes, even TH ones.)
- Some software only allow you to place *devices*, which are a fixed, pre-defined, symbol/footprint combo
  - This means that every different footprint for the same symbol require different devices -> e.g. dozens of different resistors
- Symbols, footprints, devices, and 3D models are stored in *libraries*.

Similar to how there are two views for the entire PCB, each part or component that are included in the design (even non-electrical ones) have at least two models: a symbol for the schematic, and a footprint for the layout. They may also have 3D CAD models as well for a 3D view or for exporting to mechanical design or simulation software. Symbols are the schematic representation of a part – think the squiggle of a resistor or the parallel lines of a capacitor – with pins to define the symbolic connections points. Footprints are the physical shape of the device, including the body, through-hole, or surface-mounted pads, and are digital representations of the component's package. The connection points for footprints are known as pads, even when they are through-holes. Some software requires you to link a symbol and footprint together into a single combined model called a device. All models are stored in libraries.

How do parts work?

- KiCAD does not have devices; in KiCAD, *any symbol may be associated to any footprint*
  - This allows a lot of flexibility, but also requires careful selection of footprints for every part to match pins and pads.
- Be careful to select the correct footprint

KiCAD does not use devices, meaning that any symbol may be associated with any footprint. This allows a lot of flexibility – you don't need to have a thousand different devices for resistors alone, and you can use the same footprint for many symbols very easily – but it also opens the door for designers to accidentally select the wrong package, and requires careful alignment of the symbol's pins with the footprint's pads.

How do parts work?

- Many standard components (e.g. Rs, Cs, Ls, diodes, transistors) have standardized symbols and footprints built into KiCAD that you can easily use
- ICs and non-standard components will likely not be
- If the symbol/footprint you need is not in KiCAD:
  1. Check the internet for them (work smarter, not harder)
  2. Draw them yourself
- We'll go through both methods later.

KiCAD, and most software, have many built-in libraries that can (and should) be used for standard parts, such as passives, diodes, and transistors. ICs and non-standardized components may not be as generic, and therefore may not be built into the software. One advantage of separating symbols and footprints is that a part may have a standard symbol or footprint but maybe not both; in KiCAD, that's easily handled, but in device-based software, it would be more challenging, and would generally require creating the other half of the device model.

In general, you should not be creating your own models; that should be the last resort because it's tedious, time-consuming, and prone to errors. There are dedicated companies out there that will create electrical models for free. Use these services instead to reduce the amount of busy work you have to do. I'll discuss both methods later.

Design Rules – ERC and DRC

- Both the schematic and the layout have a set of rules for allowable designs
  - Generally, the schematic rules (“electrical rules”) are fixed
  - The layout rules (“design rules”) can be changed depending on the fabrication house and/or requirements
- All CAD software can check your schematic/layout against the rule sets – the *ERC* and *DRC*, respectively

It’s critical to run these checks throughout the process to avoid incompatible or nonfunctional designs!

Lastly, there are two sets of rules that your board must adhere to to be fabricated successfully. The first are the electrical rules that apply to the schematic, things like whether wires are connected and if pin types match. These are typically fixed and don’t need adjusting. The second are design rules that apply to the layout, such as hole sizes and copper separation distances, and these come from your chosen fabrication house and design requirements. Any e-CAD software will be able to check your designs against the electrical rules, calls an ERC, and the specified design rules, called a DRC. It’s super important to both make sure that your design rules are correct by reading your fab house’s instructions carefully, and to run these checks multiple times throughout your design iteratons to avoid error propagation and incompatible or nonfunctional designs.

Why KiCAD?

- Relatively low barrier to entry
- Good for introducing concepts without too much overhead
- Free and open-source
  - + Active community of support and developers
  - + Cross-platform operation
  - + No cloud operation for you to get locked into
  - - No big company support
  - - No fancy integrations with external software
  - - A little more rough at the edges than industry-standards
- Customization with Python scripting. (Not covered here.)

KiCAD, as I've mentioned, it just one such software suite that can make circuit boards. It has a number of advantages that I think makes it a good tool for learning this process. There is a relatively low barrier to entry relative to the industry titans, making it good for introducing concepts without all the extra overhead. It's free and open-source, meaning you can use it even if you're not with a large company, with a large an active community of support and development for cross-platform operation. Additionally, there is no cloud storage to lock your designs into. The lack of big company support may be detrimental for certain reasons, however, such as missing external integrations and advanced functionality, and it can be a little more rough at the edges due to the lack of dedicated development engineers. This missing functionality, and much more, can, however, be added through the use of custom plugins and modules, written in standard Python.

KiCAD tips and tricks

- Rolling the scroll-wheel will zoom.
- Clicking-and-dragging with the scroll-wheel will pan.
- The “Insert” key will typically repeat the last command.
- Most text inputs allow for math equations and unit conversion. Useful between metric and imperial.
- CTRL + F1 shows all the available hotkeys. Pretty nice.

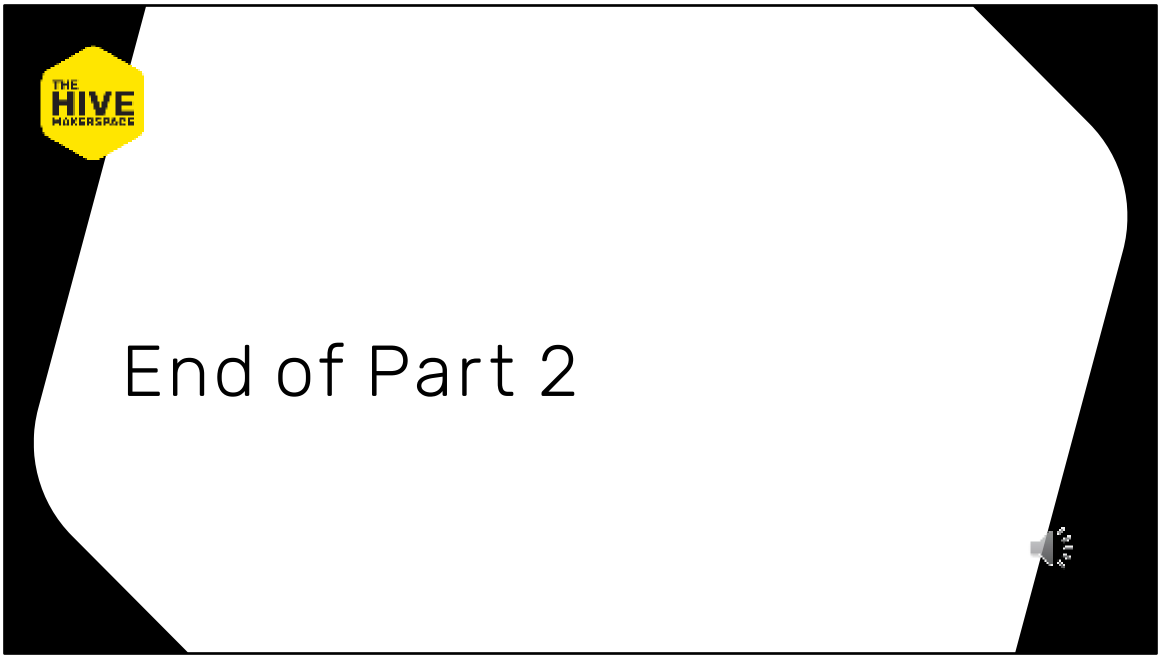
KiCAD has a few program-wide shortcuts that are good to know about before we get into it. Like more CAD software, a three-button mouse is highly advantageous, with the scroll-wheel being used to zooming and panning. The insert key will typically repeat the last command (useful for placing multiple parts), and most text input will hande math expressions and unit conversions. Finally, there are many shortcuts, most if not all of which are customizable, and CTRL + F1 will show them all.

EDA Design Flow (Broadly)

1. Conceptualize the circuit. Breadboard/simulate, perhaps.
2. Part selection
3. Generate libraries and determine design rules
4. Create your schematic
   1. Add symbols to the schematic
   2. Connect symbols with wires
   3. Assign footprints to symbols (ERC!)
5. Layout your PCB
   1. Arrange footprints in layout
   2. Connect footprints with traces and planes
   3. Add finishing touches (silkscreen, teardrops, etc.) (DRC!)
6. Gerber/assembly files

Lastly, while designing any board has it's own special requirements and every tool has its own quirks, PCB design broadly follows the following sequence.

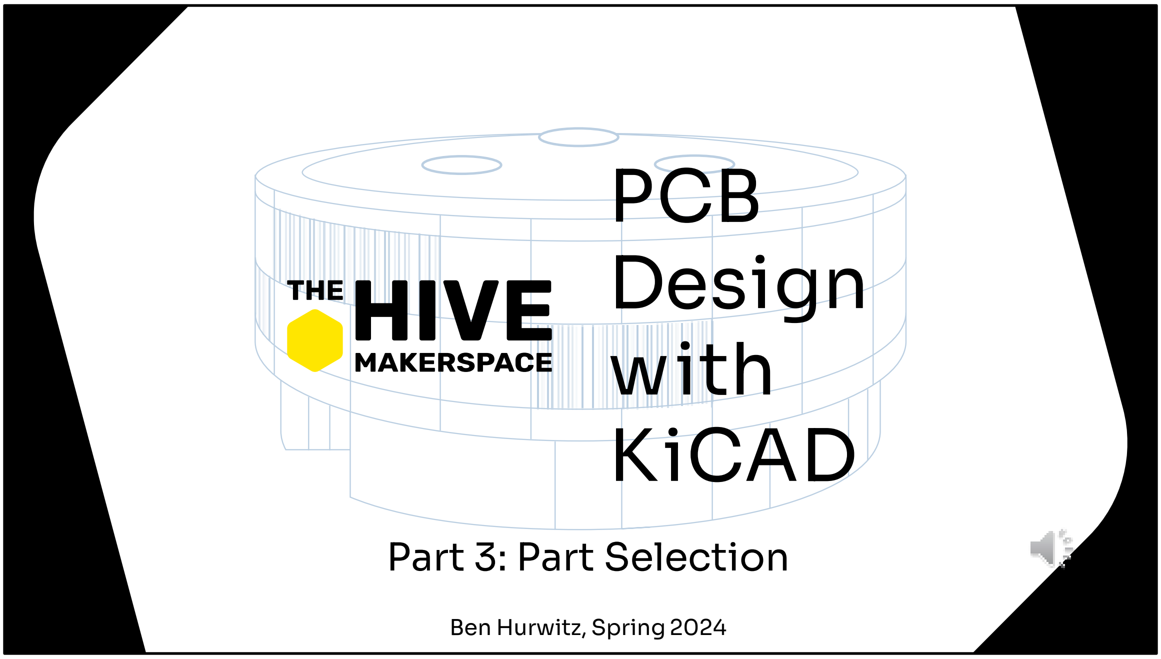
First, you conceptualize and ideate the circuit, with breadboards or simulations.
Second, you select and obtain your key parts, and anything that's either specific to the design or hard to get.
Third, create your libraries and setup your project, including determining a fab house and locating their design rules and requirements.
Fourth, create the schematic with symbols, wires, footprints, and ERC using an iterative process to finalize your design into something cohesive and coherent.
Fifth, lay it out, spending the majority of your time on placing the components, then routing, and finally adding any finishing touches, while running your DRC liberally and frequently to avoid any last-minute major errors.
And finally, sixth, once you've iterated enough, satisfied your design requirements, and get a clean bill of health from your ERC and DRC, plot the required gerber and assembly files and send to fabrication.
Of course, there's more after that, like waiting, testing, debugging, more iteration, and programming, but that's generally what it takes to go from an idea to a board.



And with that, we close the book on Part 2. We covered EDA software and how it works with a lot more terminology, introduced KiCAD, and gave a broad PCB design flow that we'll look to follow through the coming videos. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In Part 3, we'll start that process with an introduction to the circuit we'll be developing into a PCB, no electrical engineering knowledge required, and going through the process of part selection.

See you there!



Hi, and welcome to The Hive's series on PCB Design with KiCAD.

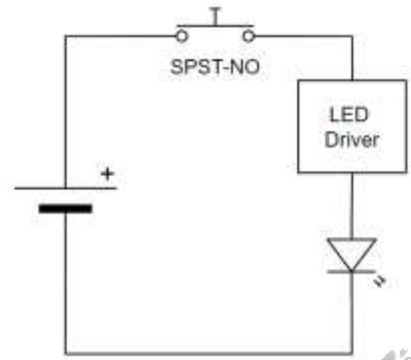
My name is Ben, and in this part 3, I'll be giving an overview of the circuit we're going to design a board for, and go through the part selection process.

I don't think you need any electrical engineering knowledge to understand this material since I'm not doing any actual theory; most of it is just why I picked the various parts, and how you might for your own projects.

Let's get into it.

What is the circuit?

- A battery-powered switched-LED circuit (a flashlight)
- An LED driver is an IC used to supply a stable current to 1-or-more LEDs
- Push-button SPST-NO switch will enable/disable
- Single coin-cell-type battery



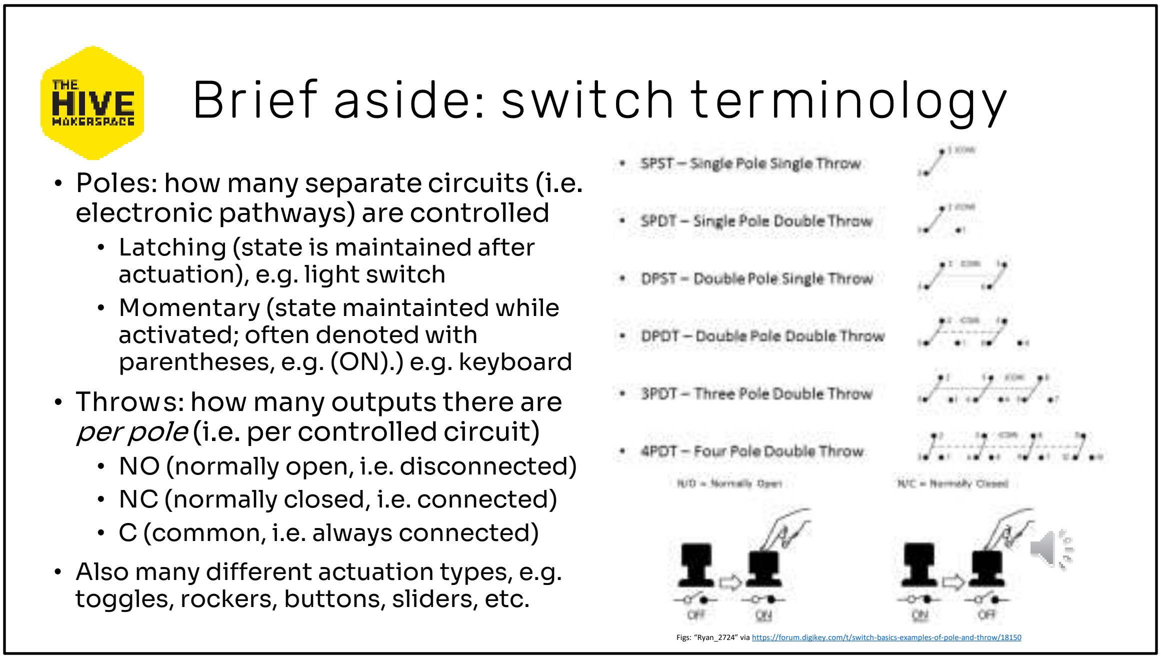
So what are we actually designing a board for?

It’s basically a flashlight, or more technically a battery-powered switched LED circuit.

An LED drive is an IC that provides a fixed and stable current and voltage at the output for driving a number of LEDs. It’s more stable than a battery and a resistor, and it can boost the input voltage up to drive many LEDs both in series and parallel.

A simple push-button tactile switch in a SPST-NO configuration will turn on and off the circuit.

The power will be provided by a single coin-cell battery.



Okay, so for those who don't know (or, like me, always forget) about switch jargon, here's a rundown.

The number of poles describes how many circuits the switch controls. This is not how many outputs, but literally how many circuits, how many electronic pathway selections. You should be able to see the difference between the first two on the right, which are both single pole, and the second two, which are double pole. Each pole can either be latching, where the state is maintained like a light switch, or momentary, where the state only changes for the duration of the actuation, like a keyboard key.

The second important term is the number of throws, which described the number of output per pole. So the first switch, the single pole single throw, controls one circuit with a single output. The second switch, single pole double throw, has one circuit with two outputs. The third is double pole single throw, so two circuits each with one output. And so on. Throws can be normally open, meaning disconnected, like the button on the lower right, or normally closed, meaning connected, like the button on the lower left. One of the terminals will be common as well, meaning always connected, and is mostly relevant for double throw (two output) switches.

Switches are typically named by the number of poles then the number of throws, single pole single throw, with the shorthand SPST. Single throw switches will often have a notation for whether they're normally open or normally closed.

Part Selection

- Before we begin CAD'ing, you want to either have your parts on hand, or know they're available.
  - Otherwise, you'll be sad once you've spent hours routing only to find the package you need doesn't exist.
- By having the parts ready, you are ready to generate or find the correct footprint as well, saving you future headaches.
- This is often very time consuming, but also critical to an operational board!

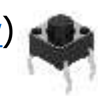
Part Selection

- This design was driven by a few factors (in no order):
  - I arbitrarily wanted to use a coin cell battery
    - Small, available, non-standard footprints for practice
  - I wanted to use an IC to demo locating symbol/footprints online
  - I wanted a low but varied component count
  - Low-cost
  - Conceptually easy

Part Selection

- How did I locate these parts?
  - I tried to find parts that are readily available in The Hive
    - These included Rs, Cs, Ls, LEDs, the switch, the battery, and its holder
  - Other components were sourced from Digikey
    - I'm familiar with them and navigating their website, but I'm sure I could have found most of this from Amazon, Mouser, or other supplier

Part Selection

- Battery: CR2032 
  - Standard coin cell (lithium-primary, 3V nominal), Hive-available
- Battery holder: MPD BC2032-E2 (Digikey) 
  - Hive-available (had to hunt a bit for a drawing)
- Switch: 6mm pushbutton tactile switch (e.g. Digikey) 
  - Standard, boring ol’ momentary on button, Hive-available
- LEDs: 5mm clear domed through-hole (e.g. Digikey) 
  - Again, standard, boring, Hive-available through-hole LED

I went with a coin cell because it’s small

Part Selection

- The LED driver was a bit more involved, as ICs usually are
- Digikey filtered for:
  - “in stock” and “active” status (always a good filter set)
  - a minimum supply voltage of ≤3V (so it works with the battery)
  - a surface-mounted package (arbitrary) that is hand-solderable (so no super-tiny or BGA-style packages)
  - Boost topoliges only (need to increase the voltage for the LED)
- This resulted in 26 options

Part Selection

- How to select from these when they are all similar?
  - Cheap and available is good
  - Read the datasheets for details – supplier can be wrong!
- I chose the Richtek USA Inc. RT4526GJ6 because:
  1. It's on the cheaper and available side
  2. It has an "enable" pin (shutting off power is less clean)
  3. The datasheet provides recommended components
  4. There is no available symbol/footprint (good for this tutorial, not for normal design work)

Part Selection

- The “typical application” circuit
- I_LED = 0.3/Rset -> for 10mA, Rset = 30 ohm
- For simplicity, Rset, Cin, Cout, and L will be Hive-available
- Diode is unavailable on Digikey; similar selected

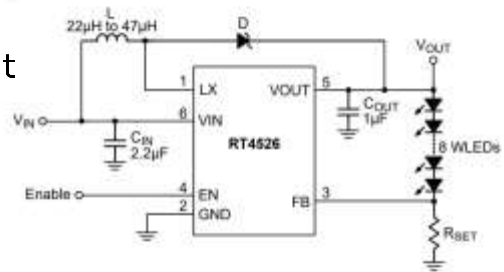
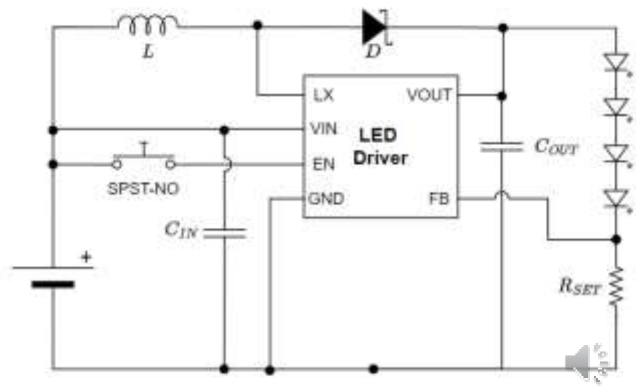


Table 1. Recommended Components for Typical Application Circuit

Reference	Qty	Part Number	Description	Manufacture
D	1	SR26	Schottky Diode	PANJIT
C_{IN}	1	EMK107BJ225MA-T	Capacitor, Ceramic, 2.2μF/16V X5R	Taiyo Yuden
C_{OUT}	1	GMK107BJ105KA	Capacitor, Ceramic, 1μF/50V X5R	Taiyo Yuden
R_{SET}	1	RC0603FR	Resistor 15Ω, 1%	[illegible]
L	1	NR4018T220M	Inductor, 22μH	Taiyo Yuden

Complete circuit and parts list

- LED driver IC
- Battery + holder
- SPST-NO switch
- Cin, Cout (caps)
- L (inductor)
- D (Schottky diode)
- Rset (resistor)
- 4x LEDs

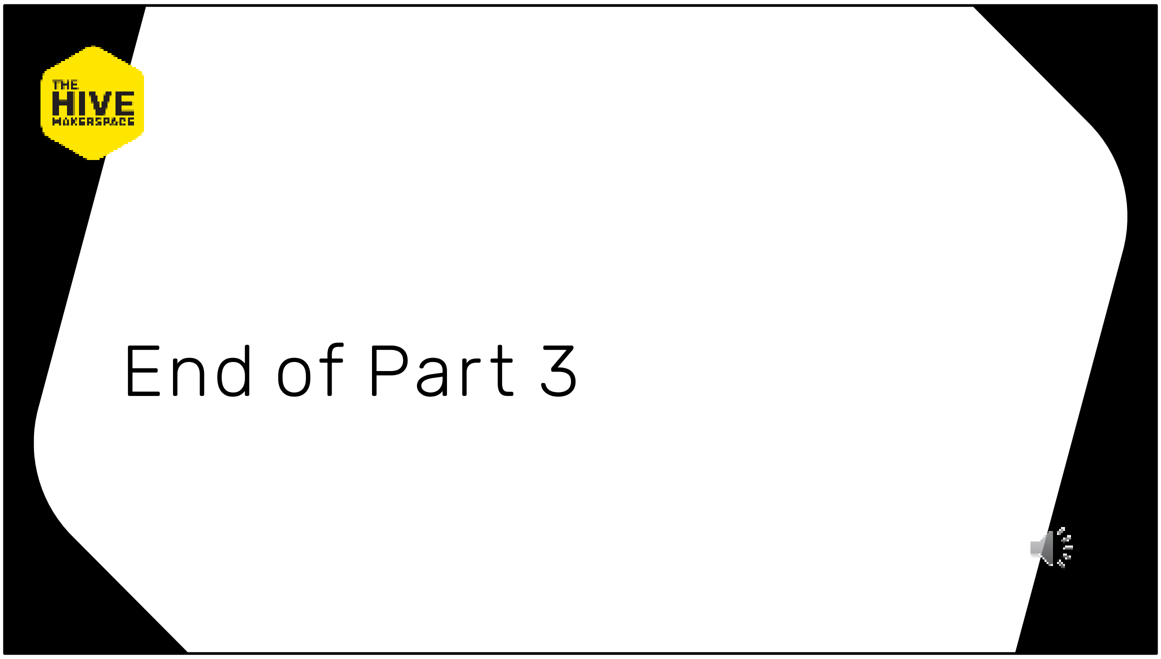


Here's the full circuit and parts lists. Don't worry too much about memorizing this or anything. I'll bring it back frequently as needed during the upcoming videos (though feel free to print it or whatever).

Full[er] BOM

Description	Part Num.	Mounting	Footprint
LED drive IC	RT4526GJ6	SMD	TSOT-23-6 (≤ 3.1 x 1.8 x 1 mm)
Battery holder	BC2032-E2	TH	Custom
Switch	TS02-66-70-BK-160-LCR-D	TH	4-TH 6mm x 6mm
Cin, 2.2uF	C3216X5R1C225KT	SMD	1206/3116 (3.1 x 1.6 x 0.55 mm)
Cout, 1uF	C3216X7R1C105KT	SMD	1206/3116 (3.1 x 1.6 x 0.55 mm)
L, 22uH	LBR2518T220M (22uH)	SMD	1008/2518 (2.5 x 1.8 x 1.8 mm)
D	PMEG6030ELPX	SMD	SOD-128 (4 x 2.7 x 1.1 mm)
Rset, 30 Ω	Unknown (from kit)	SMD	1206/3116 (3.1 x 1.6 x 0.55 mm)
LED	C512A-WNN-CZ0B0151	TH	5mm diam, 0.6mm lead holes

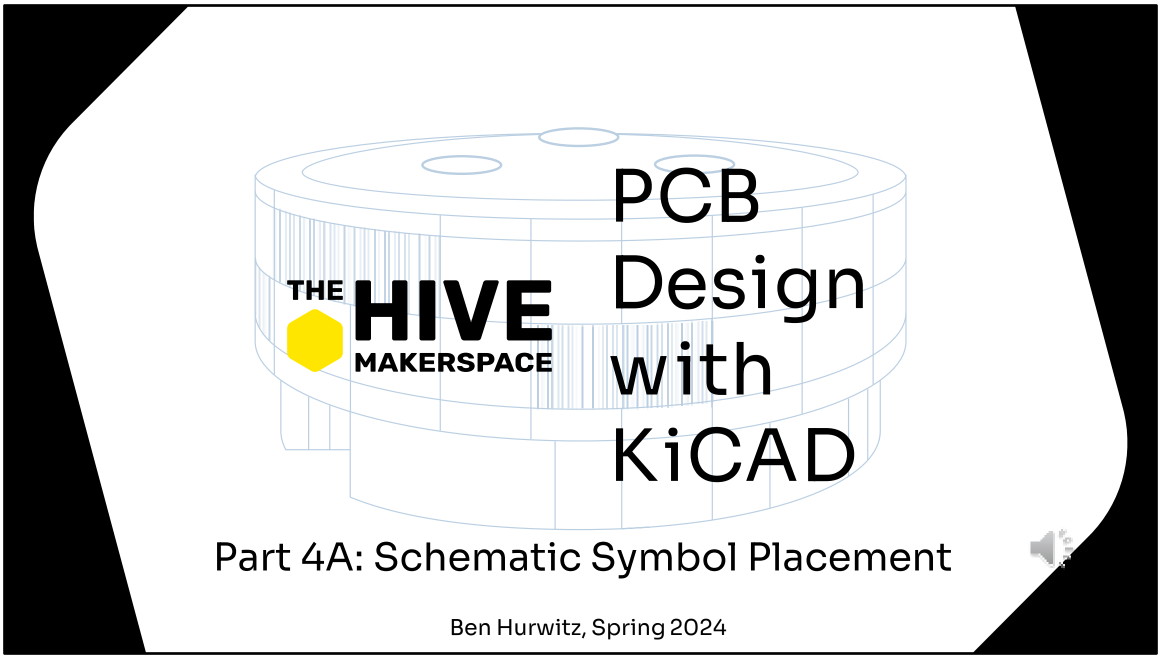
And this is a large bill of materials without any of the "bill" portion. We'll see this again later as well.



And with that, we've reached the end of Part 3, in which I introduced the circuit and went through the process of part selection. Hopefully this provided some insight on doing this process on your own. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In the next video, part 4A, we'll finally get into actual design with an introduction to KiCAD's schematic capture view and placing basic symbols.

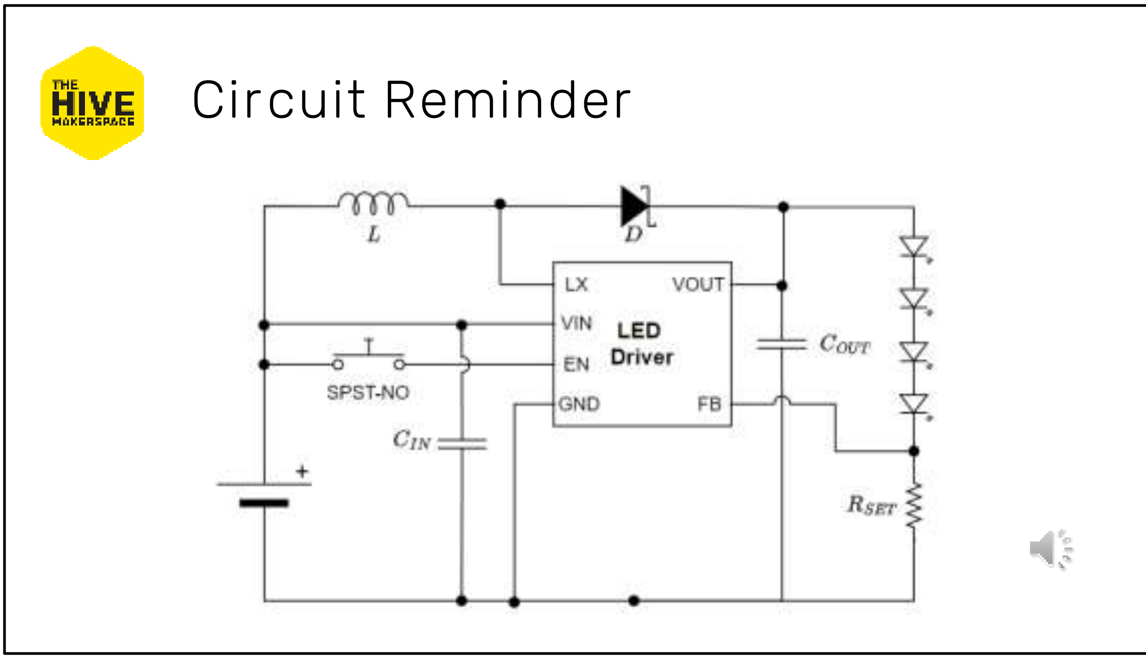
See you there!



Hi, and welcome to part 4a of The Hive's PCB Design With KiCAD series. My name is Ben, and I'll be your guide here. Part 4 as a whole will cover the entirety of the schematic creation. Part 4A will look at the schematic capture window, and detail how to add built-in symbols to the design.

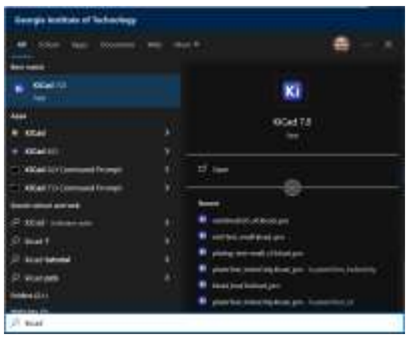
If you watched part 3, you might remember that in our design flow, there was a project setup and library creation step. Due to the relatively simplicity of this design, and that library management was not covered during the original workshop this tutorial is based off of, we're going to leave the library creations to parts 6 and 7. I strongly advise you to watch through those if you're considering doing design more seriously.

Anyway, let's get into KiCAD.



Before we get into KiCAD, just a reminder of the flashlight circuit we're developing. Note that this image was not take from KiCAD, and therefore the symbols and graphics are different from those you are about to see.

Open KiCAD



As an aside, if you're using Mac or Linux, some of these screens will look different since I did this all on Windows 10, but the concepts are the same.

Also, this is for KiCAD version 7. I'm sure a lot of the concepts will translate to future versions, but it may look different.

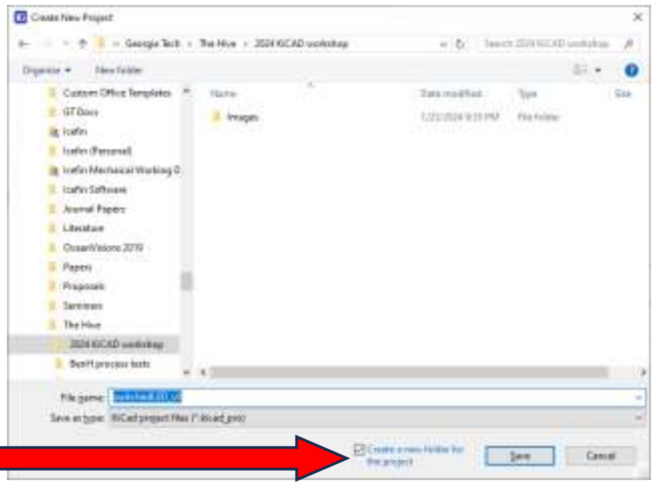
Open KiCAD however you normally open software.

As an aside, I did these slides in Windows 10 using KiCAD 7. If you're using a different operating system, or a different version of KiCAD, some of the graphics and visuals may look different or have slightly different wording from these slides, but the concepts should still hold.

Main Window

This is the first window that opens, called the Project Manager. From here, you, well, manage your project. It usually opens to some old project you've done, so we'll go ahead and create a new project under File > New Project.

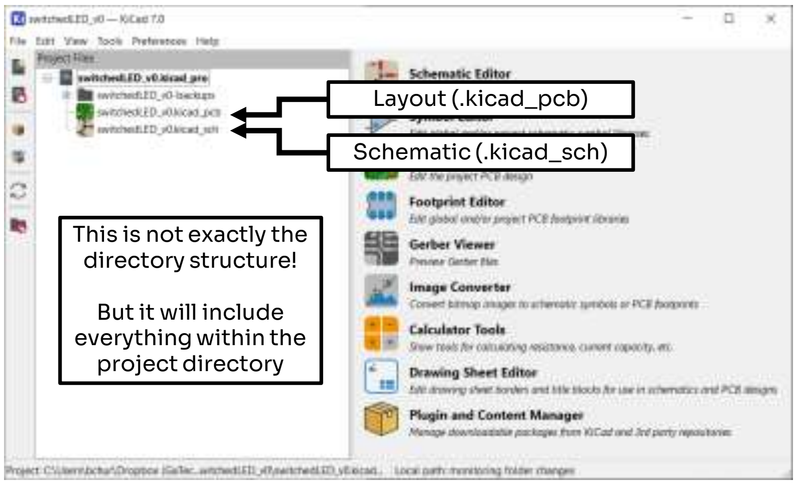
File > New Project (Ctrl+N)



Unfortunately, this name was picked for a simpler version of this circuit.

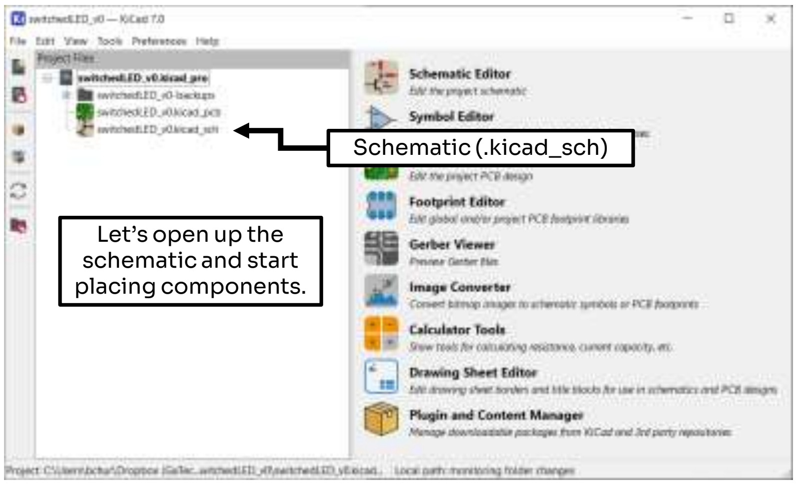
Name is and save it wherever you'd like. I don't this name was the best choice, but whatever. I also always like to check the box at the bottom that says "create a new folder for the project". Obviously I could create the folder myself, but it's easy to do.

Main Window - Working Project

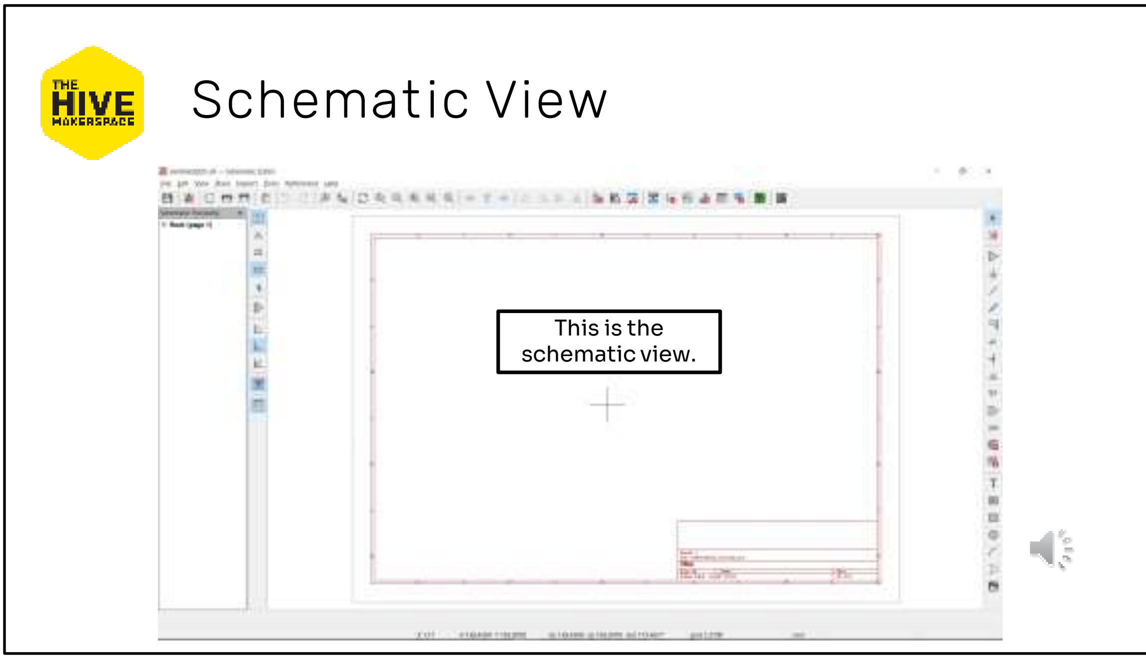


Great, so this brings our project into the Project Manager and shows us the project structure. *This isn't really a directory structure, though it looks like one. KiCAD will make backups of your project as pre-defined intervals, which can be set within the preferences, and are saved in the backups folder. On the right are shortcuts to the various tools that KiCAD offers. The two keys files are the *schematic, with the extension .kicad_sch, and the *layout, with the extension .kicad_pcb.

Main Window - Working Project

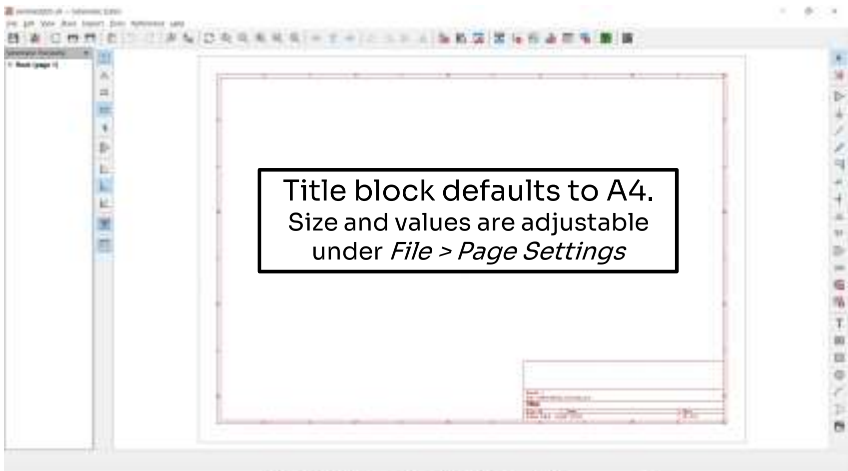


Let's open the schematic file by double-clicking to open the Schematic Editor.



This is the blank schematic view.

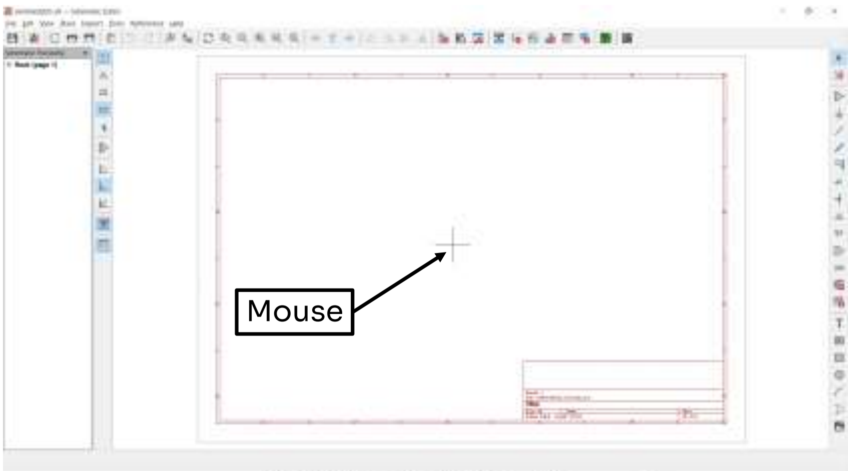
Schematic View



The title block defaults to A4. You can change that and adjust the text within it under the page settings. You can also safely ignore it and place whatever you'd like outside of it; it's not a limit at all. There is a way to remove it, but it requires making a custom drawing sheet. Not hard, just beyond the scope of this video. A link is found in the PDF:

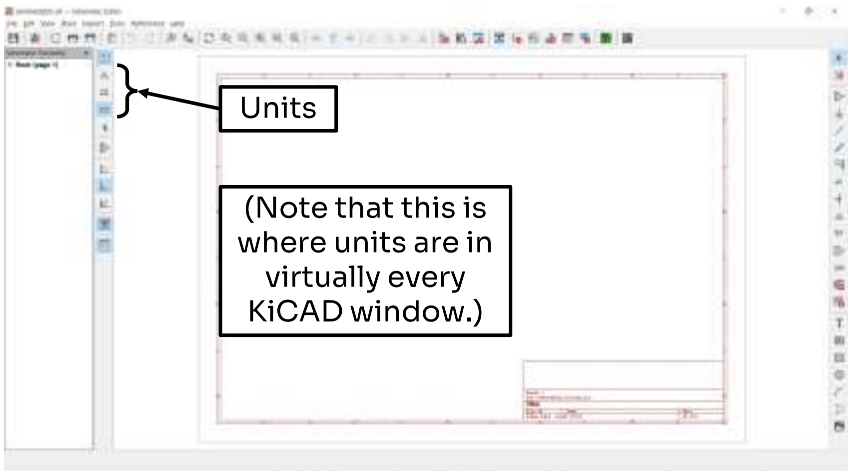
https://forum.kicad.info/t/hide-title-bar-from-existing-drawings/46213

Schematic View

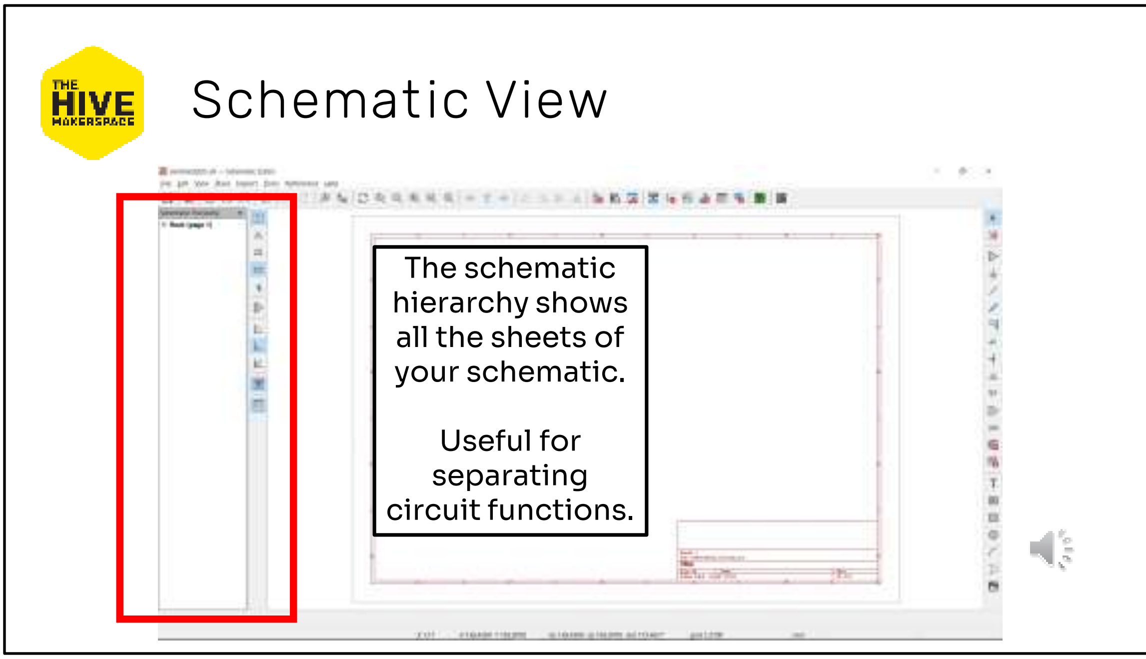


The mouse is the crosshair icon.

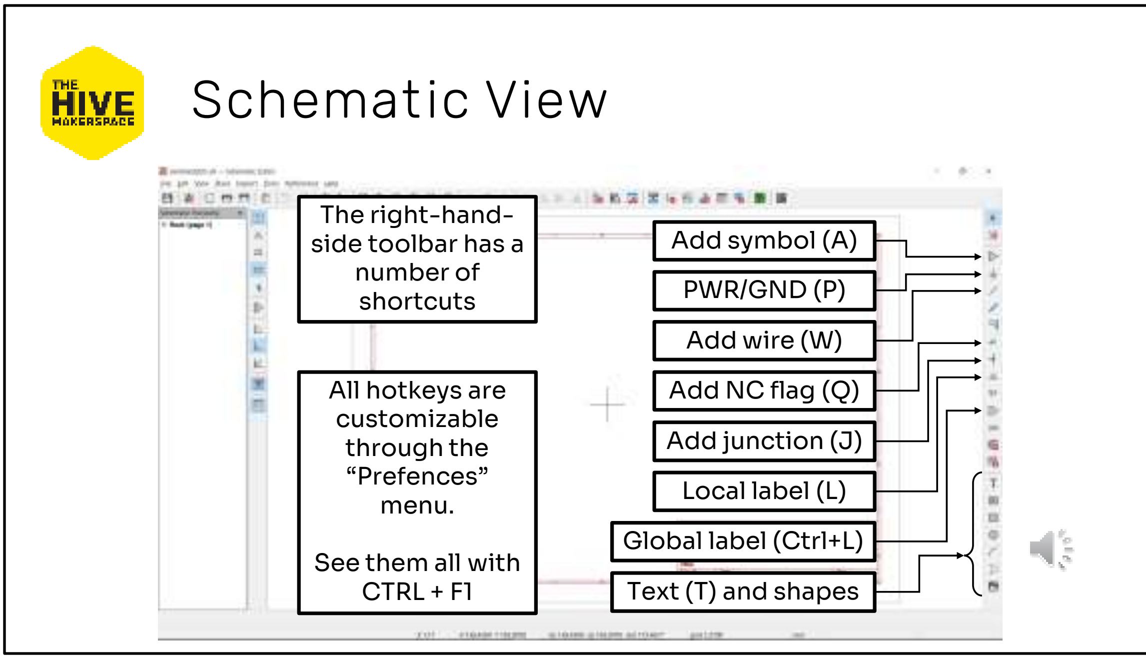
Schematic View



The units and grid settings are located in the upper-left. This is pretty much where these settings will be in every window in KiCAD.

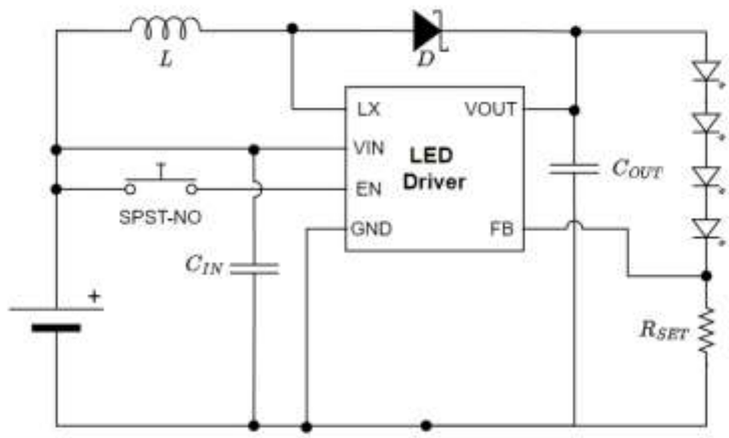


The left-hand side of the editor is the schematic hierarchy. For larger or more complex designs, it can be useful to separate the schematic into multiple sheets. Each sheet would then show up here.



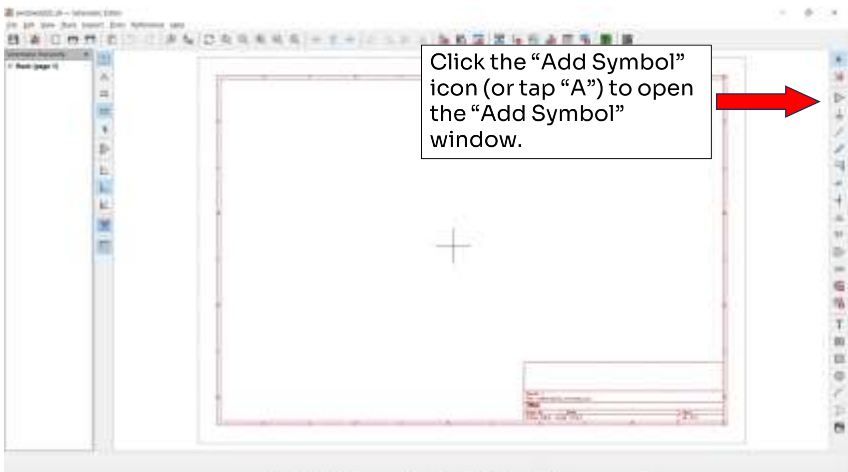
The right-hand side of the window has a toolbar that I'll call the action toolbar, with shortcuts to many different useful actions. These include *adding symbols, *adding power symbols, *adding wires, *no connection flags, *junctions, *lables, *global labels, *text, and shapes. All these actions have hotkeys assigned to them, which can be adjusted in the preferences menu. Hit CTRL + F1 to get a list of all the currently available shortcuts.

Circuit Reminder



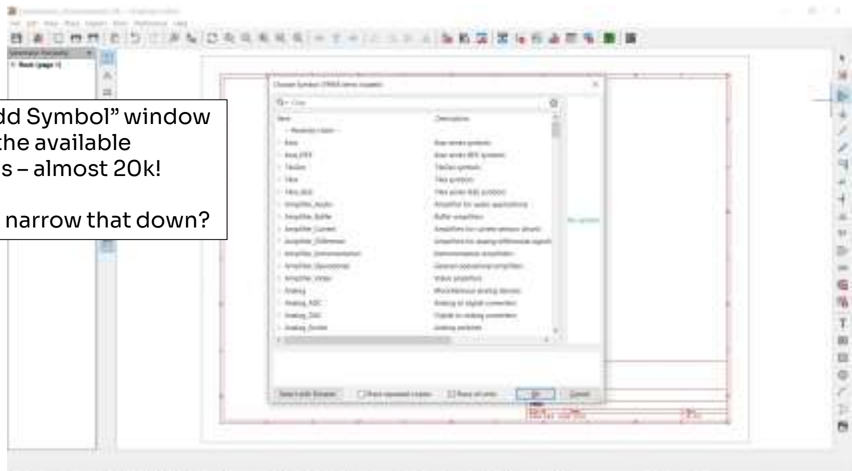
We're going to start by adding all the symbols now. We'll start with the generic symbols, like the resistor, LEDs, battery, and so on, and then do the switch. The IC will come last.

Adding components - R



To add a symbol to the schematic, tap the “A” key, or click the icon identified in the actions toolbar.

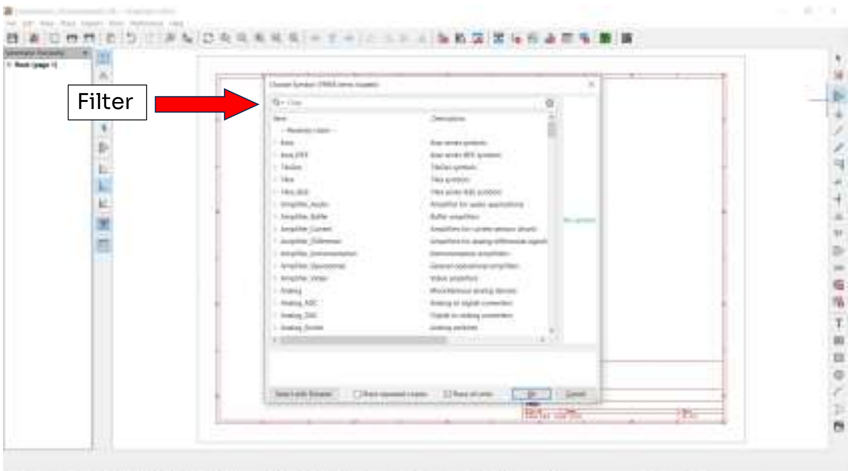
Adding components - R



This will open up the "Add Symbols" window.

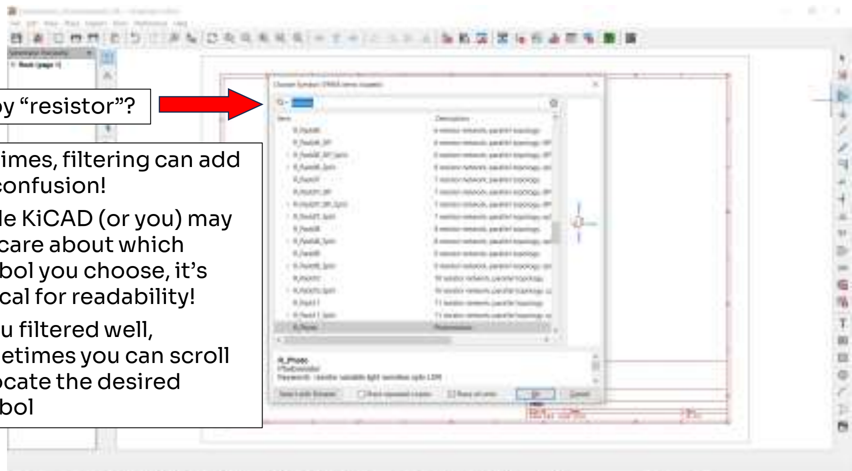
There are nearly 20 thousand symbols built-in to KiCAD!

Adding components - R



We can use the filter at the top of the window to narrow this down…

Adding components - R

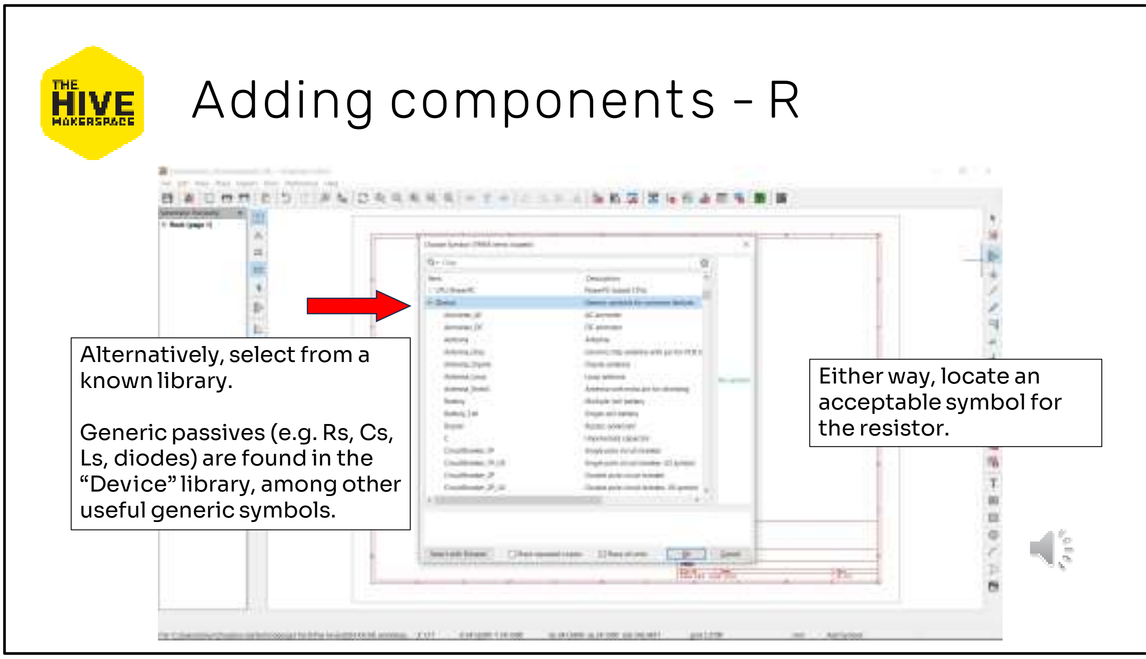


Filter by “resistor”?

Sometimes, filtering can add more confusion!

- While KiCAD (or you) may not care about which symbol you choose, it’s critical for readability!
- If you filtered well, sometimes you can scroll to locate the desired symbol

… although that may not be as helpful as we’d like, and can potentially be even more confusing. Scrolling through the filtered options can help, but only if there aren’t a thousand of them.

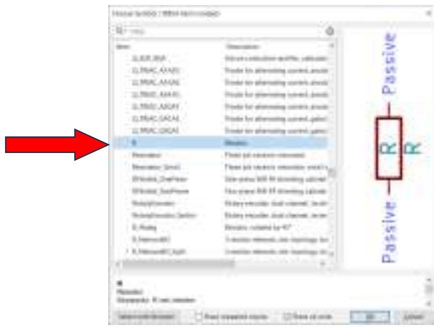


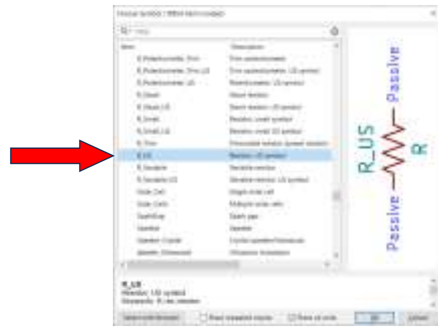
Alternatively, if you know the library, you can just open that directly.

Many, even most (though not all, as we'll see), generic components can be found in the Device library. You'll still have to scroll, but by filtering and locating that library, it can help to narrow it down when you don't know exactly what you're looking for.

Go ahead and pause the video and locate the resistor symbol.

Adding components - R





If you locate the symbol "R", you've found the IEC (internation) version

If you locate the symbol "R_US", you've found the ANSI (US) version

Either is fine!

Click "OK" to select it.

There are two resistor symbols you may have found.

If you just found the one named "R", you might have noticed that it didn't look right. That's because the box-type symbol is the international version of the resistor symbol. In the US, this would be seen as an unknown impedance.

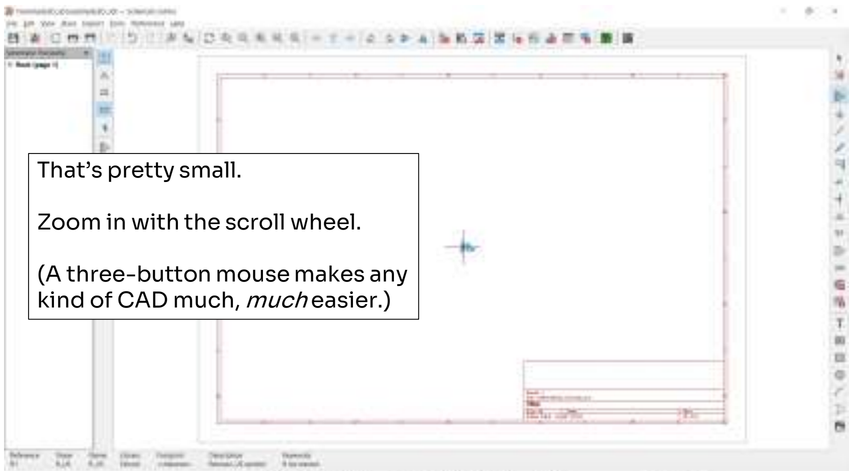
If you looked a bit harder, you might have located the US version of the symbol named "R_US".

Either is totally fine, especially for this tutorial.

Remember that the schematic is for you, your colleagues, your boss, and your future selves to read, so use whichever symbols you'll best be able to communicate your intentions to your boss with.

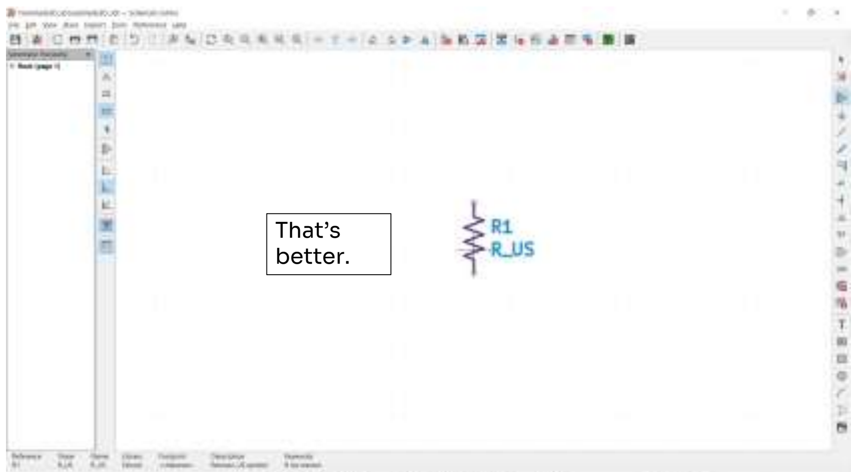
Click OK to select it.

Adding components - R

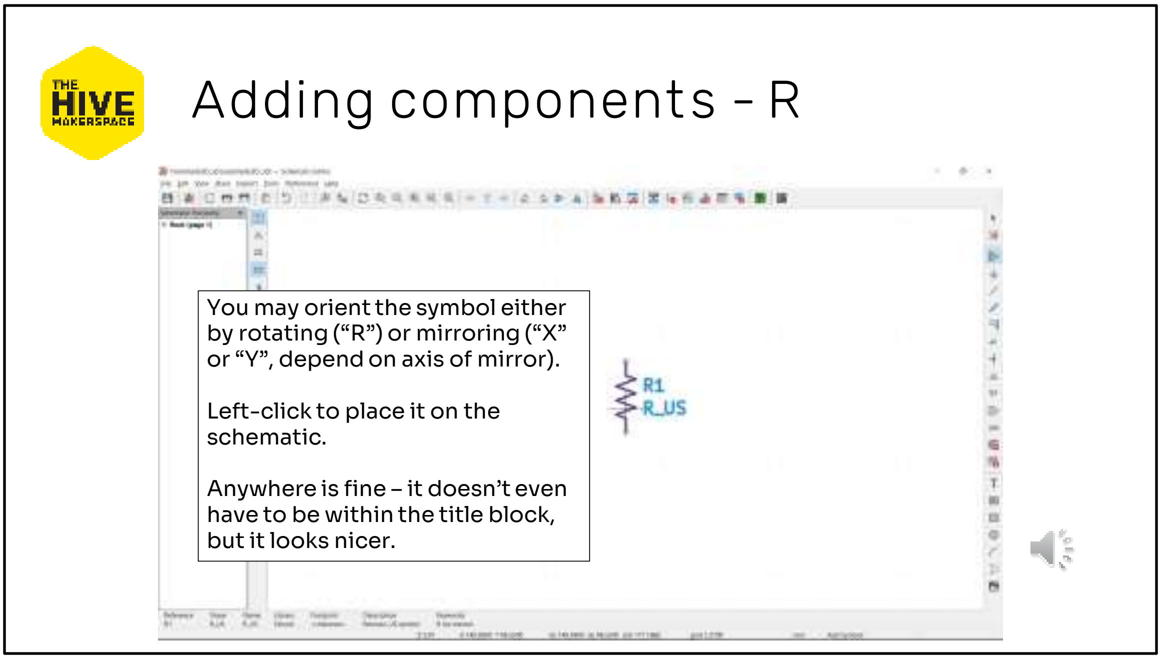


The “Add Symbol” window will return you to whatever zoom depth and viewing location you were at when you opened it. Let’s zoom in here with the scroll-wheel, or whatever trackpad combination you use for zooming. Note that a three-button mouse is really nice to have for all CAD software, KiCAD included.

Adding components - R

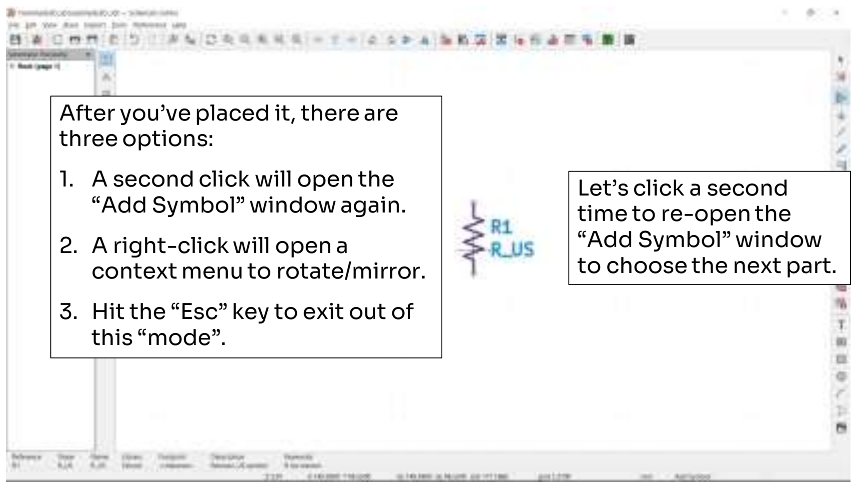


That’s better.



You can orient the symbol prior to placement by rotating with R or mirroring with X or Y. Place it with a left-click. Outside the title block is find if you'd like.

Adding components



There are three options after you've placed it.

*Left-click to open the add symbol window again.
*Right-click to open a context menu for things like rotating, properties, or duplicating.
*Hit "ESC" to exit out of this "add symbol mode".

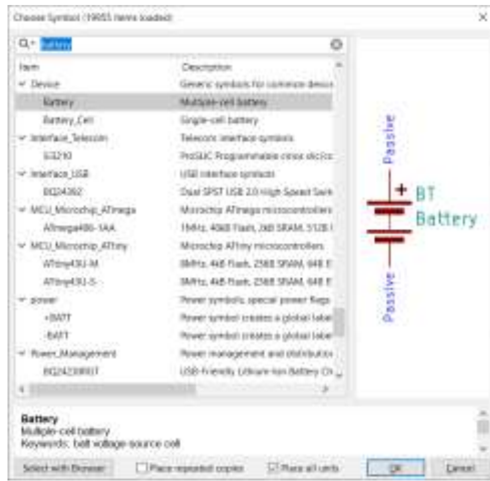
*Let's just click again to add the next symbol, the battery

Adding components - Battery

Filtering by "battery", we can scroll to locate two generic devices in the "Device" library.

Select the single-cell option, and add it to the schematic near the resistor.

(Remember: symbols are just for people to read, so use whichever symbol will make that easiest. If it's important that it's multi-cell, use that.)

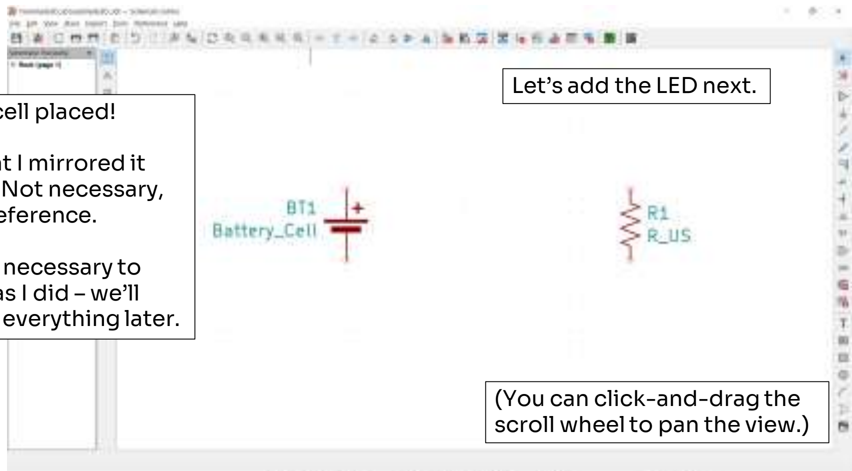


Like the resistor, the battery is a standard symbol, so there's a reasonable expectation that we can find a symbol for it in the Device library,

If we filter by battery, we'll see there are actually two – "battery" for a multi-cell battery, and "battery_cell" for single-cell battery.

Remember, since the schematic is for you and your boss to read (and your future selves), use whichever is most informative. If it's important that it's a multi-cell battery, use that one. Otherwise, use the single-cell symbol.

Adding components - Battery



Battery cell placed!

Note that I mirrored it about Y. Not necessary, just a preference.

Also not necessary to place it as I did – we'll connect everything later.

Let's add the LED next.

(You can click-and-drag the scroll wheel to pan the view.)

Place the battery symbol on the schematic. It's not critical where right now since we'll move most of this around later. I mirrored by symbol because I preferred the text to the left.

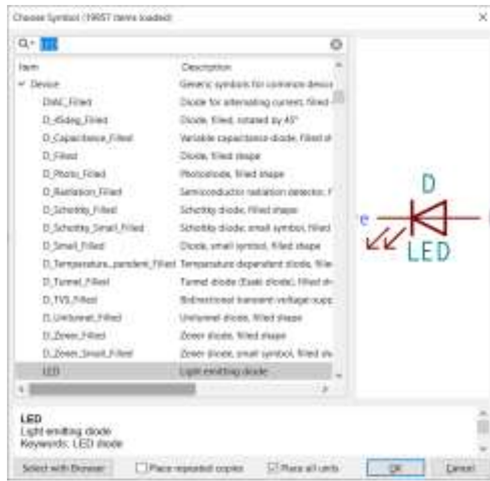
Left-click again to add an LED.

Adding components - LEDs

How convenient that there's a generic LED symbol as well!

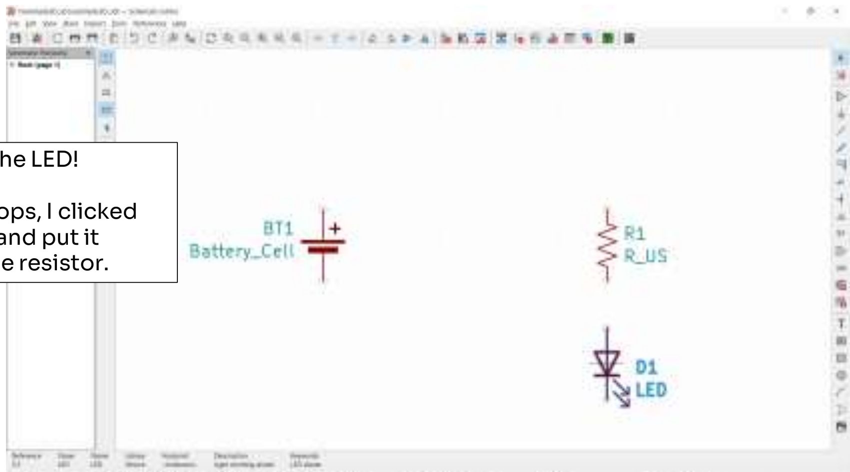
Let's add that to the schematic.

(Most non-IC components have a generic symbol that's acceptable to use, but be aware of the number of pins – that should match your actual device.)



Looking in the Device library again gives us an LED symbol. Add one to the schematic.

Adding components - LEDs

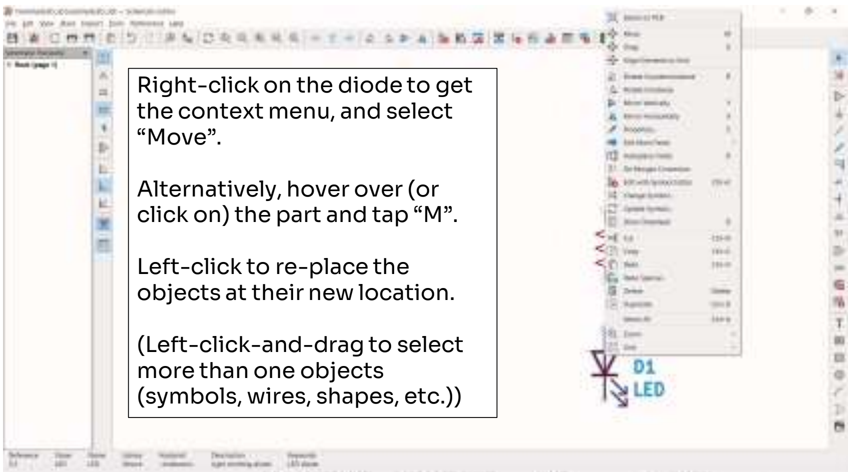


There's the LED!

But whoops, I clicked too fast and put it below the resistor.

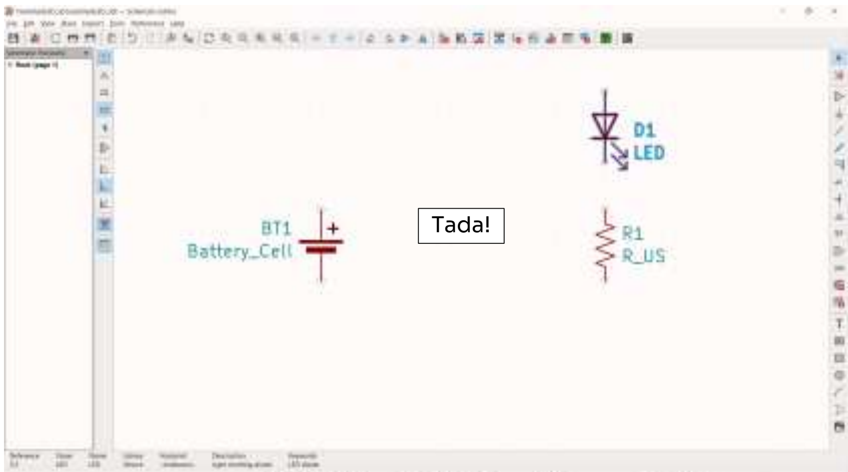
Got too excited and place it below the resistor. Whoops.

Adding components - LEDs



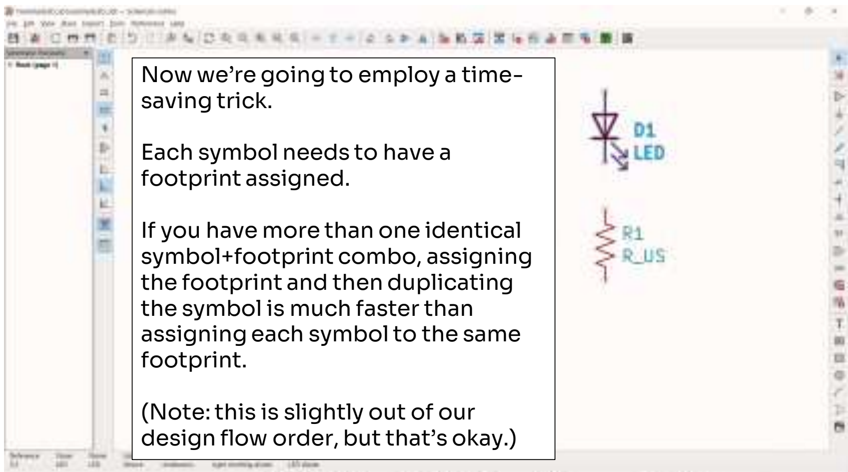
Thankfully it's easy to move. You can left-click-and-drag to move it, click or hover and hit "M", or find move in the right-click context menu. Left-click again to place it at the new location.

Adding components - LEDs



Great.

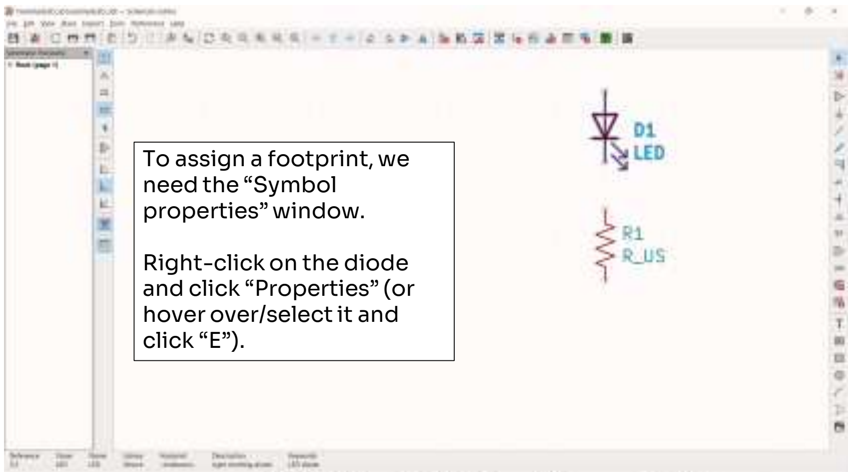
Adding components - LEDs



Now I'm going to use the duplicate feature to make the other LEDs.

But wait! If we add a footprint to the LED first (and a value if we wanted to), we wouldn't have to add footprints (or values) to the other LED symbols later. Adding a footprint can be a touch of a lengthier process (takes like five clicks and two windows), so this can be quite time-saving if you have many duplicate components.

Adding components - LEDs



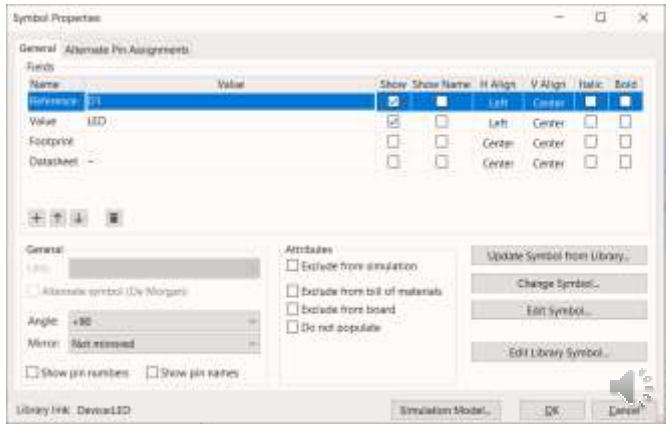
To assing a footprint (and a value), we’ll open the symbol’s properties window.

Select/hover over the symbol and press “E”, or select “properties” from the context menu (right-click).

Adding components - LEDs

There's a lot going on in this window.

Most of this we don't care about right now.



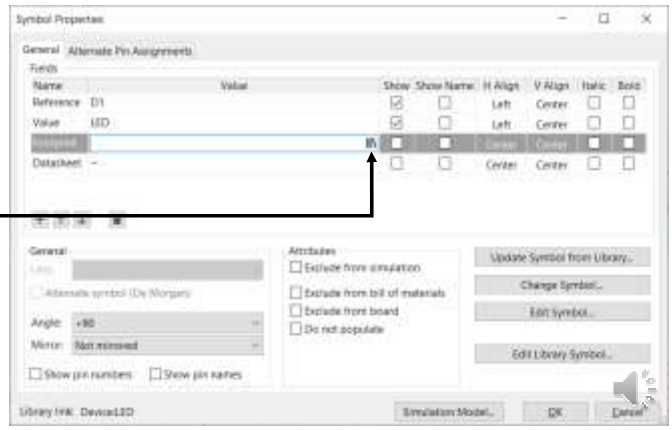
This window has a lot of options.

You can adjust the symbol's "value" here by adjusting the "value" field's value (this is confusing, but it's where it currently says "LED"). For an LED, you might put "green" or "red" in here; for a resistor, maybe 1k; for a capacitor, maybe 1u. Units are never included in the symbol value; this is a historical artifact, I think, coupled with the fact that you can't add symbols like omega into this field.

Most of this window isn't relevant right now.

Adding components - LEDs

Clicking into the “value” box for the “Footprint” field, a little icon pops up on the right (sort of looks like three books). Click it to open the footprint browser.



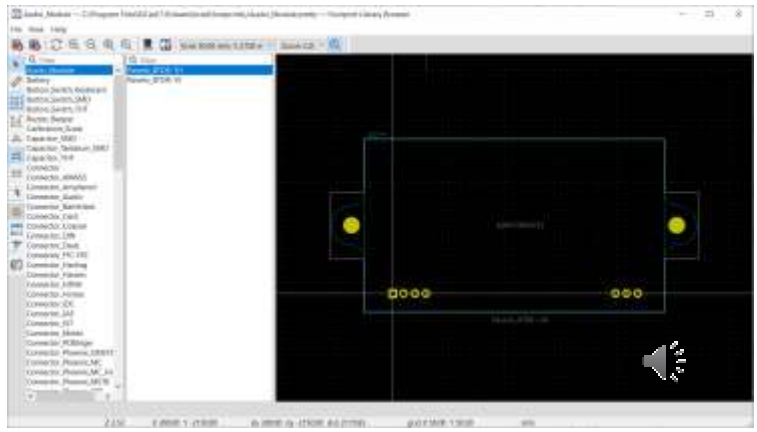
To assign or change a footprint, click the footprint box and then the little three book icon on the right.

Adding components - LEDs

This window shows all the footprints in all the libraries that are available to this project.

(Recall: footprints are the physical layout of the device.)

Filter by LED to see what there is.



This will open up the Footprint browser, which shows all the available footprints.

Filter on the top left by “LED” to see what’s available.

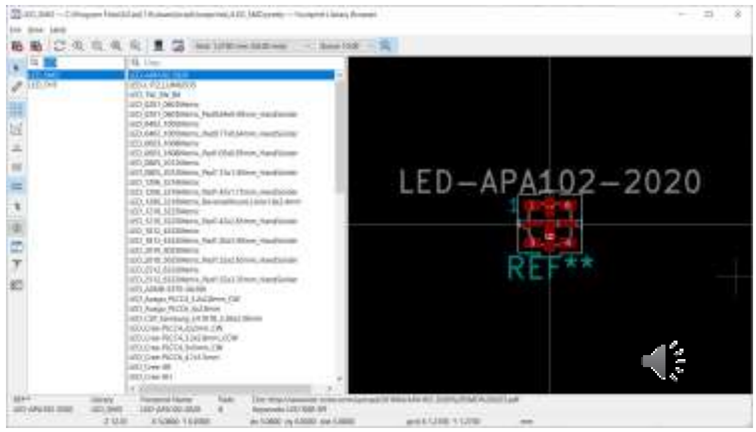
Adding components - LEDs

Wow, a lot of LEDs.

Picking the correct footprint is critical! Otherwise the part may not fit.

(Note that there may be multiple footprints that can work for a part.)

We need the 5mm through-hole LED… let's find it in the "LED_THT" library.



A lot of footprints here still.

It's critical to make sure the footprint selected will actually fit the part you're using, though there may be multiple options that will work.

Since we're using a through-hole LED, let's select the "LED_THT" library.

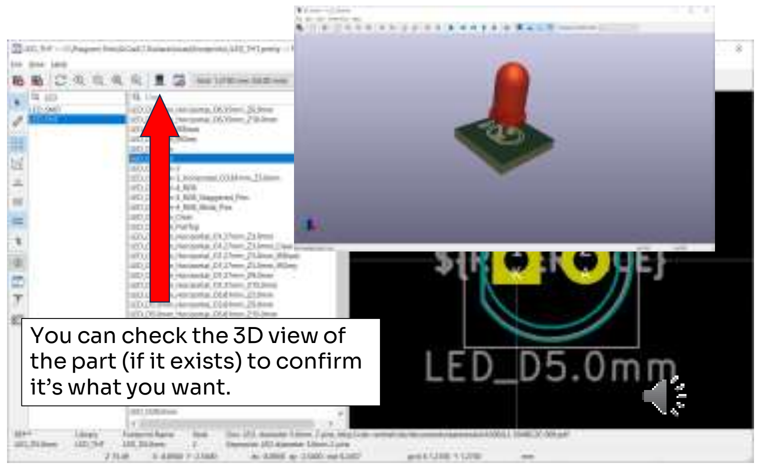
Adding components - LEDs

Lots of options here.

The "D" indicated the diameter, with additional information provided after that.

I don't care about any special or specific additional info, so just the boring 5mm LED it is.

Double click to assign it.



You can check the 3D view of the part (if it exists) to confirm it's what you want.

There are still a lot of options here, but here's the trick: the "D" in the footprint name generally means "diameter". Since we know we're looking for a 5mm diameter 2-lead LED, that narrows down our search.

Some of these have additional information in the name, such as the -3 for three pins, or "clear" for a clear dome. We could also select a horizontally-oriented model here if we knew it needs to be horizontal.

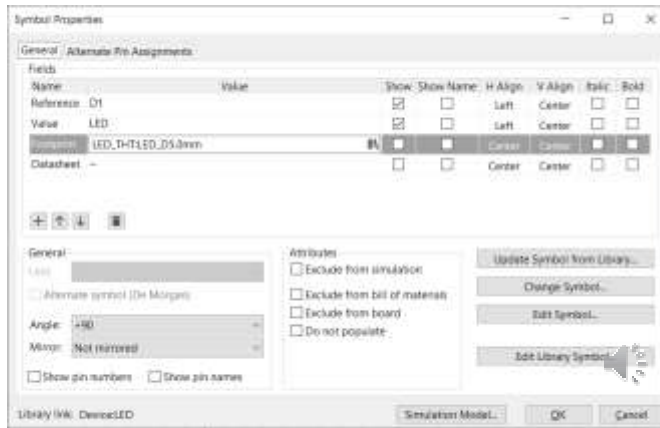
*You can also check the 3D model of the part by clicking this icon here.

Since we're just using a standard 5mm LED, I'll just double-click that one to assign it.

Adding components - LEDs

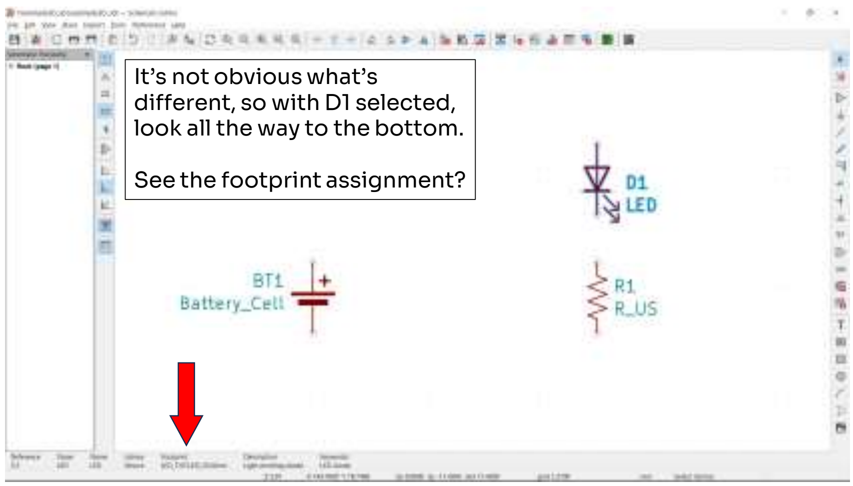
The footprint has been assigned!

Click OK.



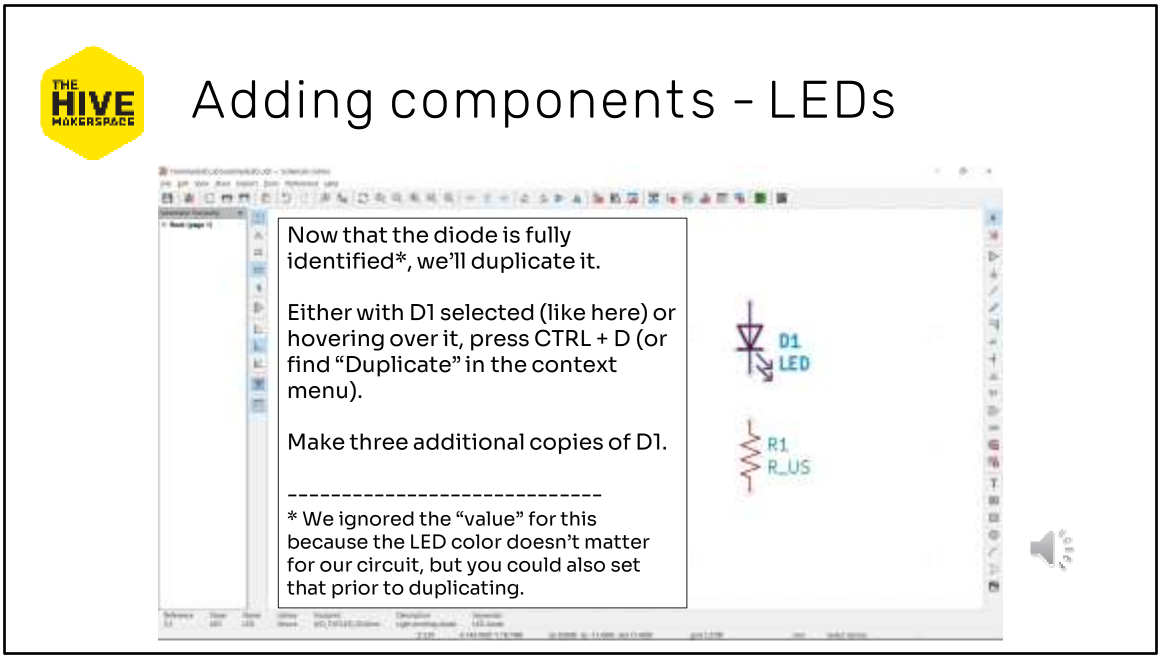
Great, the footprint is in the “footprint” value box. Click OK.

Adding components - LEDs



You can check the bottom to see that the footprint has been assigned. See it?

*It’s not super obvious.

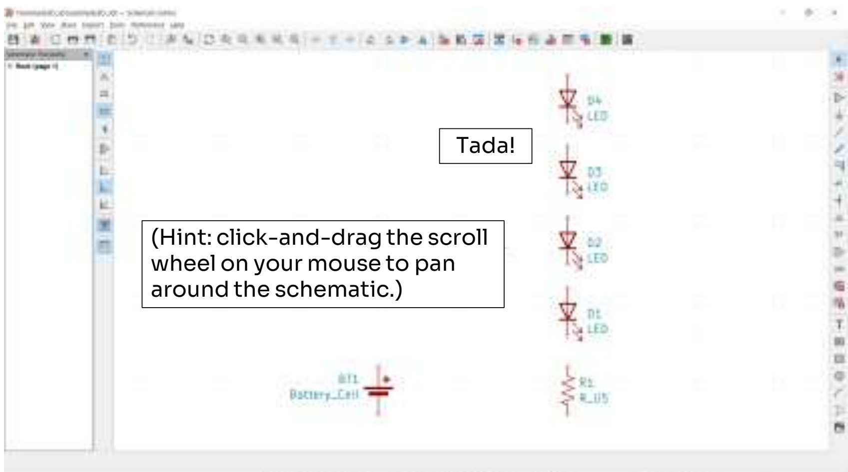


Now that we've assigned a footprint (and a value), we can duplicate it.

Select the part and press CTRL+D to duplicate, or find duplicate in the context menu. Make three copies.

Note that copy and paste will also work, and I'm not sure what the difference is between those methods.

Adding components - LEDs

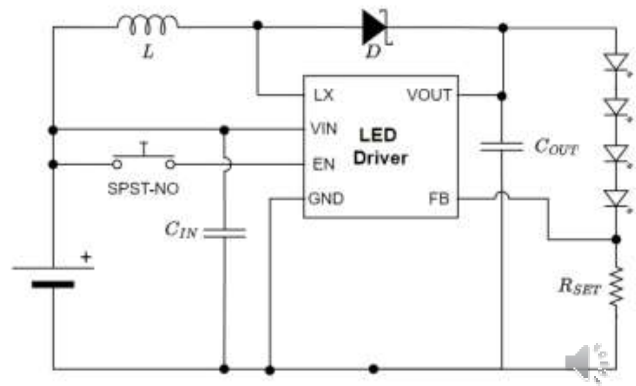


Might look like this.

Note that your schematic will almost certainly look different than mine and that's okay! There are many different ways to put a schematic together. As long as things are connected, which we'll do later, it's minimally functional.

Adding components – L, Cs, D

- Take a few minutes to add these parts.
- The L and Cs are just like the R we did before.
- The diode might require a bit more searching…

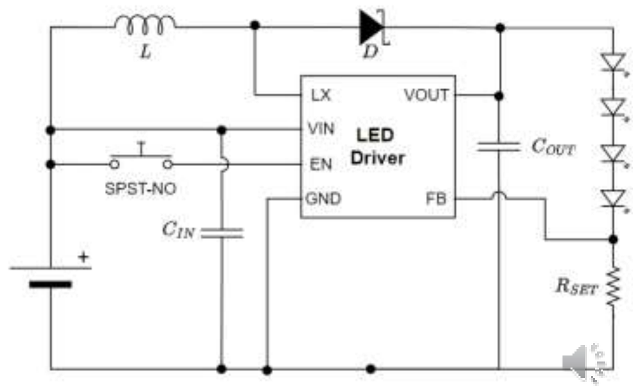


At this point, I'll suggest pausing this video and placing the inductor, capacitors, and the diode (L, C, and D, respectively).

All three can be found in the Device library, but be careful about the diode…

Adding components – L, Cs, D

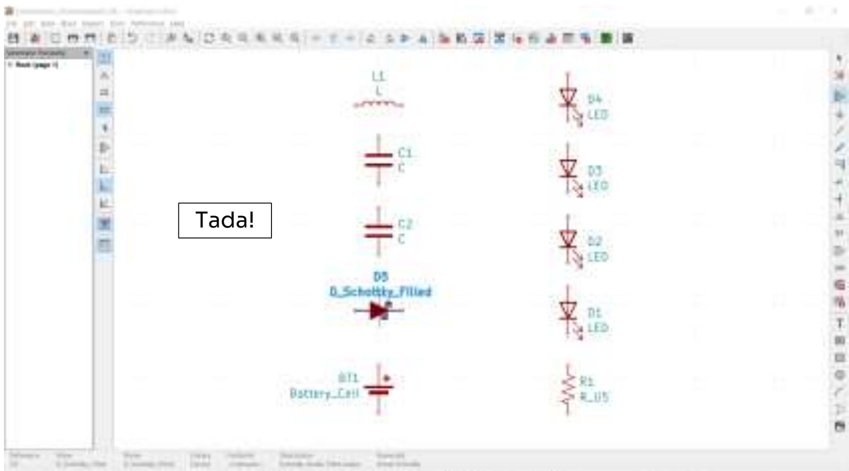
- Hint: the diode is a Schottky diode (with the weird backwards integral symbol), not a regular diode, and yes, it's a standard component.



… because the diode is a Schottky diode, meaning the symbol is slightly different than a regular diode.

Anyway, take a minute or two to find and add these four parts.

Adding components – L, Cs, D

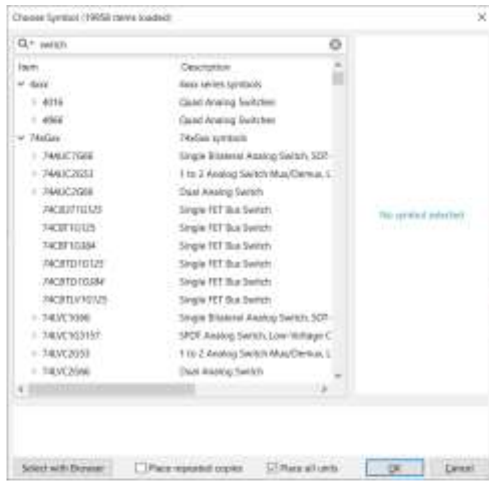


Hopefully you didn't struggle too much with that, but you should now have ten symbols.

Adding components – Switch

The last non-IC part is the switch.

Filtering by “switch” gives us a huge number of options again. Too many.

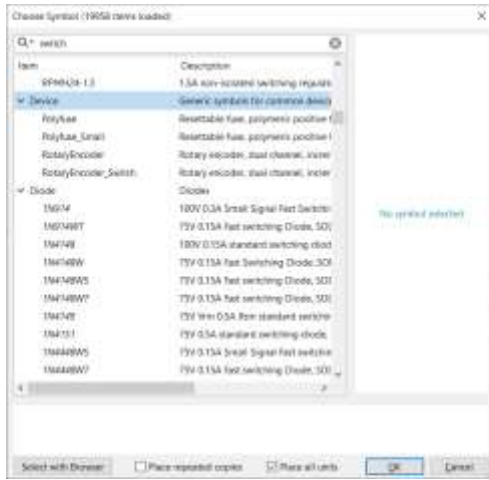


Back into the “Add Symbol” window, we’ll look for the switch next.

Lots of switches available…

Adding components – Switch

The “Device” library doesn’t have any switch symbols! Just a rotary encoder (i.e. a dial).

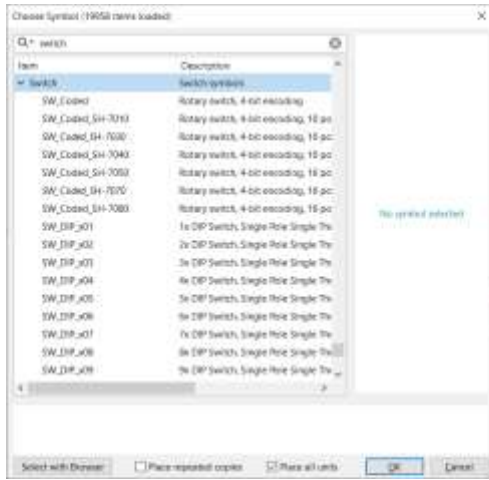


…. And none in the Device library, outside of a single rotary encoder.

Adding components – Switch

For switches, we will actually use the well-named "Switch" library instead.

But which one?



Switches are actually in their own library called, cleverly, Switches.

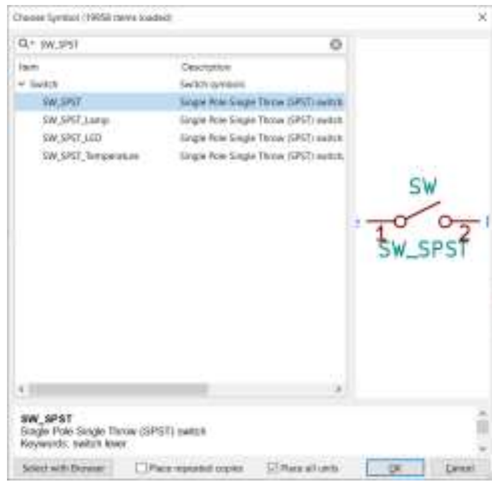
Adding components – Switch

The switch we're interested in is a SPST NO switch, the most basic switch type.

(Recall: this is a single controlled circuit with just a single output that is normally open (i.e. normally disconnected))

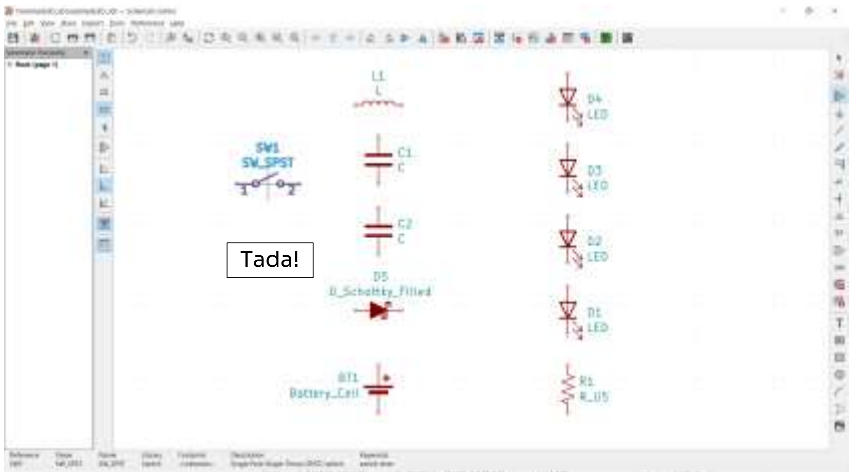
Because we know the specific type, we can actually directly filter by "SW_SPST".

Add it to the schematic.

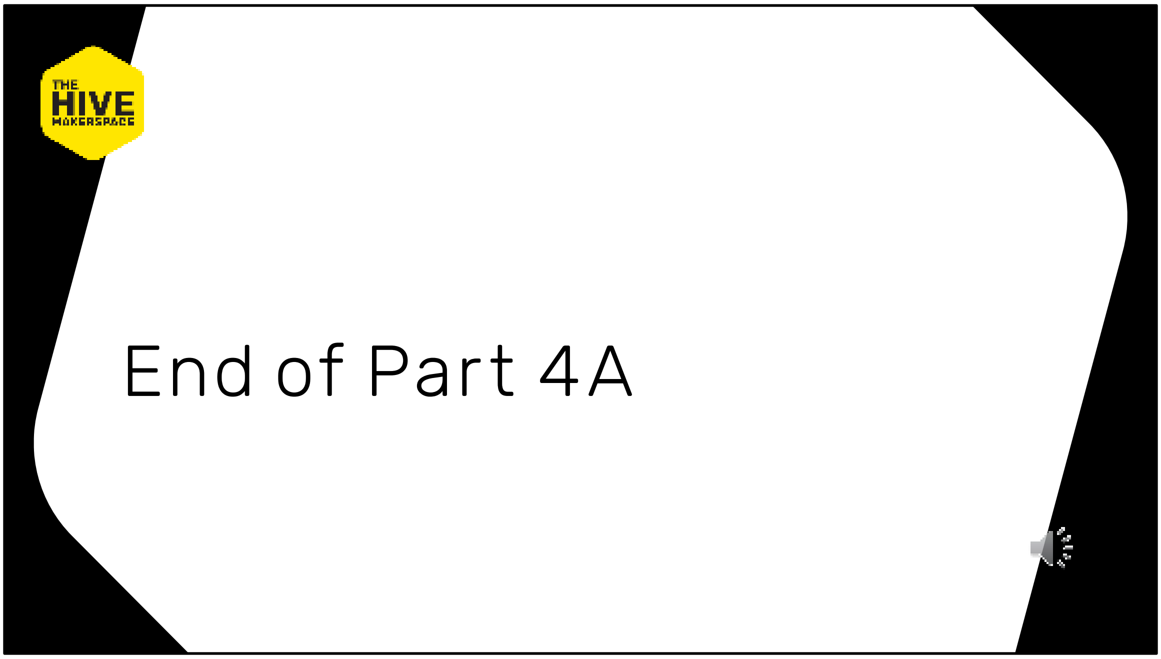


Recall the switch is a single pole single throw, normally-open, switch. Which we can filter for directly to find the right symbol. Add it to the schematic.

Adding components – Switch

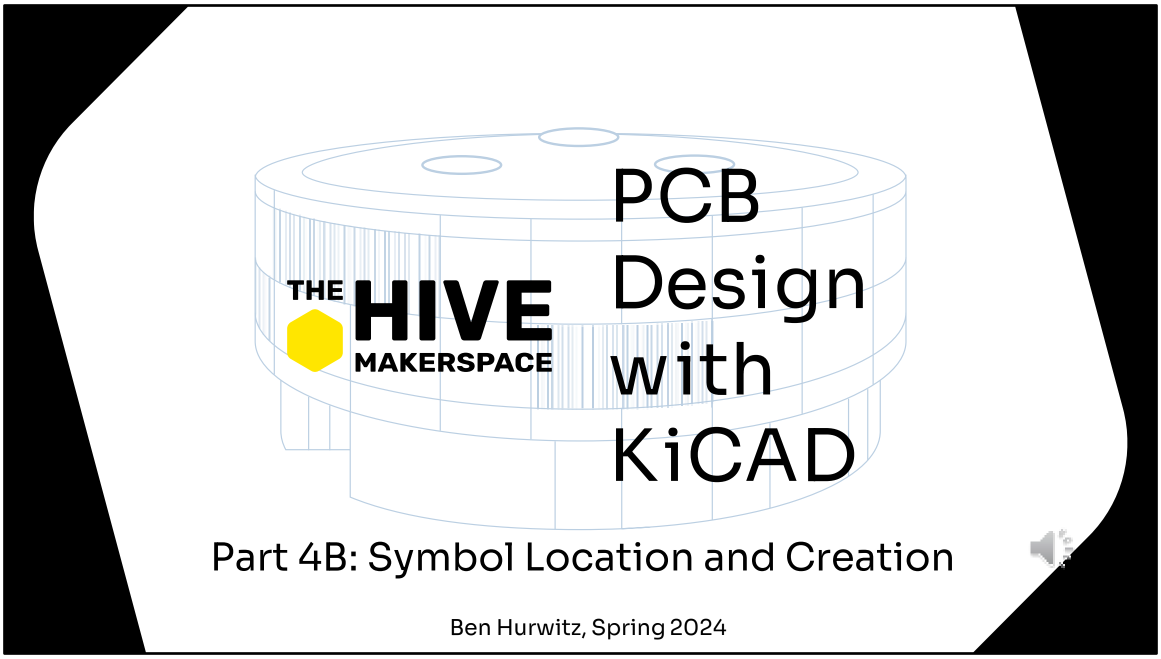


Great.



And with that, we're done with part 4A of this tutorial series. Here we took our first look at the schematic capture videw, and added the basic components to our schematic. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

Please join us for the next video, part 4B, in which we'll cover how to handle the IC, which (spoiler alert) doesn't have a standard symbol. See you there.

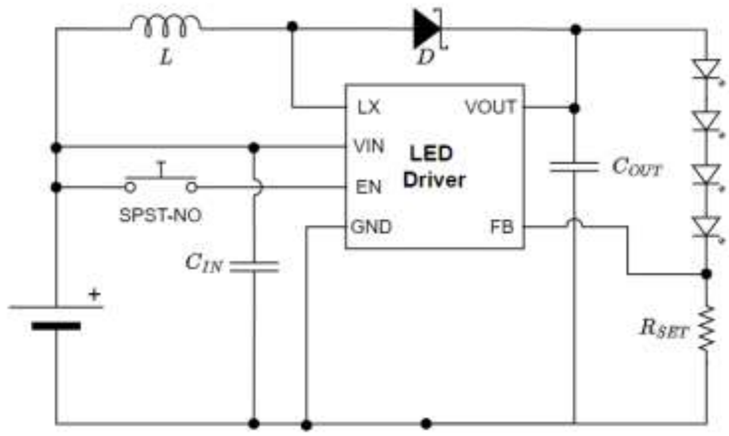


Hi, and welcome to part 4B of The Hive's PCB Design With KiCAD series. My name is Ben, and I'll be walking with you through this process. Part 4 as a whole will cover the entirety of the schematic creation. In this section, we'll cover how to locate device models online, and how to create symbols in KiCAD.

If you watched part 3, you might remember that in our design flow, there was a project setup and library creation step. Due to the relatively simplicity of this design, and that library management was not covered during the original workshop this tutorial is based off of, we're going to leave the library creations to parts 6 and 7. I strongly advise you to watch through those if you're considering doing design more seriously.

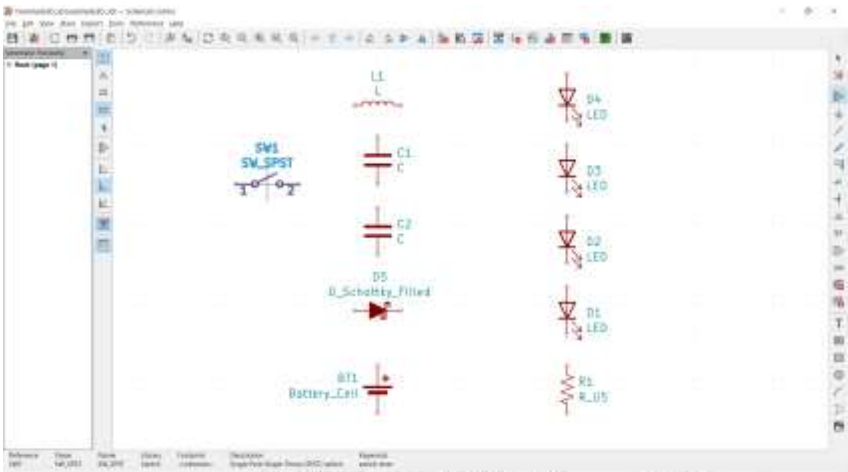
Anyway, let's get into KiCAD.

Circuit Reminder



Before we get into KiCAD, just a reminder of the flashlight circuit we're developing. Note that this image was not take from KiCAD, and therefore the symbols and graphics are different from those you are about to see.

Schematic Reminder



And this is a reminder of the schematic as it stood at the end of part 4A. We had added all the standard components, and were about to add the IC. If you've forgotten anything, I suggest you at least skim through that video (or the associated PDF).

Adding components – IC

- The last component to add is the IC
- The IC is slightly different than the other components we have put down, for two reasons:
  1. The symbol is "unique", or at least, a symbol with the same pinout would be difficult to locate in the built-in libraries
  2. The footprint may or may not be in the libraries (although it is a standard package, so it likely is... but we'll pretend.)

Adding the IC is the last step.

*It's different than the previous symbols for two big reasons.

*First, the symbol itself is unique, or if it's not unique, locating an IC with the same symbol that is already in KiCAD will likely be as much work as making a new symbol.

*Second, the footprint may or may not be in KiCAD either. It's a standard package, so maybe, but we'd have to see.

Adding components – IC

- The first thing to do with ICs or other non-standard components is to see if they're already in the Symbol Libraries.
- Click "A" to add a symbol and filter for the RT4526.
- Did you find it?

It's always a good idea to check if they happen to be in one of the built in libraries though, so *go ahead and open the "Add Symbol" window and filter for the RT4526.

*Is it there?

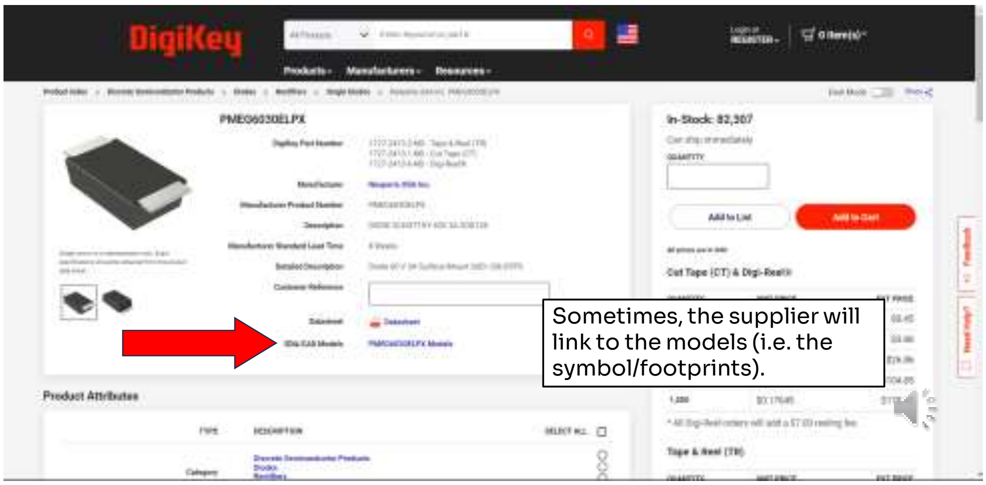
Locating a model for the IC

- You shouldn’t have actually found anything, which is pretty normal for most ICs and non-standard parts.
- That’s okay!
- We’re now going to look to see if someone else has already done the work of generating a symbol and footprint.

Totally okay that it’s not.

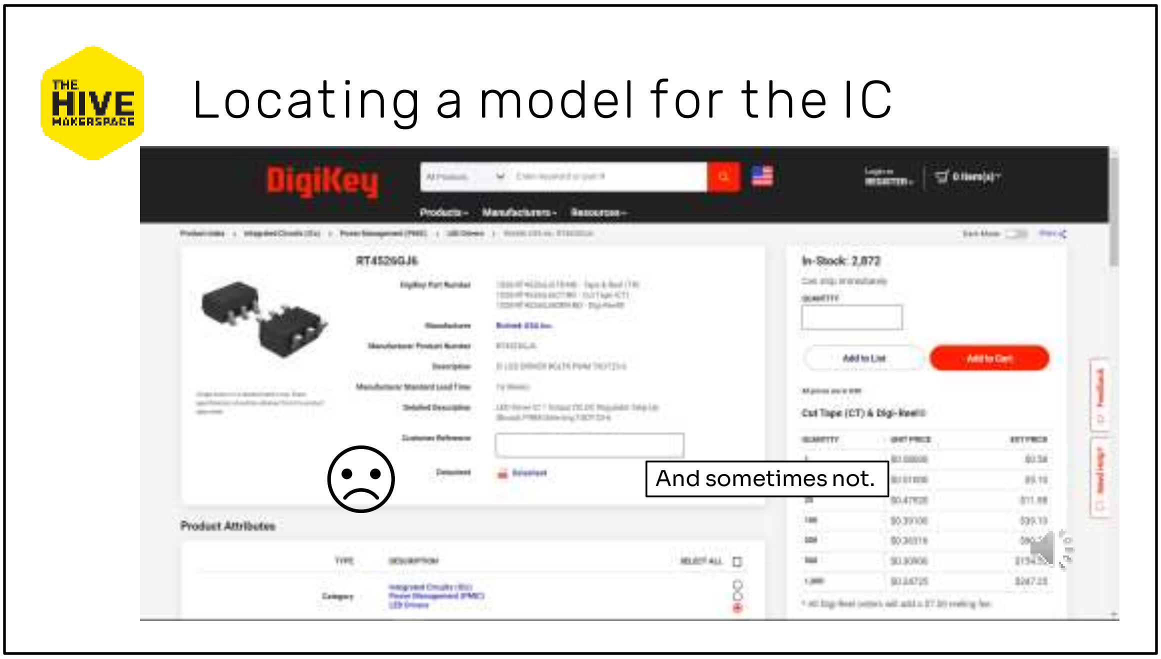
The next check is to see if the models have been professionally generated already. Always try to work smarter, not harder.

Locating a model for the IC

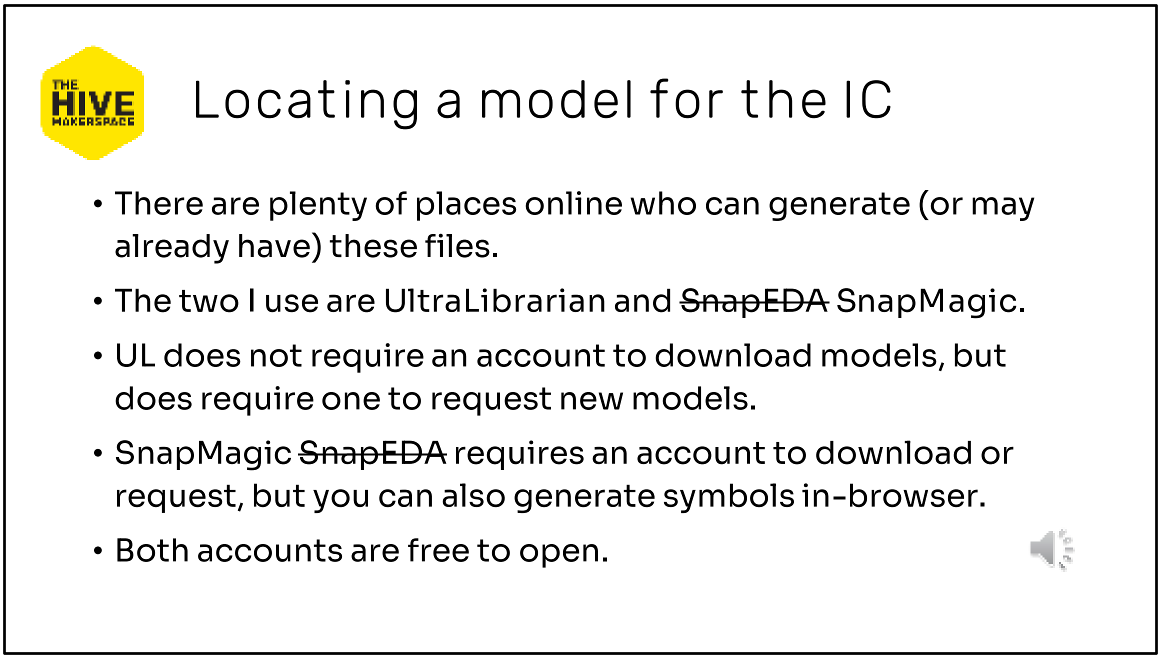


Sometimes, the supplier will link to the models (i.e. the symbol/footprints).

Sometimes, the suppliers will link to models.



Sometimes not.



If the supplier doesn't link, we can go look for them manually.

*The two places I have had good results with are Ultra Librarian and SnapMagic, which used to be called SnapEDA.

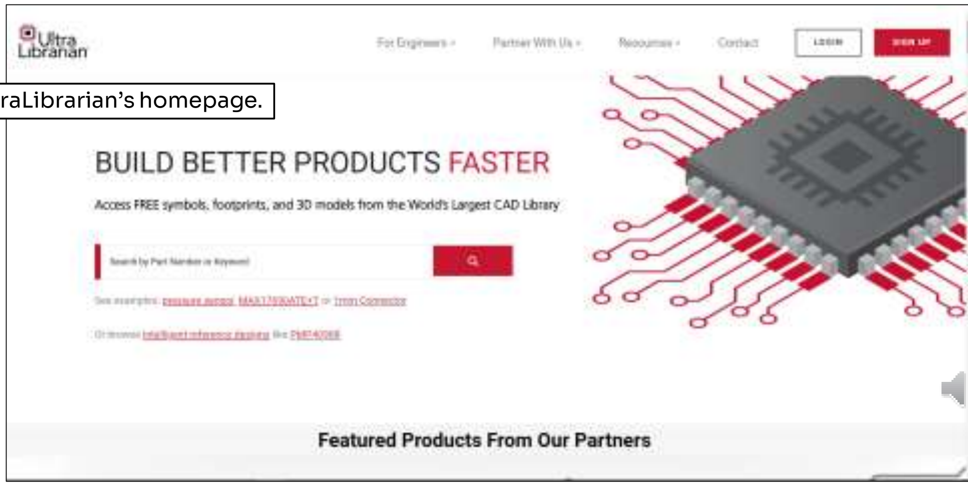
*Both require free accounts to request new models, but Ultra Librarian allows you to download pre-made ones without one.

*SnapMagic has a few additional tools that are available for registered accounts as well, like in-browser symbol and footprint generation.

**

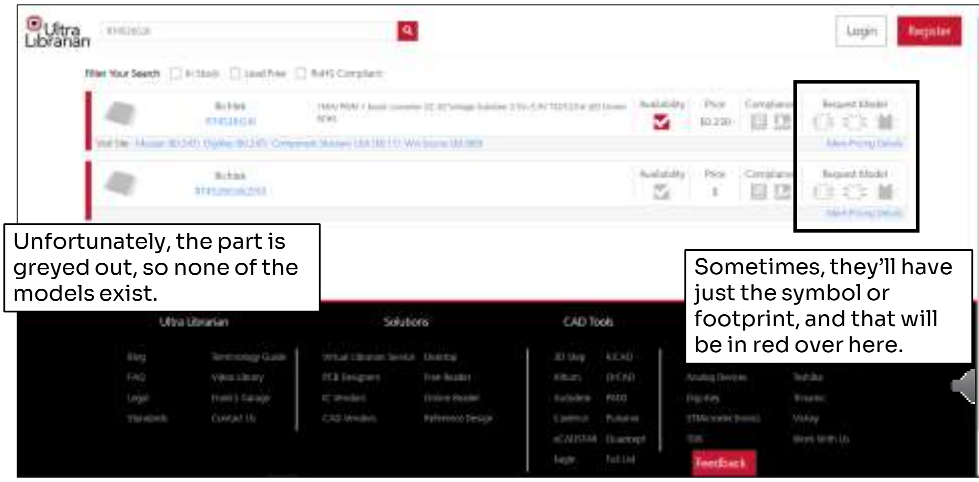
Locating a model for the IC

UltraLibrarian's homepage.



We'll start arbitrarily with Ultra Librarian. Search for the symbol you'd like a model for.

Locating a model for the IC



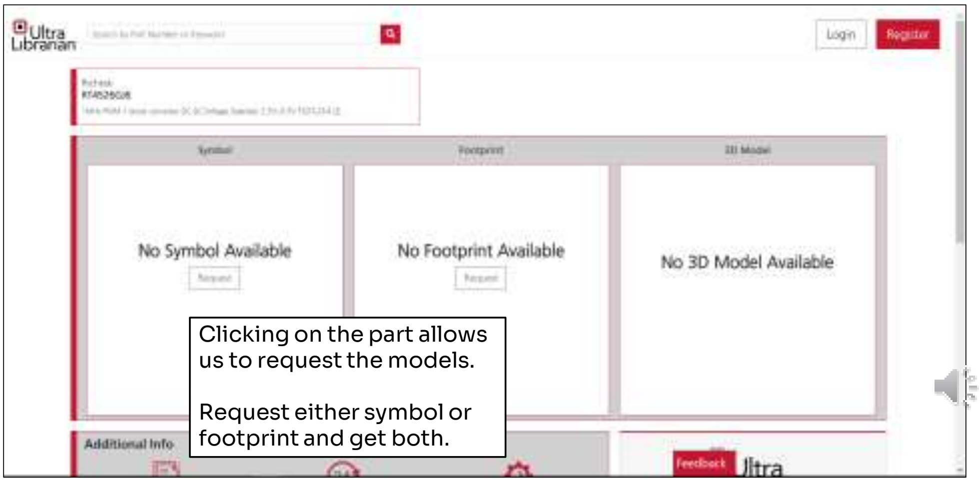
Unfortunately, the part is greyed out, so none of the models exist.

Sometimes, they'll have just the symbol or footprint, and that will be in red over here.

Unfortunately, the part is grayed, so no models exist.

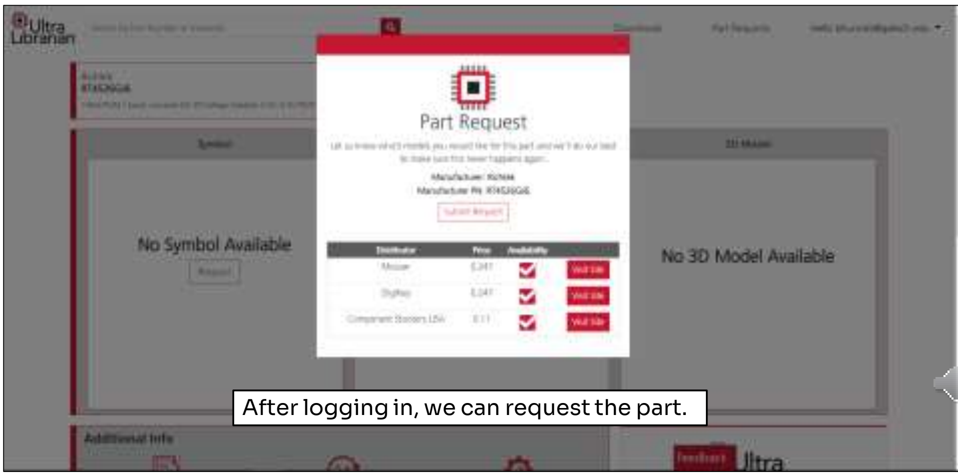
*Sometimes, the icons on the right will be filled in, indicating that they have a footprint or a symbol, but not today.

Locating a model for the IC



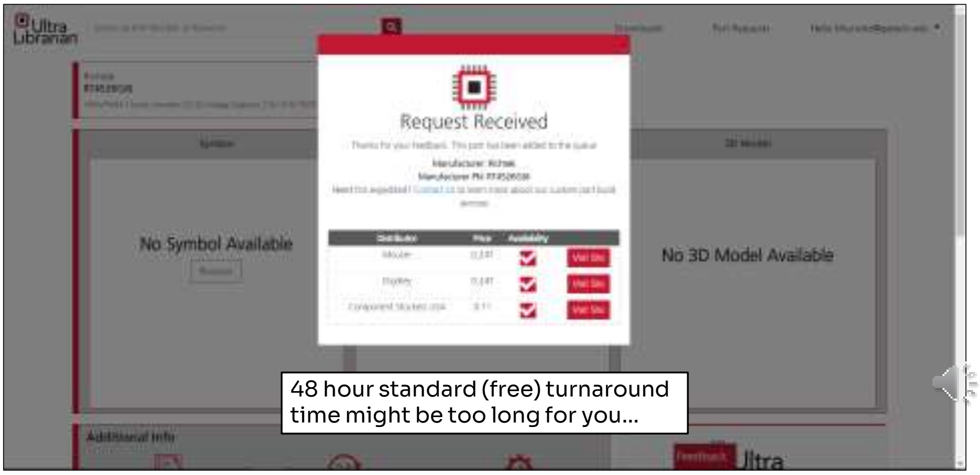
Clicking the part gets us to the part page, where we can ask them to create a model for us. Request either to get both.

Locating a model for the IC



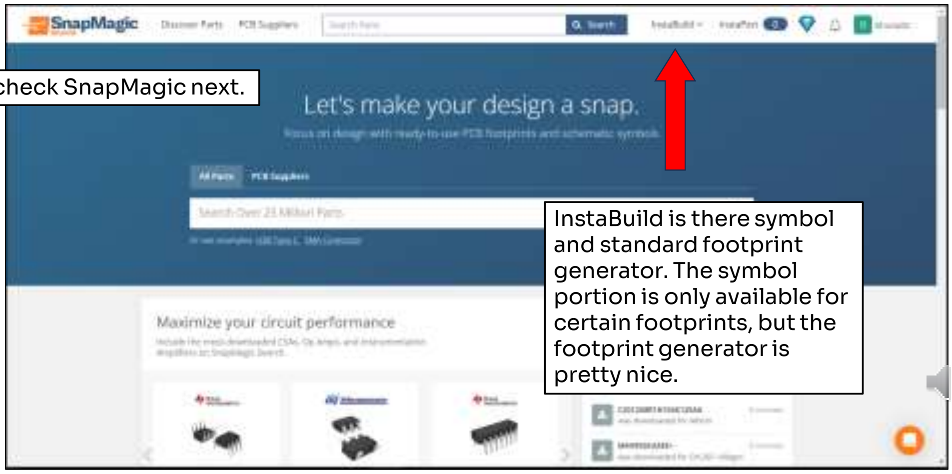
Looks like this

Locating a model for the IC



Standard turnaround guarantee is 48 hours, which may or may not be too long for you.

Locating a model for the IC



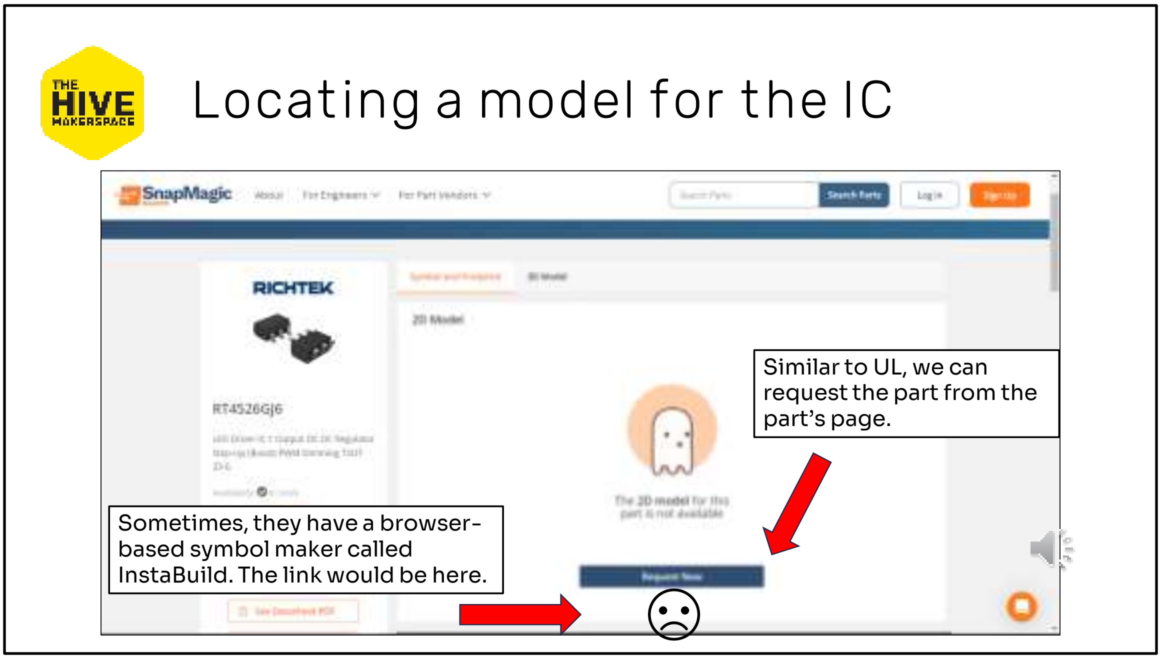
Let’s check our other source, SnapMagic.

*SnapMagic’s in-browser model generator is linked on their homepage, but while most standard IC footprints can be generated, the symbols are limited to select footprints only.

Locating a model for the IC

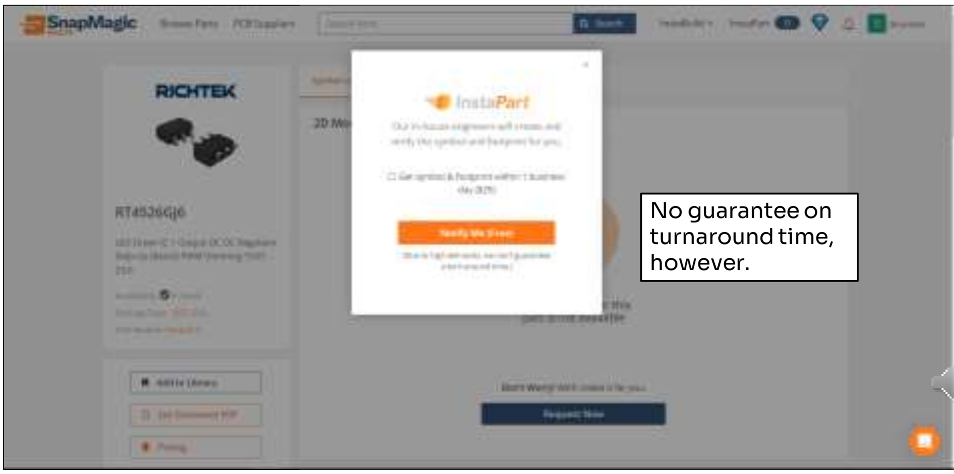
Searching for our part brings us to this page, *but be careful! Sometimes they recommend a part at the top that isn't right.

*Still, no models available – we can see this with the empty icons on the right.



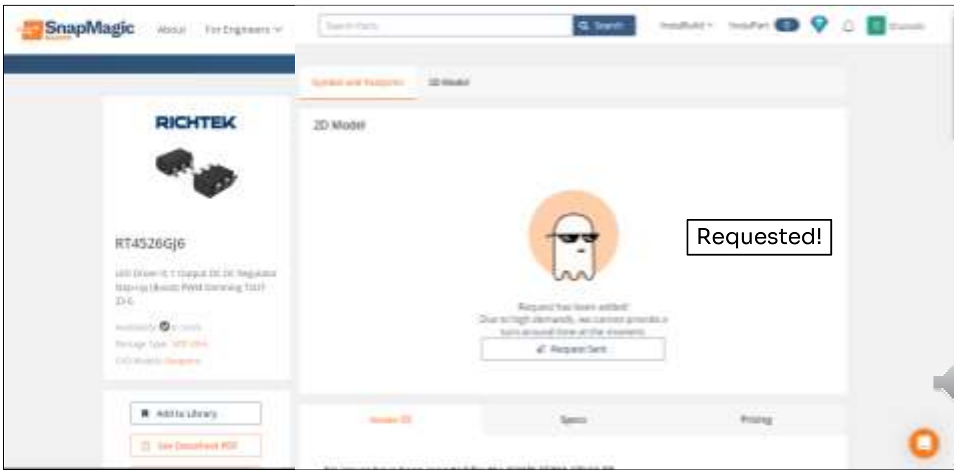
We can request the models from the parts page. *There would also be a link here if the in-brower symbol generator was available for this part, but it's not. Sad.

Locating a model for the IC



Unfortunately, SnapMagic doesn't have a guarantee on turnaround time.

Locating a model for the IC



But you get a cool ghost when you request the part.

~~Locating~~ Creating a model for the IC

- Struck out thrice. What to do?
- Can keep searching (someone *must* have made this before, right?), but returns may be diminishing.
- Because KiCAD stores symbols and footprints in separate libraries, we'll just go ahead and create our own symbol directly in KiCAD.
  - No need to create our own footprint (yet)

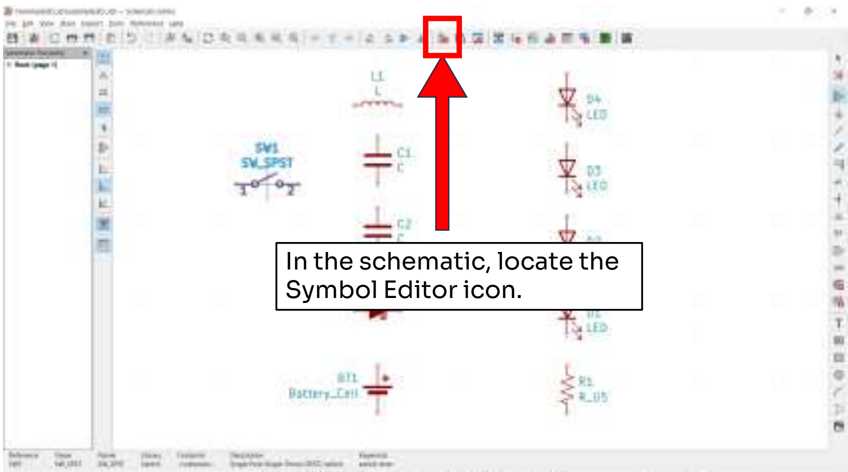
So three strikes. Are we out?

*We can keep searching randomly on the internet, but the returns will likely be diminishing, and the quality might be degraded.

*Instead, we can just go ahead and make the symbol in KiCAD directly.

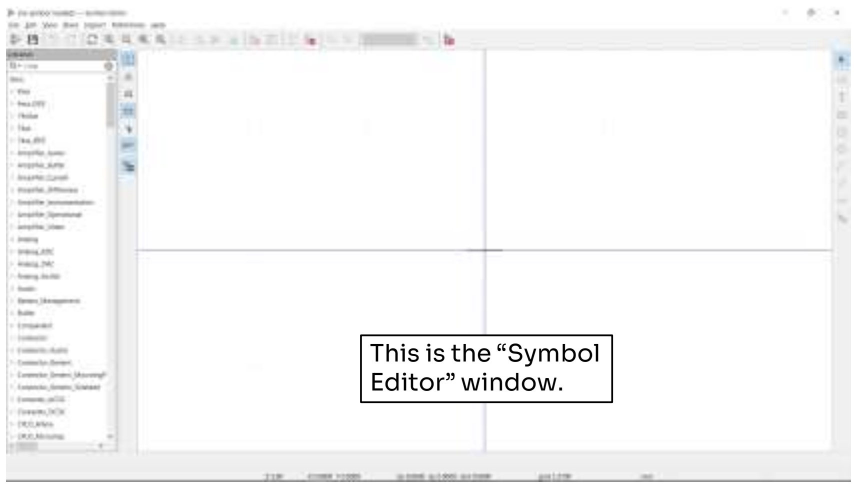
Symbols are relatively easy to make, and are less prone to errors than a custom footprint, though they can be tedious with many pins, so it's totally doable to make them without waiting for Ultra Librarian or Snap Magic to be done.

Creating a symbol for the IC

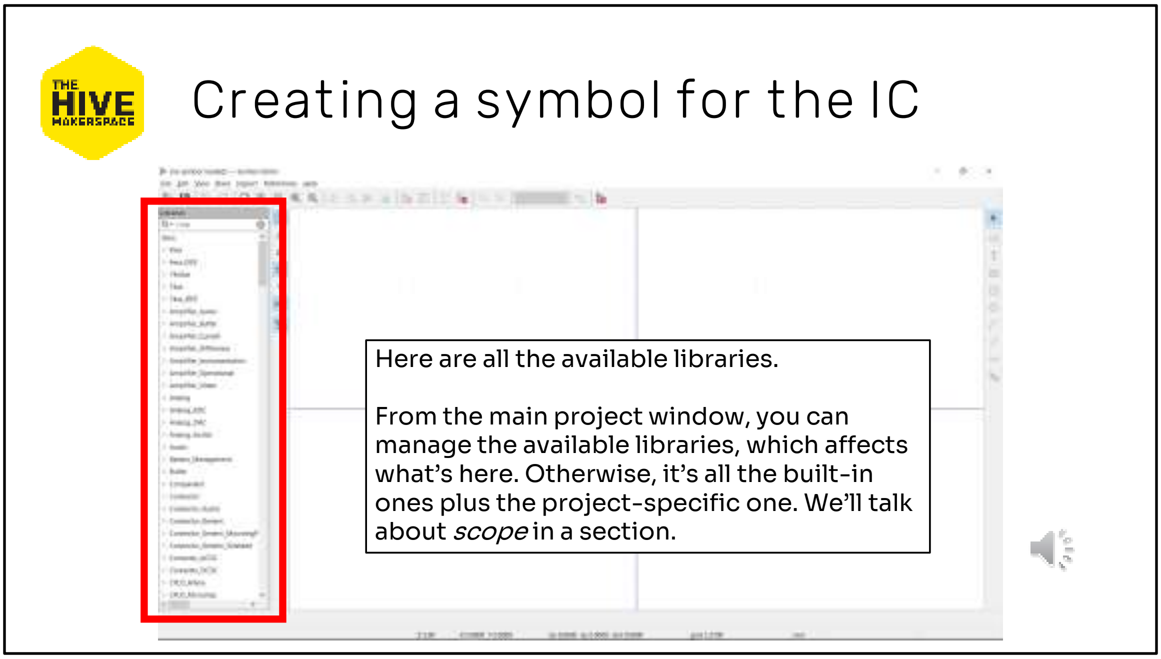


To make a symbol, we need to go into the Symbol Editor, which can be access from the project window or directly from the schematic using this icon.

Creating a symbol for the IC



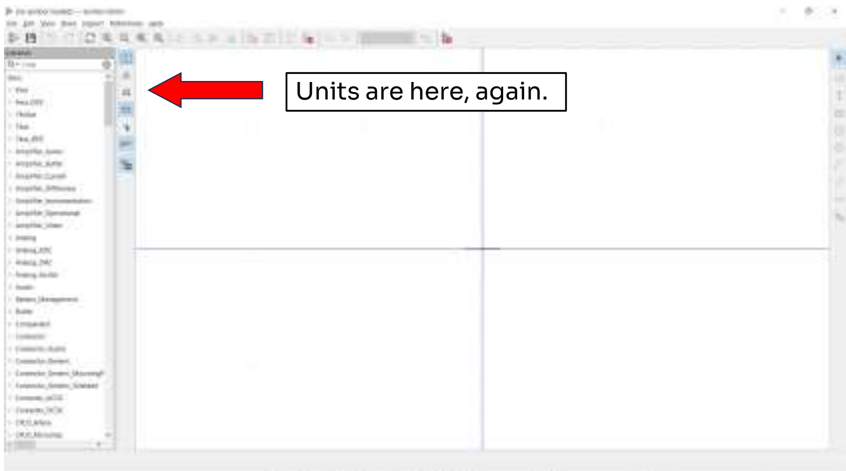
This is the blank symbol editor window.



On the left are all the available libraries. You can manage which are available through the project window under preferences.

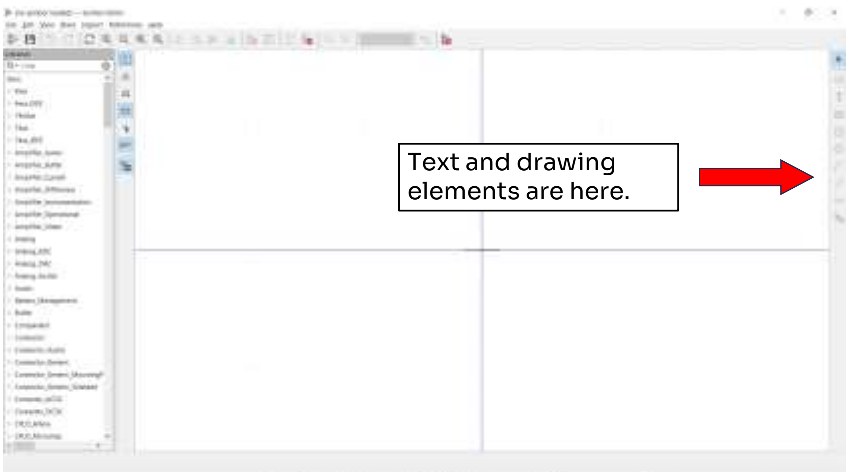
Note that all the built in libraries are read-only, so we'll need to make a new library for our new footprints.

Creating a symbol for the IC



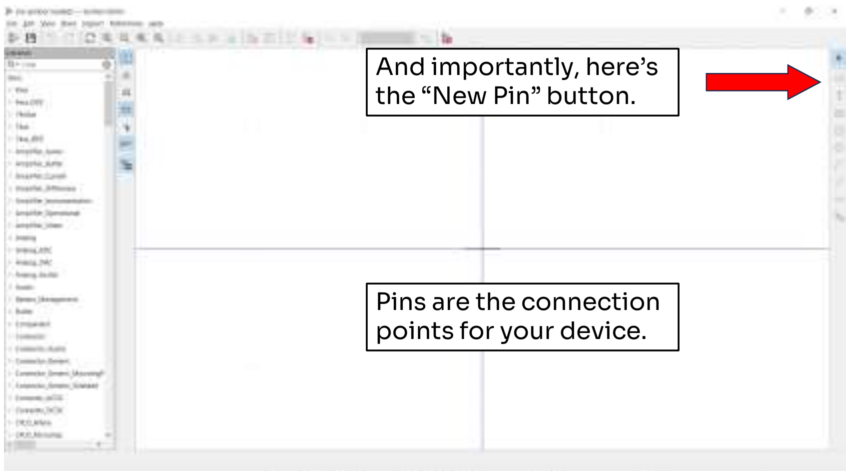
Units in the top left again.

Creating a symbol for the IC

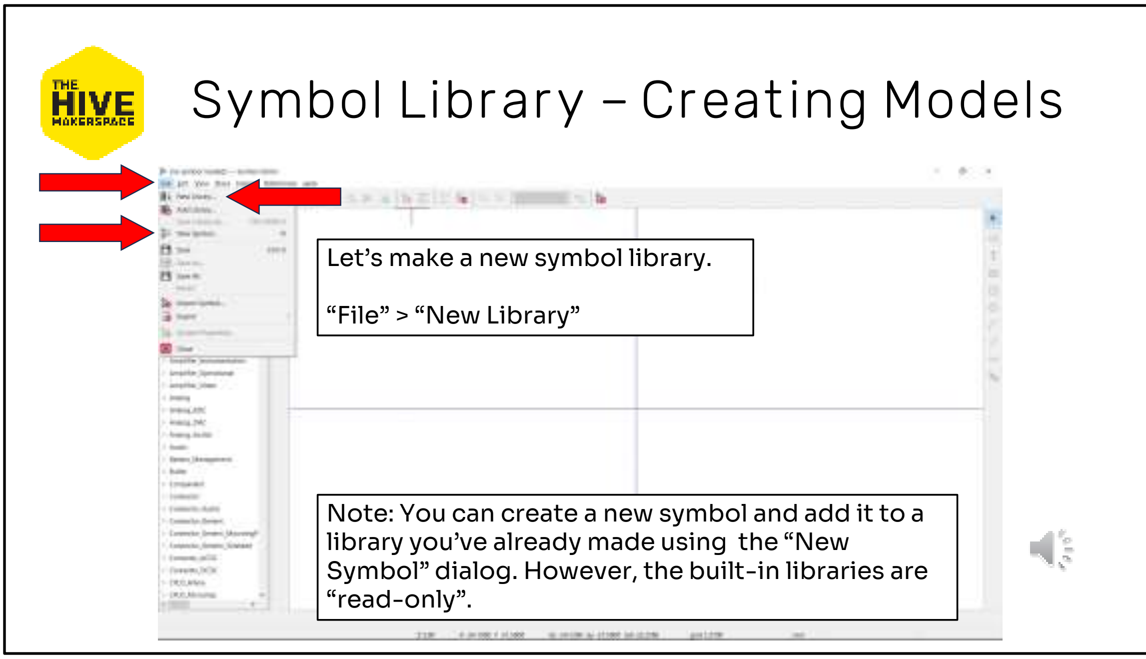


Text and drawing actions on the right.

Creating a symbol for the IC

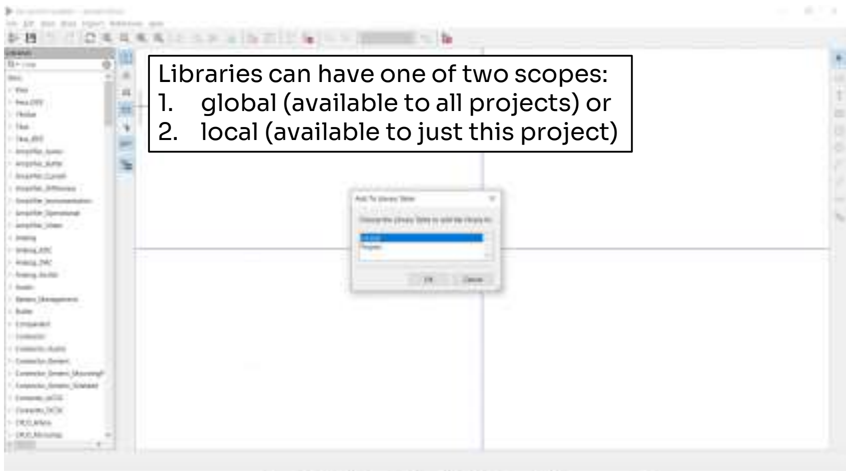


Top right has the new pin button, which will be the most used button here, likely, since pins are the connection points.



As mentioned, we need to make a new library to add our symbol into, since all the built-in libraries are read-only, and we don’t have any writeable libraries of our own.

Creating a symbol for the IC



New libraries can be set to have one of two scopes: they can be set to available for all projects, called global scope, or only be available for this project, called project or local. Project libraries can actually be made available in other projects by adding them in manually though the main project window, under preferences.

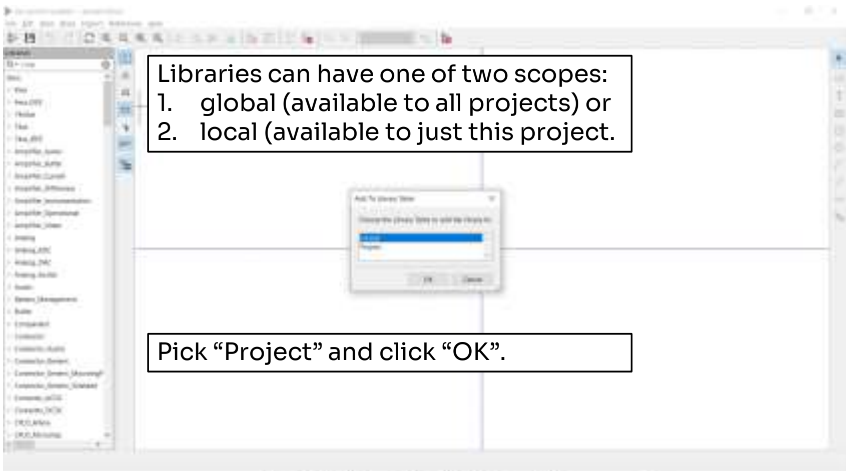
Aside: Library Management

- Library management is a whole topic, especially with larger teams with different backgrounds integrating automated backups or version control
- For KiCAD, it is strongly preferable to use local (i.e. project-specific) libraries for all projects.
  - Library objects can easily be copied into new libraries, making global libraries mostly a liability (e.g. external updates breaking your design)

Library management is an important but under-appreciated aspect of PCB design work. Or, at least, under-appreciated by beginning designers. There's a world of discussion out there about the best methods, especially when considering backups, version control, and cloud connectivity, but for KiCAD, it's nearly always better to use project-scope libraries rather than global ones. Why? Global libraries are not under your control, can change or fluctuate between revisions, and make transferring an entire project more difficult. Thus, it's strongly preferred to create a single project library and copy global parts into it.

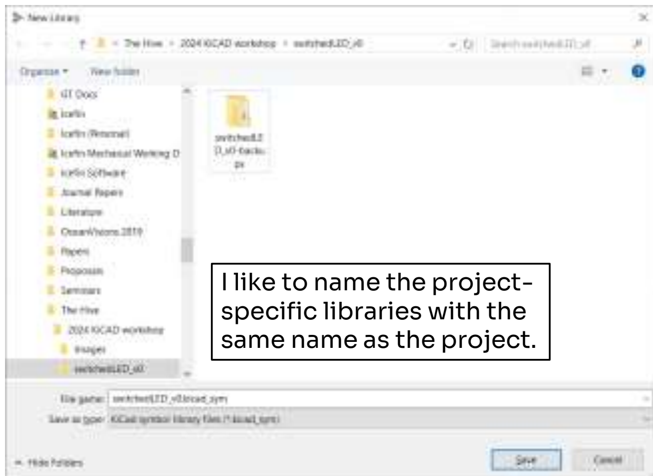
We've obviously not done that with this tutorial for a few reasons, primarily because it adds a lot of tutorial time and that it's a very simple design. If you'd like to understand how this process would work in a real design, we go through them from the beginning in parts 6 and 7.

Creating a symbol for the IC



Based on the principles of good KiCAD library management, make the new library a project-level library.

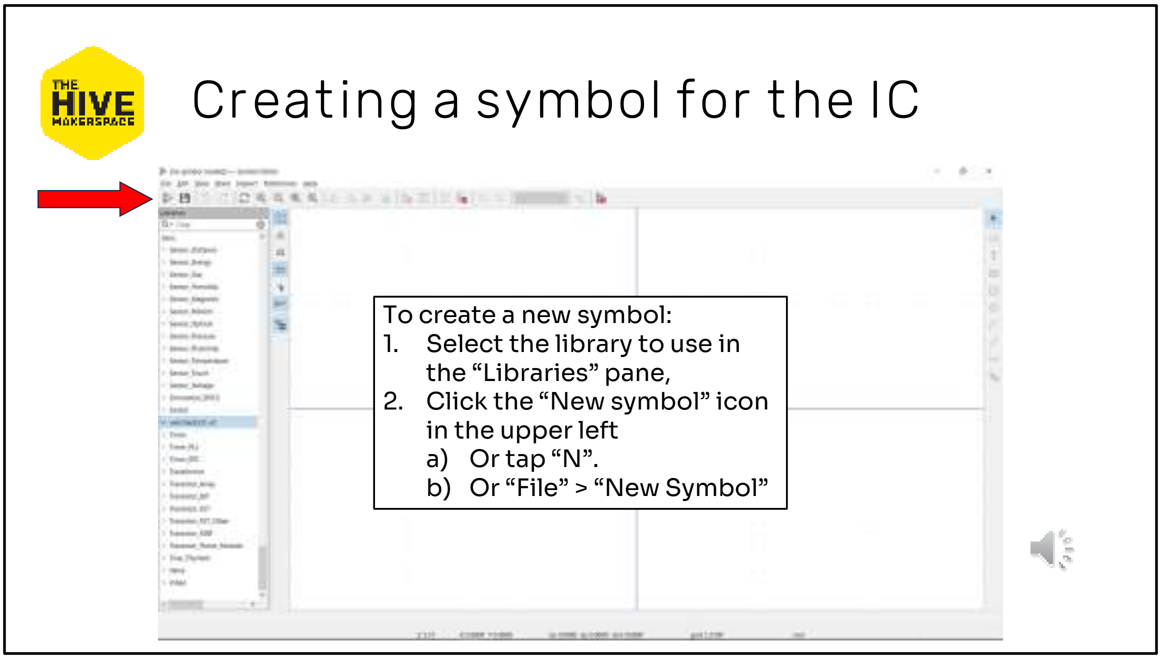
Creating a symbol for the IC



Save the library. Typically, you would save it as the same name as the project itself, and in the project directory, so that everything related to the project is nicely packaged together.

Creating a symbol for the IC

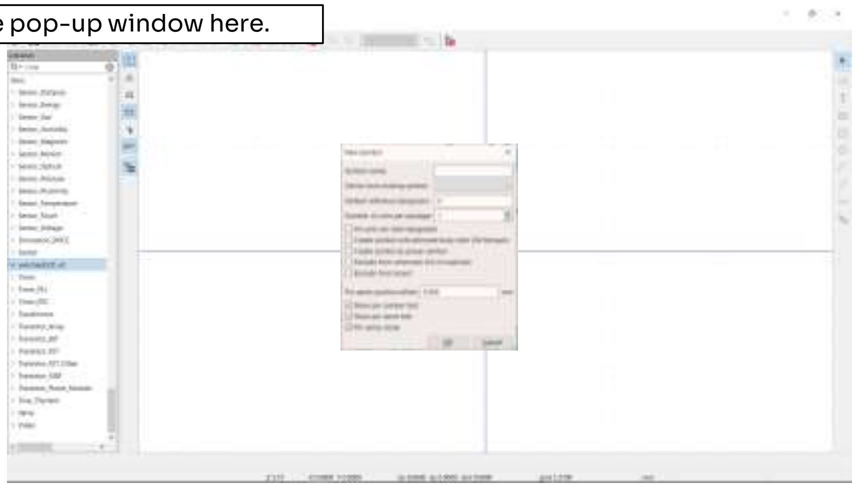
We can see the new library on the left here.



Next, we can create a new symbol just hitting the “N” key, the icon in the upper-left, or under the “File” menu.

Creating a symbol for the IC

Fill in the pop-up window here.

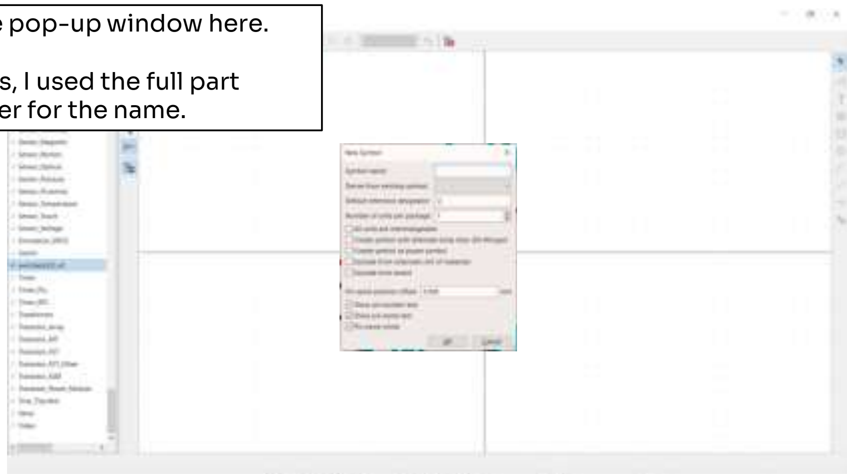


Let’s fill in the window here.

Creating a symbol for the IC

Fill in the pop-up window here.

- For ICs, I used the full part number for the name.

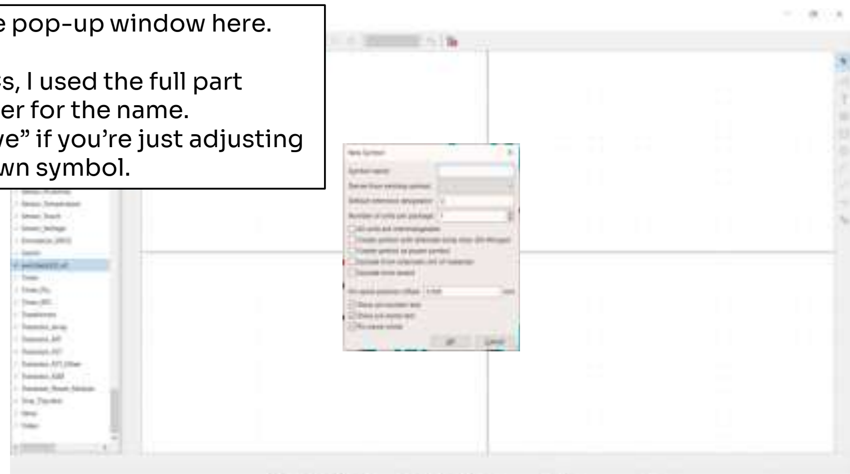


Generally, good policy is to name the symbol with the part number, which in this case would be RT4526. You can leave off the GJ6 in this case because, if you read the datasheet, those are the only options for the final three digits.

Creating a symbol for the IC

Fill in the pop-up window here.

- For ICs, I used the full part number for the name.
- "Derive" if you're just adjusting a known symbol.

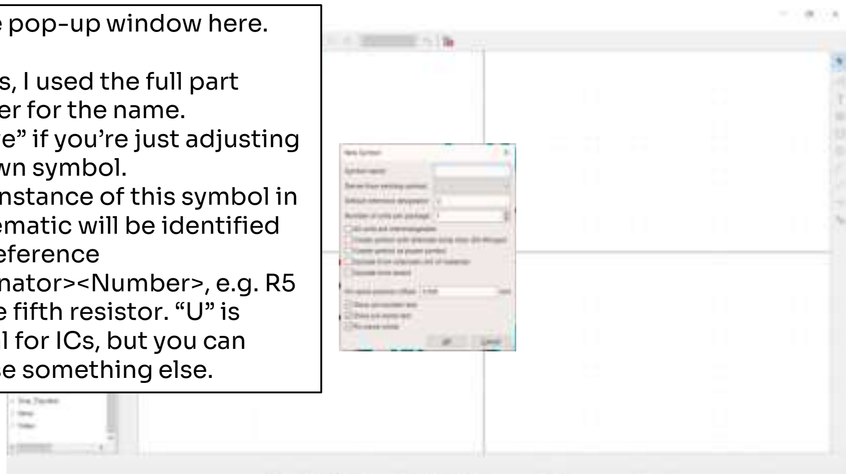


"Derive" refers to if you're adjusting a known symbol, like creating a polarized capacitor by copying the basic capacitor.

Creating a symbol for the IC

Fill in the pop-up window here.

- For ICs, I used the full part number for the name.
- “Derive” if you’re just adjusting a known symbol.
- Each instance of this symbol in a schematic will be identified by <Reference Designator><Number>, e.g. R5 for the fifth resistor. “U” is typical for ICs, but you can choose something else.



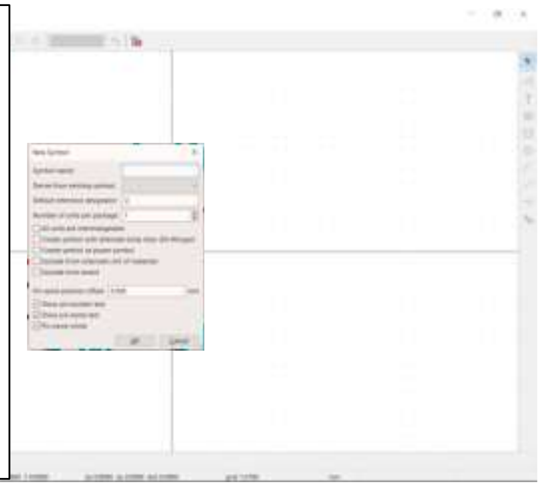
The reference designator is the letter that KiCAD uses to identify all instances of this symbol. Each instance will also get a number, so a resistor might be designated R5 for the fifth resistor of the schematic. “U” is very common for ICs, though you could choose something else if you’d like. It’s fine for multiple symbols to have the same designator character, like U or R; it just means they’re of the same type, so to speak.

Creating a symbol for the IC

Fill in the pop-up window here.

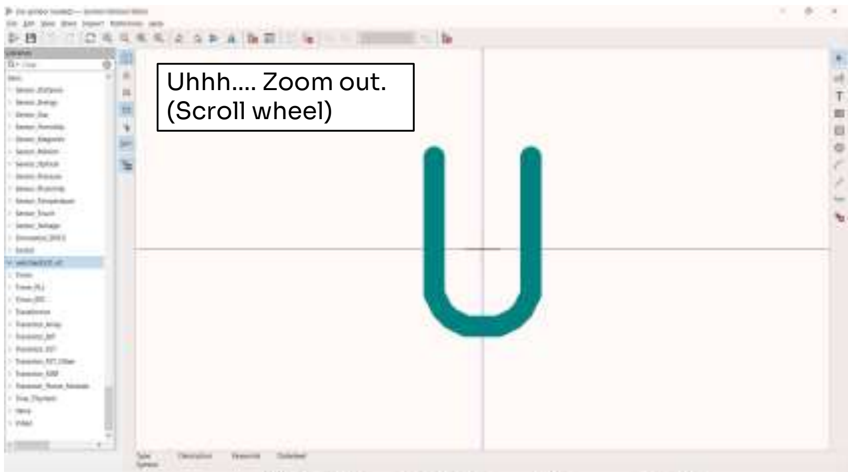
- For ICs, I used the full part number for the name.
- “Derive” if you’re just adjusting a known symbol.
- Each instance of this symbol in a schematic will be identified by <Reference Designator><Number>, e.g. R5 for the fifth resistor. “U” is typical for ICs, but you can choose something else.

The rest is not relevant now. Click OK to continue.

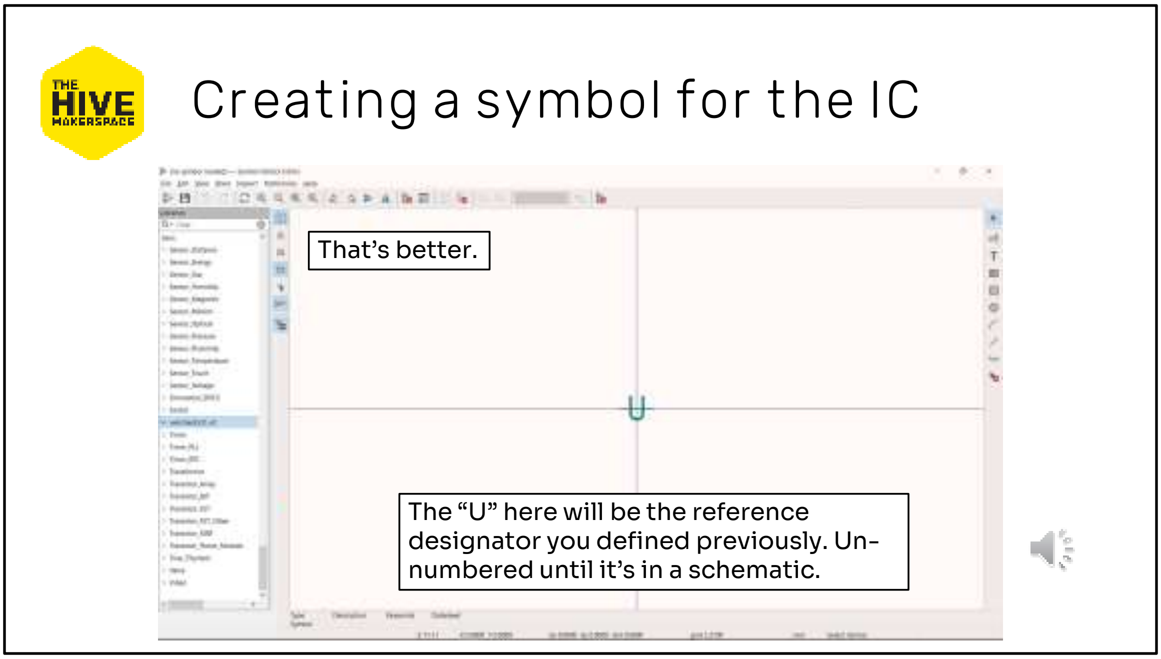


The rest doesn’t matter to us, so click “OK”.

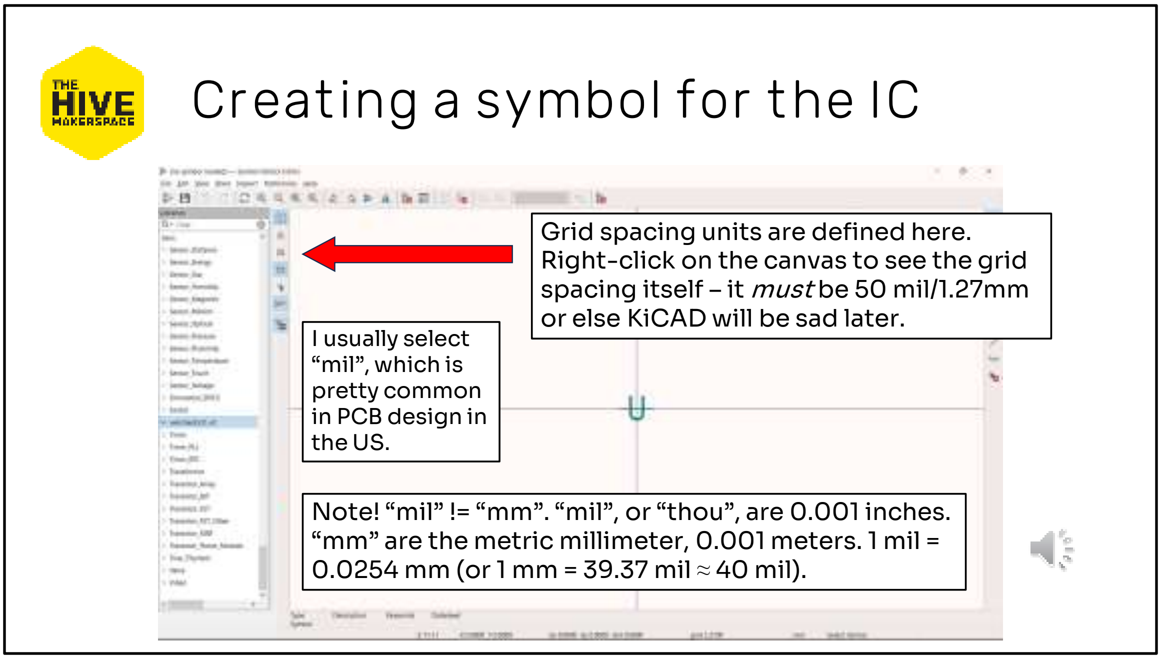
Creating a symbol for the IC



Since there’s nothing on the editor now, it will zoom automatically in to the reference designator.



Something like this is better. The “U” is currently placed at the anchor point, which is the point at which the symbol will be attached to the mouse. We’ll move those both later.

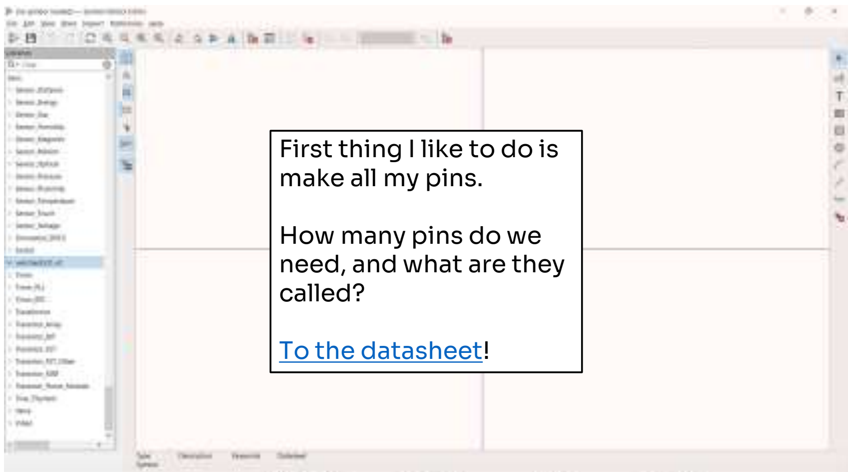


As usual, grid spacing and units are defined on the left. Check the grid by right-clicking. It should read 50 mil or 1.27 mm, or else KiCAD will not be able to attach wires to it and you'll be sad later.

I usually select mils for my units because I'm used to thinking of hole sizes and trace widths in mils, and because I'm used to imperial units. But you can readily use metric units as well. KiCAD is based on metric values, after all.

Note that mils and millimeters are not the same. Mils, which are also known as thous, are a thousandth of an inch, or 0.001 inches. Millimeters are, of course, a thousandth of a meter. One millimeter is about forty mils.

Creating a symbol for the IC

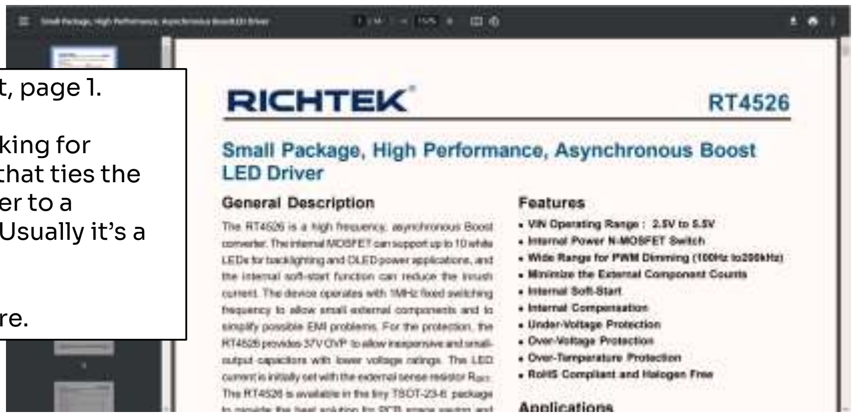


First thing I like to do is make all my pins.

How many pins do we need, and what are they called?

To the datasheet!

Creating a symbol for the IC


RICHTEK

RT4526

Small Package, High Performance, Asynchronous Boost LED Driver

General Description

The RT4526 is a high frequency, asynchronous Boost converter. The internal MOSFET can support up to 10 white LEDs for backlighting and OLED power applications, and the internal soft-start function can reduce the inrush current. The device operates with 1MHz fixed switching frequency to allow small external components and to simplify possible EMI problems. For the protection, the RT4526 provides 37V OVP to allow inexpensive and small-output capacitors with lower voltage ratings. The LED current is initially set with the external sense resistor R_{SET}. The RT4526 is available in the tiny TSOT-23-6 package

Features

- VIN Operating Range : 2.5V to 5.5V
- Internal Power N-MOSFET Switch
- Wide Range for PWM Dimming (100Hz to200kHz)
- Minimize the External Component Counts
- Internal Soft-Start
- Internal Compensation
- Under-Voltage Protection
- Over-Voltage Protection
- Over-Temperature Protection
- RoHS Compliant and Halogen Free

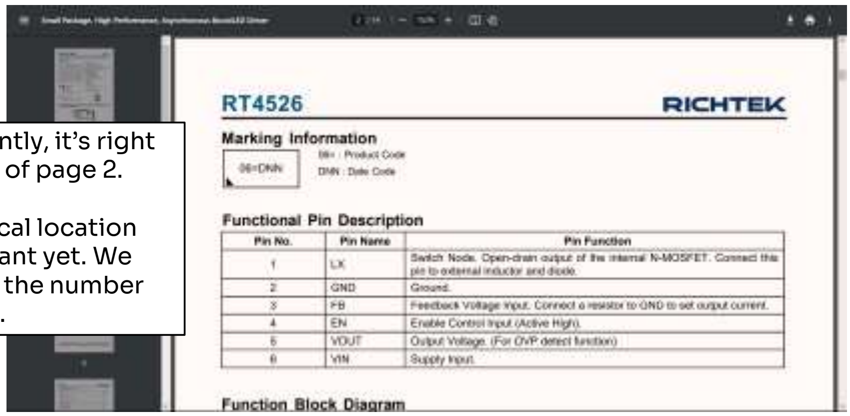
Applications

Datasheet, page 1.

We're looking for anything that ties the pin number to a function. Usually it's a table.

So not here.

Creating a symbol for the IC



RT4526 RICHTEK

Marking Information

06=DNN

06= : Product Code

DNN : Date Code

Functional Pin Description

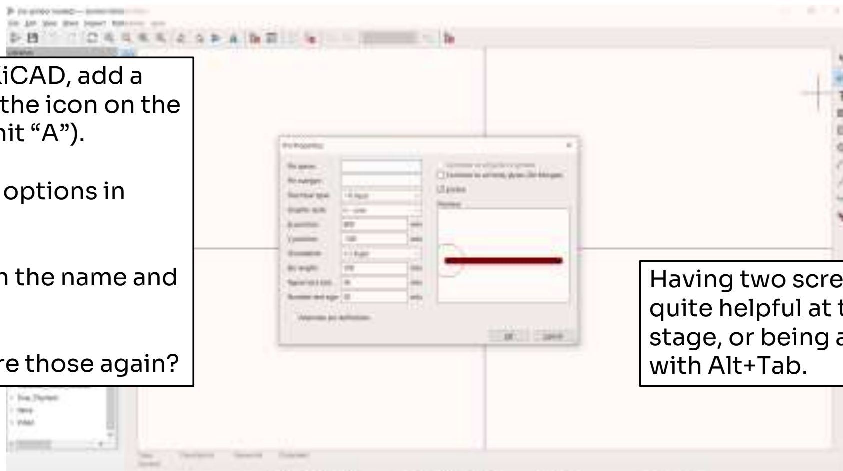
Pin No.	Pin Name	Pin Function
1	LX	Switch Node. Open-drain output of the internal N-MOSFET. Connect this pin to external inductor and diode.
2	GND	Ground.
3	FB	Feedback Voltage Input. Connect a resistor to GND to set output current.
4	EN	Enable Control Input (Active High).
5	VOUT	Output Voltage. (For OVP detect function)
6	VIN	Supply Input.

Function Block Diagram

Conveniently, it's right at the top of page 2.

The physical location isn't relevant yet. We only want the number and name.

Creating a symbol for the IC



Back to KiCAD, add a new pin (the icon on the right, or hit “A”).

Bunch of options in here.

Start with the name and number.

What were those again?

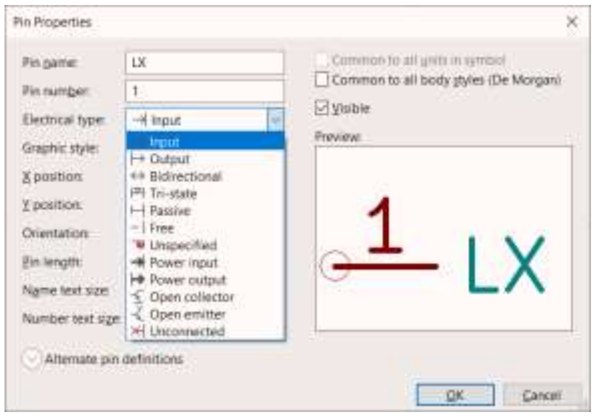
Having two screens is quite helpful at this stage, or being a wizard with Alt+Tab.

Creating a symbol for the IC

Start with pin 1.

By default, it's set as an "input". But is it?

Back to the pinout table!



Functional Pin Descripti

Pin No.	Pin Name
1	LX
2	GND
3	FB
4	EN
5	VOUT
6	VIN

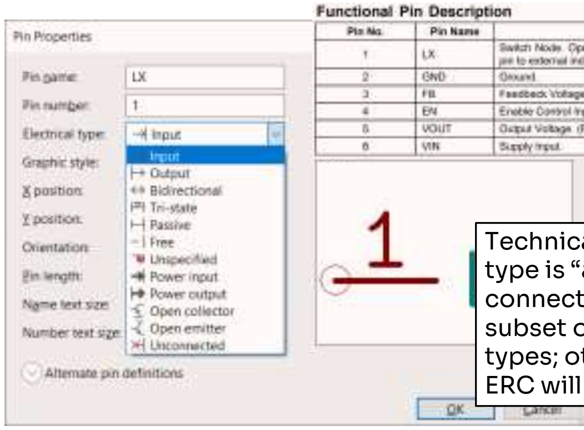
Creating a symbol for the IC

If you're not sure what it is, you can either set it to "Bidirectional" or "Unspecified" or "Passive".

The type won't cause the design to fail, but it might cause you headaches later with the ERC.

Functional Pin Description

Pin No.	Pin Name	Pin Function
1	LX	Switch Node. Open-drain output of the internal N-MOSFET. Connect this pin to external inductor and diode.
2	GND	Ground.
3	FB	Feedback Voltage Input. Connect a resistor to GND to set output current.
4	EN	Enable Control Input (Active High).
5	VOUT	Output Voltage. (For OVP detect function)
6	VIN	Supply Input.

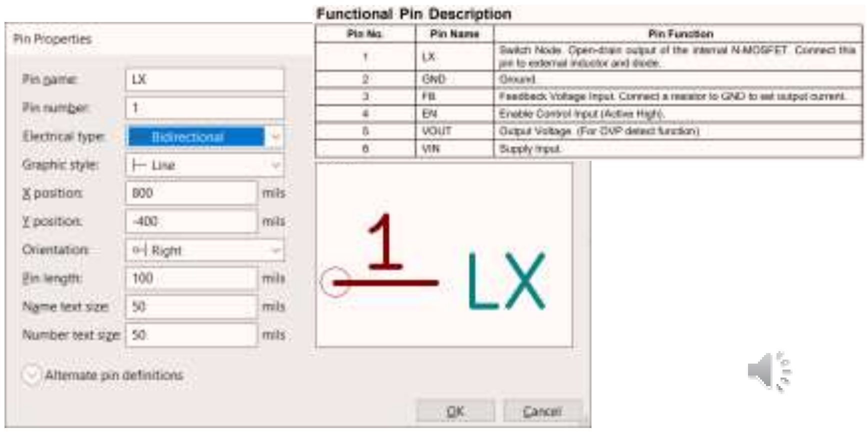


Technically, each pin type is "allowed" connected to only a subset of other pin-types; otherwise, the ERC will throw an error.

Creating a symbol for the IC

The rest of these are graphical choices, so we'll adjust them as needed later.

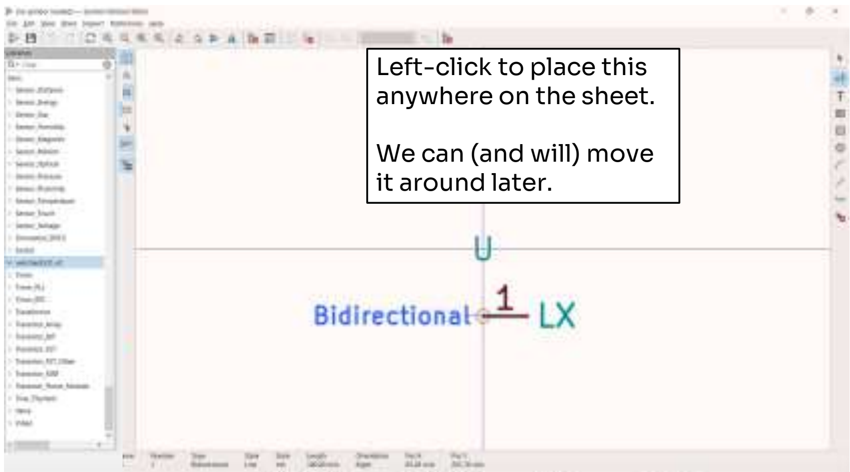
Click OK.



Functional Pin Description

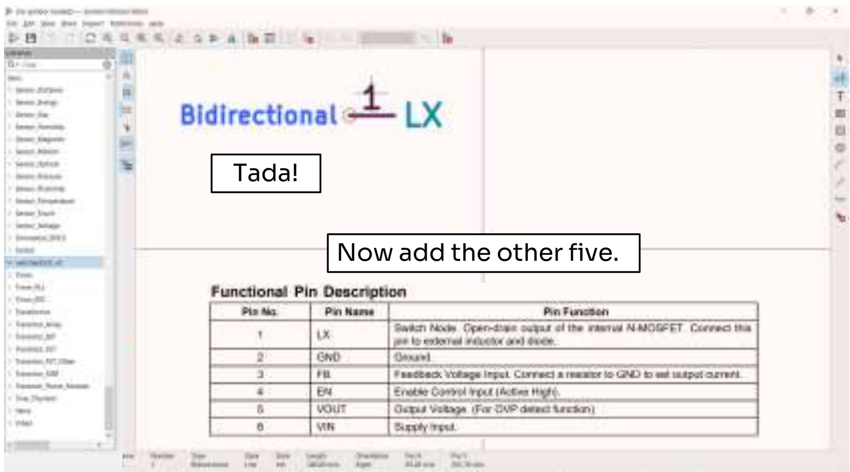
Pin No.	Pin Name	Pin Function
1	LX	Switch Node. Open-drain output of the internal N-MOSFET. Connect this pin to external inductor and diode.
2	GND	Ground.
3	FB	Feedback Voltage Input. Connect a resistor to GND to set output current.
4	EN	Enable Control Input (Active High).
5	VOUT	Output Voltage. (For OVP detect function)
6	VIN	Supply Input.

Creating a symbol for the IC

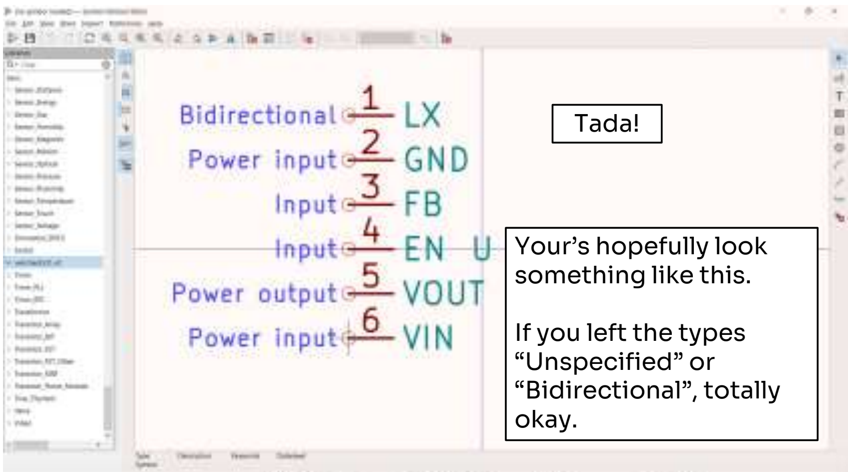


The pin type is in blue there.

Creating a symbol for the IC



Creating a symbol for the IC

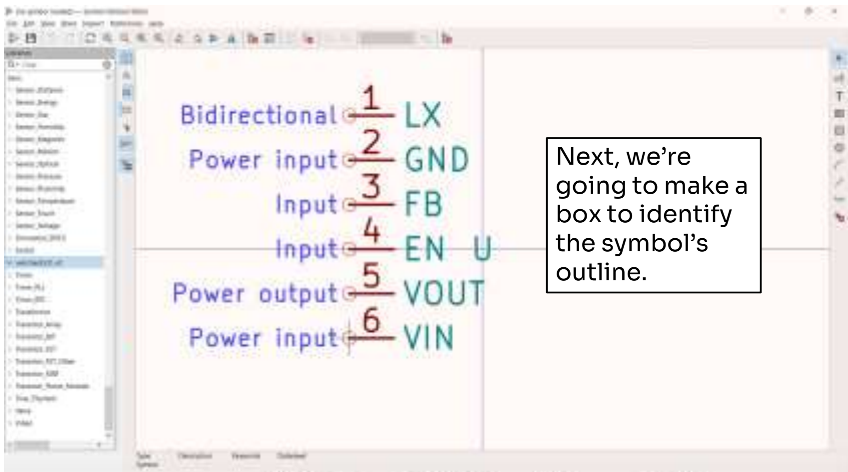


Tada!

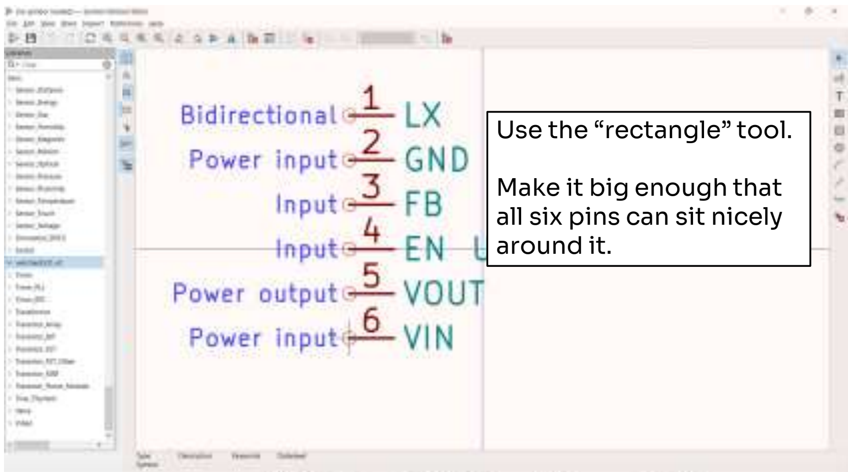
Your's hopefully look something like this.

If you left the types "Unspecified" or "Bidirectional", totally okay.

Creating a symbol for the IC



Creating a symbol for the IC



Use the “rectangle” tool.

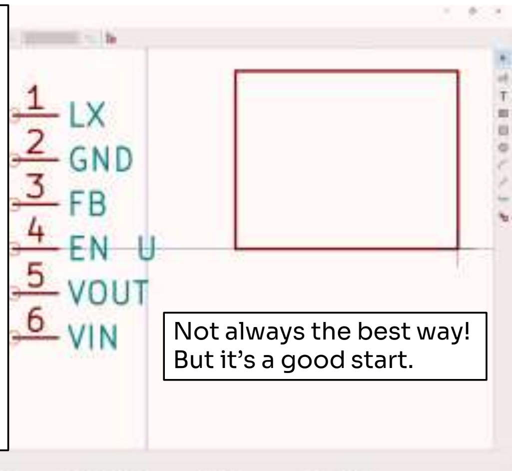
Make it big enough that all six pins can sit nicely around it.

Creating a symbol for the IC

Mine looks like this.

For height, I conceptually split the pins in half and make the box tall enough for half the pins spaced two grids apart plus two grids above and below. In this case, that's 3 pins x 2 grids per pin + 2 extra top grids + 2 extra bottom grids = 10 grids.

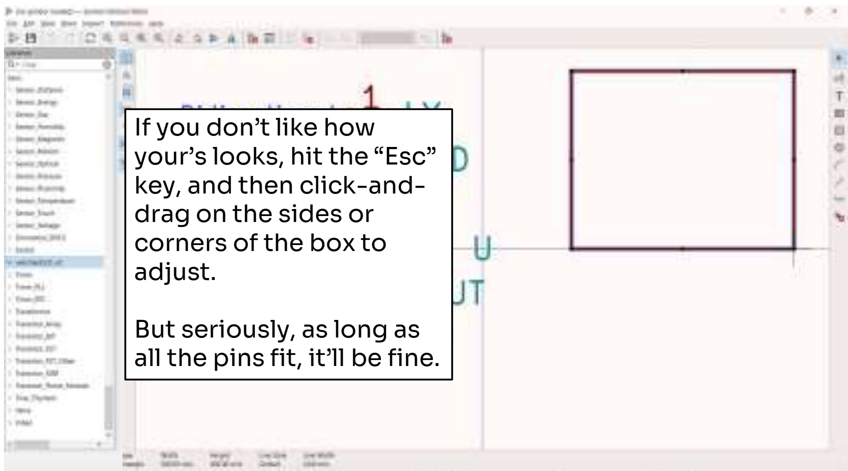
For width, I count the grides for the widest label, double it, and add two. "VOUT" is the widest at 4 grids, so that's 10 grids wide.



Not always the best way! But it's a good start.

This is the algorithm by which I spec'd my box's size, but I abandoned that size basically in the next few slides, so you're safe to ignore that.

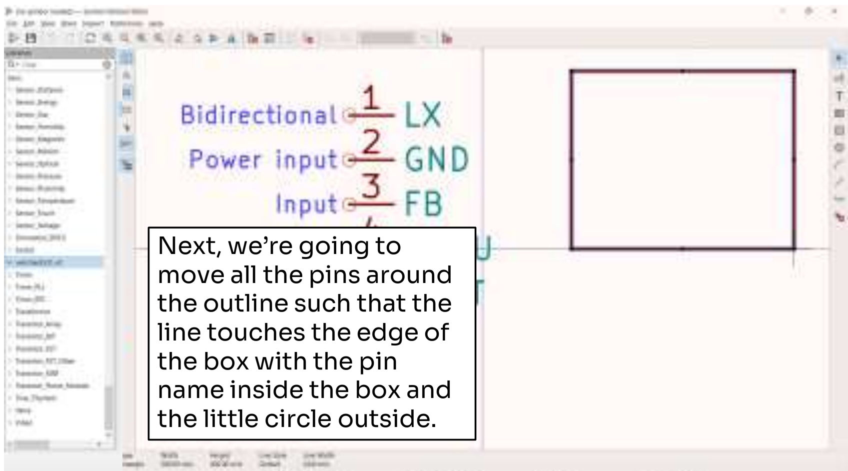
Creating a symbol for the IC



If you don't like how your's looks, hit the "Esc" key, and then click-and-drag on the sides or corners of the box to adjust.

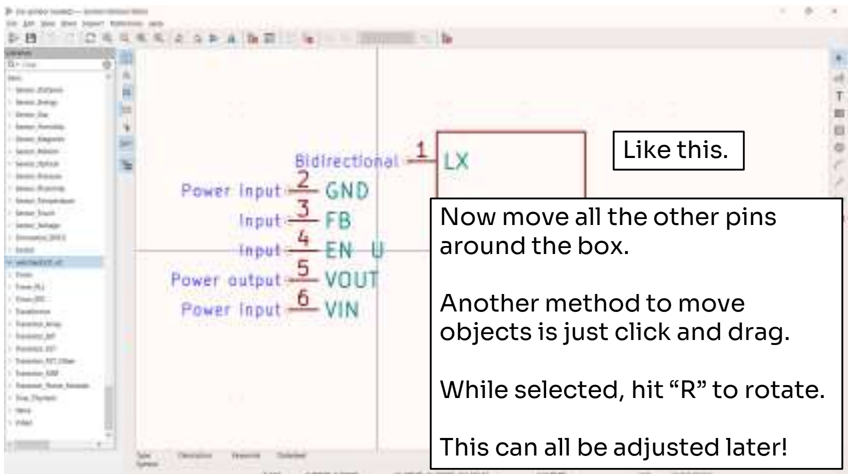
But seriously, as long as all the pins fit, it'll be fine.

Creating a symbol for the IC



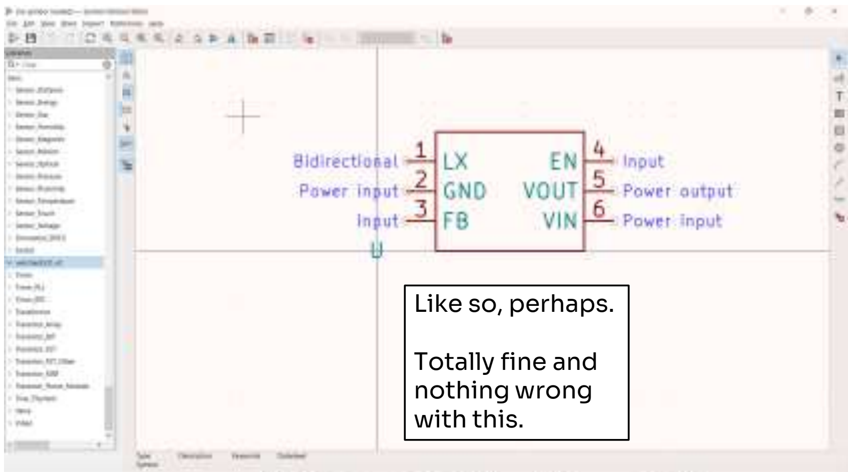
Next, we’re going to move all the pins around the outline such that the line touches the edge of the box with the pin name inside the box and the little circle outside.

Creating a symbol for the IC



You might consider pausing the video here while you place your pins before seeing what I did.

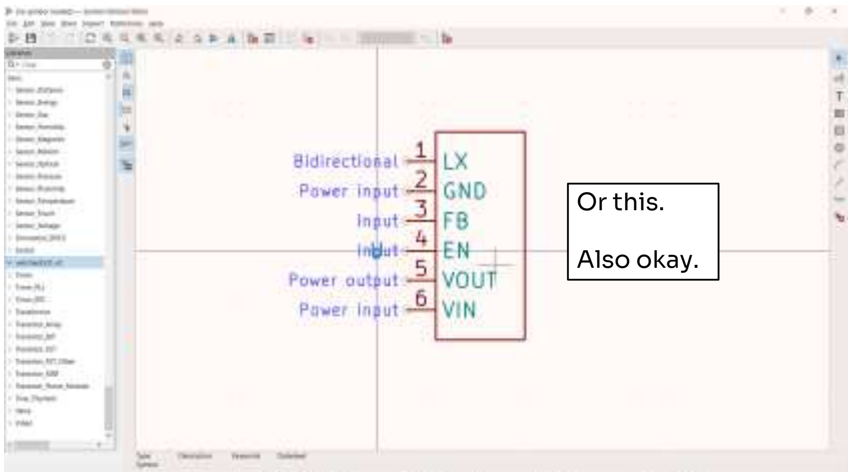
Creating a symbol for the IC



Like so, perhaps.

Totally fine and nothing wrong with this.

Creating a symbol for the IC



Or this.

Also okay.

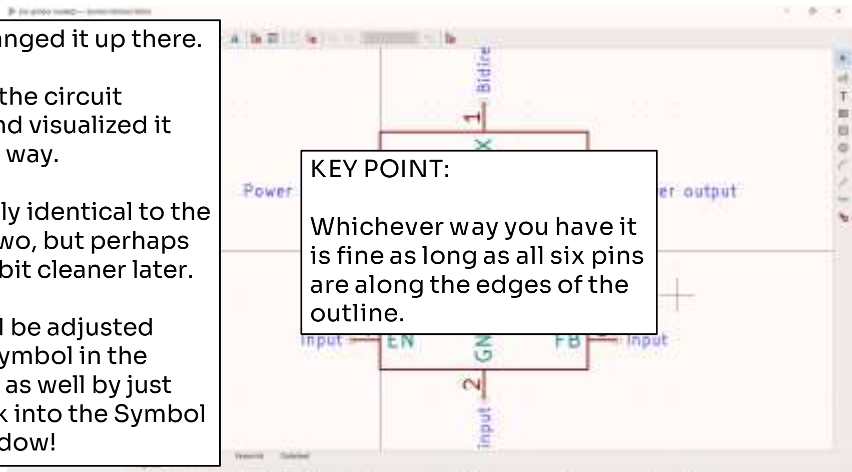
Creating a symbol for the IC

Totally changed it up there.

Looked at the circuit drawing and visualized it better this way.

Functionally identical to the previous two, but perhaps will look a bit cleaner later.

This can all be adjusted once the symbol in the schematic as well by just going back into the Symbol Editor window!



KEY POINT:

Whichever way you have it is fine as long as all six pins are along the edges of the outline.

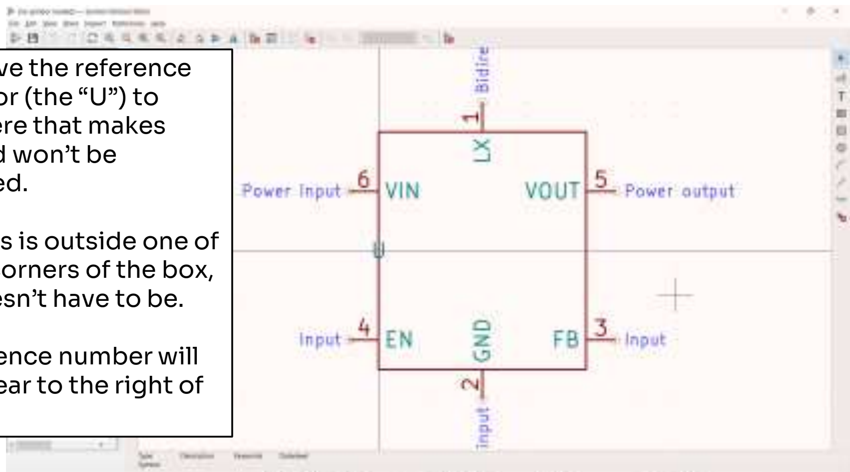
Additionally, when we learn about nets in the next video, you might understand why it doesn't matter how the symbol is laid out so much – pins can connect anywhere without a direct line.

Creating a symbol for the IC

Next, move the reference designator (the “U”) to somewhere that makes sense and won’t be obstructed.

Often, this is outside one of the four corners of the box, but it doesn’t have to be.

The reference number will later appear to the right of the “U”.



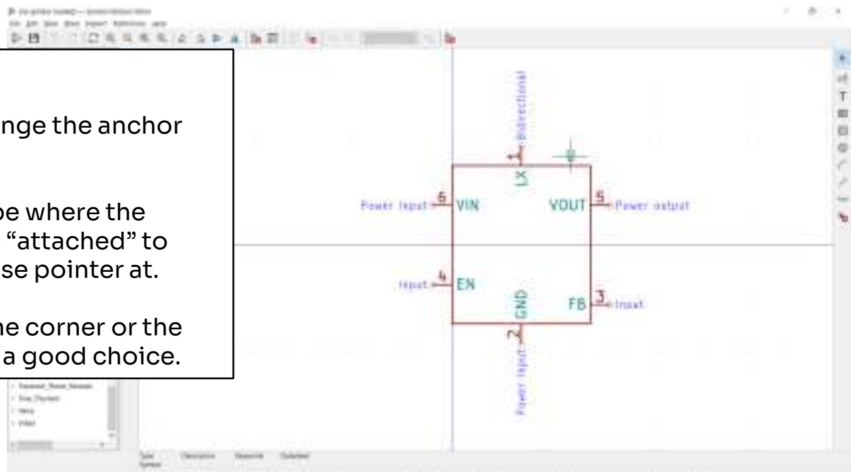
Creating a symbol for the IC

Great.

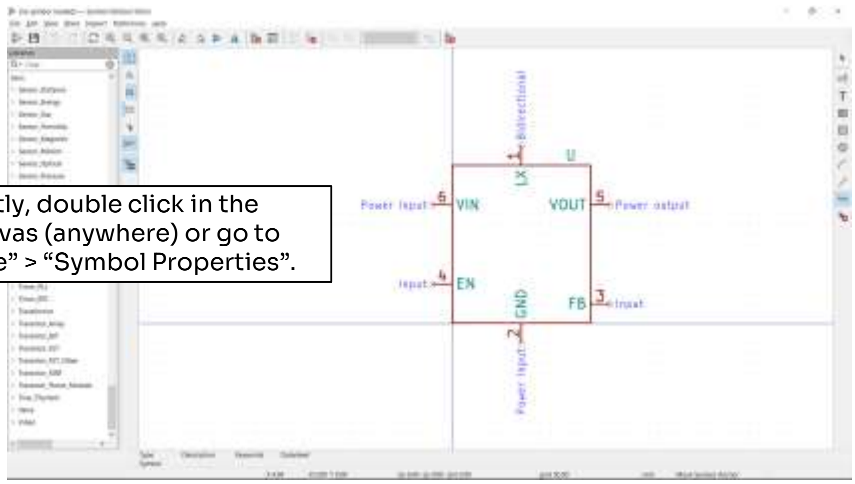
Next, change the anchor point.

This will be where the symbol is "attached" to your mouse pointer at.

Usually the corner or the middle is a good choice.



Creating a symbol for the IC



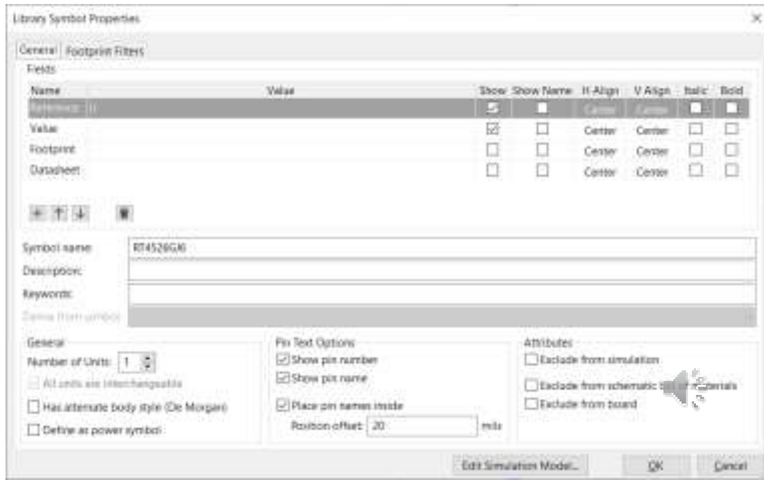
Lastly, double click in the canvas (anywhere) or go to “File” > “Symbol Properties”.

Creating a symbol for the IC

These windows often have a lot more options than we need right now, but they do offer a lot of flexibility.

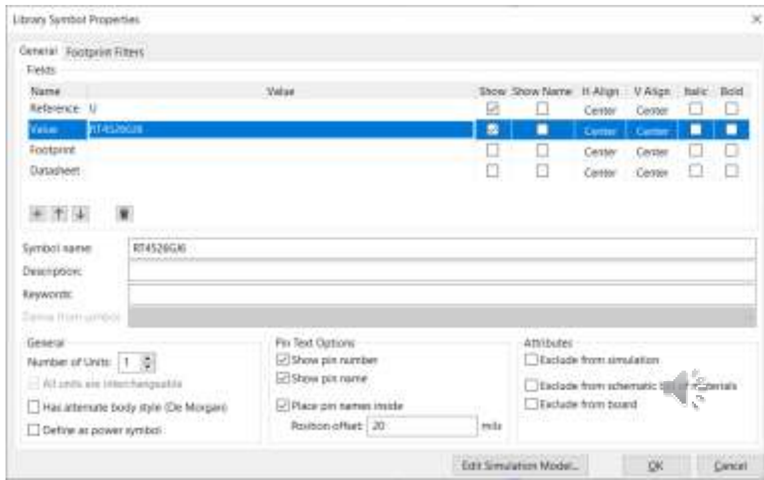
For now, just add the part number (RT4526GJ6) to the "Value" field's value.

Adding a footprint here can be done if there's a single footprint for this part. We don't have a footprint yet, so we won't add it, but later maybe.

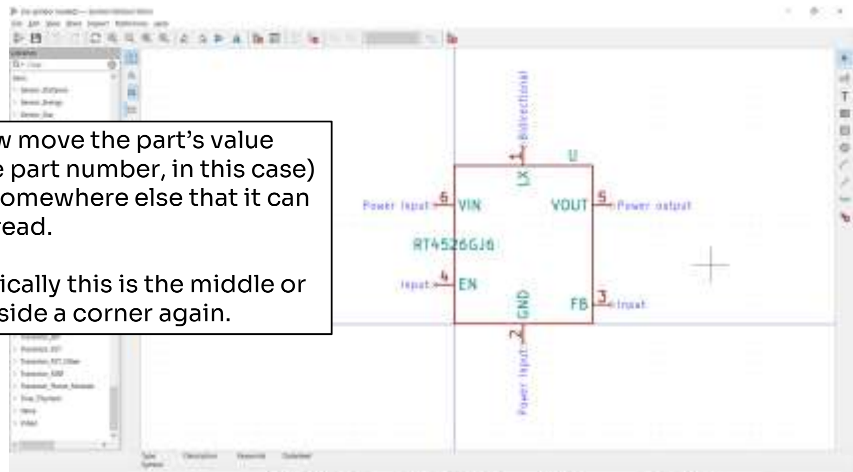


Creating a symbol for the IC

Click OK once your done.



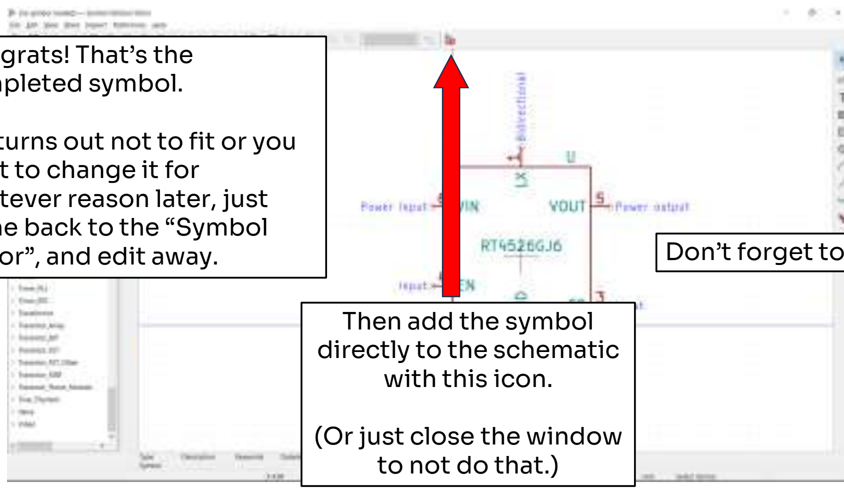
Creating a symbol for the IC



Now move the part's value (the part number, in this case) to somewhere else that it can be read.

Typically this is the middle or outside a corner again.

Creating a symbol for the IC



Congrats! That's the completed symbol.

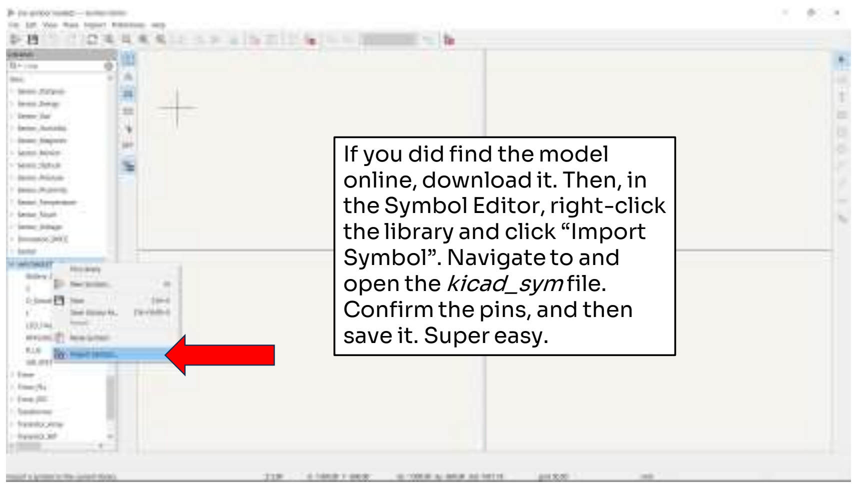
If it turns out not to fit or you want to change it for whatever reason later, just come back to the "Symbol Editor", and edit away.

Don't forget to save!

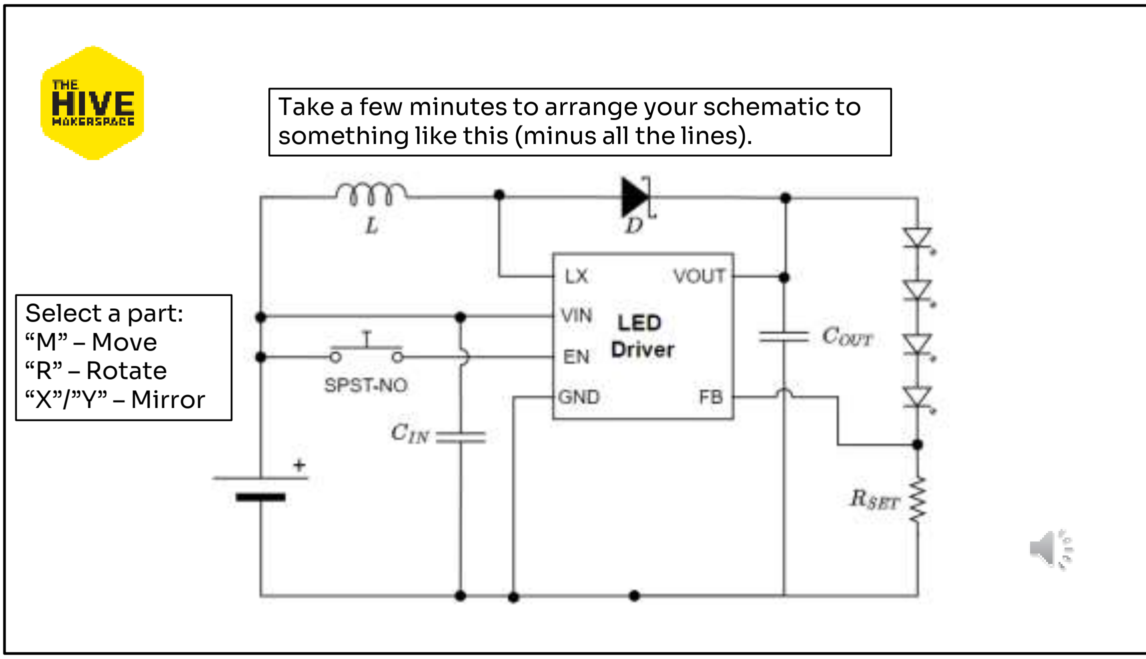
Then add the symbol directly to the schematic with this icon.

(Or just close the window to not do that.)

Aside: Importing a model



If you did find the model online, download it. Then, in the Symbol Editor, right-click the library and click “Import Symbol”. Navigate to and open the *kicad_sym* file. Confirm the pins, and then save it. Super easy.

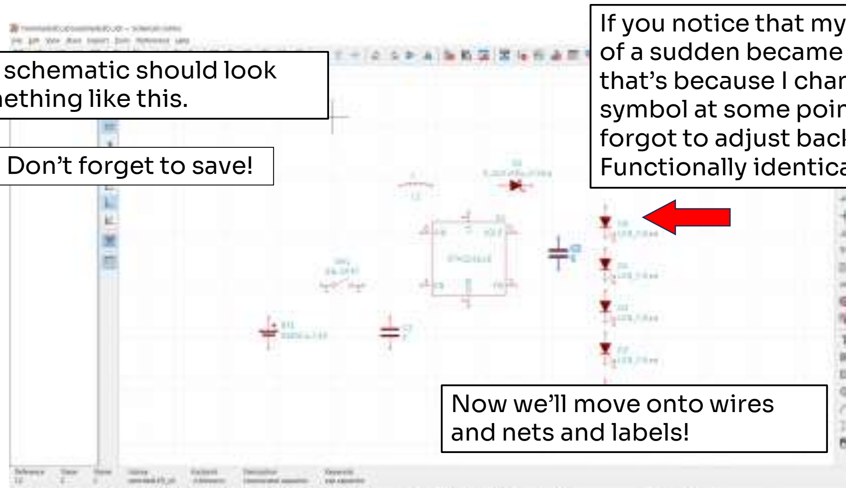


Again, pause the video here and take a few minutes to arrange your schematic. You can arrange it like this if you'd like to, but you don't have to – connections between components can be made without actual lines, as we'll cover in the next video.

Schematic with all the parts

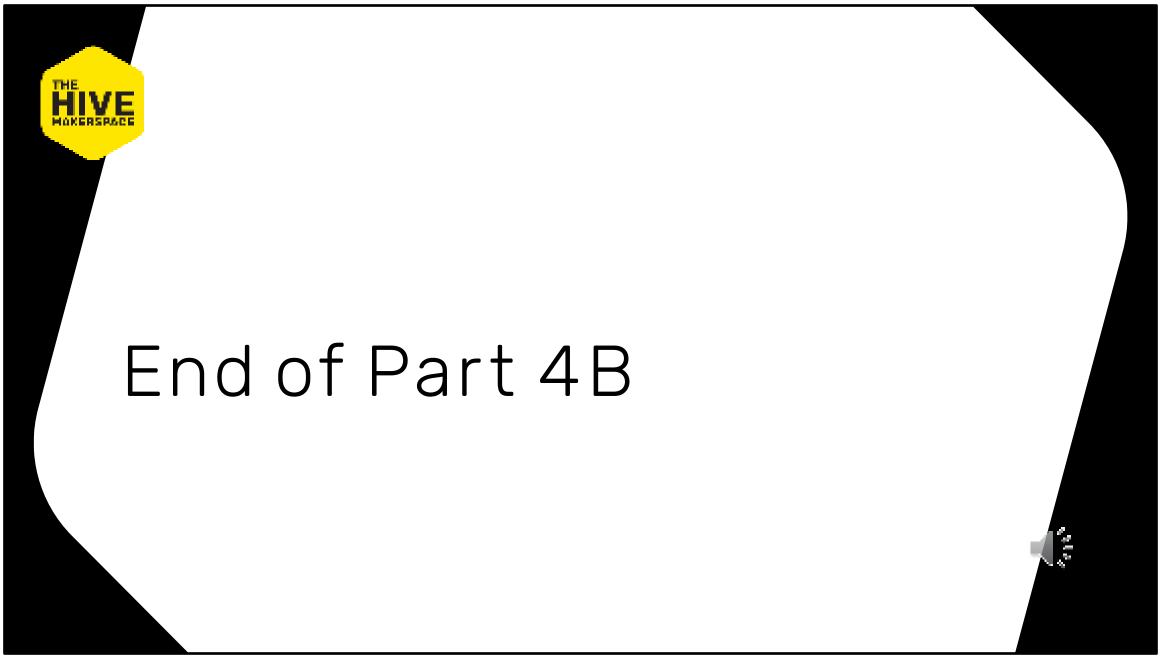
You schematic should look something like this.

Don't forget to save!



If you notice that my LEDs all of a sudden became filled, that's because I changed the symbol at some point and forgot to adjust back. Sorry. Functionally identical though!

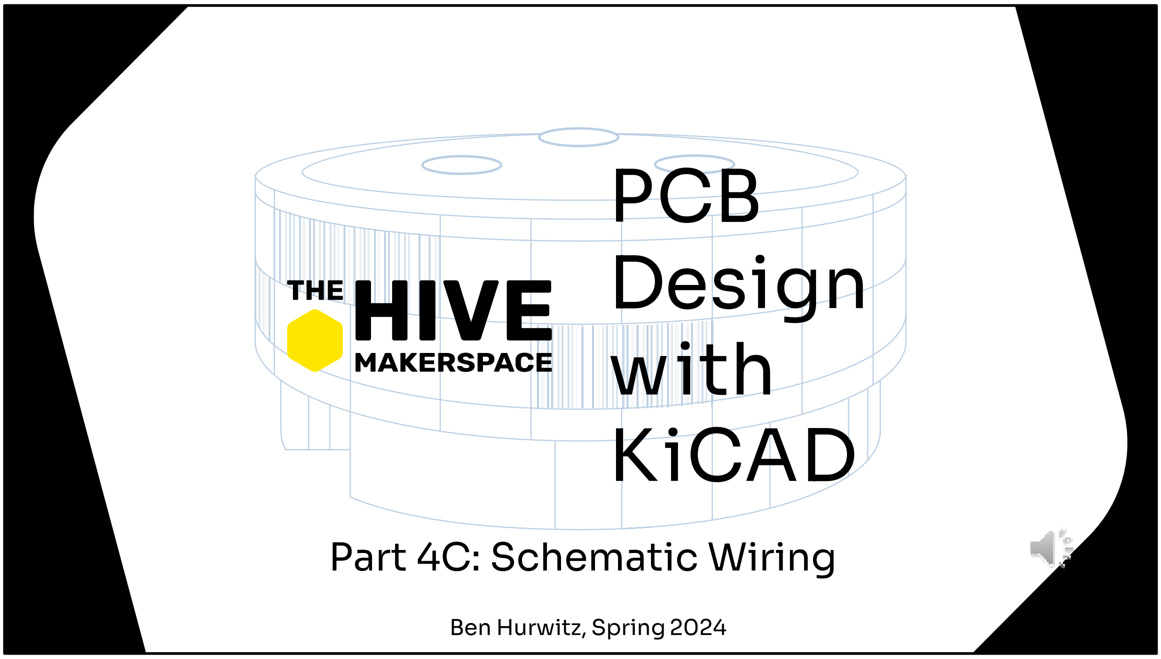
Now we'll move onto wires and nets and labels!



And that ends part 4B of this video series in which we covered locating device models online, and making a symbol for our integrated circuit component. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In the next video of this series, part 4C, I'll teach you how to connect components together.

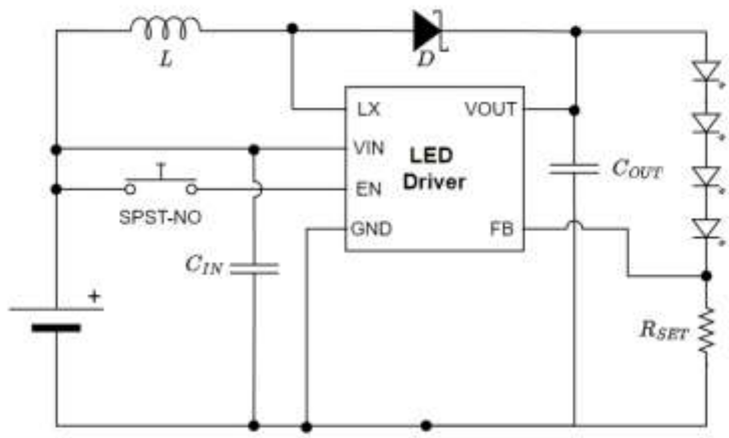
See you then.



Hi, and welcome to part 4a of The Hive's PCB Design With KiCAD series. My name is Ben, and I'll be your guide today. Part 4 as a whole will cover the entirety of the schematic creation. The segment, part 4C, will cover wiring the symbols together.

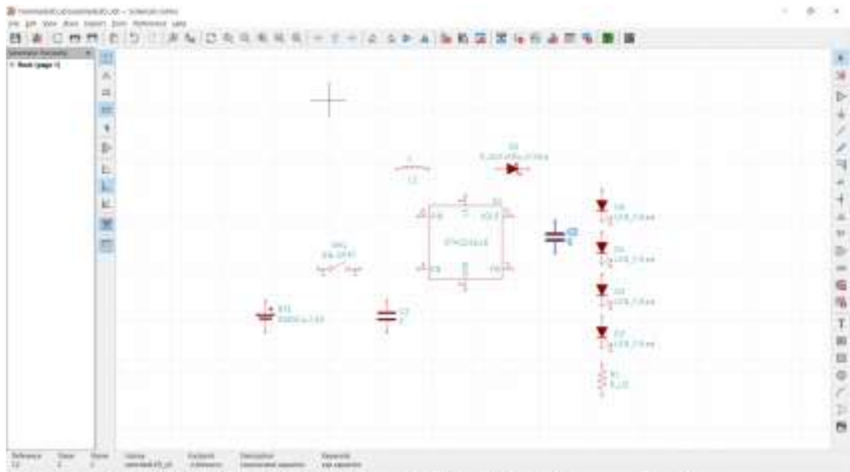
As with previous videos, it's recommended that you follow along and pause the playback frequently.

Circuit Reminder



Before we get into KiCAD, just a reminder of the flashlight circuit we're developing. Note that this image was not take from KiCAD, and therefore the symbols and graphics are different from those you are about to see.

Schematic Reminder



And this is a reminder of the schematic as it stood at the end of part 4B. All the components were down and arranged to look like the schematic in the previous slide (and like the one in the datasheet). If you've forgotten anything, I suggest you at least skim through that video (or the associated PDF).

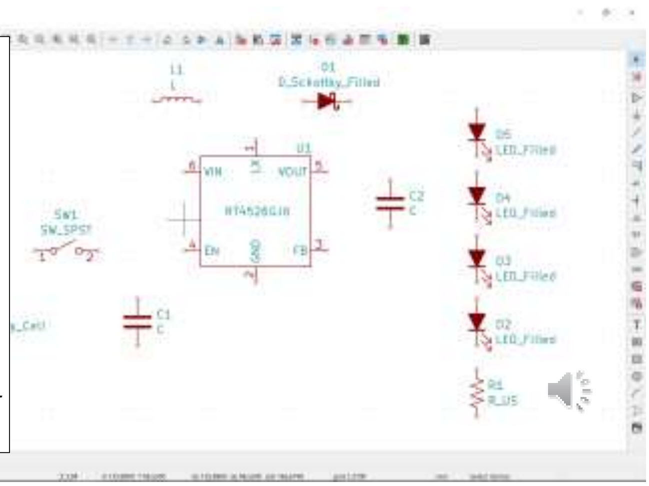
Wires and Labels and Nets – Oh My!

Components in a schematic can be connected either directly or indirectly.

Directly means connecting two pins with a *wire*.

Indirectly means using *labels* to name the nodes.

Any pin attached to nodes that have the name labeled name are electrically connected through a *net* of that name.



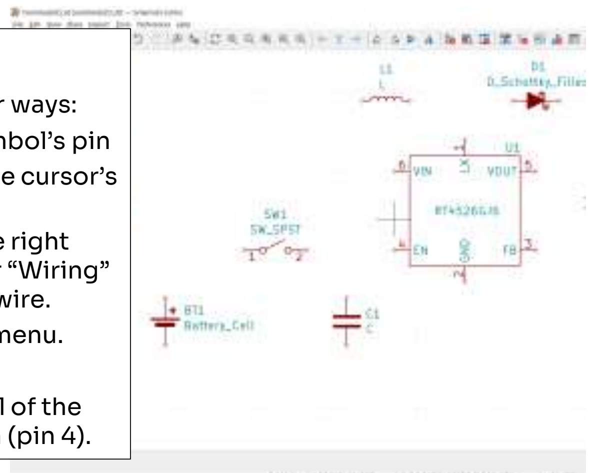
Wires and Labels and Nets – Oh My!

We'll start with wires.

Wires are created in one of four ways:

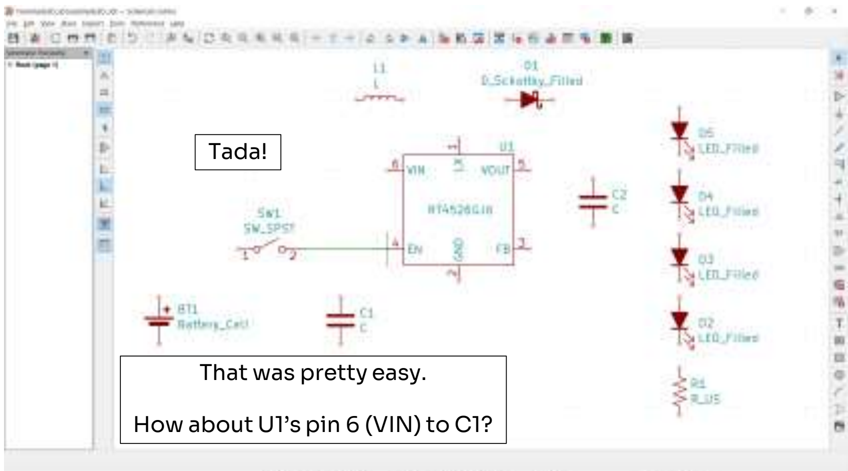
1. Click (and release) on a symbol's pin
2. Tap "W" to start a wire at the cursor's current location.
3. Click the "Wire" icon on the right (fifth from the top) to enter "Wiring" mode, then click to start a wire.
4. Click "Wire" in the "Place" menu.

Let's connect the right terminal of the switch (pin 2) to the IC's EN pin (pin 4).



You might pause the video before continuing.

Wires and Labels and Nets – Oh My!

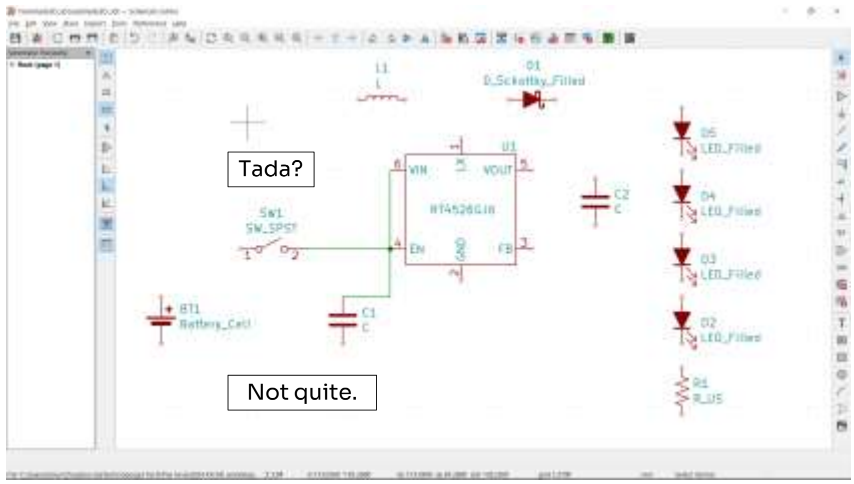


Tada!

That was pretty easy.

How about U1's pin 6 (VIN) to C1?

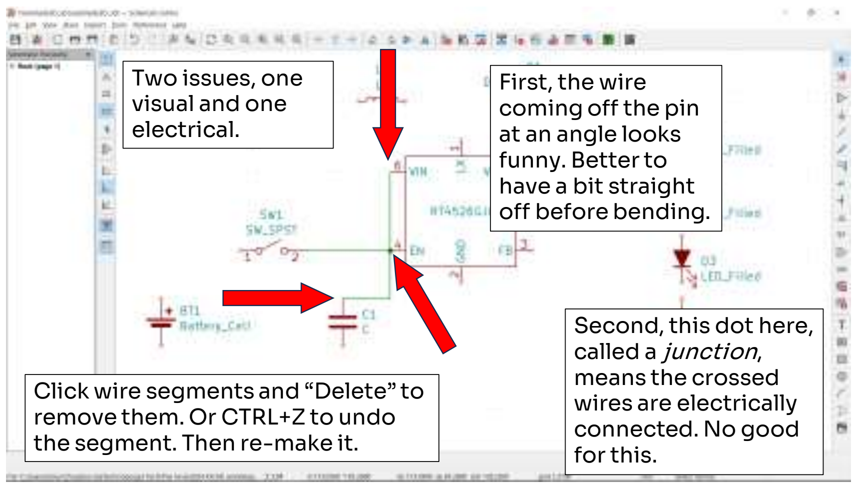
Wires and Labels and Nets – Oh My!



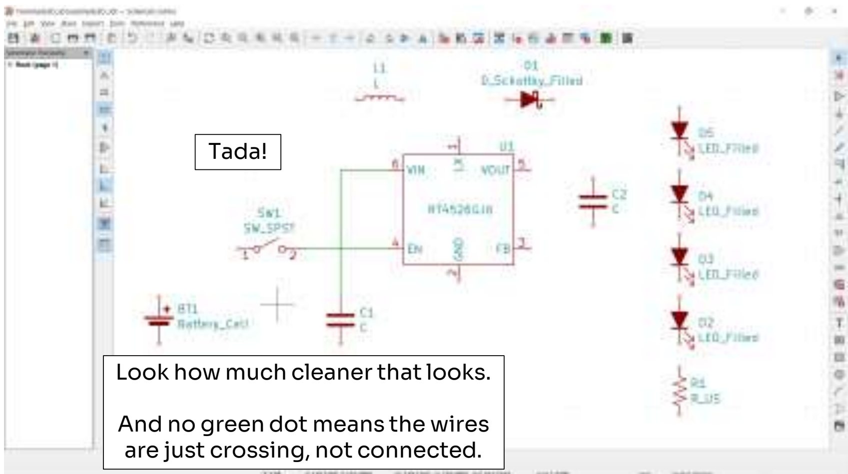
How’s this look?

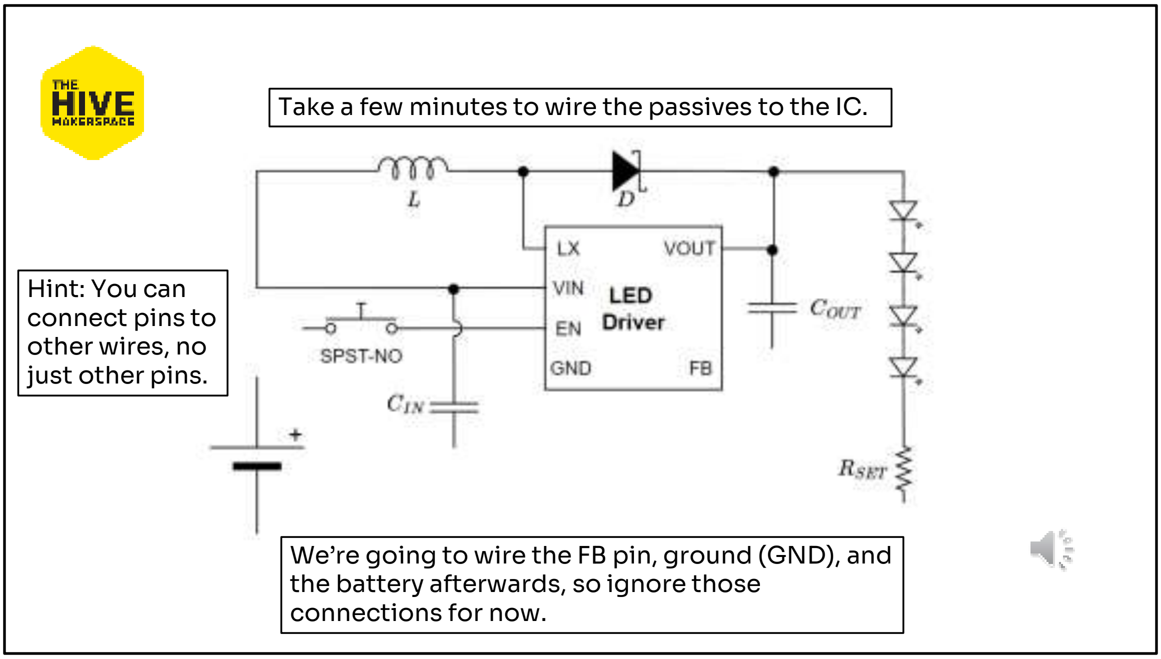
*Not quite.

Wires and Labels and Nets – Oh My!



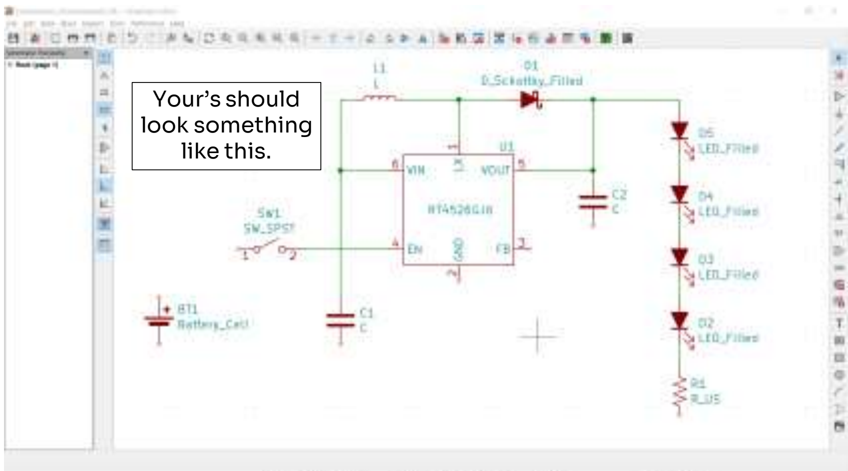
Wires and Labels and Nets – Oh My!





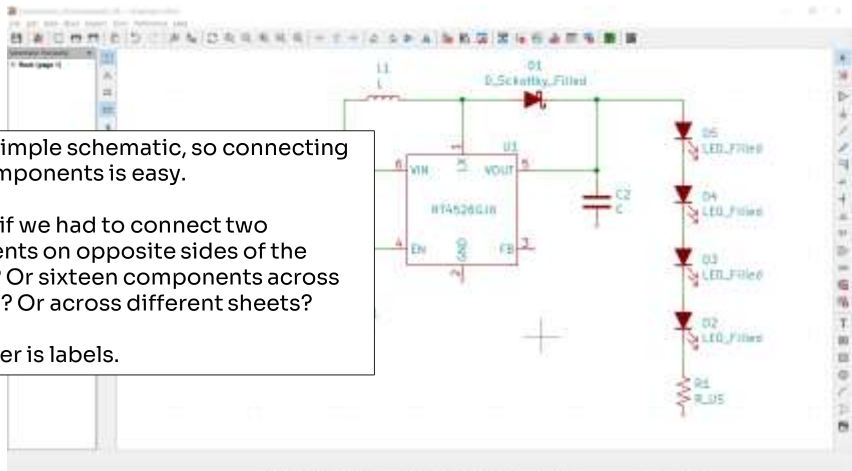
I suggest pausing the video here to try this on our own before continuing.

Wires and Labels and Nets – Oh My!



Again, it doesn't /have/ to look like this at all. But as long as the components are connected, you're fine.

Wires and Labels and Nets – Oh My!



This is a simple schematic, so connecting these components is easy.

But what if we had to connect two components on opposite sides of the diagram? Or sixteen components across the sheet? Or across different sheets?

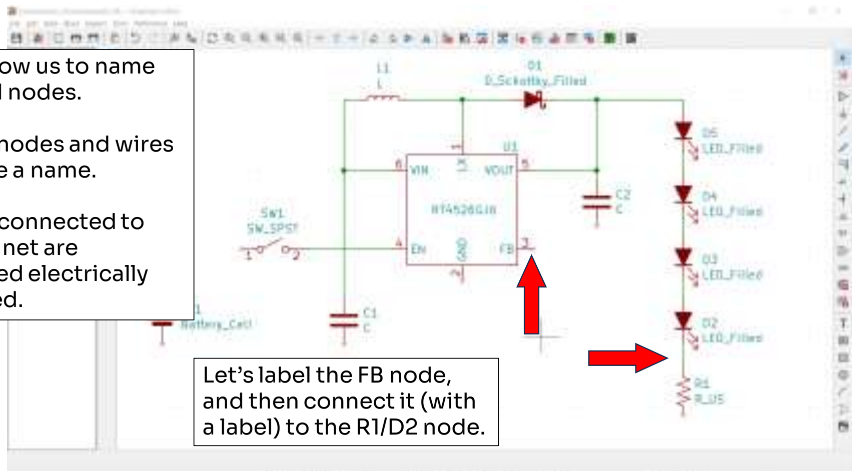
The answer is labels.

Wires and Labels and Nets – Oh My!

Labels allow us to name wires and nodes.

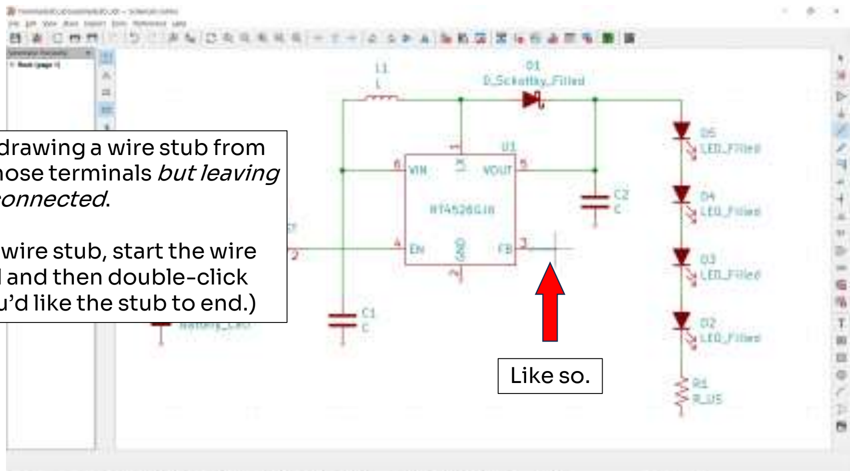
Nets are nodes and wires that share a name.

Any pins connected to the same net are considered electrically connected.



Let’s label the FB node, and then connect it (with a label) to the R1/D2 node.

Wires and Labels and Nets – Oh My!



I start by drawing a wire stub from each of those terminals *but leaving them unconnected*.

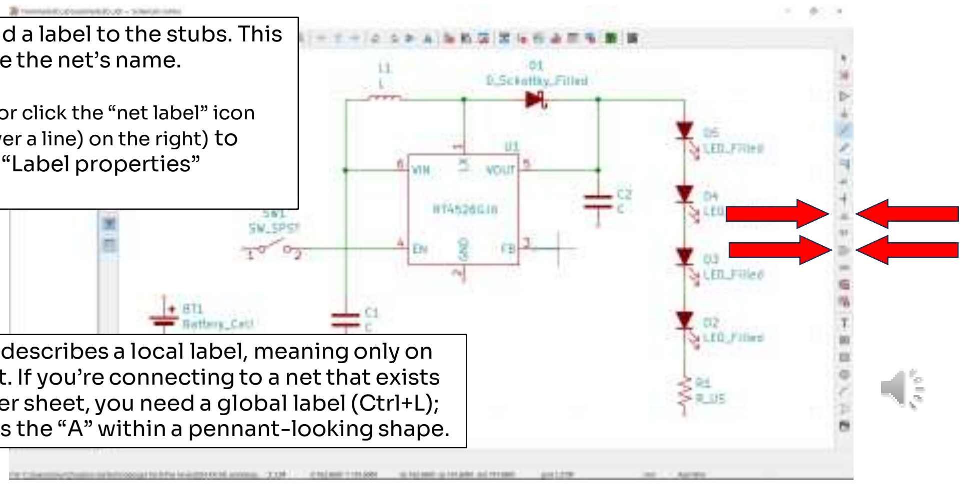
(To end a wire stub, start the wire as normal and then double-click where you'd like the stub to end.)

Like so.

Wires and Labels and Nets – Oh My!

Next, I add a label to the stubs. This will define the net's name.

Tap "L"[1] (or click the "net label" icon (the "A" over a line) on the right) to open the "Label properties" window.



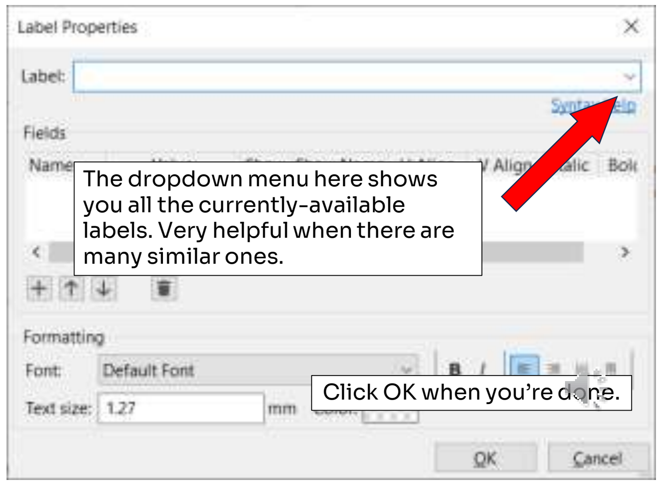
Note: "L" describes a local label, meaning only on this sheet. If you're connecting to a net that exists on another sheet, you need a global label (Ctrl+L); the icon is the "A" within a pennant-looking shape.

Wires and Labels and Nets – Oh My!

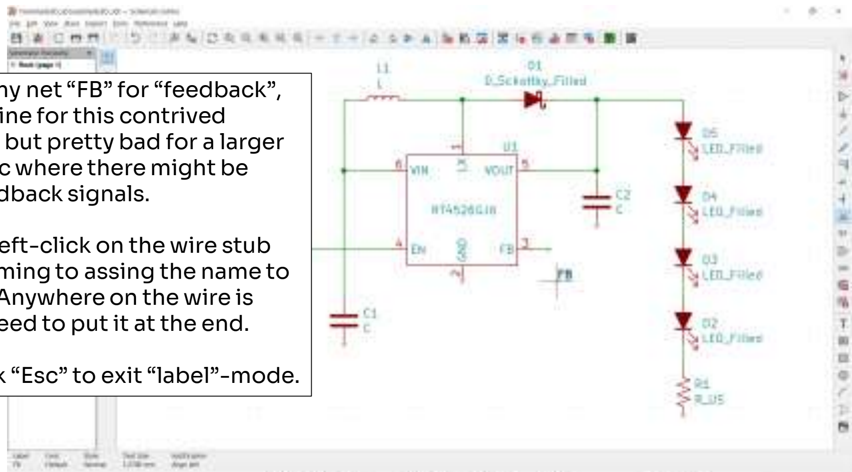
Type the name of the connecting net into the “Label” field and click “OK”.

The restrictions on names center around some symbols, and some globally-defined names like “gnd”.

(You can get fancy if you want here, but boring labels are fine, too.)



Wires and Labels and Nets – Oh My!

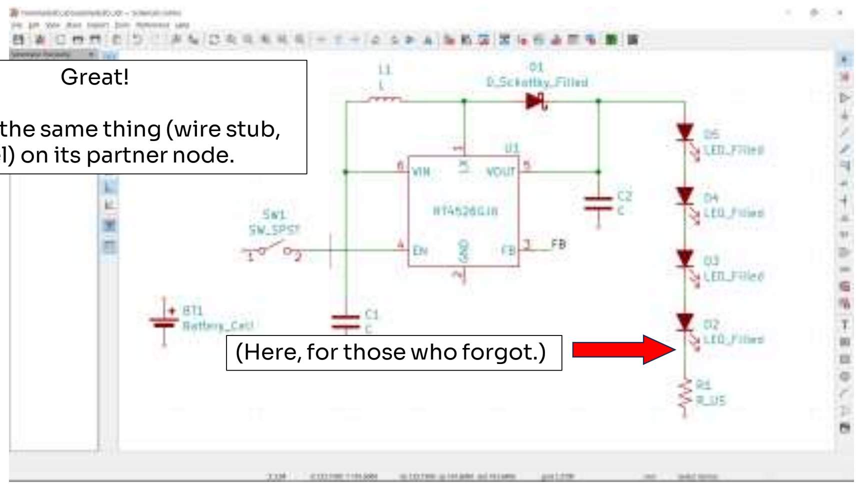


I named my net "FB" for "feedback", which is fine for this contrived situation, but pretty bad for a larger schematic where there might be many feedback signals.

Anyway, left-click on the wire stub you're naming to assing the name to the wire. Anywhere on the wire is fine, no need to put it at the end.

Then click "Esc" to exit "label"-mode.

Wires and Labels and Nets – Oh My!

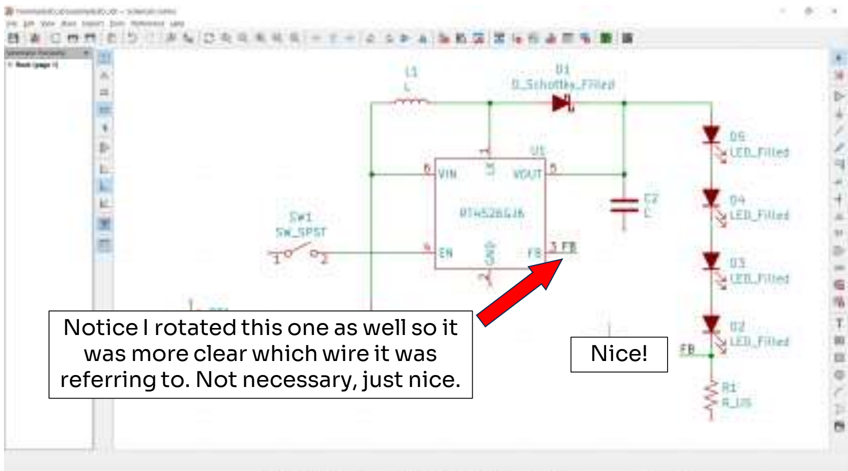


Great!

Now do the same thing (wire stub, label) on its partner node.

(Here, for those who forgot.)

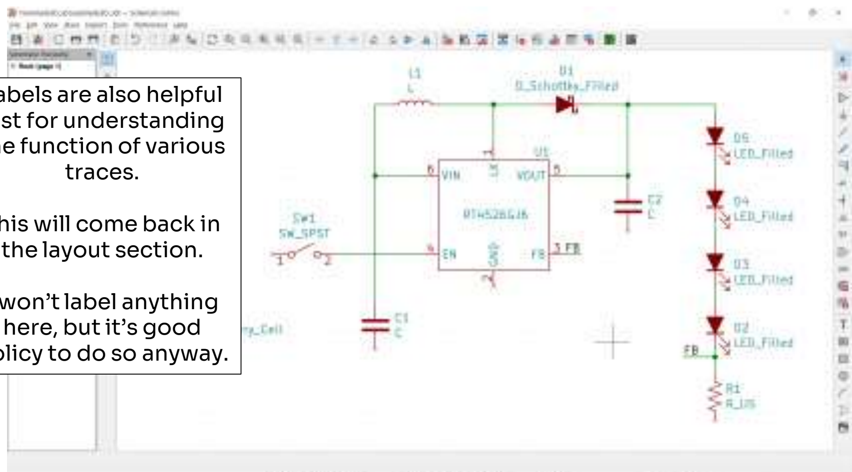
Wires and Labels and Nets – Oh My!



Notice I rotated this one as well so it was more clear which wire it was referring to. Not necessary, just nice.

Nice!

Wires and Labels and Nets – Oh My!



Labels are also helpful just for understanding the function of various traces.

This will come back in the layout section.

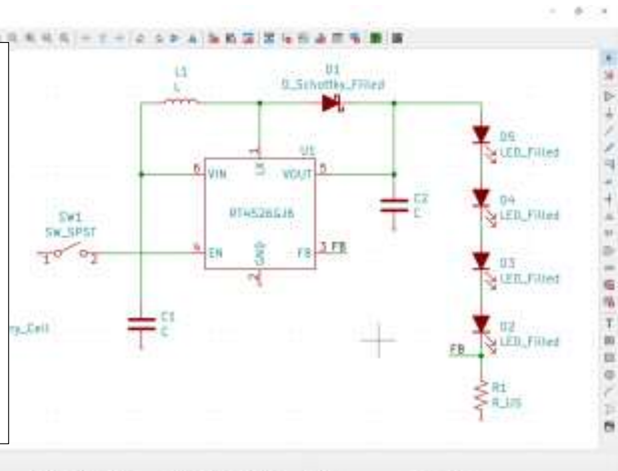
I won't label anything here, but it's good policy to do so anyway.

Wires and Labels and Nets – Oh My!

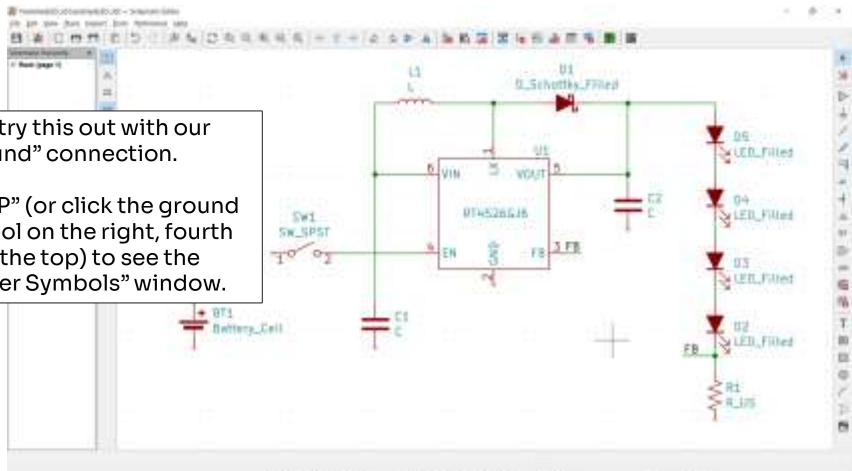
There are special nets for power and ground connections as well.

These come with their own symbols, meaning anything the symbol is connected to is automatically connected to the net of that name.

Exercise caution here! There is some subtelty with these that I'm not covering that can cause ERC failures or mistakes.



Wires and Labels and Nets – Oh My!

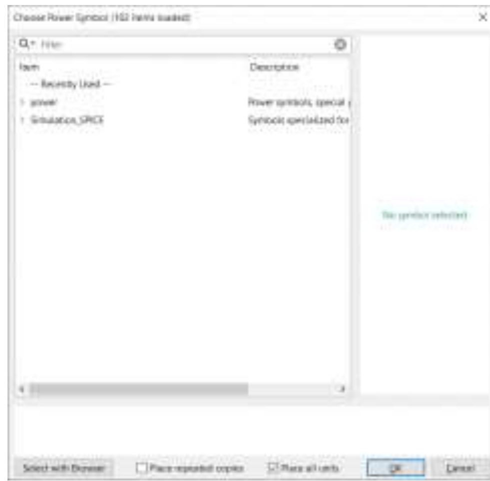


Let's try this out with our "ground" connection.

Tap "P" (or click the ground symbol on the right, fourth from the top) to see the "Power Symbols" window.

Wires and Labels and Nets – Oh My!

Basically the same as the
“Add Symbol” window.

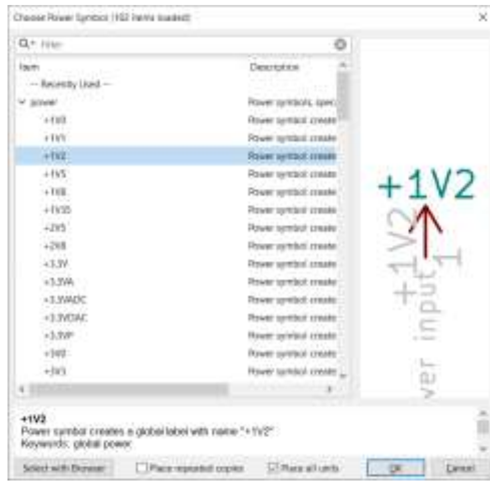


Wires and Labels and Nets – Oh My!

Opening the “power” library shows a bunch of different symbols for different voltages. Most look like this.

Note that these will also create a *global net* with the same name as the symbol, e.g. “+1V2”.

This is one of those label naming restrictions – errors will be thrown if you use these names for your other nets, or confusion will reign.

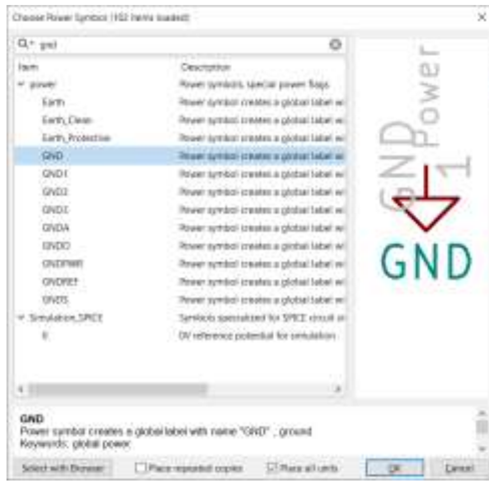


Wires and Labels and Nets – Oh My!

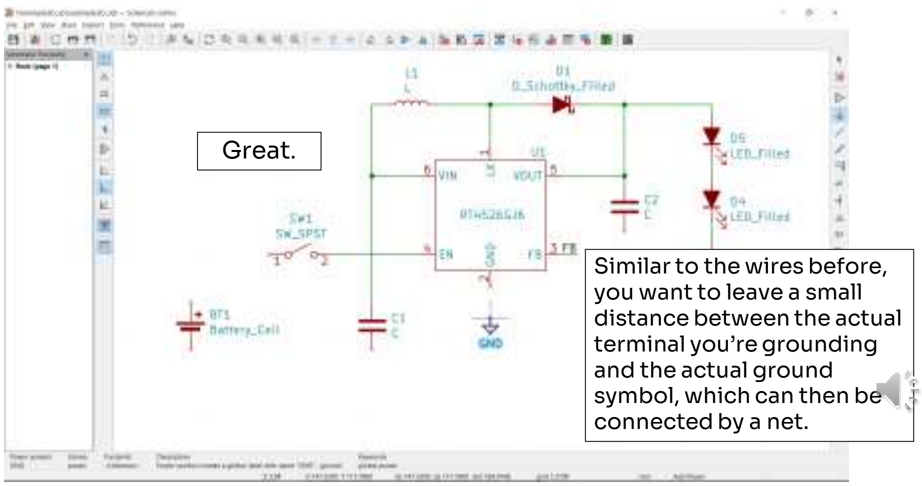
If we filter by “gnd” or “ground”, we’ll see a number of different ground net/symbol options.

For this schematic, with only a single common return path, the regular GND is what we want.

Place it below (but not connected to) the battery’s negative pin.

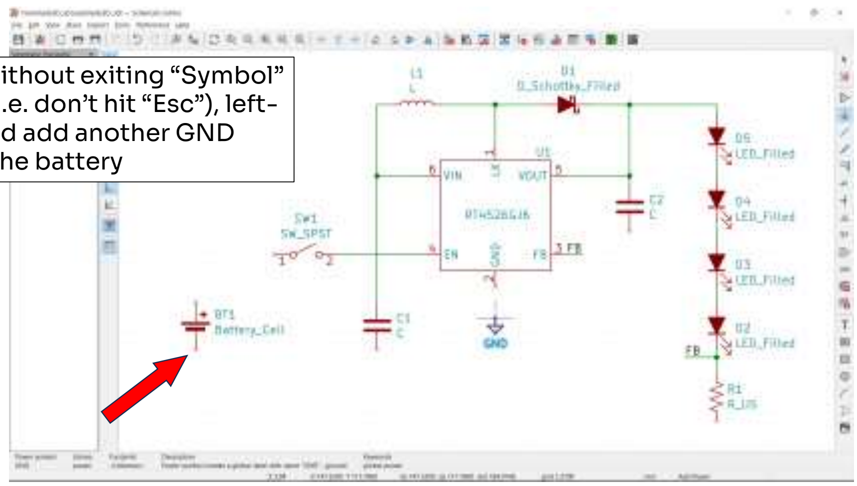


Wires and Labels and Nets – Oh My!



Wires and Labels and Nets – Oh My!

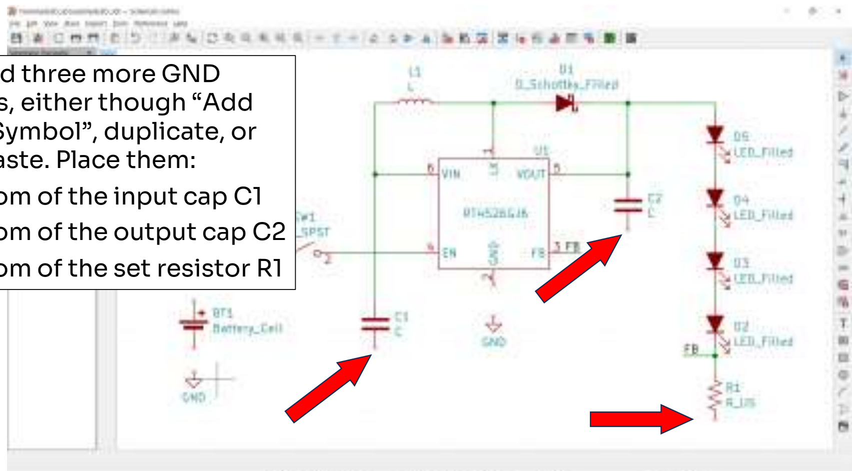
Then, without exiting “Symbol” mode (i.e. don’t hit “Esc”), left-click and add another GND below the battery



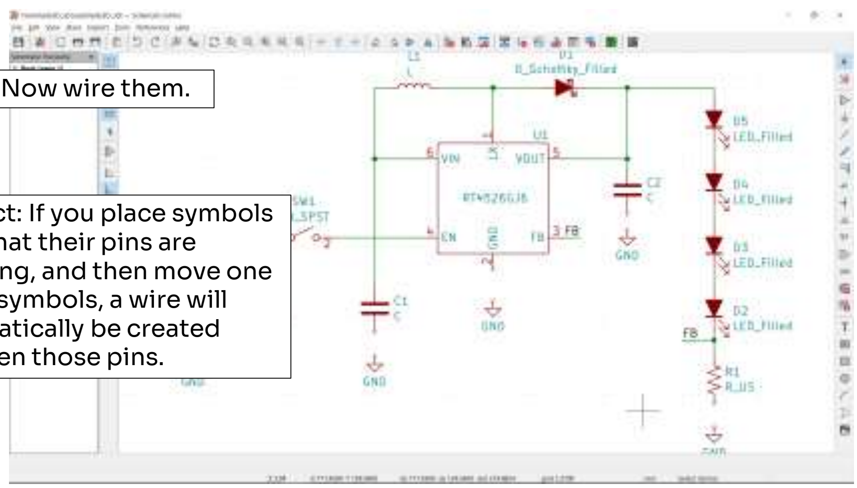
Wires and Labels and Nets – Oh My!

Now add three more GND symbols, either though “Add Power Symbol”, duplicate, or copy/paste. Place them:

- Bottom of the input cap C1
- Bottom of the output cap C2
- Bottom of the set resistor R1



Wires and Labels and Nets – Oh My!

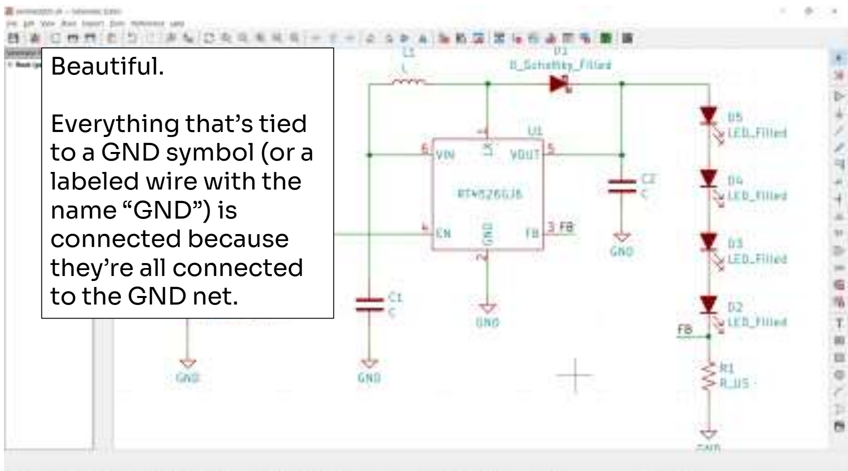


Great! Now wire them.

Fun fact: If you place symbols such that their pins are touching, and then move one of the symbols, a wire will automatically be created between those pins.

Not necessary to

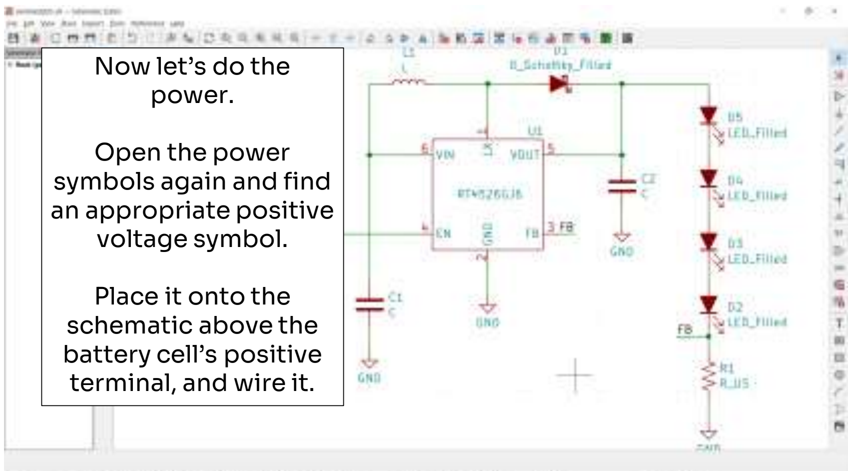
Wires and Labels and Nets – Oh My!



Beautiful.

Everything that's tied to a GND symbol (or a labeled wire with the name "GND") is connected because they're all connected to the GND net.

Wires and Labels and Nets – Oh My!

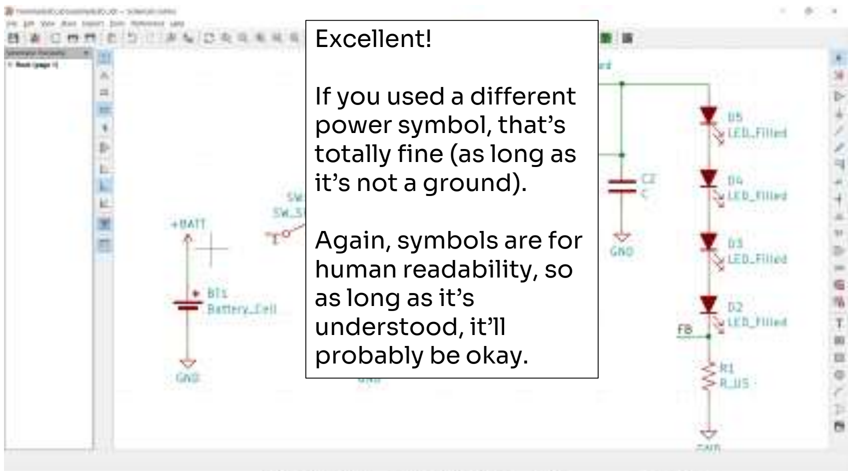


Now let's do the power.

Open the power symbols again and find an appropriate positive voltage symbol.

Place it onto the schematic above the battery cell's positive terminal, and wire it.

Wires and Labels and Nets – Oh My!



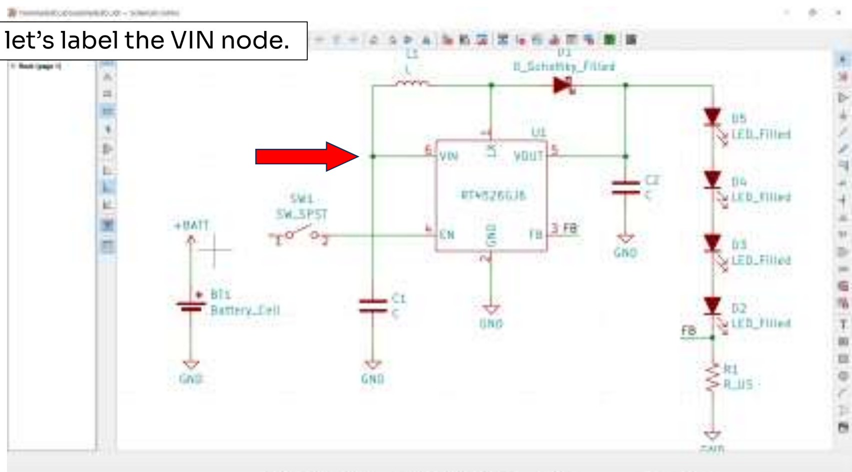
Excellent!

If you used a different power symbol, that's totally fine (as long as it's not a ground).

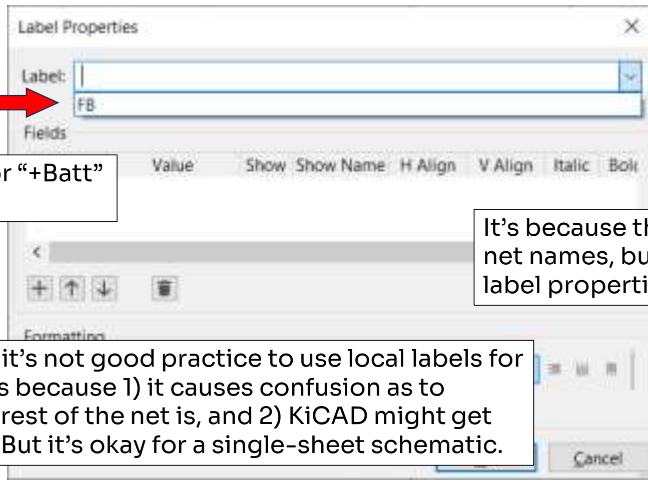
Again, symbols are for human readability, so as long as it's understood, it'll probably be okay.

Wires and Labels and Nets – Oh My!

Now let's label the VIN node.



Wires and Labels and Nets – Oh My!



Why isn't "GND" or "+Batt" listed here?

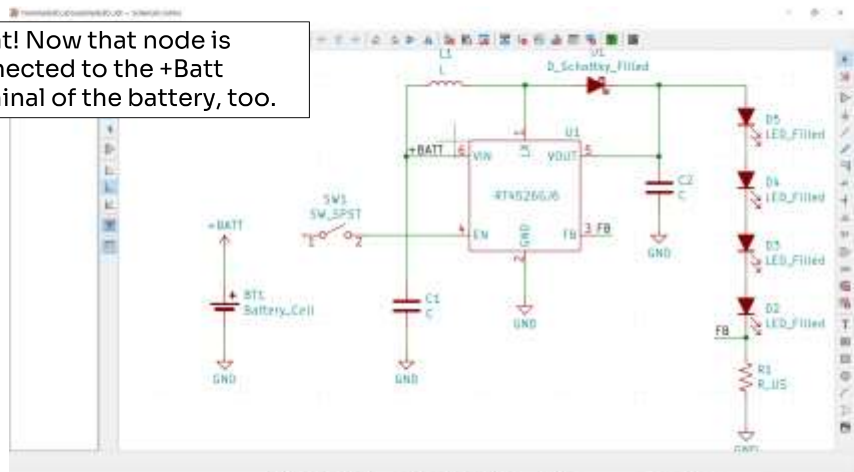
It's because those are *global* net names, but this is the *local* label properties window.

Generally, it's not good practice to use local labels for global nets because 1) it causes confusion as to where the rest of the net is, and 2) KiCAD might get confused. But it's okay for a single-sheet schematic.

You might notice in the label properties window that GND and your power net name aren't listed in the drop down. Why?

Wires and Labels and Nets – Oh My!

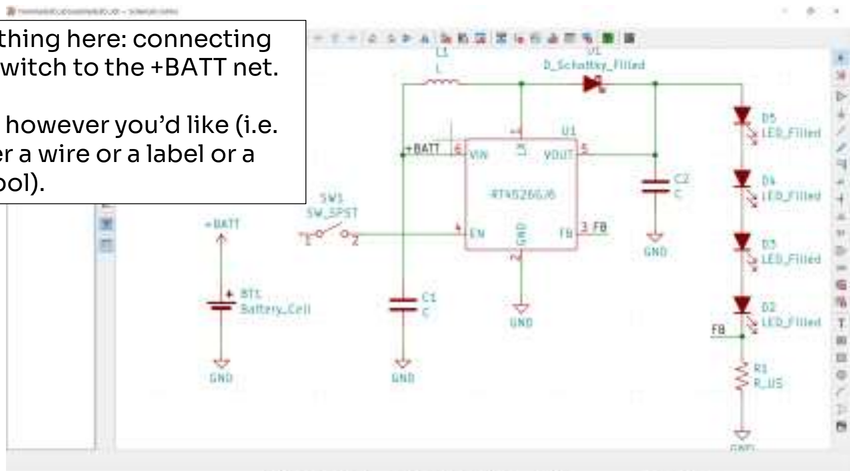
Great! Now that node is connected to the +Batt terminal of the battery, too.



Wires and Labels and Nets – Oh My!

Last thing here: connecting the switch to the +BATT net.

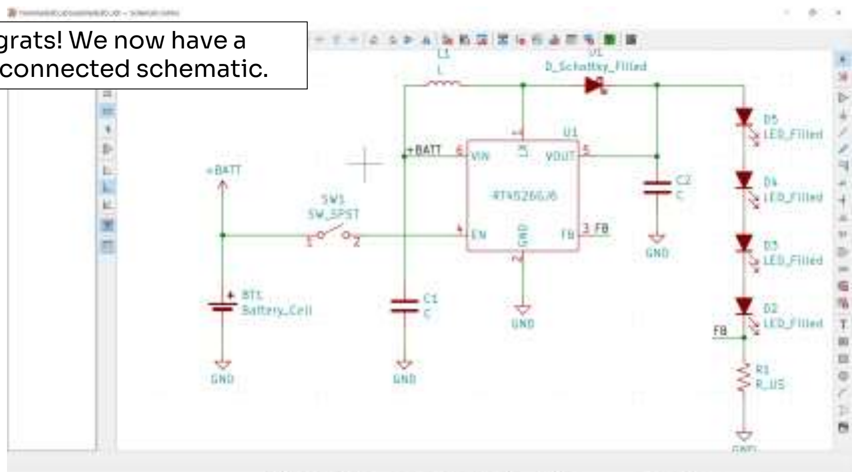
Do it however you'd like (i.e. either a wire or a label or a symbol).

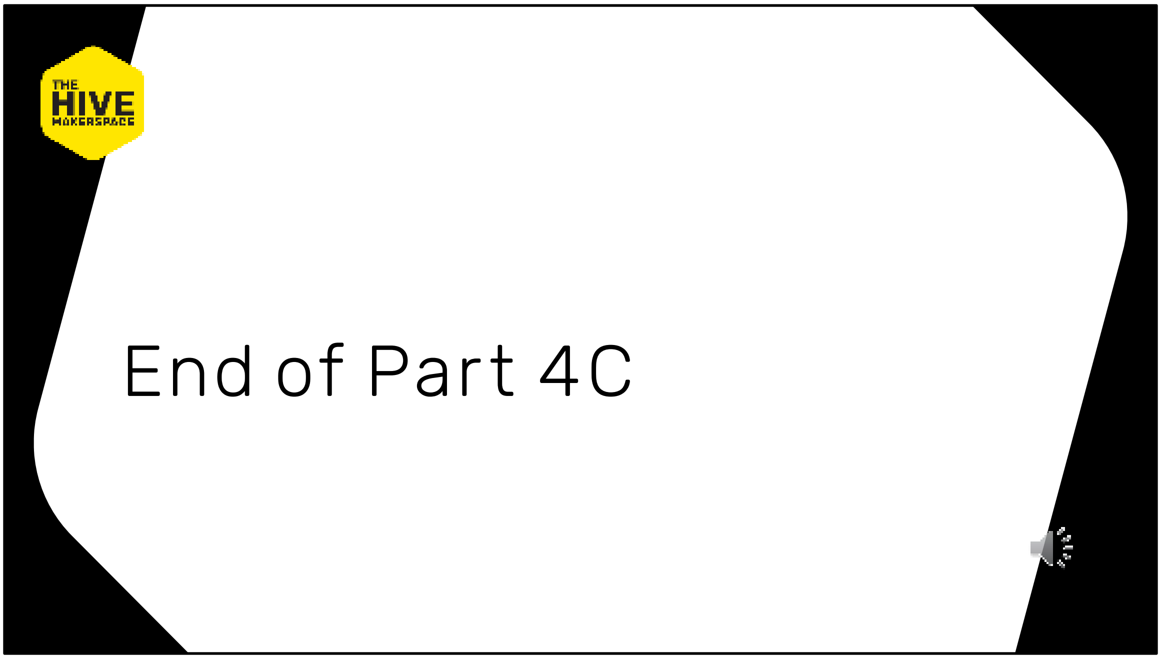


Take a second here to make this connection before continuing.

A fully connected schematic!

Congrats! We now have a fully connected schematic.

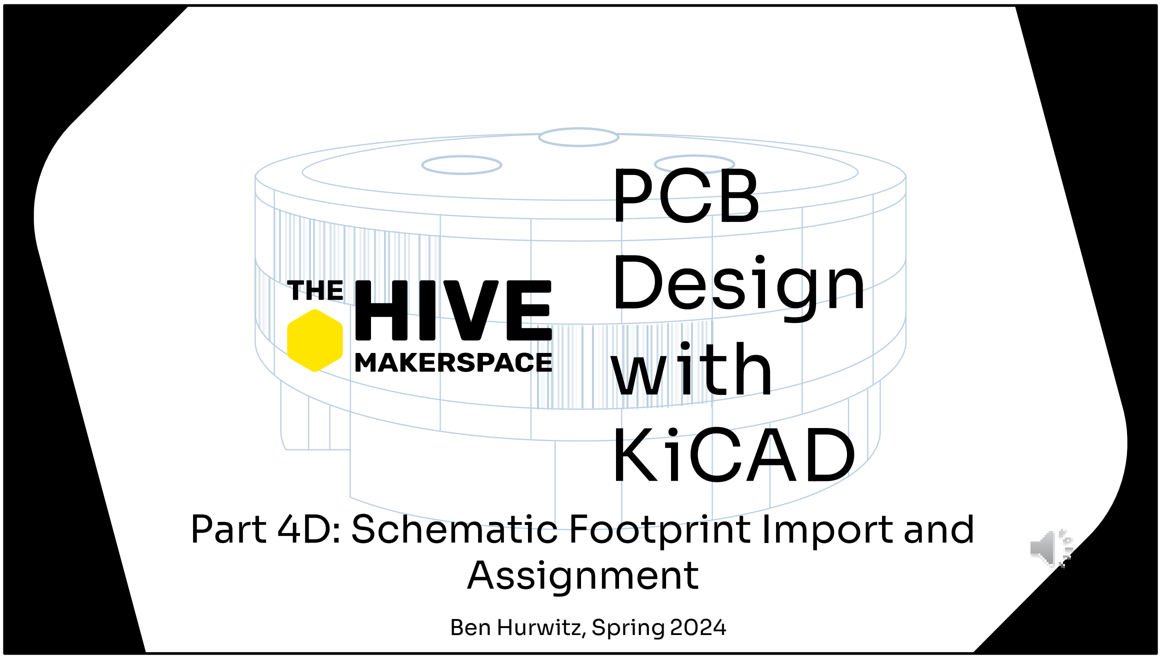




And with that, we end part 4C of this video series on KiCAD and PCB design in which I covered wiring and nets. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In the next part, 4D, we'll look at assigning footprints to the various symbols.

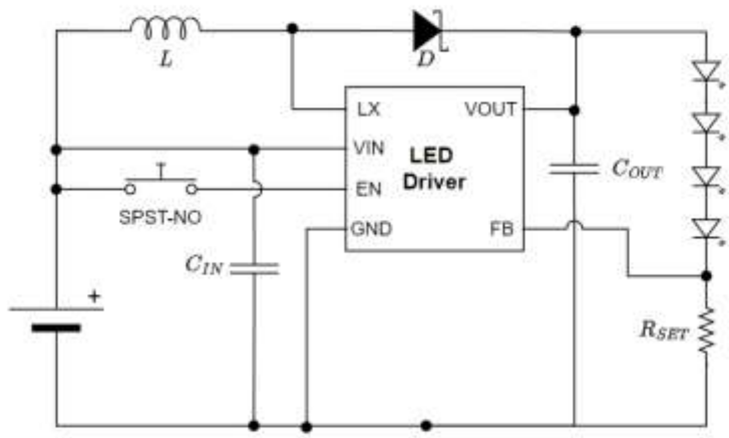
See you there.



Hi, and welcome to part 4D of The Hive's PCB Design With KiCAD series. My name is Ben, and I'll be walking you through this section. Part 4 as a whole will cover the entirety of the schematic creation. In this part 4D, I'll cover assigning footprints to symbols and how to import footprints that you download from the internet. Creation of footprints won't be covered because you should almost never have to do that, and it's far more error-prone than custom symbol creation, though I will point you towards how to do it if necessary. More details about footprint creation can be found in parts 7B and 7C.

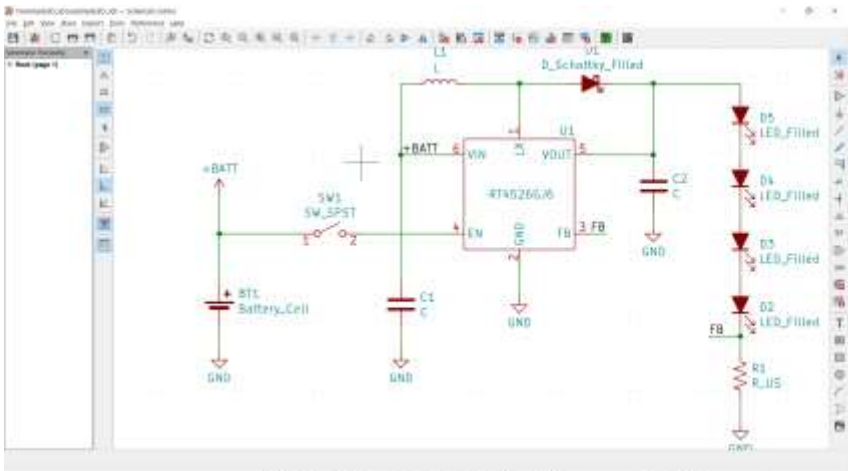
Anyway, onto footprints!

Circuit Reminder



Before we get into KiCAD, just a reminder of the flashlight circuit we're developing. Note that this image was not take from KiCAD, and therefore the symbols and graphics are different from those you are about to see.

Schematic Reminder

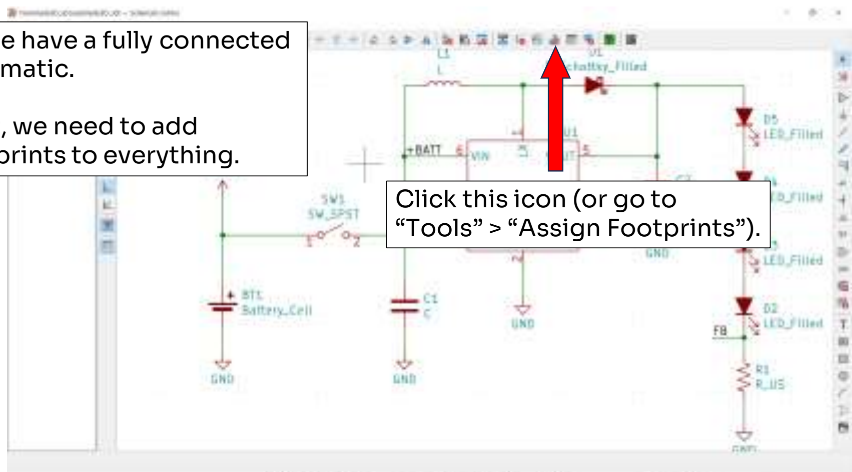


And a reminder of the schematic that we ended part 4C with, fully populated and connected.

Assigning Footprints

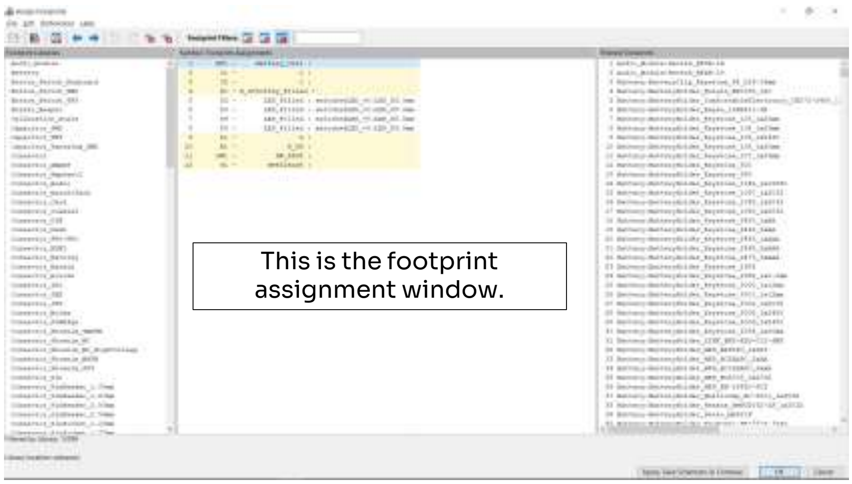
So we have a fully connected schematic.

Next, we need to add footprints to everything.



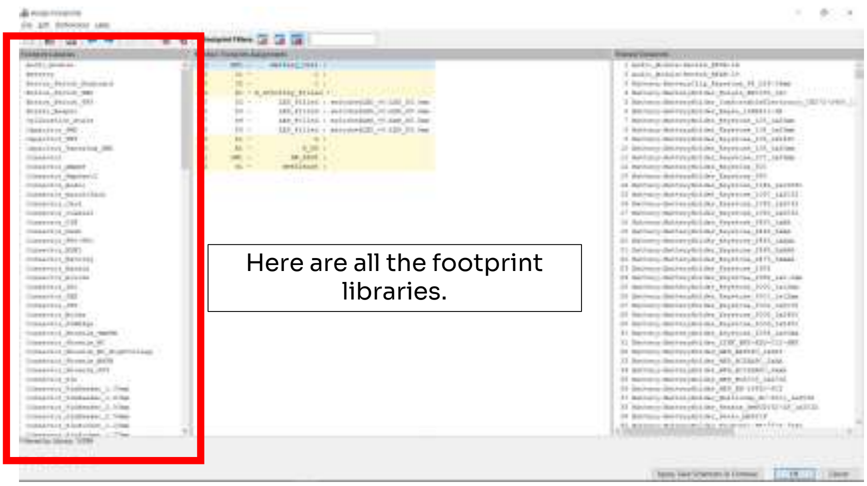
Click this icon (or go to “Tools” > “Assign Footprints”).

Assigning Footprints

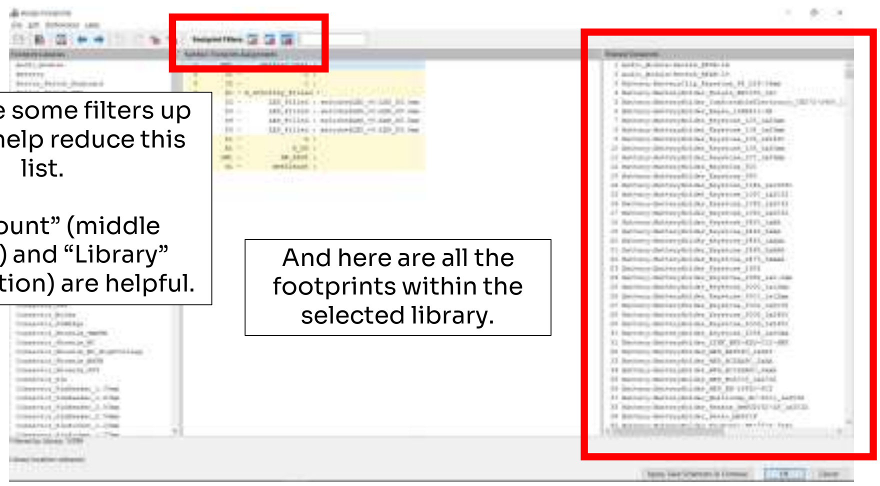


This is the footprint assignment window.

Assigning Footprints



Assigning Footprints

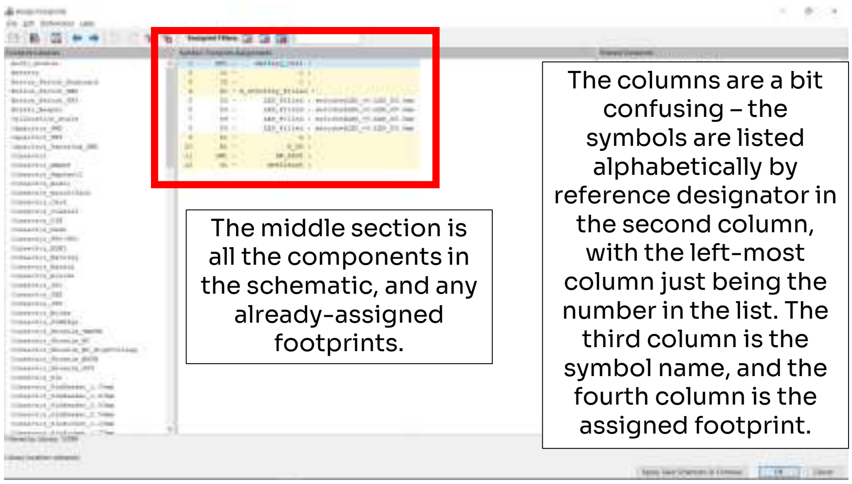


There are some filters up here to help reduce this list.

"Pin count" (middle option) and "Library" (right option) are helpful.

And here are all the footprints within the selected library.

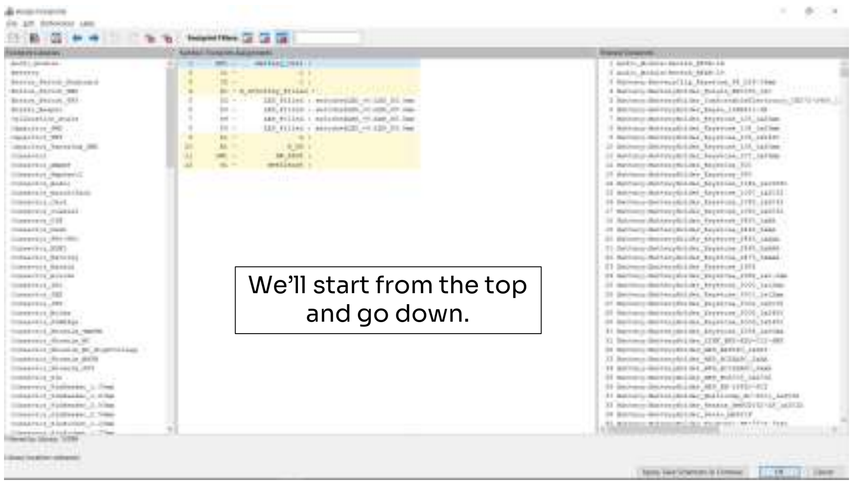
Assigning Footprints



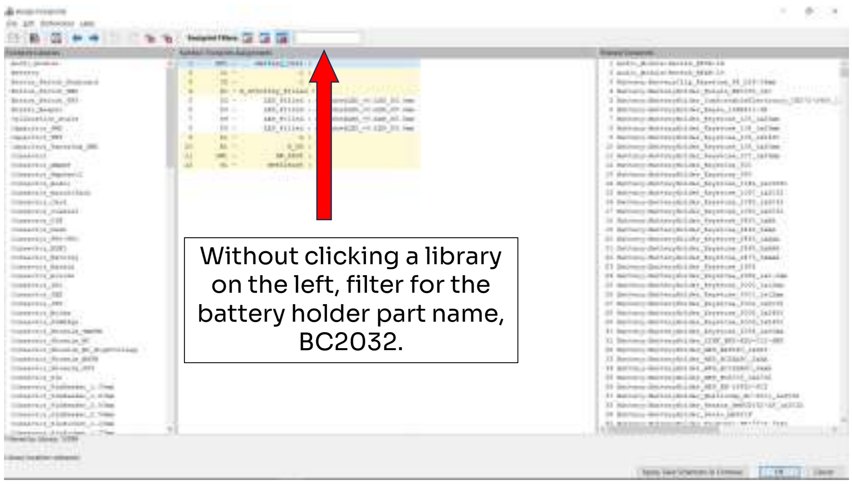
The middle section is all the components in the schematic, and any already-assigned footprints.

The columns are a bit confusing – the symbols are listed alphabetically by reference designator in the second column, with the left-most column just being the number in the list. The third column is the symbol name, and the fourth column is the assigned footprint.

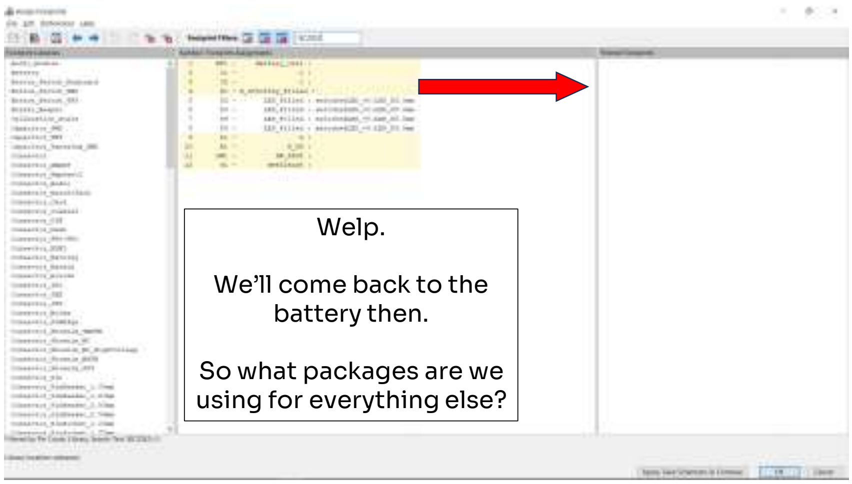
Assigning Footprints



Assigning Footprints



Assigning Footprints



Not necessary to

I didn’t remember, either.

Description	Part Num.	Mounting	Footprint
LED drive IC	RT4526GJ6	SMD	TSOT-23-6 (3.1 x 1.8 x 1 mm)
Battery holder	BC2032-E2	TH	Custom
Switch	TS02-66-70-BK-160-LCR-D	TH	4-TH 6mm x 6mm
Cin, 2.2uF	C3216X5R1C225KT	SMD	1206/3116 (3.1 x 1.6 x 0.55 mm)
Cout, 1uF	C3216X7R1C105KT	SMD	1206/3116 (3.1 x 1.6 x 0.55 mm)
L, 22uH	LBR2518T220M (22uH)	SMD	1008/2518 (2.5 x 1.8 x 1.8 mm)
D	PMEG6030ELPX	SMD	SOD-128 (4 x 2.7 x 1.1 mm)
Rset, 30 Ω	Unknown (from kit)	SMD	1206/3116 (3.1 x 1.6 x 0.55 mm)
LED	C512A-WNN-CZ0B0151	TH	5mm diam, 0.6mm lead holes

Don’t worry, no need to remember these. I’ll remind you of them as we need them.

Note! Blindly using global footprints can leave you exposed to potential issues!

Parts or suppliers may say it’s a standard footprint, but only the datasheet is truth.

It’s up to you as the designer to confirm the dimensions of your parts and footprints.

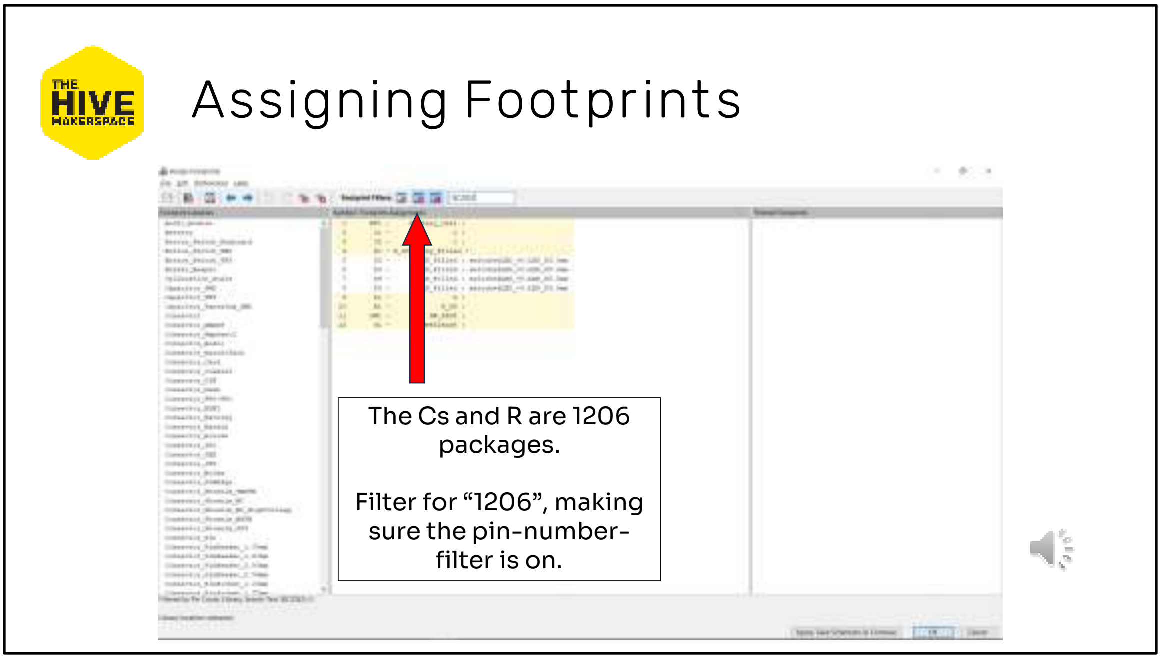
Failure to do so is at your own risk.

Assume, and make an ass out of you and me.

Aside: Package nomenclature

- Package nomenclature is *the worst*, and it's something you'll get familiar with as you design boards.
- There are so many standards, and the variation between some are so tiny. It can be really painful.
- The datasheet's mechanical drawings are the *one and only truth* for package sizes.
- Use caution!

This is a very (very) small subset of package families.

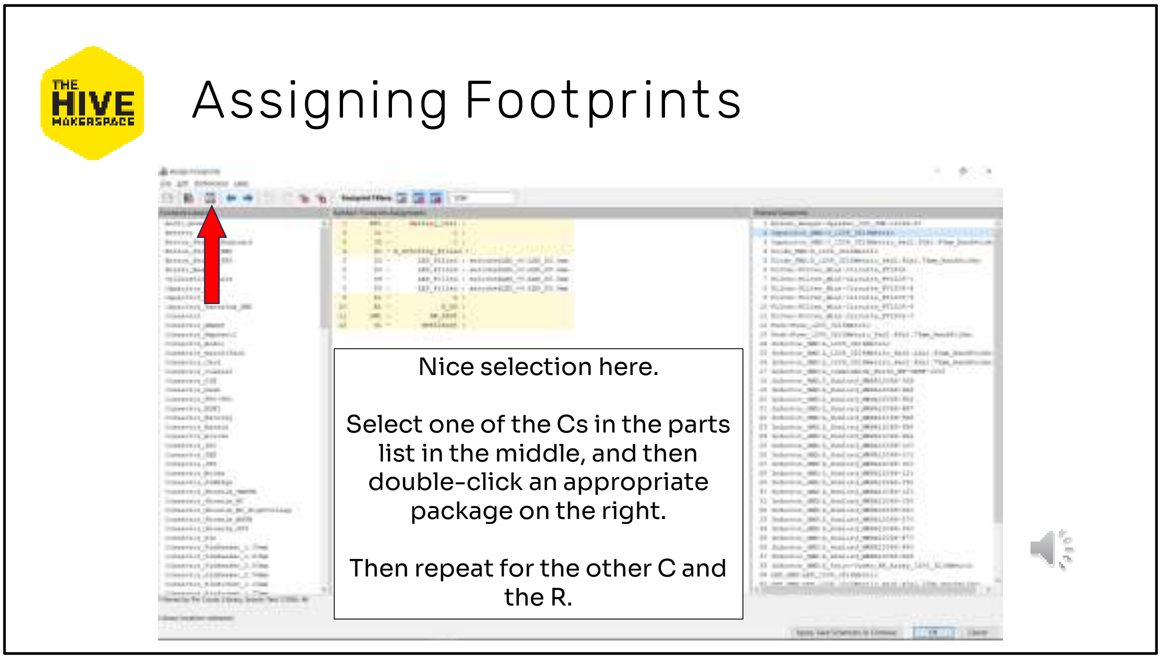


Back to assigning footprints.

If you’re unsure about which library to find footprints, or you know exactly the footprint you need, the filter box is the best way to limit your options.

The leftmost filter option uses some keywords in the symbol to narrow down the options. The middle option filters by number of pins in the symbol. The right one shows you only footprints in the libraries you’ve selected on the left, if any. I find the middle one most useful when I don’t know which library to look in, and the right one only when I know the library. I’ve selected those two in this screen here.

Then we can filter by the 1206 package size that the capacitors and the resistor are to find those.

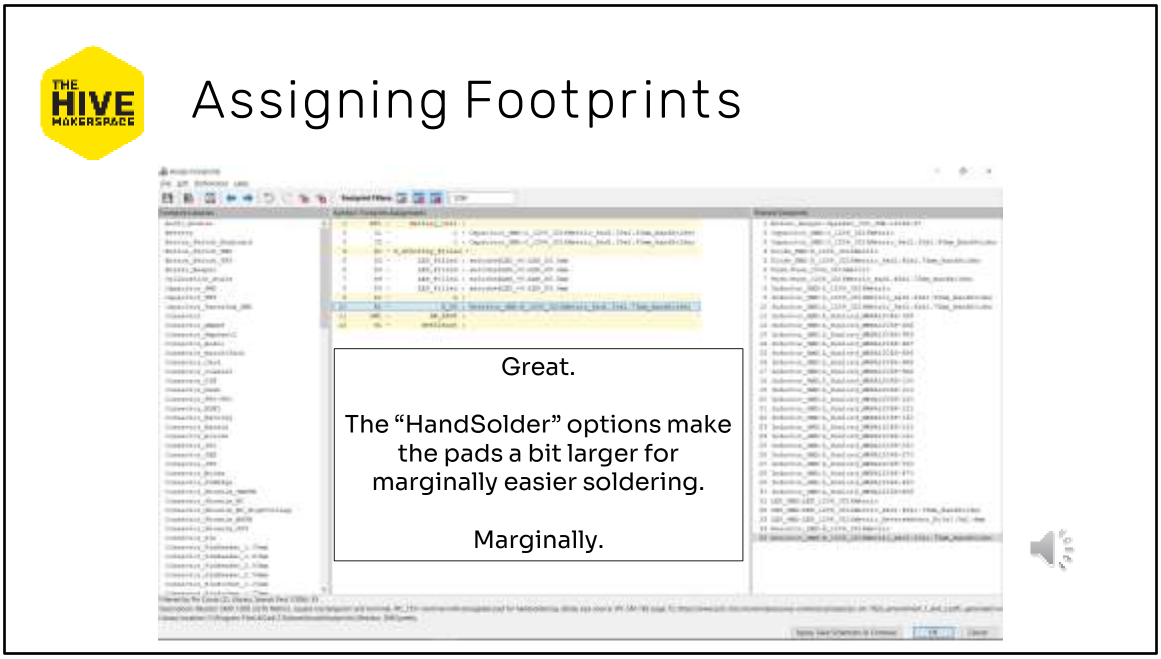


There are a number of options on the right here. Each footprint is labeled with it’s library on the left of the colon, and the footprint name on the right of the colon.

For the capacitors, we want to find something that says “capacitor” or “C” and “1206”. Similarly, for the resistor, something that says “resistor” or “R” and “1206”.

Pick the component in the middle pane to assign a footprint to, then double-click the footprint on the right to assign it.

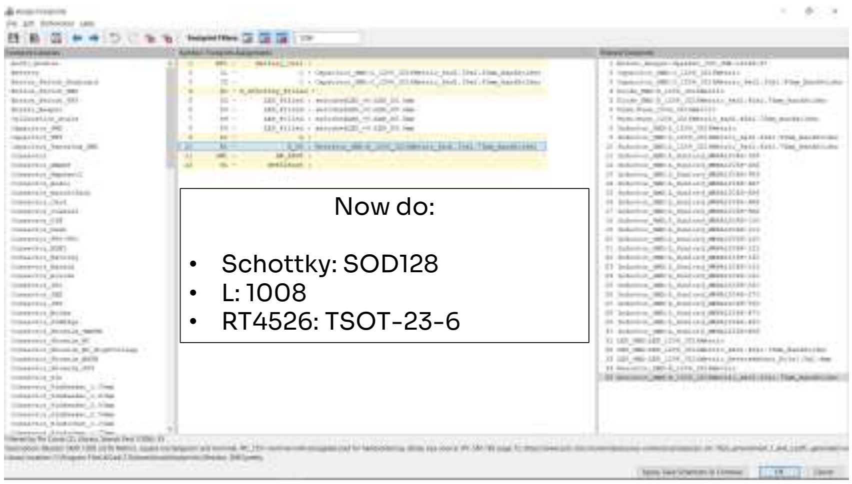
*You can view the footprints themselves through this icon here to access the footprint viewer, from which you can also view its 3D model.



Hopefully you found the C_1206 and R_1206 footprints in the Capacitor SMD and Resistor SMD libraries, respectively, and assigned them like so.

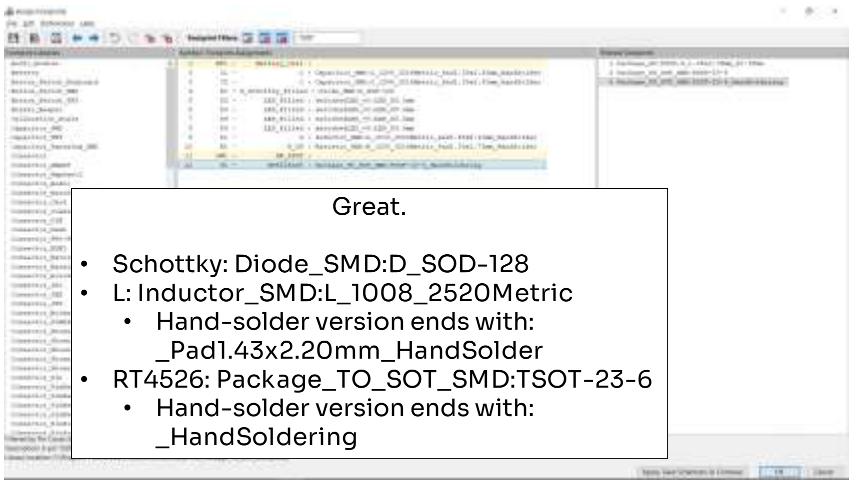
Note that the "HandSolder" version of the footprints uses a slightly larger footprint that supposedly makes them easier to hand solder. It's a pretty marginal change though.

Assigning Footprints



Go ahead and add these three footprints to their schematic components. You can filter directly by the package name, given here. Make an educated guess as to the correct option if you're unsure. Pause the video for a minute to do this on your own before continuing.

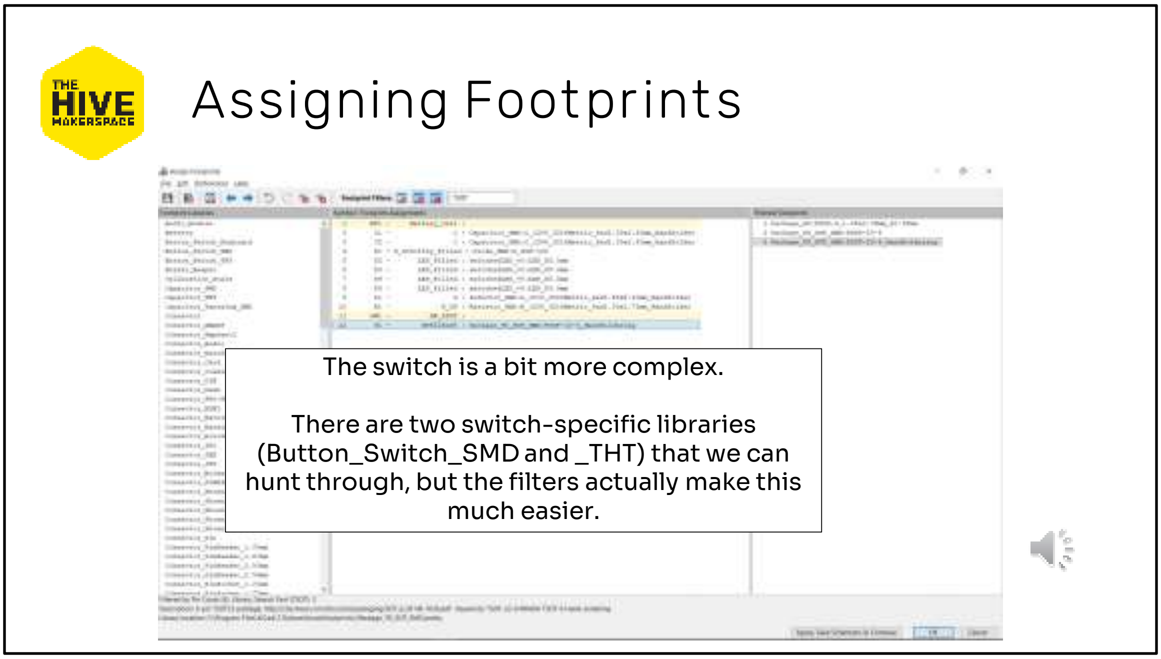
Assigning Footprints



Great.

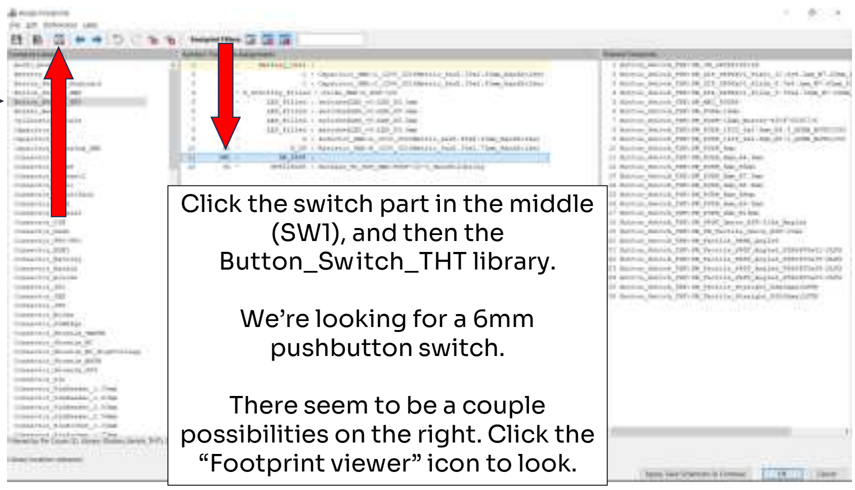
- Schottky: Diode_SMD:D_SOD-128
- L: Inductor_SMD:L_1008_2520Metric
  - Hand-solder version ends with: _Pad1.43x2.20mm_HandSolder
- RT4526: Package_TO_SOT_SMD:TSOT-23-6
  - Hand-solder version ends with: _HandSoldering

Great, hopefully you found those models okay. They're listed here if not.



The switch is slightly more complex because there's no standardize packages for switches. Or, at least, not in the same way as there are for passives and ICs.

Assigning Footprints

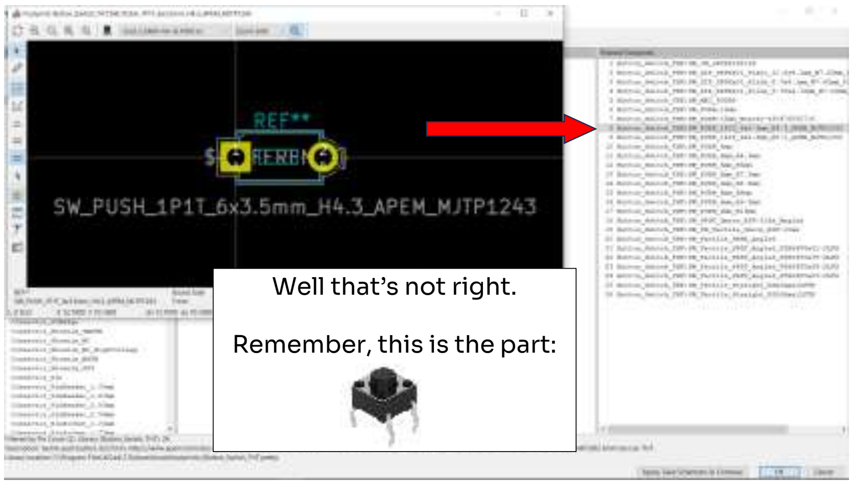


Click the switch part in the middle (SW1), and then the Button_Switch_THT library.

We're looking for a 6mm pushbutton switch.

There seem to be a couple possibilities on the right. Click the "Footprint viewer" icon to look.

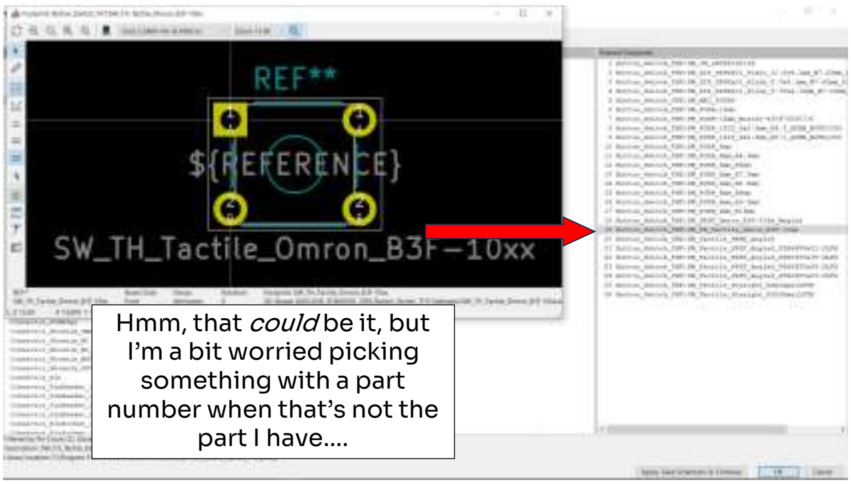
Assigning Footprints



Well that’s not right.

Remember, this is the part:

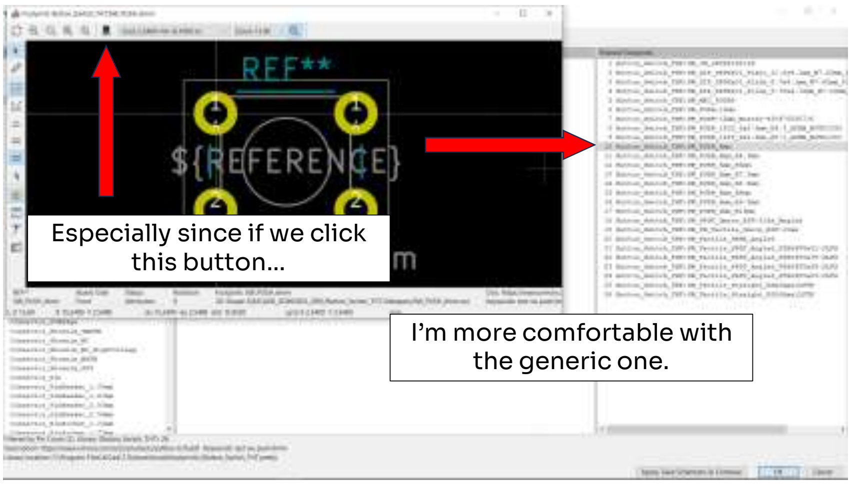
Assigning Footprints

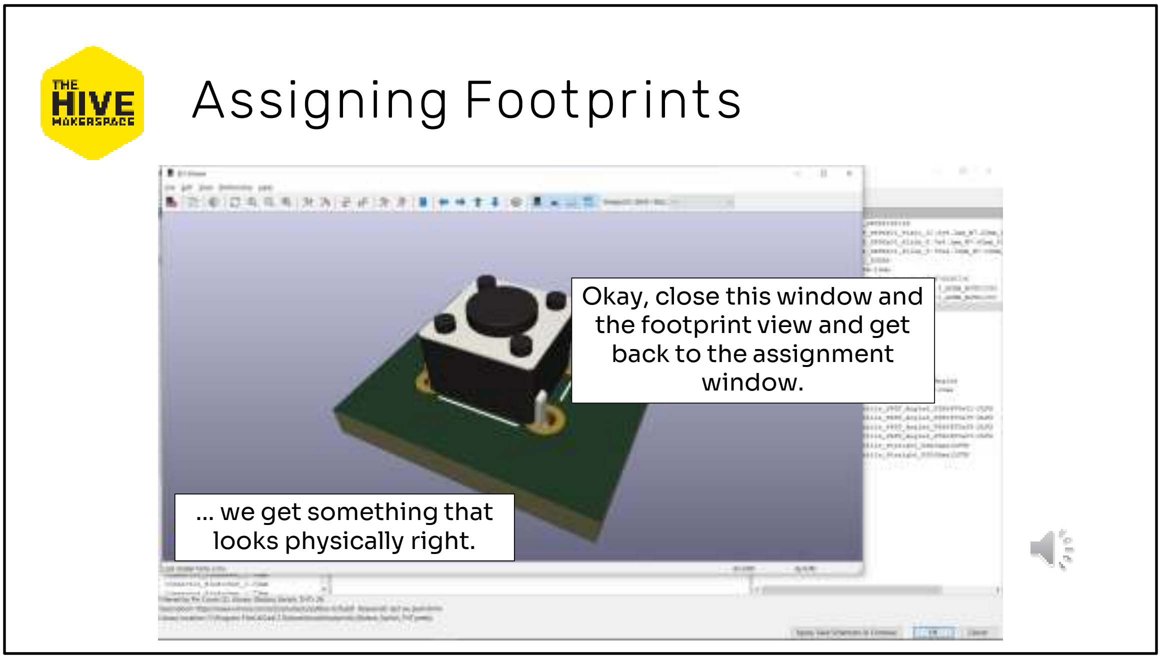


Hmm, that *could* be it, but I'm a bit worried picking something with a part number when that's not the part I have....

Notice that the part number is given at the end, the BF3-10xx bit. Omron is the manufacturer, but you probably wouldn't know that beforehand.

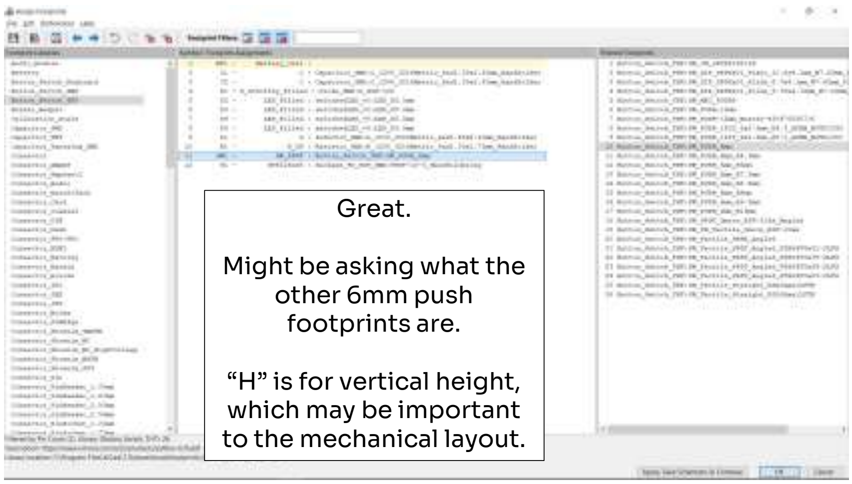
Assigning Footprints

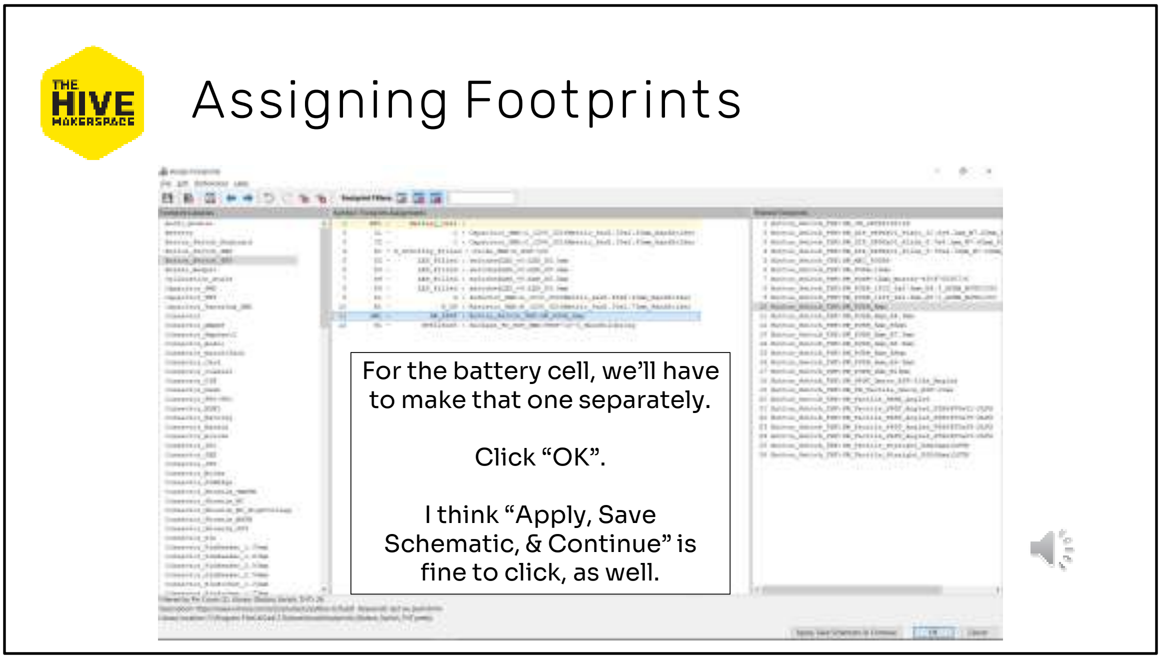




Later, we'll go over a method to physically verify the footprint, and this selection can be changed later, as well. *

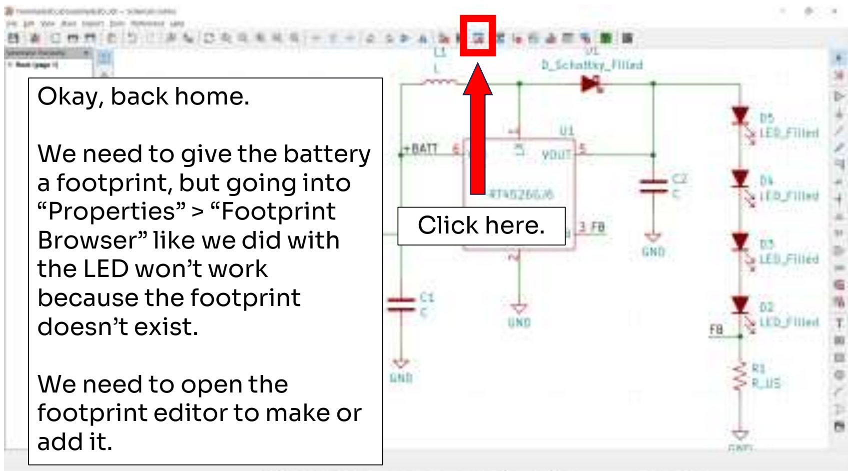
Assigning Footprints





Not necessary to

Creating a Footprint



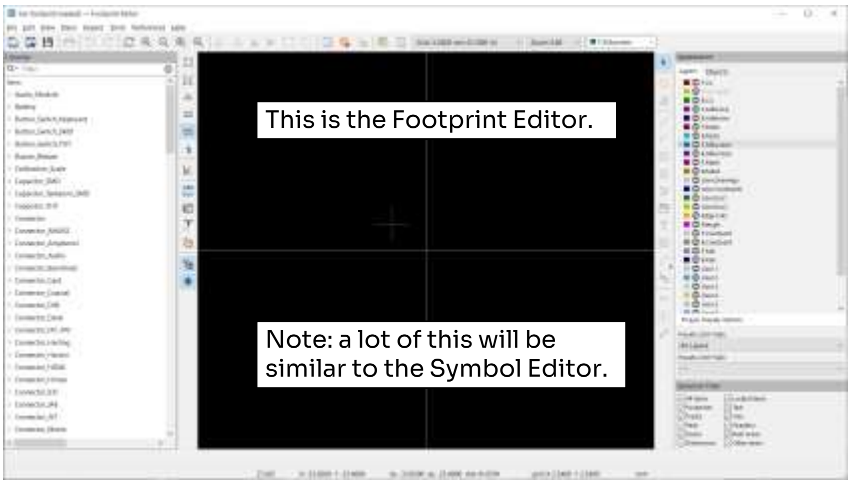
Okay, back home.

We need to give the battery a footprint, but going into “Properties” > “Footprint Browser” like we did with the LED won’t work because the footprint doesn’t exist.

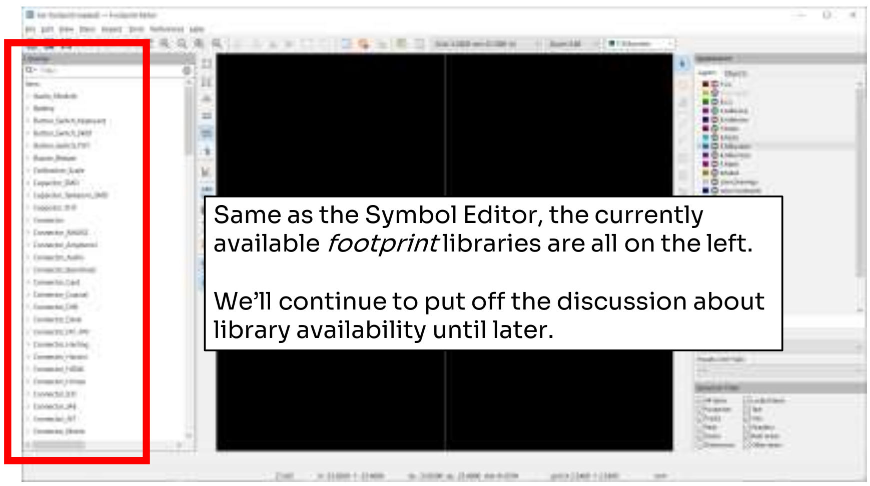
We need to open the footprint editor to make or add it.

Click here.

Footprint Library



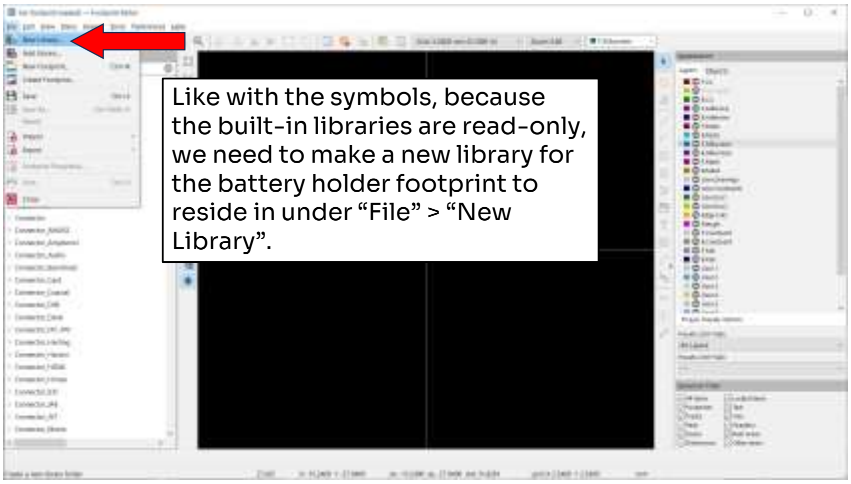
Footprint Library



Same as the Symbol Editor, the currently available *footprint* libraries are all on the left.

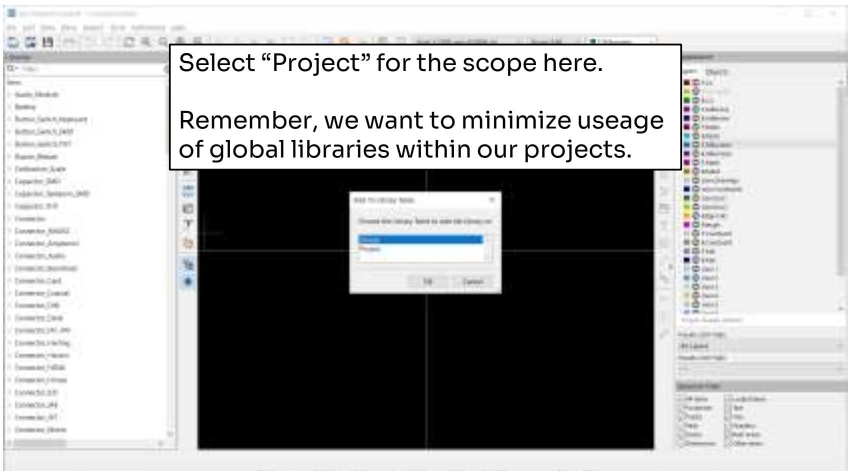
We'll continue to put off the discussion about library availability until later.

Footprint Library

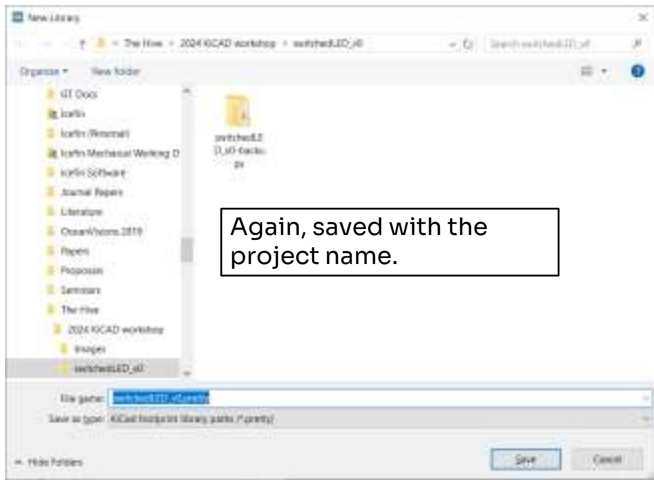


Like with the symbols, because the built-in libraries are read-only, we need to make a new library for the battery holder footprint to reside in under “File” > “New Library”.

Footprint Library

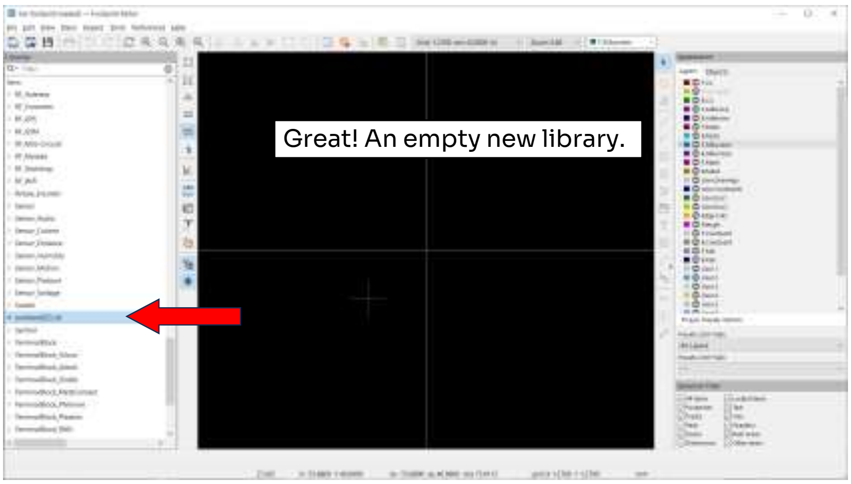


Footprint Library

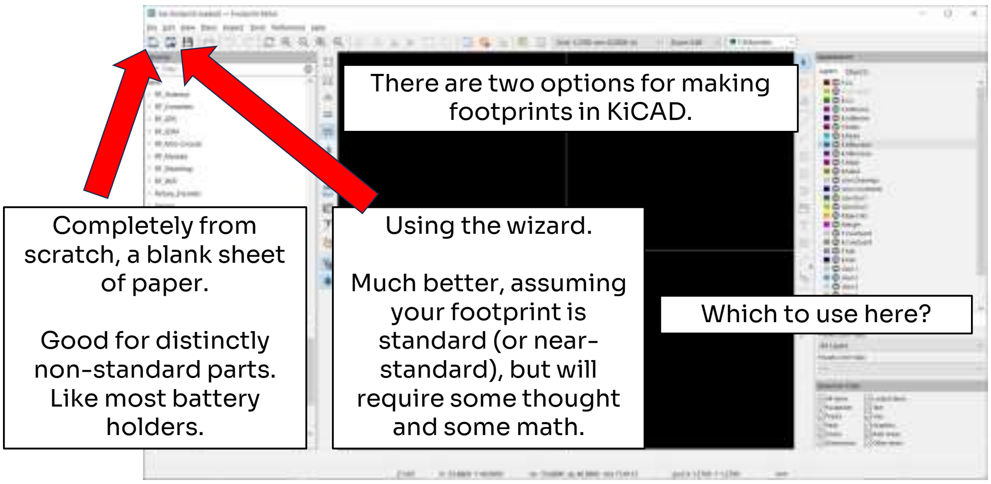


Again, saved with the project name.

Footprint Library



Footprint Library



Footprint Library

Normally, I would just go look for the footprint on UltraLibrarian or SnapMagic first.

But for this tutorial, we'll go look for the datasheet first to see what the footprint looks like, and then check online.

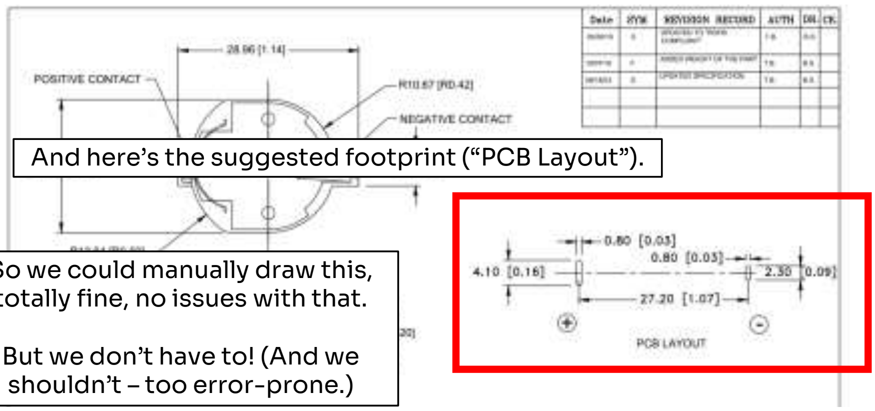
Footprint Library

I had to hunt down the technical drawing for the battery holder on the manufacturer's website because the datasheet link on Digikey gave me their catalogue.

Not helpful.

Footprint Library

Here's the technical drawing (the top half, at least).

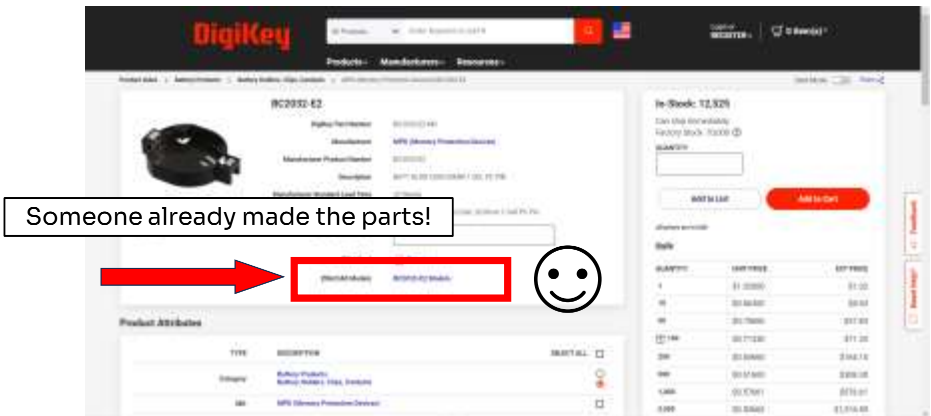


And here's the suggested footprint ("PCB Layout").

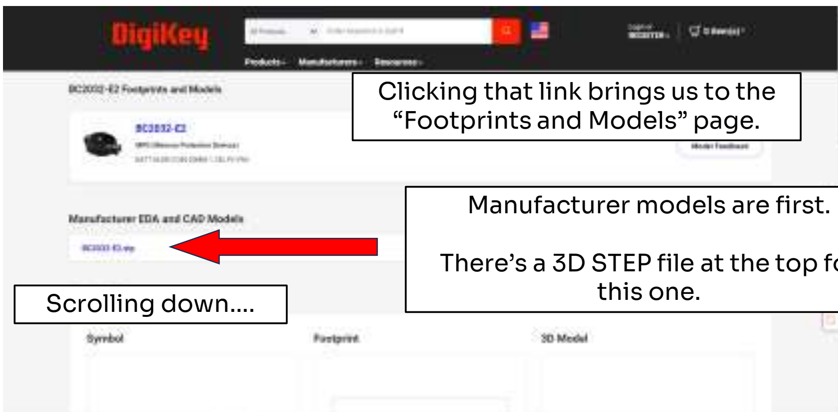
So we could manually draw this, totally fine, no issues with that.

But we don't have to! (And we shouldn't – too error-prone.)

Footprint Library



Footprint Library



Clicking that link brings us to the "Footprints and Models" page.

Manufacturer models are first.

There's a 3D STEP file at the top for this one.

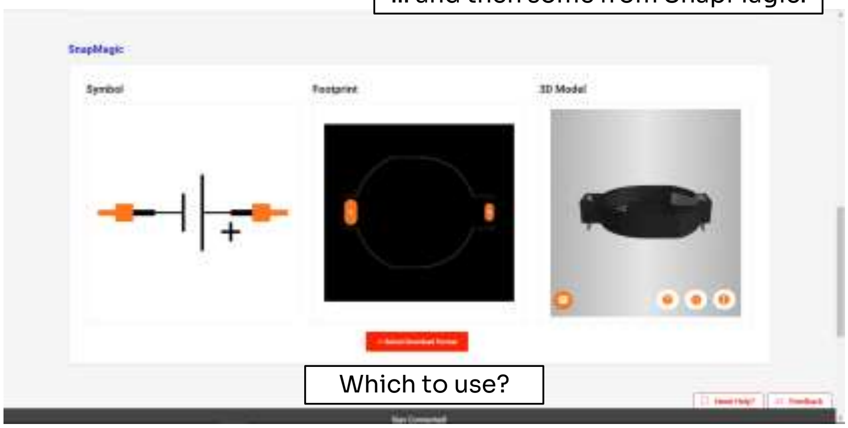
Scrolling down....

Footprint Library

... There's a set of models from UltraLibrarian...

Footprint Library

... and then some from SnapMagic.



Which to use?

Footprint Library

There is no right or wrong answer here generally. Both models are extremely likely to be sufficient.

It's on you to confirm that though!

I'm going to use SnapMagic here because I like the footprint better (and there's a 3D model).

Footprint Library

Click “Select Download Format” and select “KiCAD (V6 or later)”.

(Obviously select a different format if you’re using different software.)

I also selected the 3D CAD model because a 3D view of the designed PCB is useful.

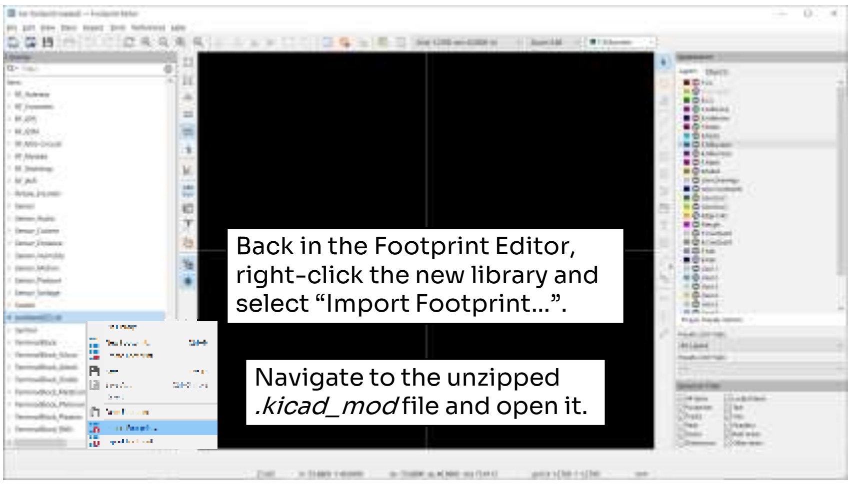
Footprint Library

Unzip the download somewhere accessible.

There will be five five files:

- a *.kicad_sym* symbol library
- a *.kicad_mod* footprint model library
- a *.step* 3D model
- a *.htm* link to a "How to import" webpage
- a license textfile

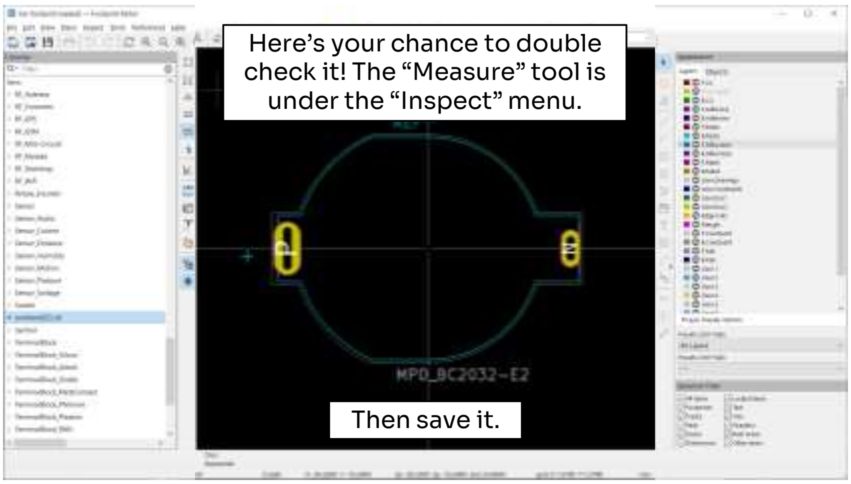
Footprint Library



Back in the Footprint Editor, right-click the new library and select “Import Footprint…”.

Navigate to the unzipped *.kicad_mod* file and open it.

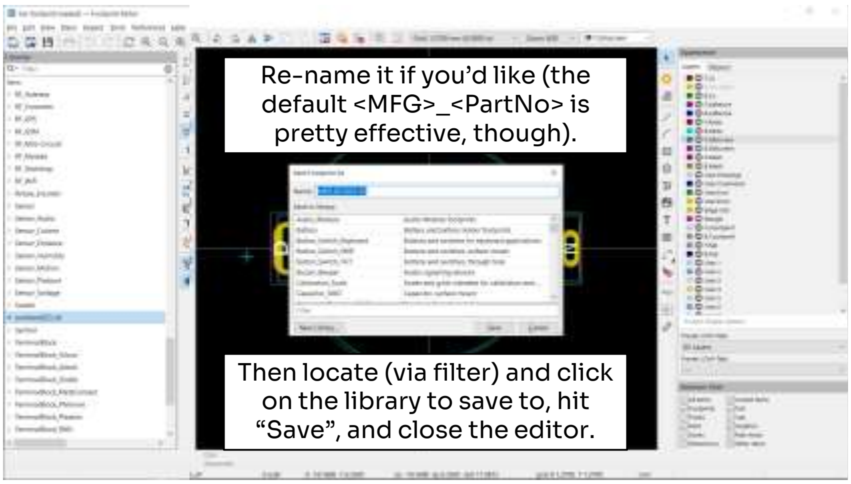
Footprint Library



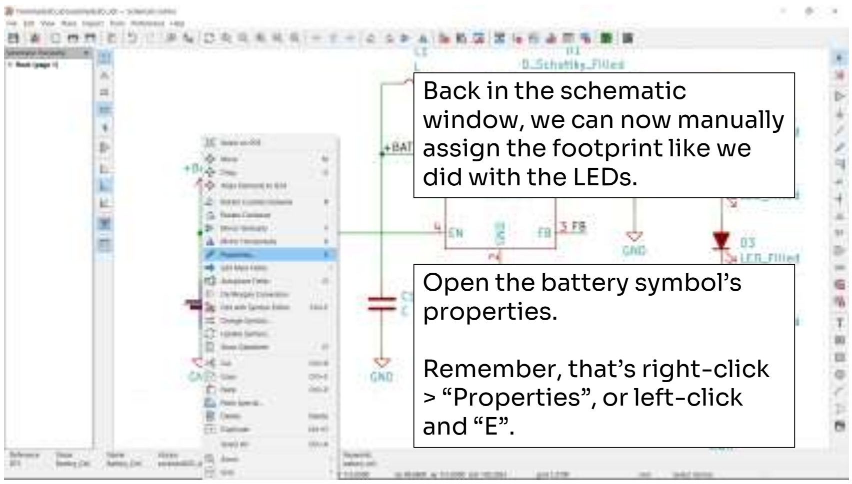
Good opportunity also to check the pin orientation with the datasheet. Is the positive pin actually the left one?

*Once you're confirmed, save it

Footprint Library



Manual Footprint Assignment

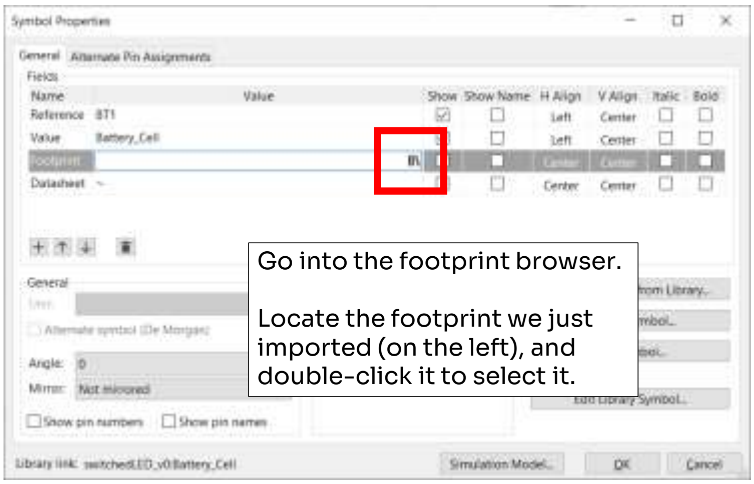


Back in the schematic window, we can now manually assign the footprint like we did with the LEDs.

Open the battery symbol's properties.

Remember, that's right-click > "Properties", or left-click and "E".

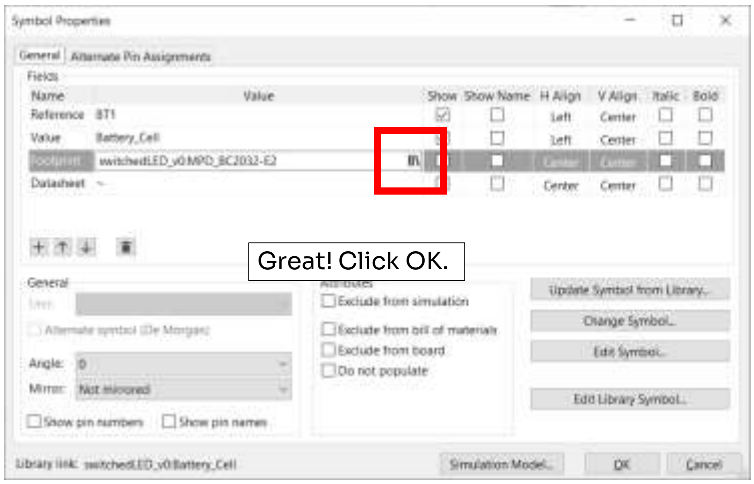
Manual Footprint Assignment



Go into the footprint browser.

Locate the footprint we just imported (on the left), and double-click it to select it.

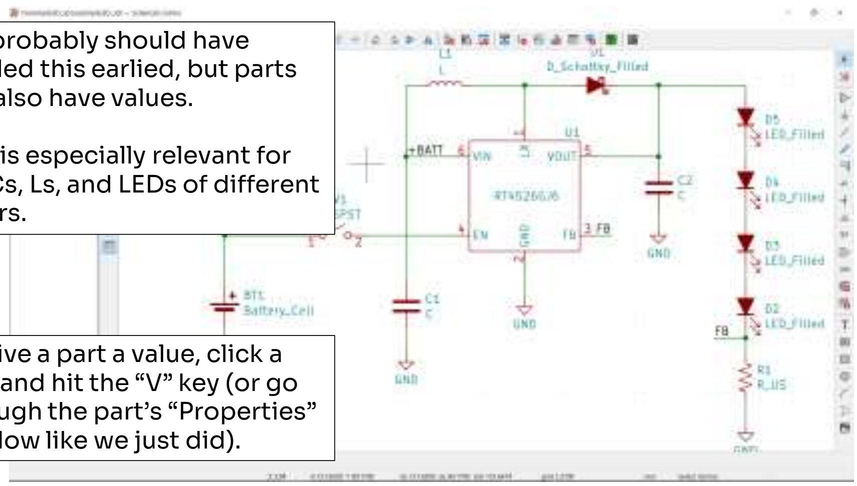
Manual Footprint Assignment



Values

We probably should have tackled this earlied, but parts can also have values.

This is especially relevant for Rs, Cs, Ls, and LEDs of different colors.

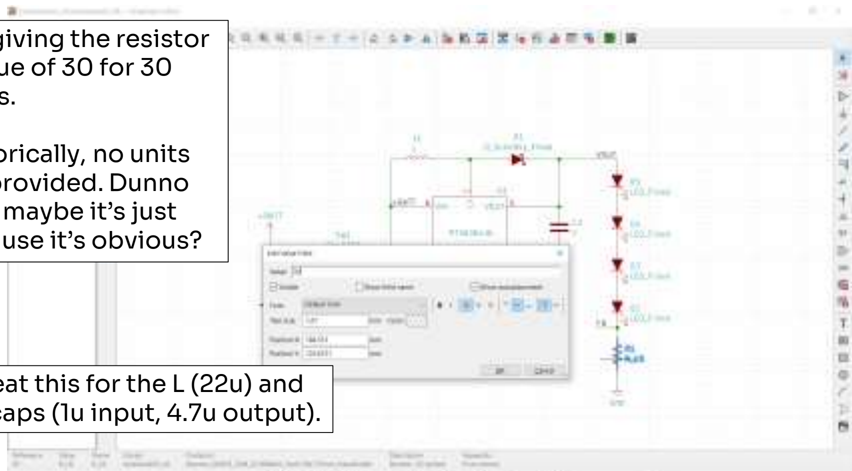


To give a part a value, click a part and hit the “V” key (or go through the part’s “Properties” window like we just did).

Values

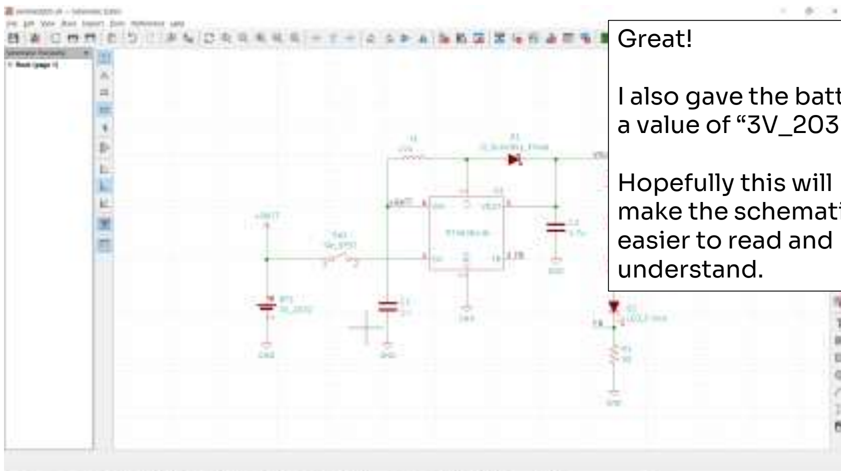
I've giving the resistor a value of 30 for 30 ohms.

Historically, no units are provided. Dunno why, maybe it's just because it's obvious?



Repeat this for the L (22u) and the caps (1u input, 4.7u output).

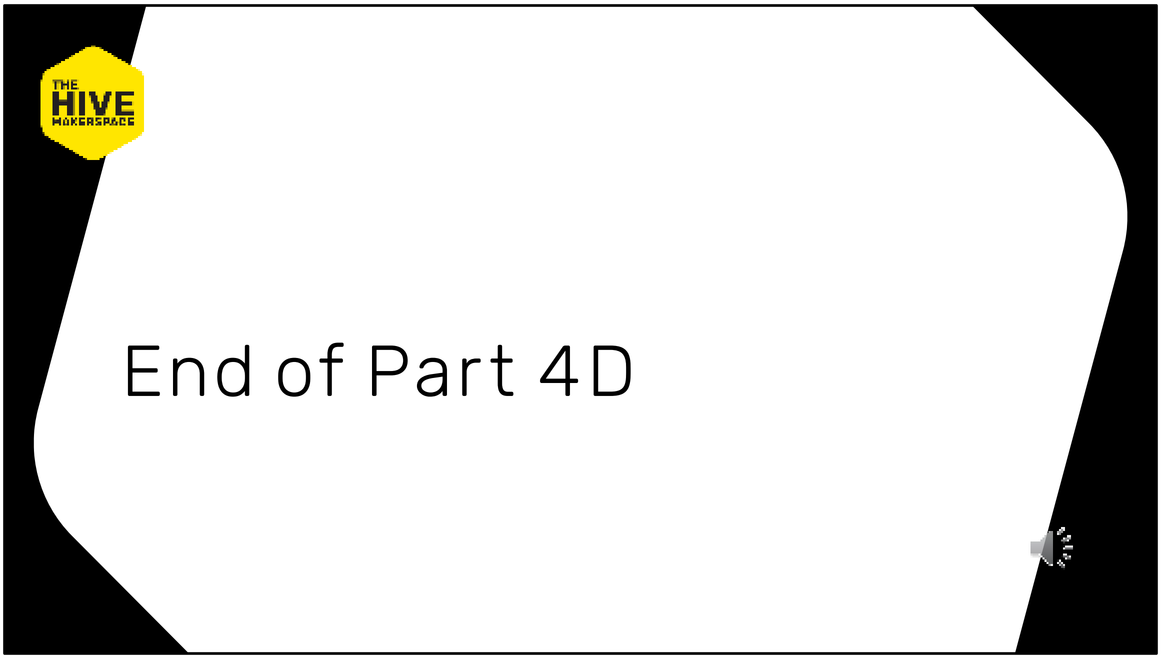
Values



Great!

I also gave the battery a value of “3V_2032”.

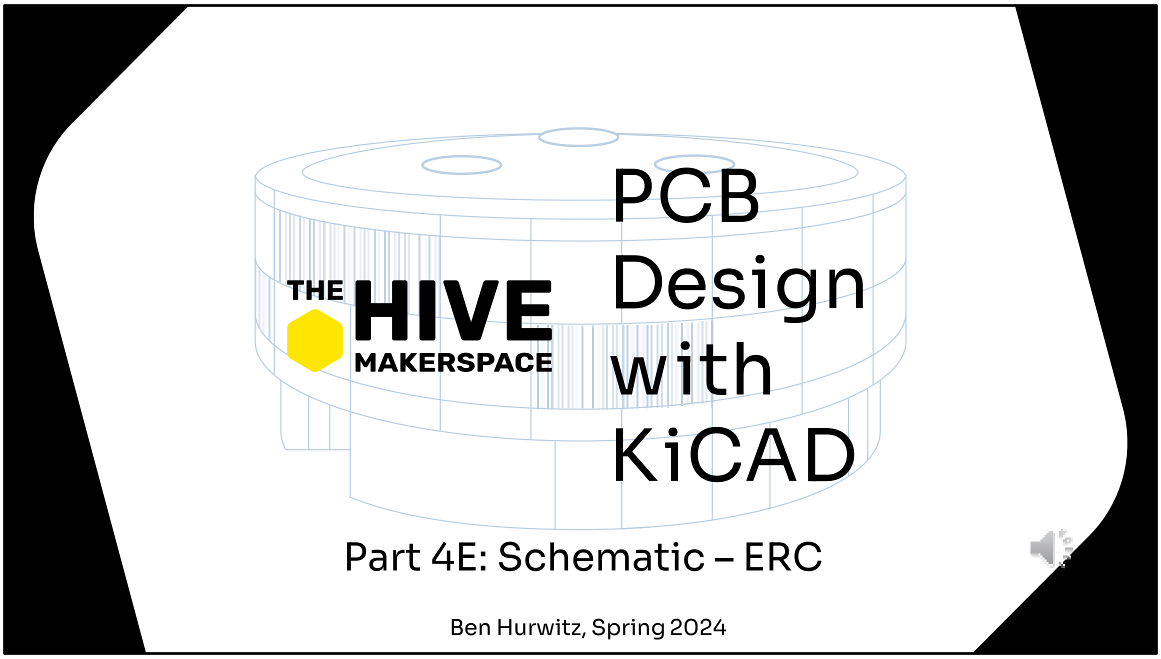
Hopefully this will make the schematic easier to read and understand.



And that ends part 4D of our video series on KiCAD, in which I covered assigning footprints, footprint libraries, importing pre-designed modles, and adding part values.

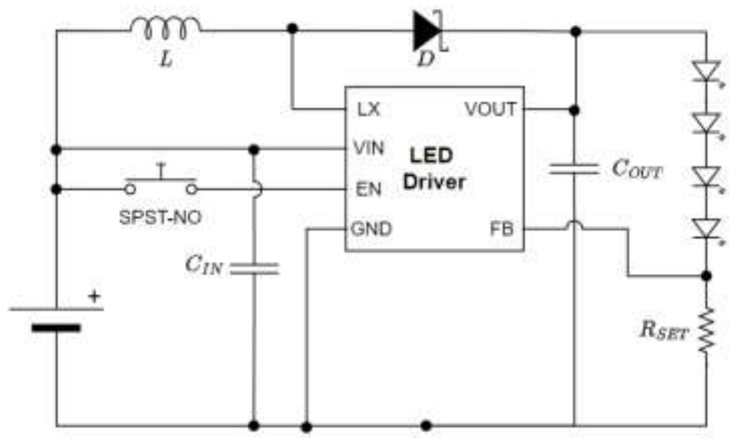
A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In the next segment, part 4E, we'll wrap up the schematic drawing portion of the series with a discussion of ERC and some miscellaneous schematic tools you might want to be vaguely aware of. See you there.



Hi, and welcome to part 4E of The Hive's PCB Design With KiCAD series. My name is Ben, and I'm your host today. Part 4 as a whole has been covering the entirety of the schematic creation, and we'll wrap this portion up with a discussion of the ERC, understanding and dealing with warnings and errors, and some miscellaneous schematic tools you might be interested in being vaguely aware of. Let's get into it.

Circuit Reminder



Before we get into KiCAD, just a reminder of the flashlight circuit we're developing. Note that this image was not take from KiCAD, and therefore the symbols and graphics are different from those you are about to see.

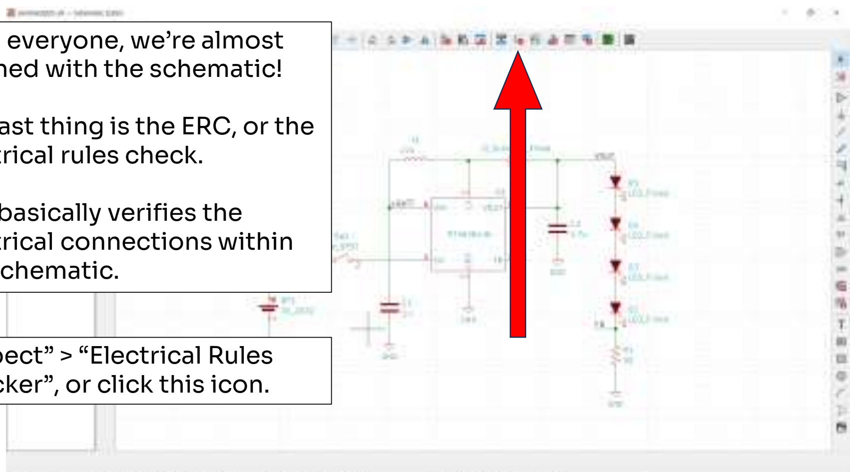
ERC

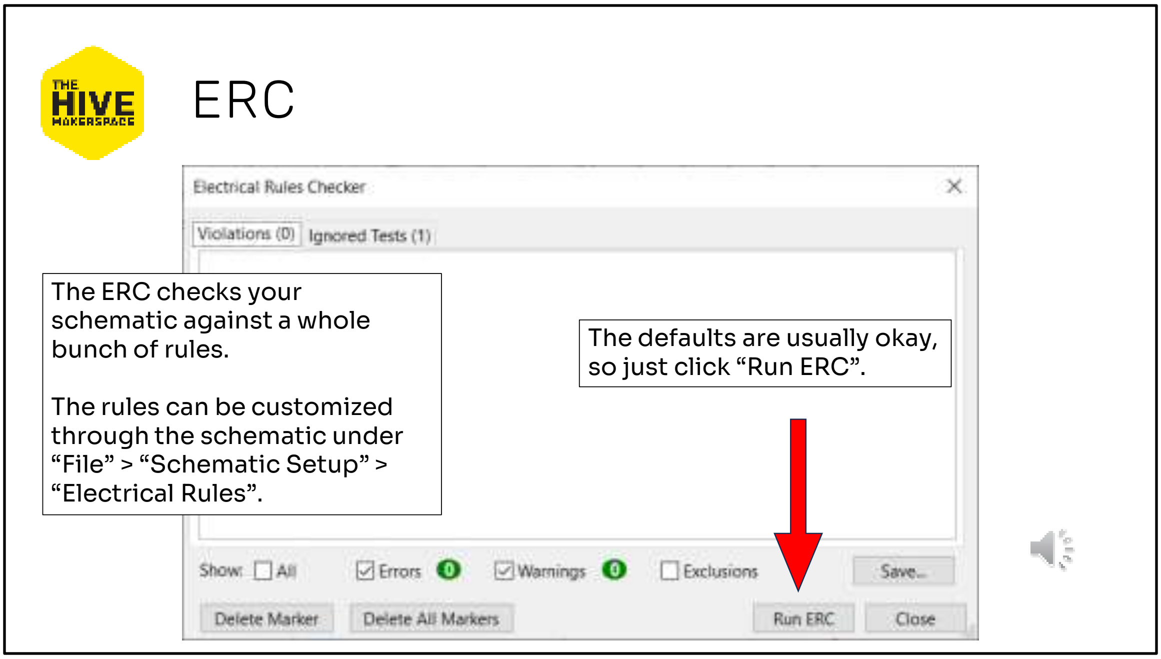
Omg everyone, we're almost finished with the schematic!

The last thing is the ERC, or the electrical rules check.

This basically verifies the electrical connections within the schematic.

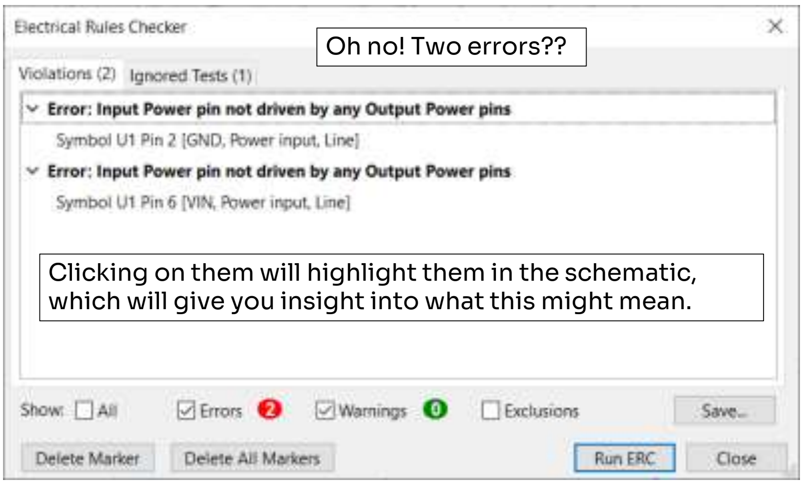
"Inspect" > "Electrical Rules Checker", or click this icon.





The ERC should, ideally, have been run a few times throughout the schematic design process because it can be lengthy and iterative, so it's valuable to see if you've made errors early before they pile up and start building on each other.

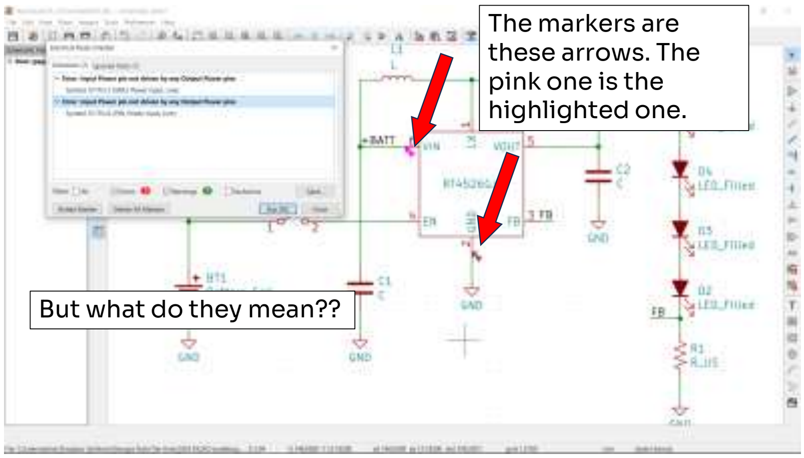
ERC



Oh no! Two errors??

Clicking on them will highlight them in the schematic, which will give you insight into what this might mean.

ERC

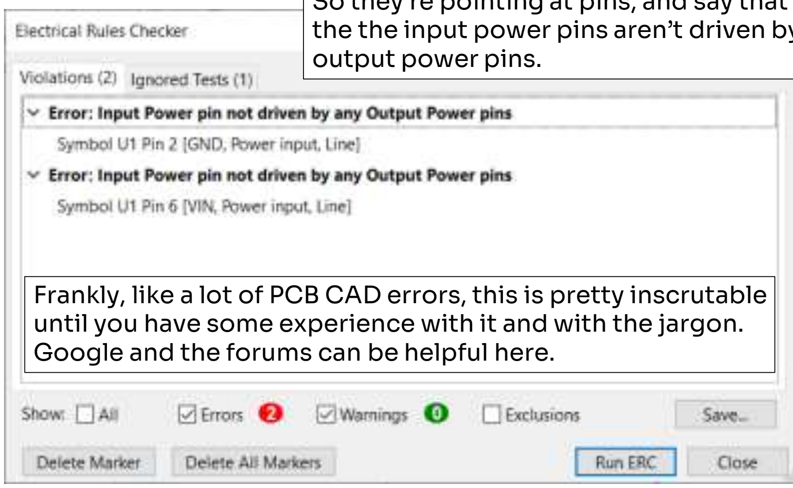


The markers are these arrows. The pink one is the highlighted one.

But what do they mean??

ERC

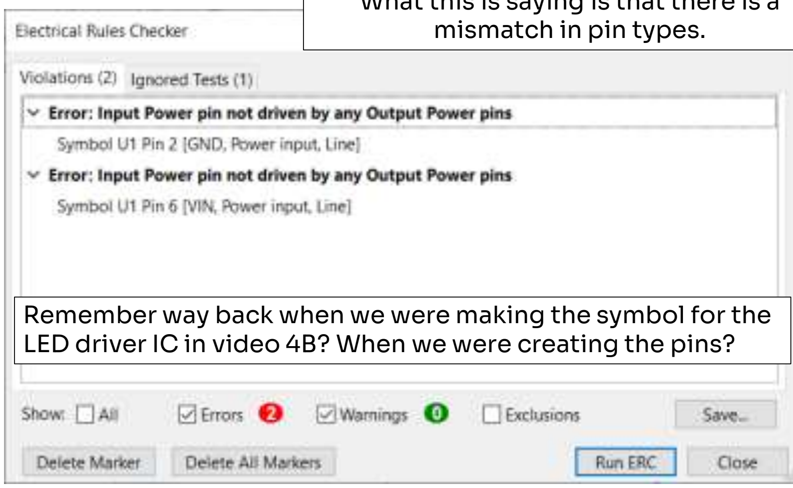
So they're pointing at pins, and say that the the input power pins aren't driven by output power pins.



Frankly, like a lot of PCB CAD errors, this is pretty inscrutable until you have some experience with it and with the jargon. Google and the forums can be helpful here.

ERC

What this is saying is that there is a mismatch in pin types.

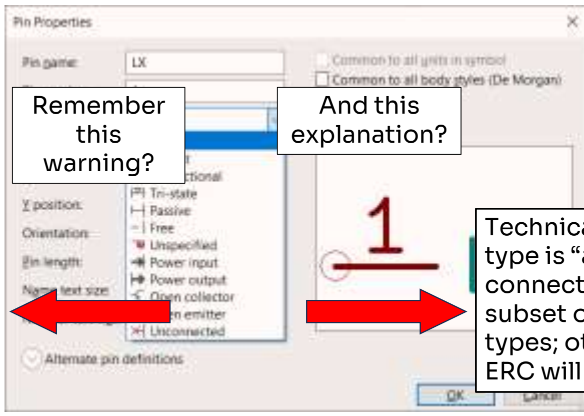


Remember way back when we were making the symbol for the LED driver IC in video 4B? When we were creating the pins?

(For those who don't remember)

If you're not sure what it is, you can either set it to "Bidirectional" or "Unspecified" or "Passive".

The type won't cause the design to fail, but it might cause you headaches later with the ERC.

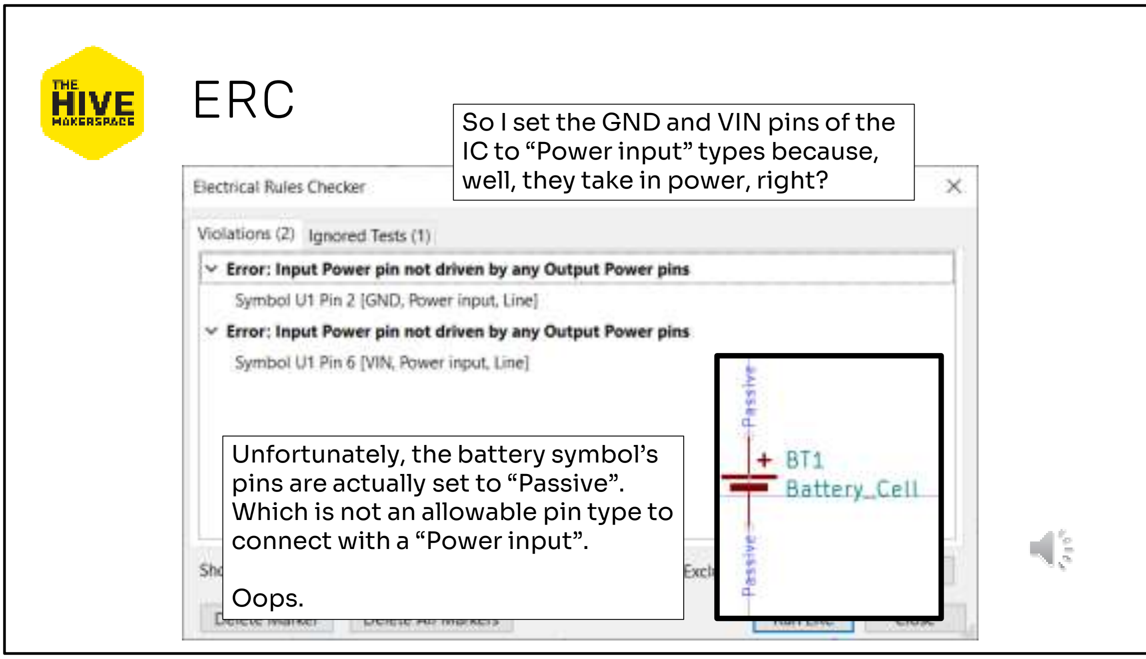


Remember this warning?

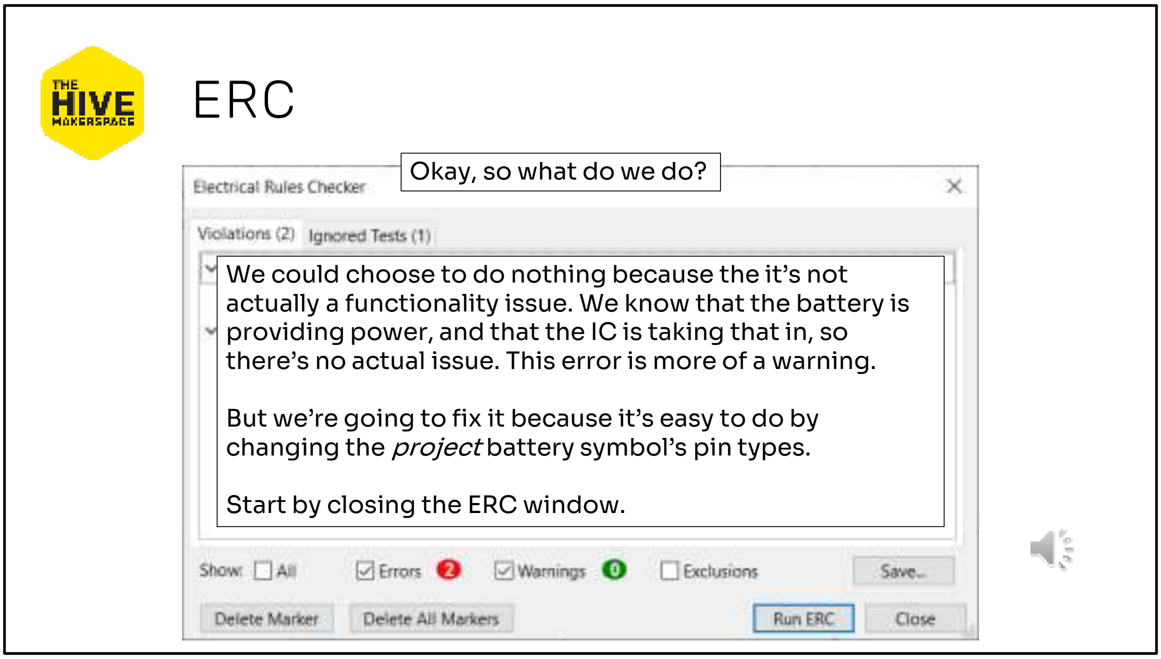
And this explanation?

Technically, each pin type is "allowed" connected to only a subset of other pin-types; otherwise, the ERC will throw an error.

*Remember this warning?
*And this explanation? This probably didn't make any sense back then, but this is exactly what we're seeing.

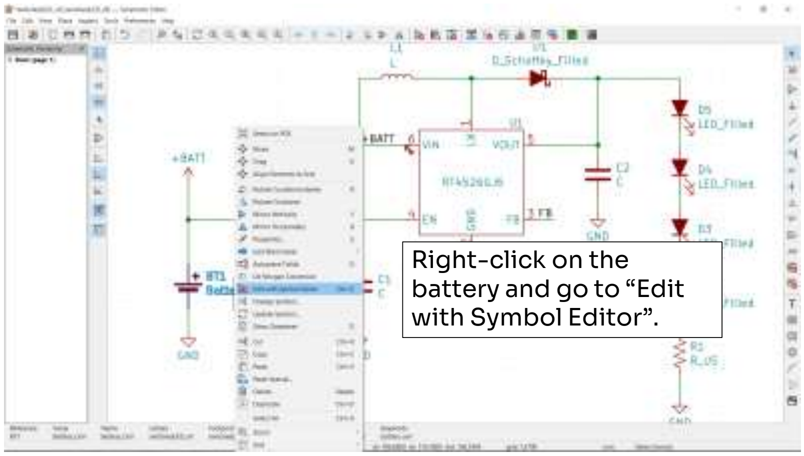


How would you have known this? You wouldn't, necessarily, and I certainly didn't, but the error suggests that whatever is on the other side of the problematic VIN and GND pins (which is what the markers point at) is not correct. So I went and looked at the battery symbol and at it's pin types to discover this mismatch.



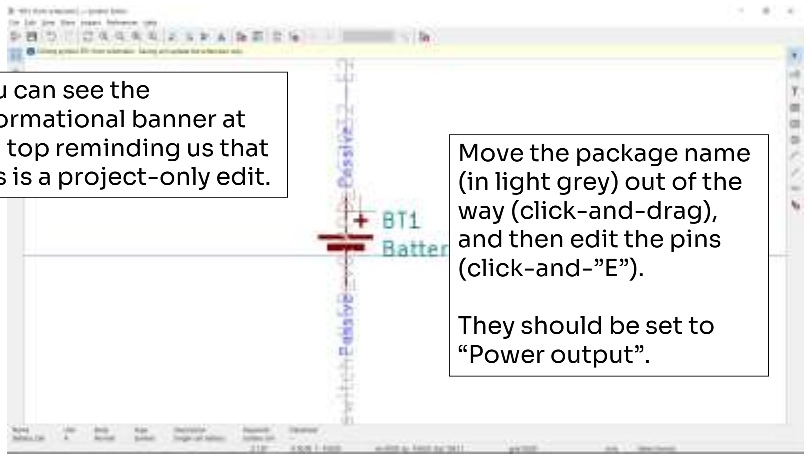
By "project" here, I mean that we're just editing the project's version of the footprint, not the global part model. Therefore, we're not interfering with any other battery symbols in other schematics, though it will update other batteries within this schematic.

ERC



Right-click on the battery and go to "Edit with Symbol Editor".

ERC



You can see the informational banner at the top reminding us that this is a project-only edit.

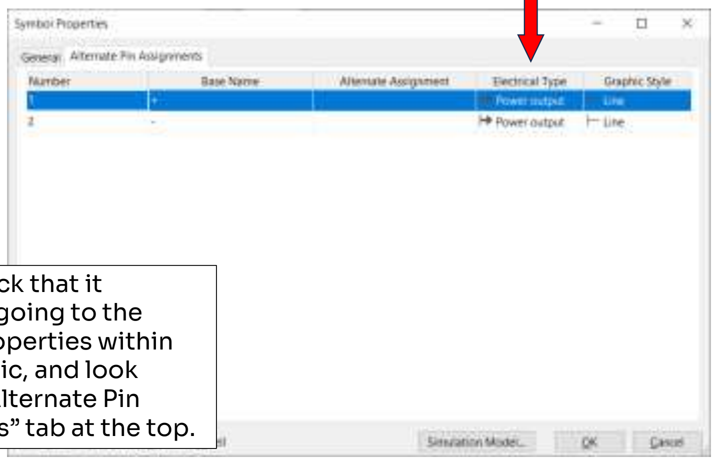
Move the package name (in light grey) out of the way (click-and-drag), and then edit the pins (click-and-”E”).

They should be set to “Power output”.

ERC

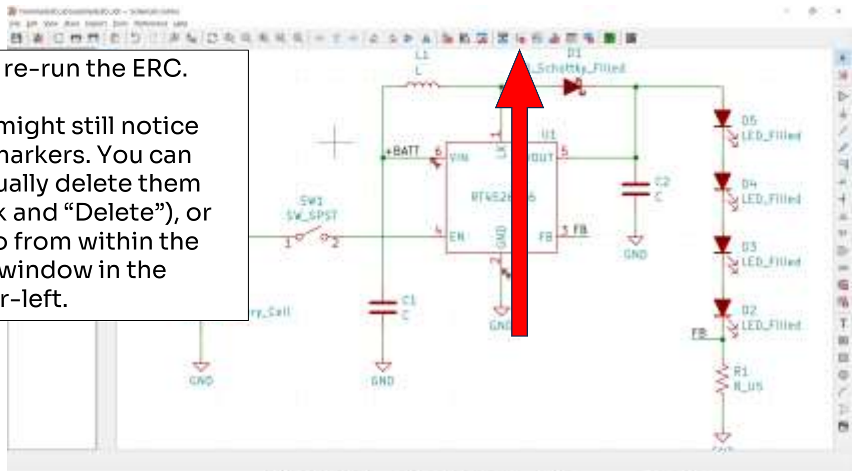
Not necessary to

ERC



You can check that it updated by going to the symbol's properties within the schematic, and look under the "Alternate Pin Assignments" tab at the top.

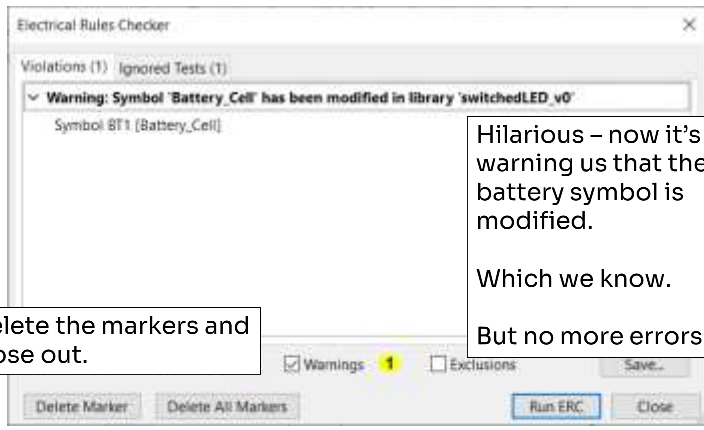
ERC



Now re-run the ERC.

You might still notice the markers. You can manually delete them (click and "Delete"), or do so from within the ERC window in the lower-left.

ERC



Hilarious – now it's warning us that the battery symbol is modified.

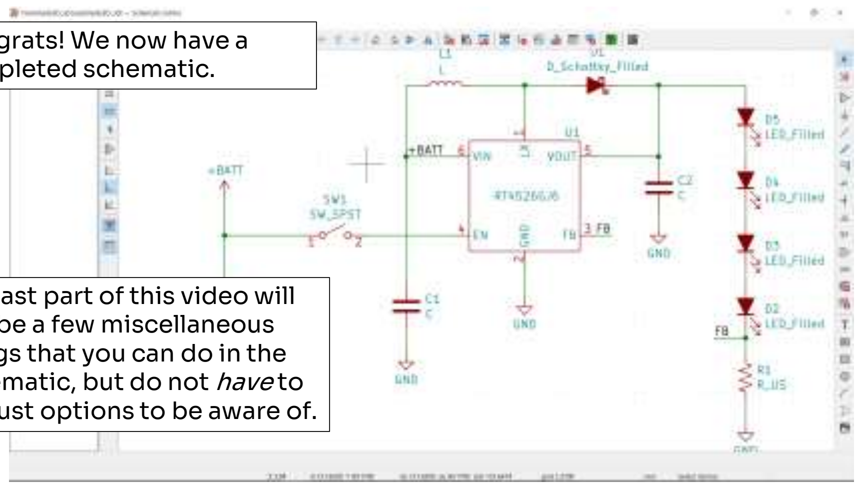
Which we know.

But no more errors!

Delete the markers and close out.

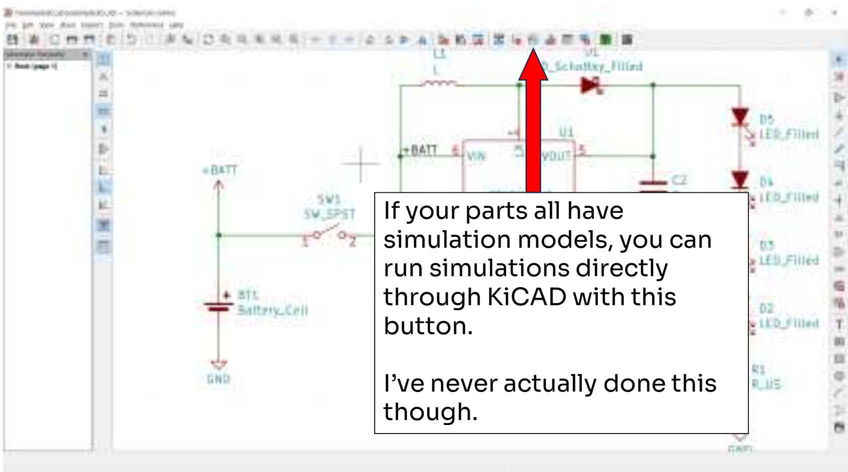
Final schematic and misc.

Congrats! We now have a completed schematic.

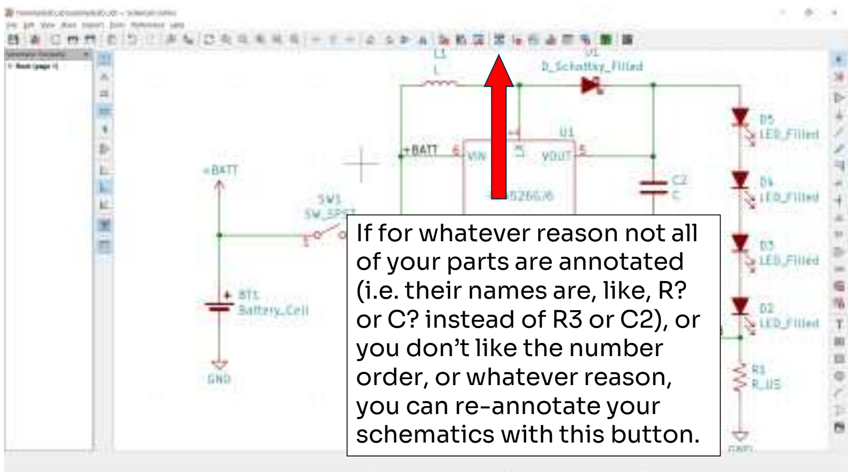


The last part of this video will just be a few miscellaneous things that you can do in the schematic, but do not *have* to do; just options to be aware of.

Final schematic and misc.

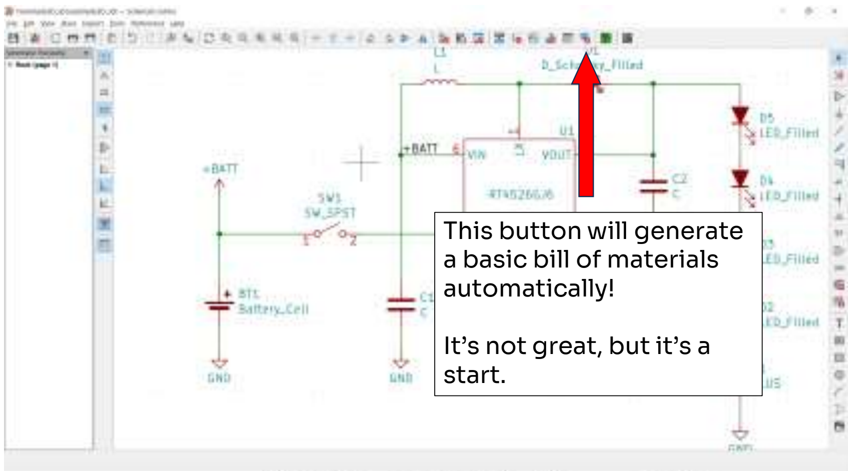


Final schematic and misc.



If for whatever reason not all of your parts are annotated (i.e. their names are, like, R? or C? instead of R3 or C2), or you don't like the number order, or whatever reason, you can re-annotate your schematics with this button.

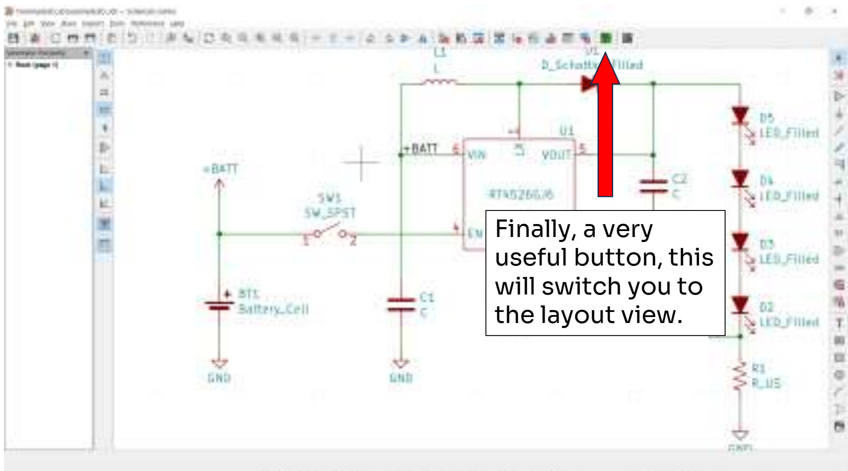
Final schematic and misc.

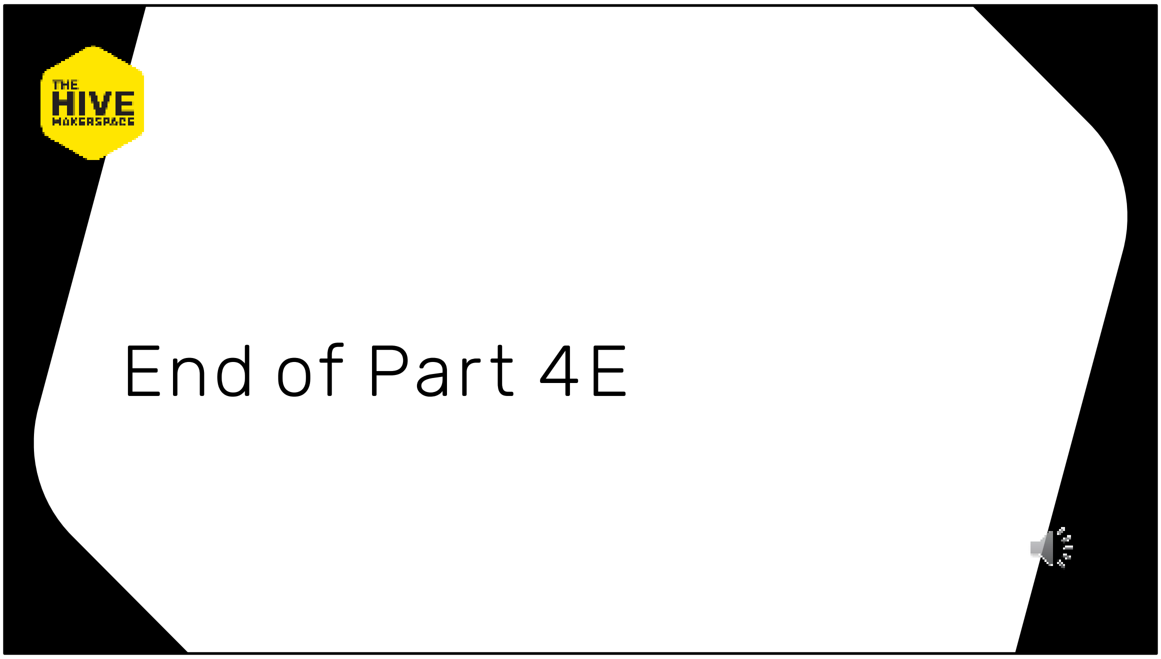


This button will generate a basic bill of materials automatically!

It's not great, but it's a start.

Final schematic and misc.

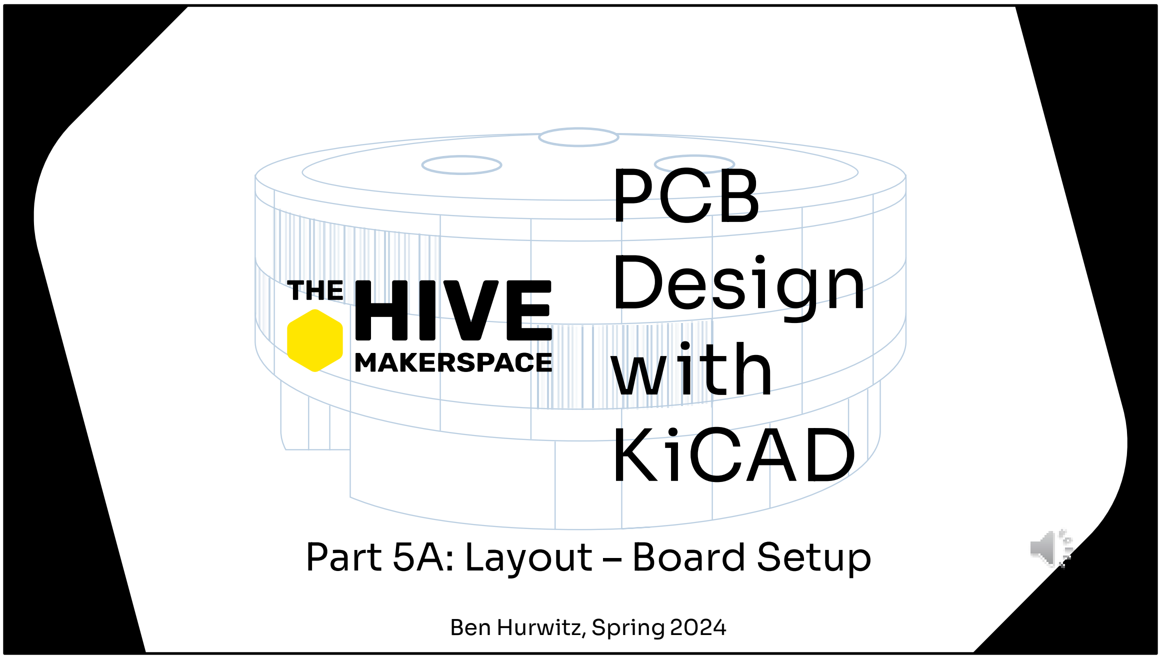




And with that, we've completed part 4E, in which we covered ERC and some additional miscellany. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

This also brings us to the end of the schematic capture portion of the design. Congratulations!

In the next video in our PCB Design with KiCAD series, part 5A, we'll move over the the layout, known in KiCAD as the PCB view, and begin with setting up some defaults and the design rules.



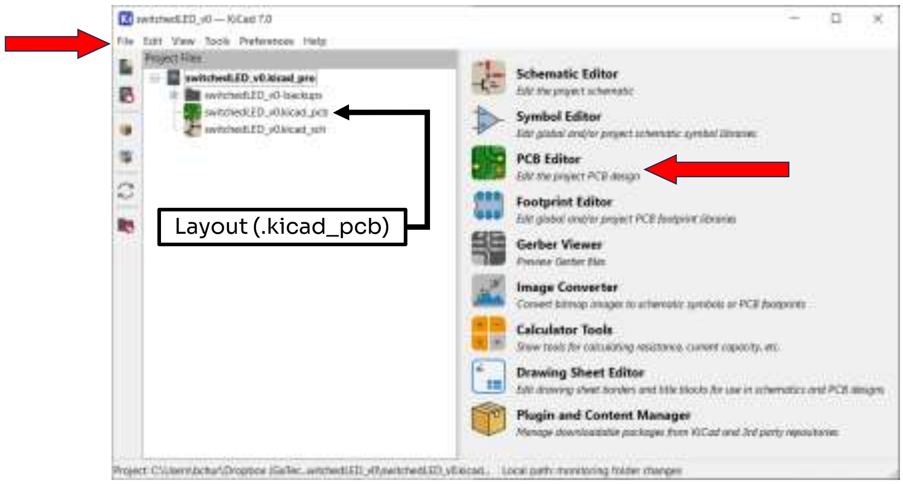
Hi all, welcome to The Hive's series on PCB Design with KiCAD. My name is Ben Hurwitz, and in this series, we've been walking through the PCB design process using KiCAD as our electronics design software.

In the last video, part 4E, we finished off the schematic capture with a look at the ERC, or the electrical rules check, and a few bits of schematic miscellany.

In these next few videos, we'll shift focus to the layout portion of the design process, which KiCAD calls the PCB view, in which we'll actually physically align and orient the actual components on the board and connect them with traces.

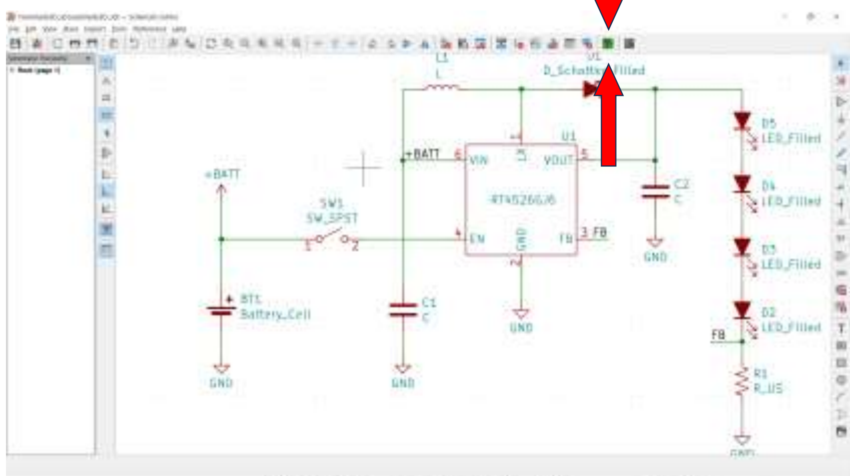
This section, part 5A, will introduce the board editor and look at setting up the board's defaults and design rules. Let's get started.

Main Window – Working Project



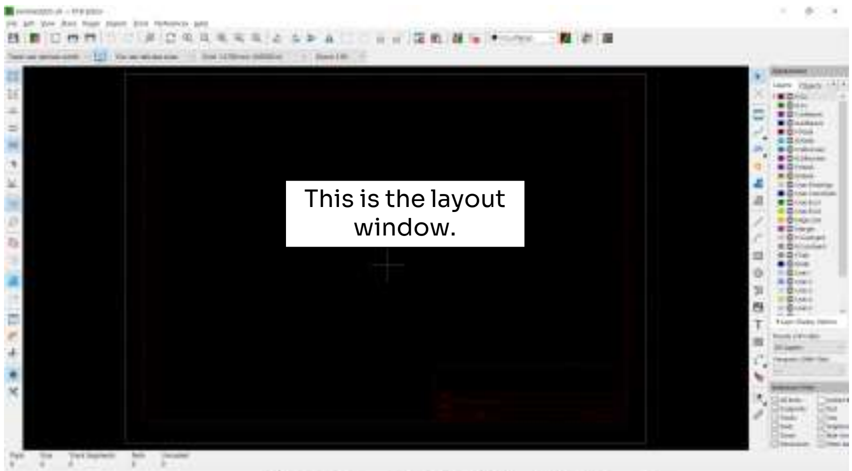
If you're just opening KiCAD for this video, go ahead and *load the project via the "File" menu, and open the PCB editor by ether *double clicking on the layout file with extension .kicad_pcb, or by *clicking on the PCB editor icon on the right, which will open the current project's board view by default.

Layout quick-switch

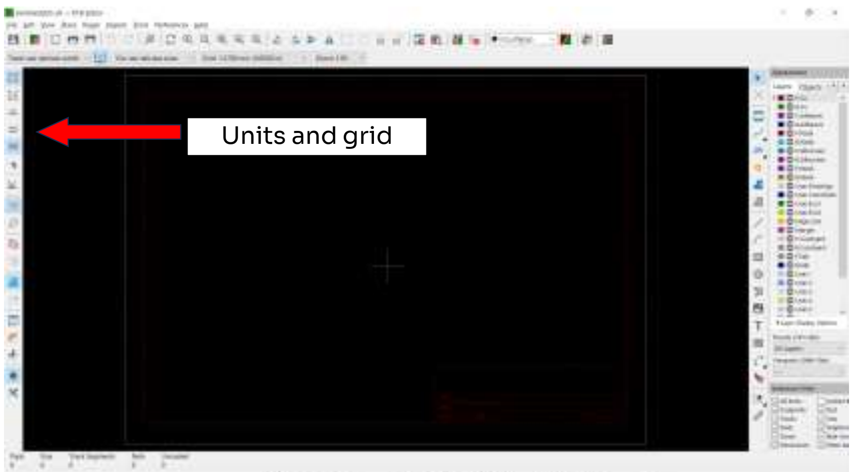


If you're coming from the previous video, part 4E, or you have the schematic editor up for whatever reason, you can also use the icon highlighted here to open the PCB view.

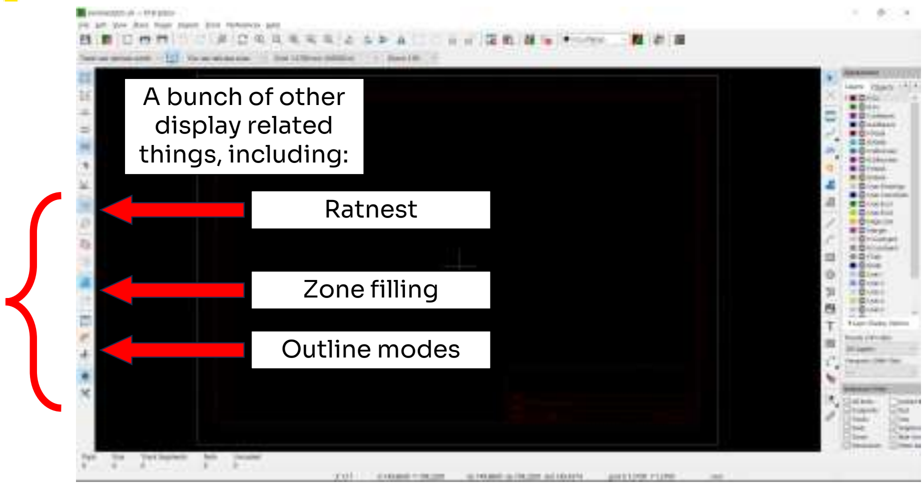
Layout



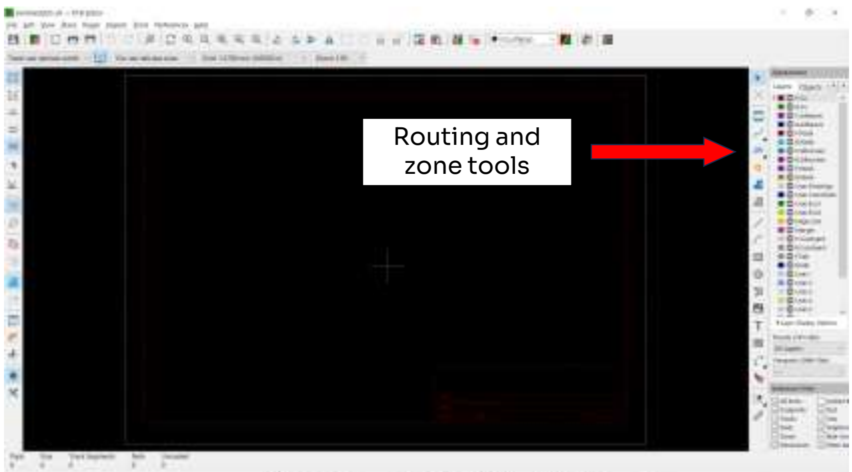
Layout



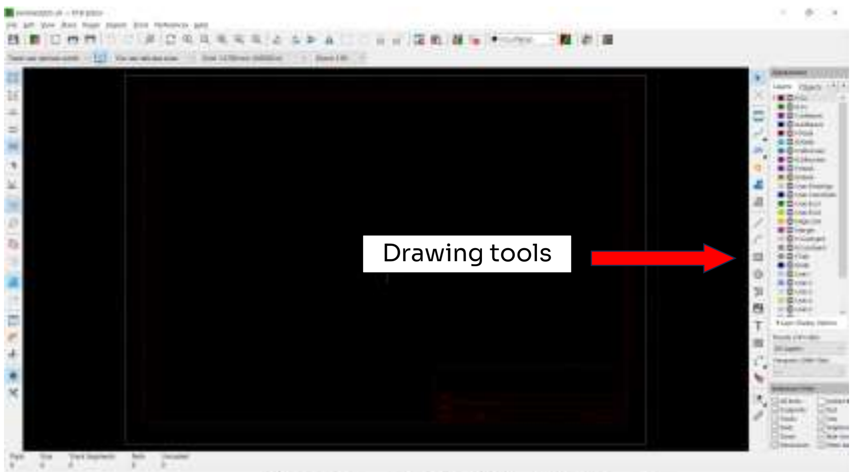
Layout


A bunch of other display related things, including:
Ratnest
Zone filling
Outline modes

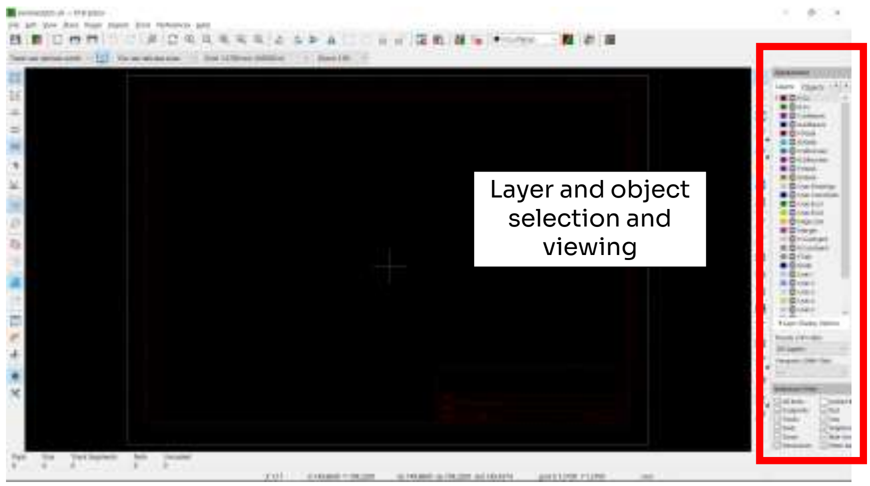
Layout



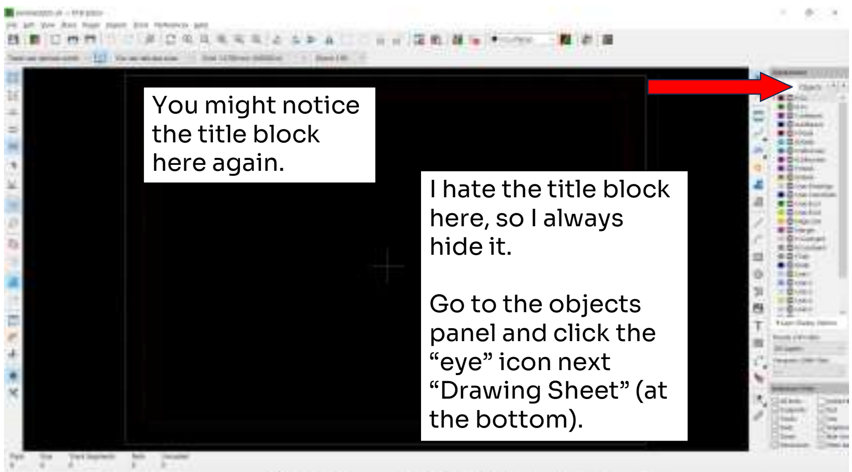
Layout



Layout

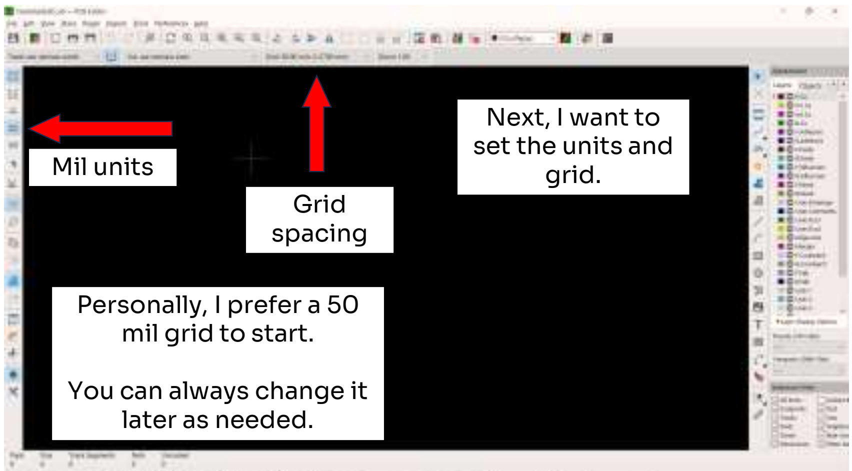


Layout



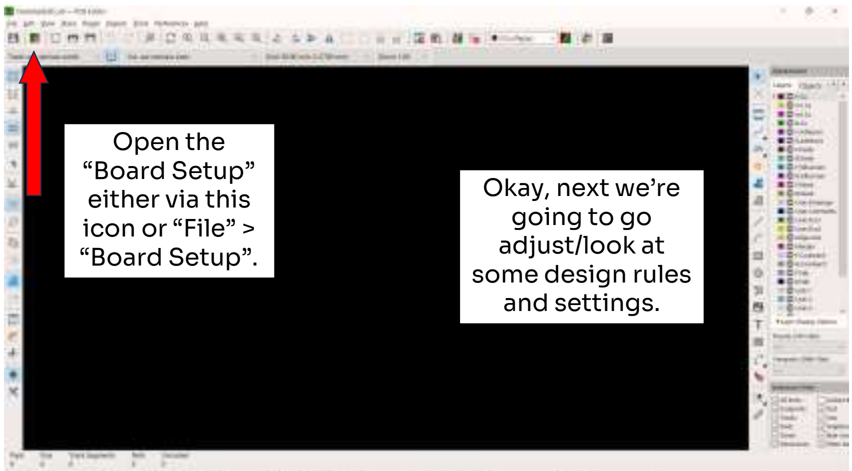
Similar to the Schematic editor, removing the title block requires making a new drawing sheet, which is beyond the scope of this tutorial.

Layout



Unlike the schematic in which the grid is really important to KiCAD's functionality, the grid is highly flexible in the layout editor, and can (and often will) be changed frequently.

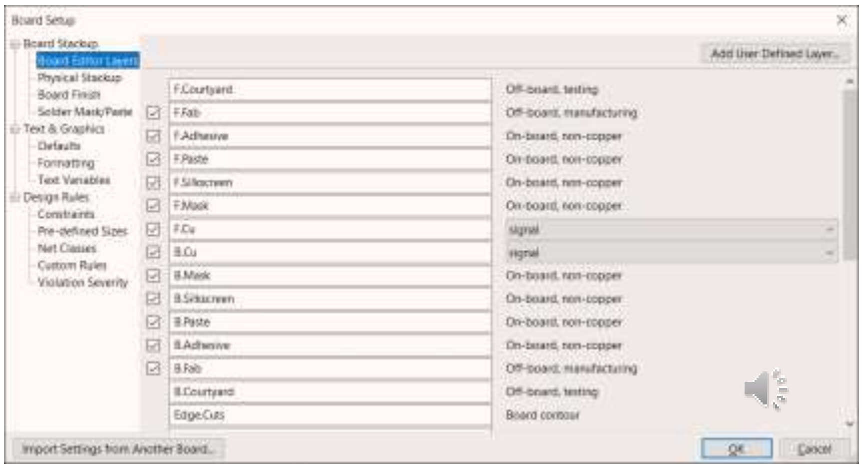
Layout


Open the
“Board Setup”
either via this
icon or “File” >
“Board Setup”.
Okay, next we’re
going to go
adjust/look at
some design rules
and settings.

Layout – Board Setup

The “Board setup” window offers you many settings to adjust for your design.

There is a similar window for the schematic, but it’s much more important to understand some of these settings in the layout view.



Layout – Board Setup

The first section allows you to select the layers available to your design.

You can select/deselect these as you see hit in your design.

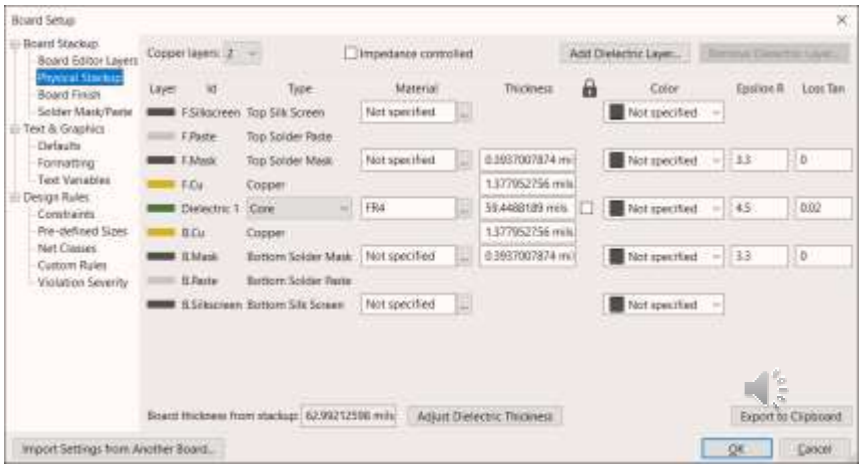
I believe they will not be used at all if they're deselected, versus just not being visible.

This can be useful if it's important that you don't put silkscreen or copper on the backside of the board, for example. By removing them from the layers entirely, you minimize changes of error. You can also add user-defined layers here if you want or need, perhaps for a custom assembly process, required by your fab house, or for specific additional information.

Layout – Board Setup

“Physical stackup” refers to the the physical (rather than digital) layers of the boards.

This is where you would set your board to be 4 layers, or 8 or whatever, using the “Copper layers” drop down.



You can do some really advanced setup in here, though I’m honestly not sure how much of it is used in other parts of the software versus just being for your (and you boss’s) knowledge.

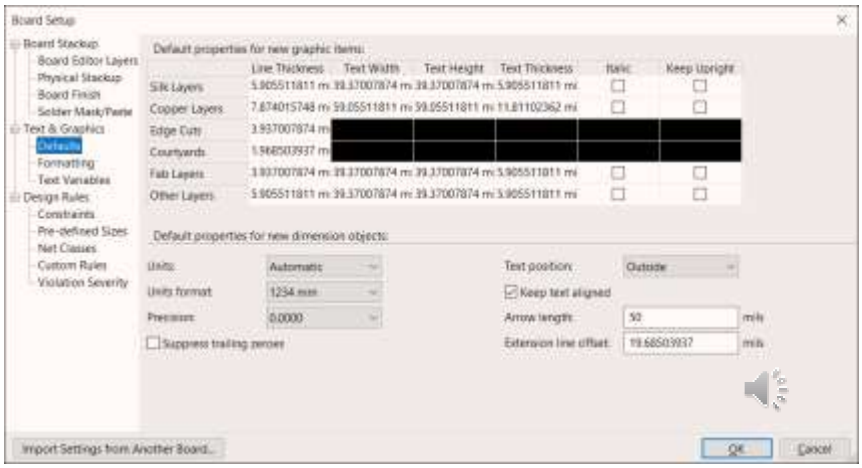
The units in this pane and in all other panes are derived from the units specified in the main layout editor window. KiCAD is a metric-based design tool, which is why the values are so wonky in mils.

Layout – Board Setup

Here you can set some defaults for text and graphics.

The defaults are normally okay, but it can be hard to gauge what size you need.

Good rule of thumb is that height = width = 6*thickness.

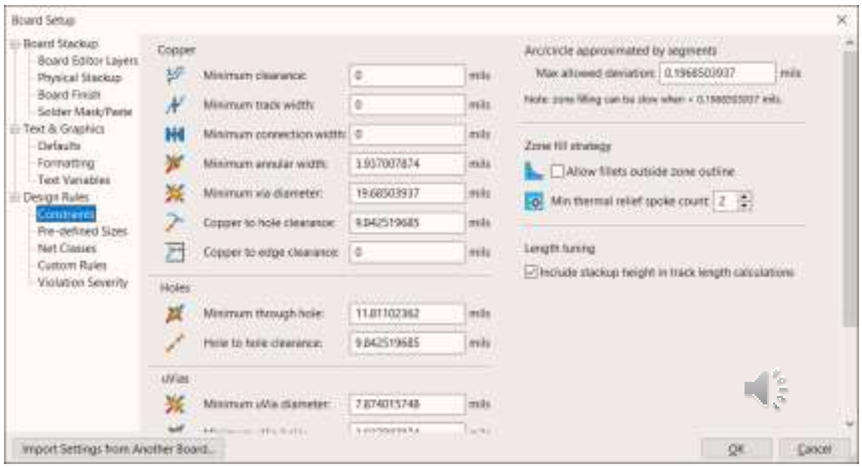


It can be a useful exercise to make a dummy layout only with text on it at various height/width and thickness ratios for your reference. If you print that board (CTRL + P, or "File" > "Print"), it can be handy to keep nearby for scale.

Layout – Board Setup

The "Constraints" pane is where all our design rules go, things like minimum spacings and sizes.

These are defined by your chosen fabrication house (i.e. where you're getting the board made). Read their instructions closely.

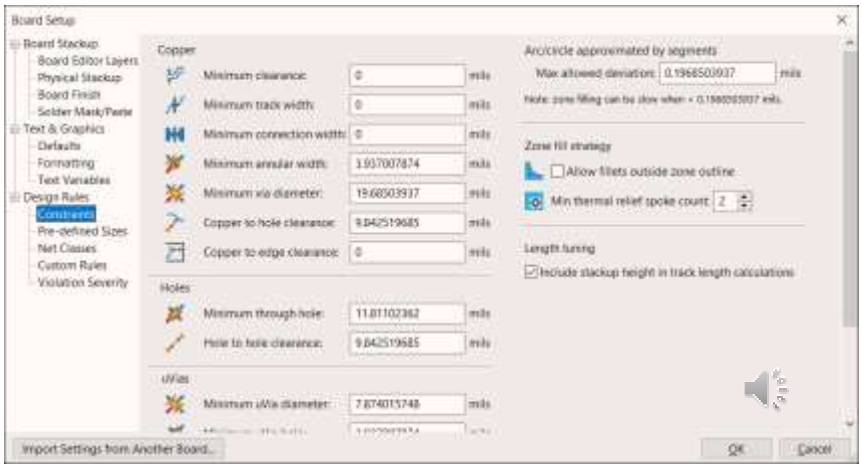


Many fab houses have software-specific instructions these days. Read them carefully! This is an easy way to make your design un-fabricatable, and require re-doing the layout.

Layout – Board Setup

I'm not going to adjust any of this now because I'm not getting this fabbed, but do not forget to set this.

If you set it at the end, that's okay, but be prepared for a bunch of DRC errors and potentially significant re-design.



Layout – Board Setup

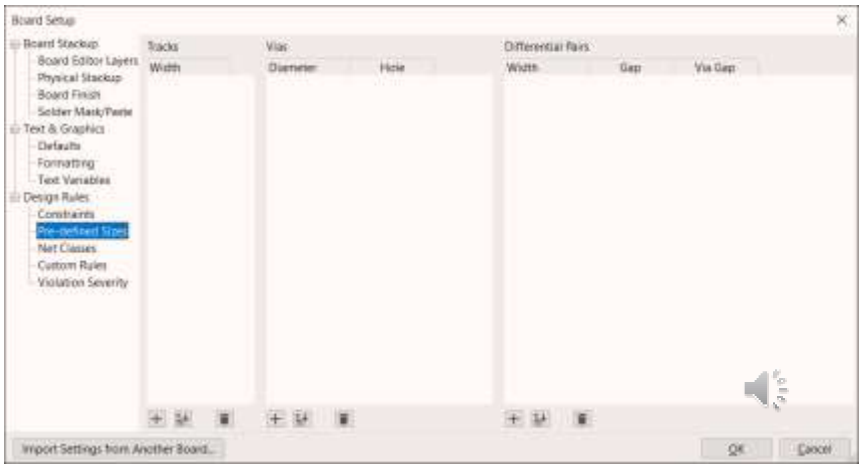
You can adjust the severity of the DRC rules (warning, error, or ignore) under the "Violation Severity" tab.

Don't do this unless you're confident of the result, or you may end up with a bad board.

Layout – Board Setup

In “Pre-defined Sizes”, we can set some pre-defined sizes for traces and holes.

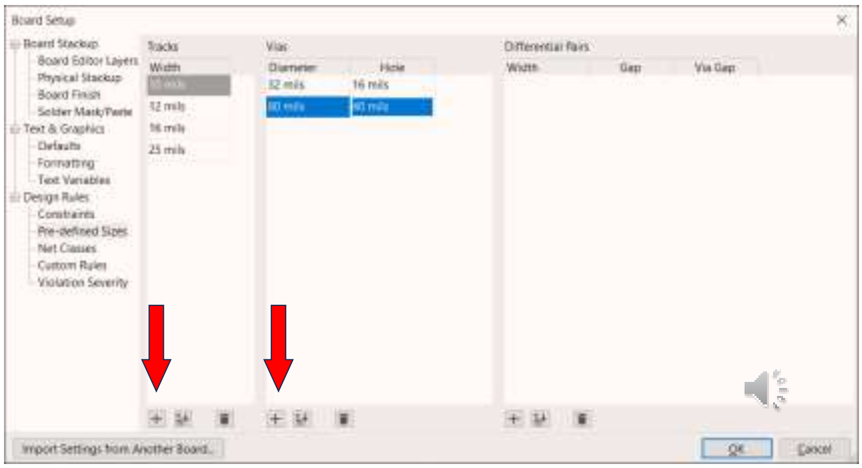
This will make your lives easier later by allowing hot-changing of sizes.



Layout – Board Setup

Use the “+” button to add:

- Track widths of 10 mil, 12 mil, 16 mil, and 25 mil.
- Via diam/holes of 32/16 mil and 80/40 mil



The parameter “diameter” for holes and vias refers to both the drill hole diameter itself plus the annular ring. It’s generally a good starting point to have the ring diameter to be twice the hole diameter, and larger can make soldering easier.

These default values are not required by anything.

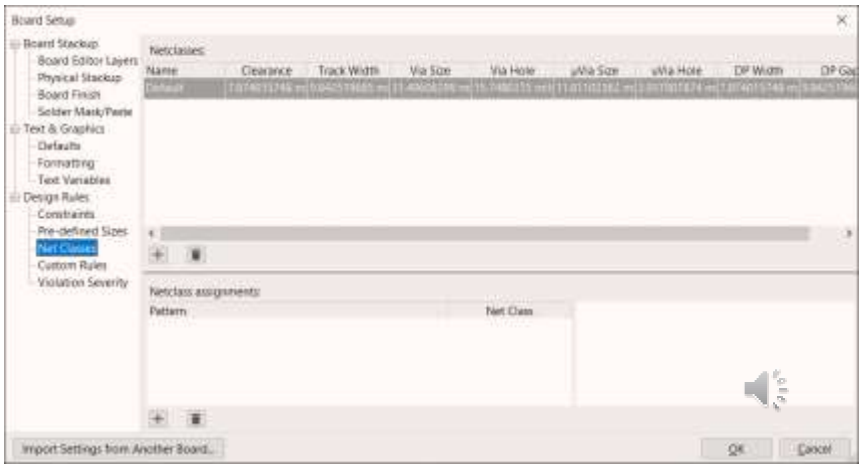
I like to keep my traces above the minimum allowable if possible, which is normally around 6 mils, to reduce the possibility of broken or damaged minimum-width traces that are more likely to have issues. For signals, I try to use 12 mils generally and neck-down, meaning shrink briefly, to 10 mils. Power traces should have the largest reasonable width possible. 25 mils is a good balance, and according to the trace-width calculator at Advanced Circuits, can be used to pass 1A on external 0.5 oz copper with just a 10 degree Celsius rise in temperature. 16 mils is a good necking size for brief lengths over which 25 mils can’t fit.

For the vias, the 16 mil diameter hole for vias is a classic default, and is good for small currents and signals. The 40 mil hole is about 1mm, and will fit a standard pin header if you need a via but do not have electroplating. It’s also good for a lot more current. Not necessary typically, since you can use multiple small vias instead of one large one.

Layout – Board Setup

You can group nets together into *netclasses* to give them all the same default settings.

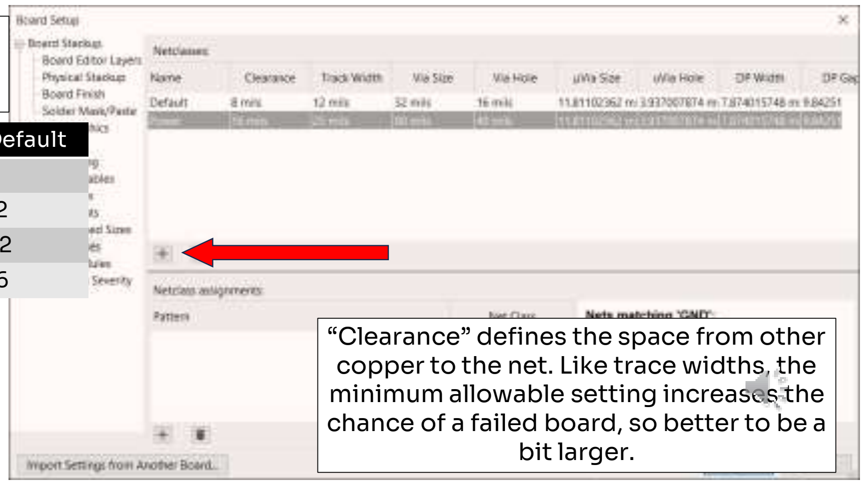
This is very useful if you want all your power traces to be wide, or some traces to be very narrow.



Layout – Board Setup

Let's define a "Power" netclass, and adjust "Default":

Parameter	Power	Default
Clearence	16	8
Track Width	25	12
Via Size	80	32
Via Hole	40	16



"Clearance" defines the space from other copper to the net. Like trace widths, the minimum allowable setting increases the chance of a failed board, so better to be a bit larger.

*Clearance… blah blah blah….

For The Hive's tool, because we don't offer a protective soldermask layer, it's important that this clearance is set quite large, 30-50 mils, to reduce the chances of accidentally jumping traces or a plane when soldering.

The uVia, or microvias, are extra-tiny vias that are typically used with BGA-style components with high-density ball-style pads underneath the package. These will add to your board cost, and are usually not needed unless you really can't find an alternative package.

DP stands for differential pairs, which are used when impedance matching is important, like for USB data or antennas.

Layout – Board Setup

Assign your power net (the one out of your battery) and the ground net to the “Power” class.

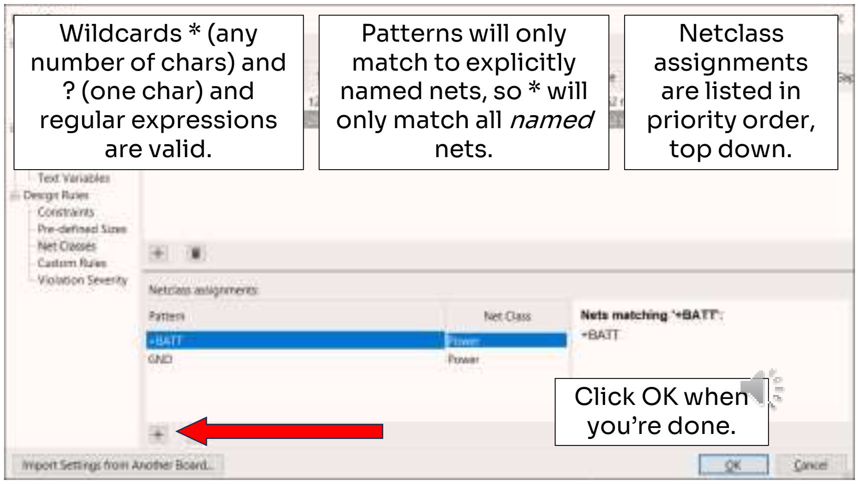
Must be one pattern per assignment line.

ALT+Tab into the schematic to double-check the names.

Wildcards * (any number of chars) and ? (one char) and regular expressions are valid.

Patterns will only match to explicitly named nets, so * will only match all *named* nets.

Netclass assignments are listed in priority order, top down.

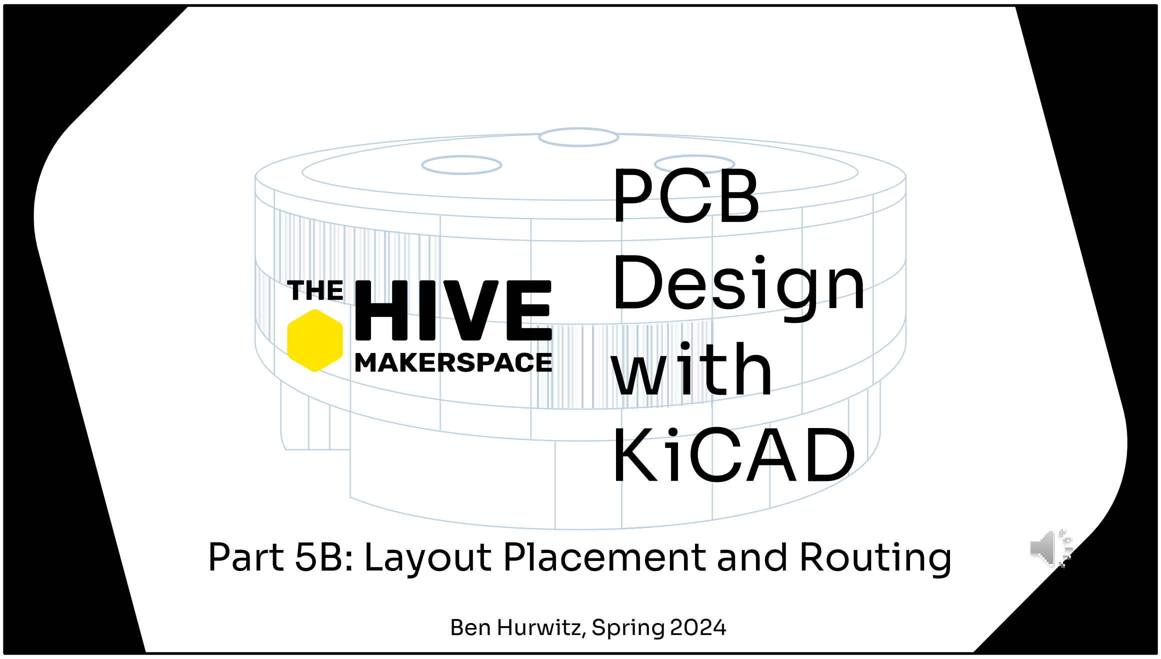


Click OK when you’re done.

Netclass assignments can actually also be defined in the schematic by right-clicking a node or net.

And that ends part 5A of the KiCAD design tutorial series in which we introduced the layout editor and a number of pre-layout board setup settings. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In the next video, part 5B, we'll go through placement of the components and routing. See you then.



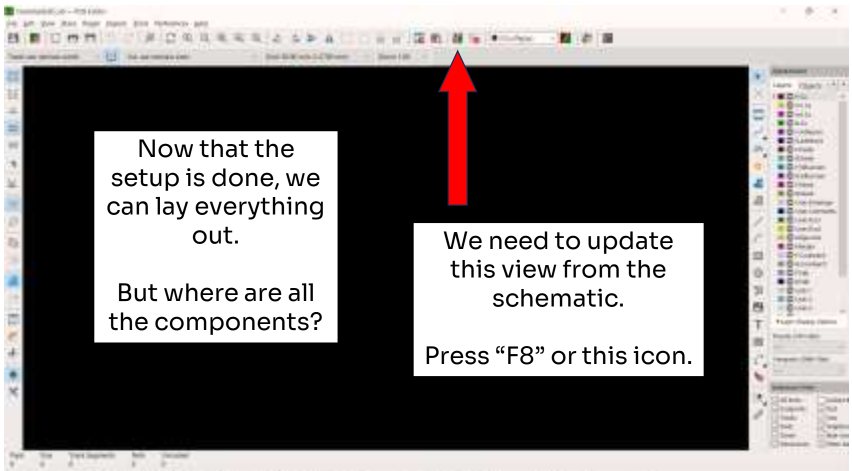
Hi all, welcome to The Hive's series on PCB Design with KiCAD. My name is Ben, and in this series, we've been walking through the PCB design process using KiCAD as our electronics design software. Part 5, where we are, is focused on the layout portion of the design process.

In the last video, part 5A, we introduced the layout editor, also called the PCB editor, and how to setup your design constraints and sizings.

In part 5B, I will take you through the basics of placing your compoentns, called placement, and connecting them together, called routing. This video will primarily be focused on showing you the gist and then having you do most of these processes, so definitely follow along and pause to do the work.

Let's get started.

Layout

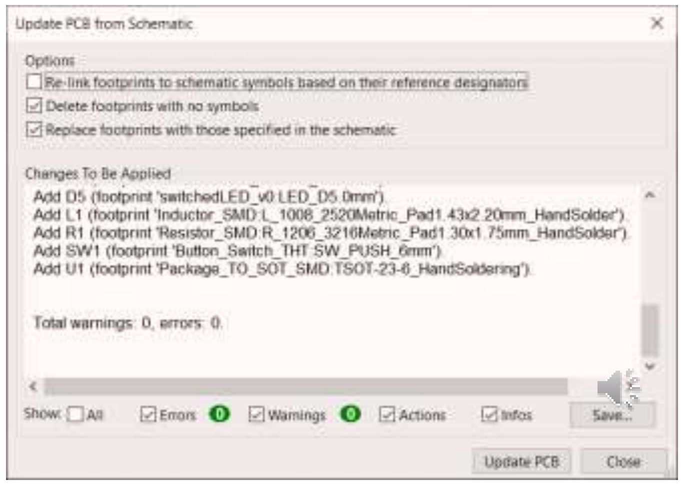

Now that the setup is done, we can lay everything out.
But where are all the components?
We need to update this view from the schematic.
Press “F8” or this icon.

Layout

It'll take a second, but you'll get this window.

It will list the changes to be applied, which you can read through.

Click "Update PCB" to bring the footprints into the layout view.



Layout

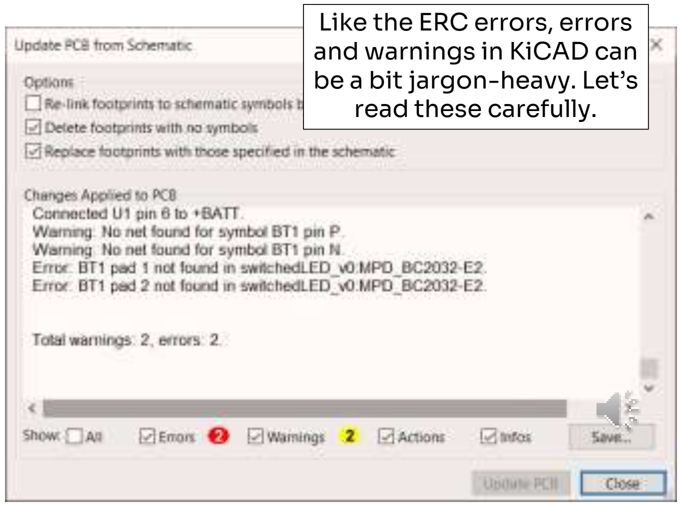
Well phooey. We got some warning and errors.

What are they?

The warnings say that the battery symbol (BT1) has no nets connected to its pins P and N.

The errors say that the BT1 footprint does not have a pad 1 or a pad 2.

.... What?



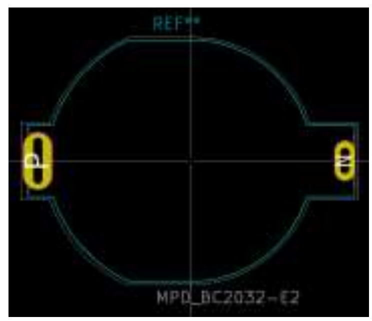
Like the ERC errors, errors and warnings in KiCAD can be a bit jargon-heavy. Let's read these carefully.

Layout

When you assign a footprint to a symbol, KiCAD attempts to match the symbol pins to the footprint's pads.

In this case, the symbol, which we got from the global built-in library, has pins 1 and 2, which KiCAD attempted to match to the footprint's pads, which are names P and N.

Let's take a quick look at the warnings and errors again.

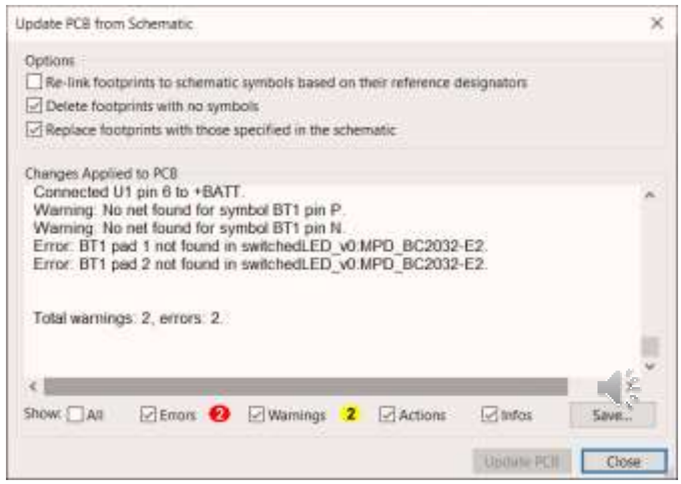


Layout

After linking pins 1 and 2 with pads P and N, KiCAD was looking in the schematic's battery symbol for two pins called P and N because that's what the footprint says they should be called.

Obviously, it didn't find them, and thus there are no nets associated with those pin (because those pins don't exist).

That's the warning – no nets.

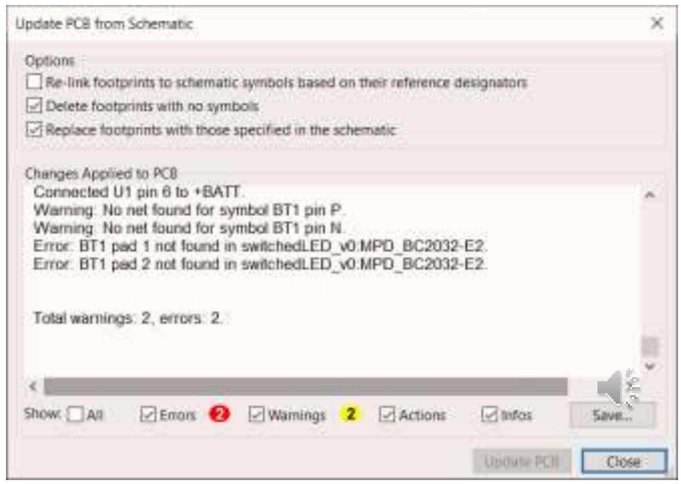


Layout

Similarly, KiCAD went looking for pads 1 and 2 in the footprint because the symbol said those should exist.

Of course, the pads are actually called P and N, so pads 1 and 2 weren't found in the footprint.

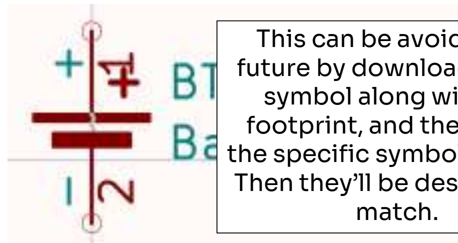
Hence, the errors – no pads.



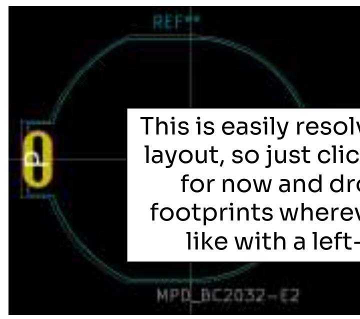
Layout

Recall that I brought this concern up waaaaaay back when I said that KiCAD not having devices improved flexibility, but meant you have to be careful about pins.

And now we know – symbol pin numbers must match footprint pad numbers.

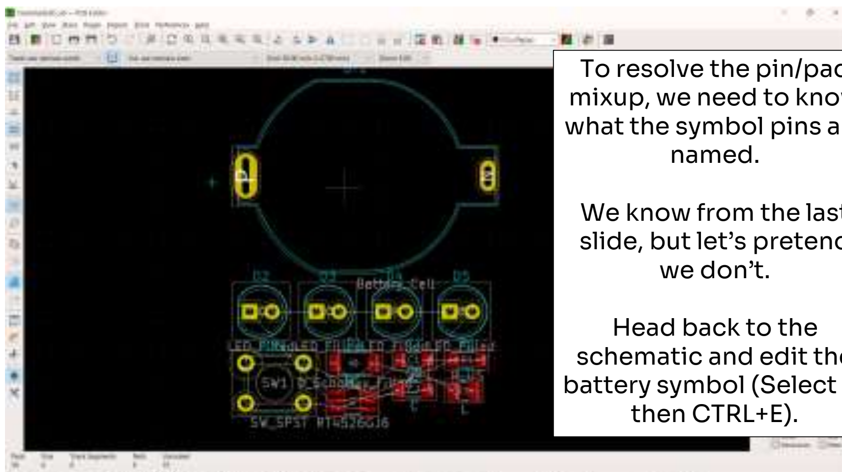


This can be avoid in the future by downloading the symbol along with the footprint, and then using the specific symbol instead. Then they'll be designed to match.



This is easily resolved in the layout, so just click "Close" for now and drop the footprints wherever you'd like with a left-click.

Layout

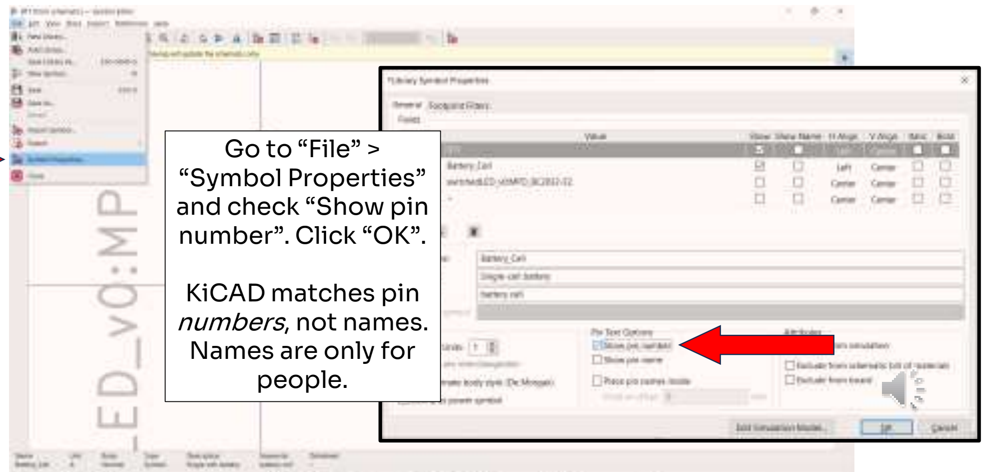


To resolve the pin/pad mixup, we need to know what the symbol pins are named.

We know from the last slide, but let’s pretend we don’t.

Head back to the schematic and edit the battery symbol (Select it, then CTRL+E).

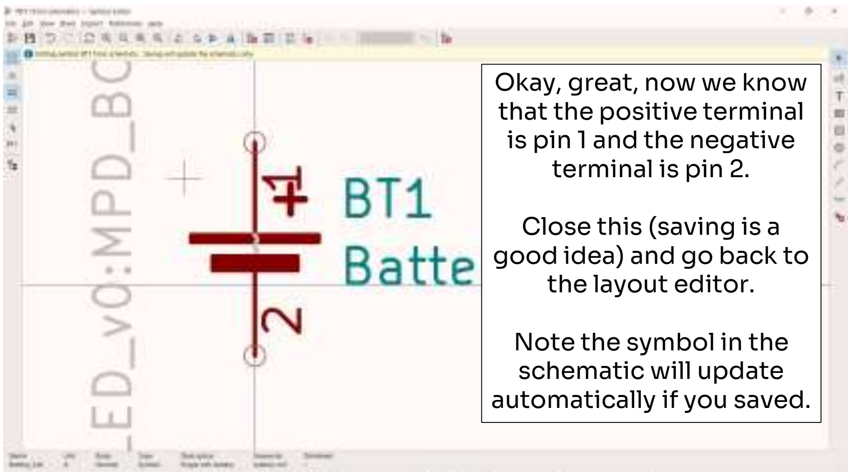
Layout



Go to “File” > “Symbol Properties” and check “Show pin number”. Click “OK”.

KiCAD matches pin *numbers*, not names. Names are only for people.

Layout

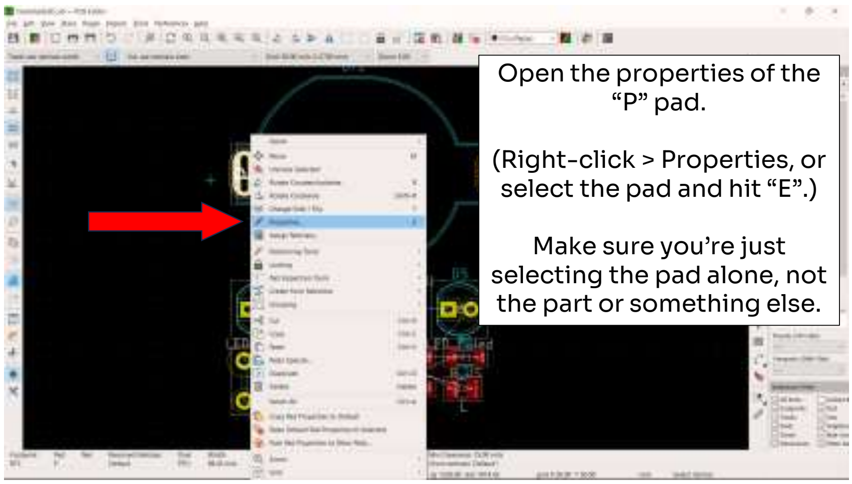


Okay, great, now we know that the positive terminal is pin 1 and the negative terminal is pin 2.

Close this (saving is a good idea) and go back to the layout editor.

Note the symbol in the schematic will update automatically if you saved.

Layout

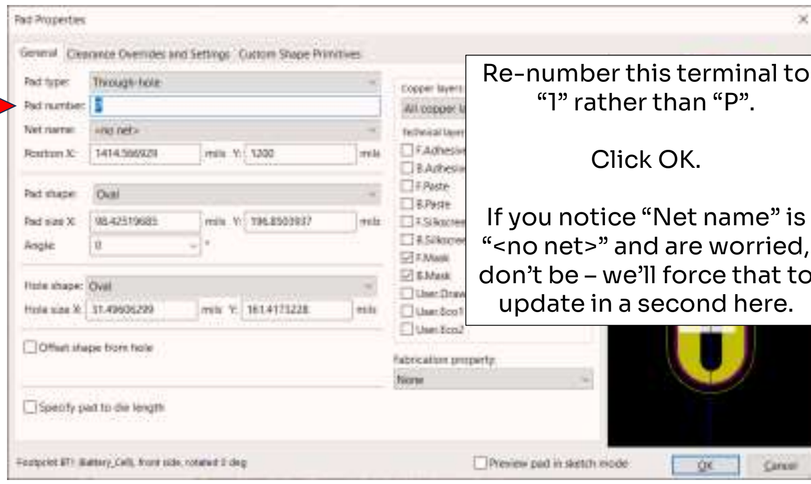


Open the properties of the "P" pad.

(Right-click > Properties, or select the pad and hit "E".)

Make sure you're just selecting the pad alone, not the part or something else.

Layout

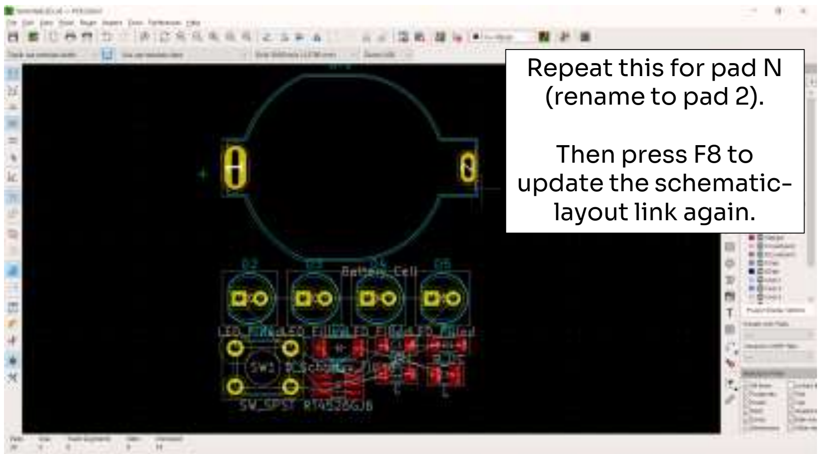


Re-number this terminal to “1” rather than “P”.

Click OK.

If you notice “Net name” is “<no net>” and are worried, don’t be – we’ll force that to update in a second here.

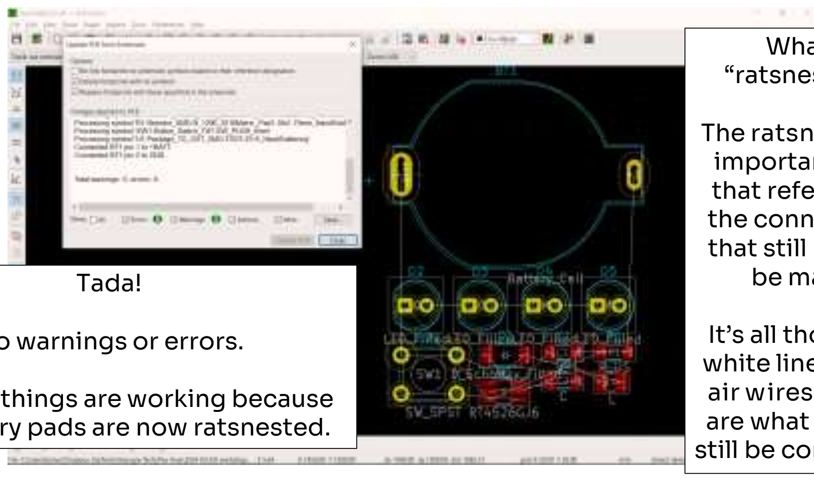
Layout



Repeat this for pad N (rename to pad 2).

Then press F8 to update the schematic-layout link again.

Layout



Tada!

No warnings or errors.

We know things are working because the battery pads are now ratsnested.

What's "ratsnested"?

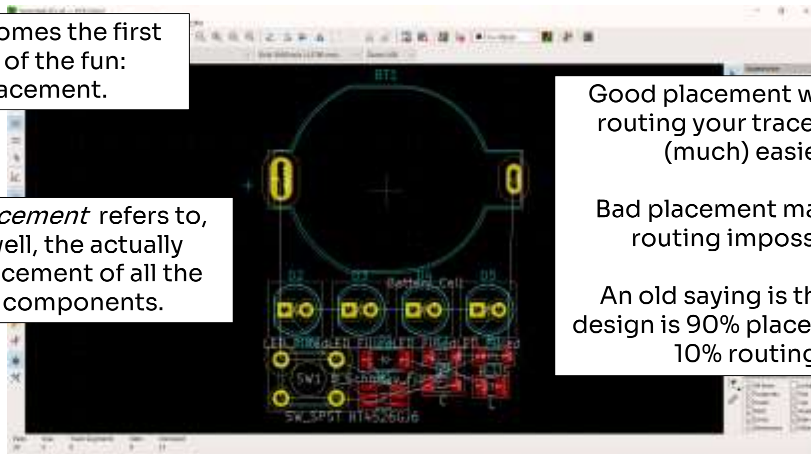
The ratsnest is an important term that refers to all the connections that still need to be made.

It's all those thin white lines called air wires – those are what need to still be connected.

Layout

Now comes the first half of the fun: placement.

Placement refers to, well, the actually placement of all the components.



Good placement will make routing your traces much (much) easier.

Bad placement may make routing impossible.

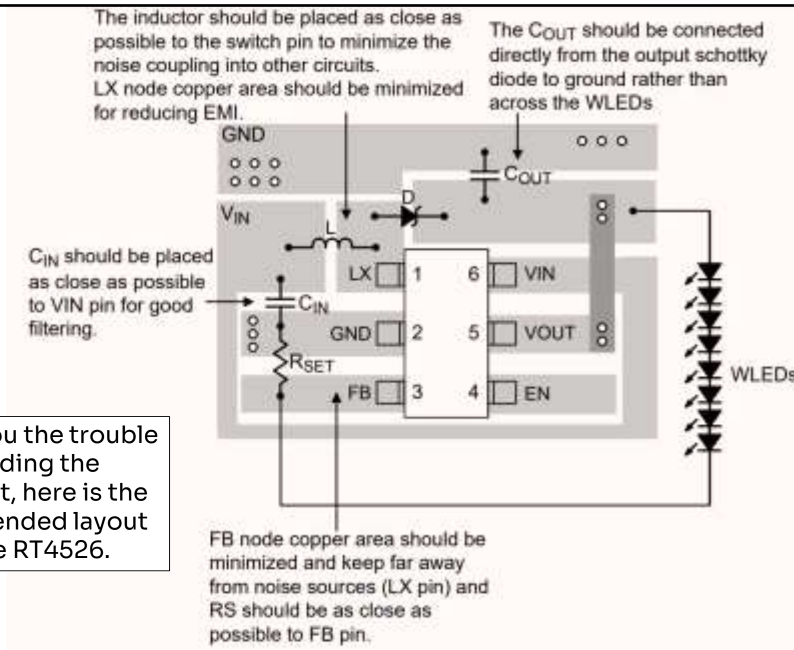
An old saying is that PCB design is 90% placement and 10% routing.

Layout - Shortcuts

- Go ahead and try placing the components now.
  - Move by click-and-drag or select-and-“M”
  - Rotate with “R”
  - Flip to the other side of the board with “F”.
  - If your component turns red, it means it’s illegally positioned
  - Hold CTRL to get ultra-fine positioning grid
- It may be helpful to hide the fab layers
  - It’s a lot of distracting text.
  - The little “eye” icons on the right under “Layers”
- Next slide has a few more pointers…

Layout – Tips and Tricks

- One thing you may notice is why the net names are so useful. What is "Net-(D1-A)" anyway?
- IC datasheets often have layout recommendations. These can be very helpful because they're known working arrangements!
- Again, two screens is helpful here.
- Watch the ratsnest as you move parts around.
- Conceptualize how to circuit flows together and cluster related componens.
- Good placement takes time! No need to rush.
- Your layout will very likely not look anything like your schematic, *and that's okay.*
  - Remember: the schematic is for people, and the layout is for electrons



To save you the trouble of finding the datasheet, here is the recommended layout for the RT4526.

I suggest that you paused the video here and take some time to arrange your components. As I mentioned already, placement is most of the work. With good placement comes easy routing. Arranging the IC based on the datasheet's suggestion here is good practice, if possible.

Layout

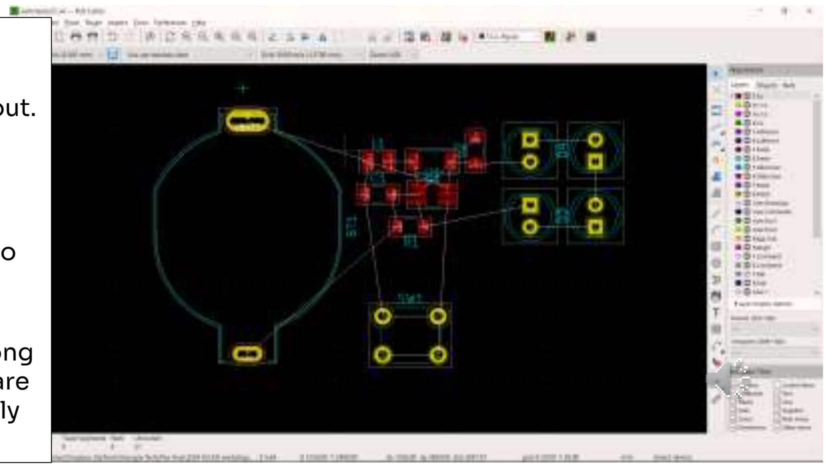
This is MY layout.

It may not be YOUR layout.

And that's okay.

There are many ways to layout a board.

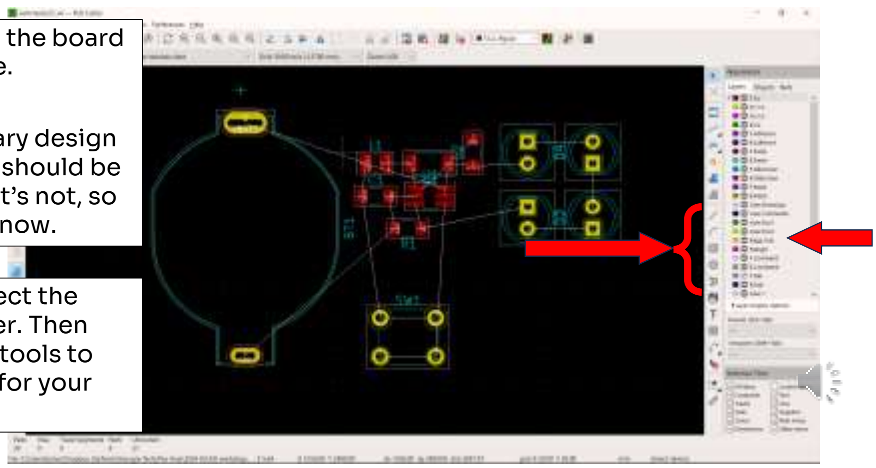
For simple boards, as long as all the connections are made, the rest probably doesn't matter.



Layout

Next is defining the board outline.

If this is a primary design constraint, this should be done first. But it's not, so we'll do it now.

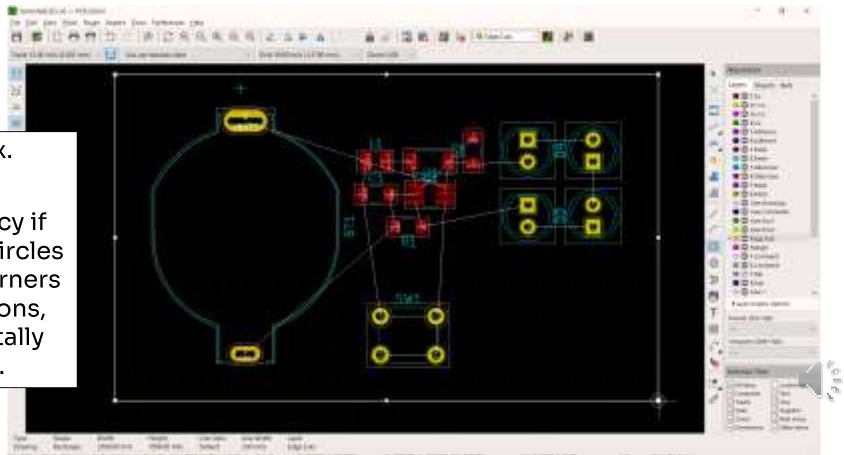


On the right, select the "Edge-Cuts" layer. Then use the drawing tools to make an outline for your board.

Layout

I made a box.

You can be fancy if you want, with circles and rounded corners or weird polygons, but a box is totally acceptable.

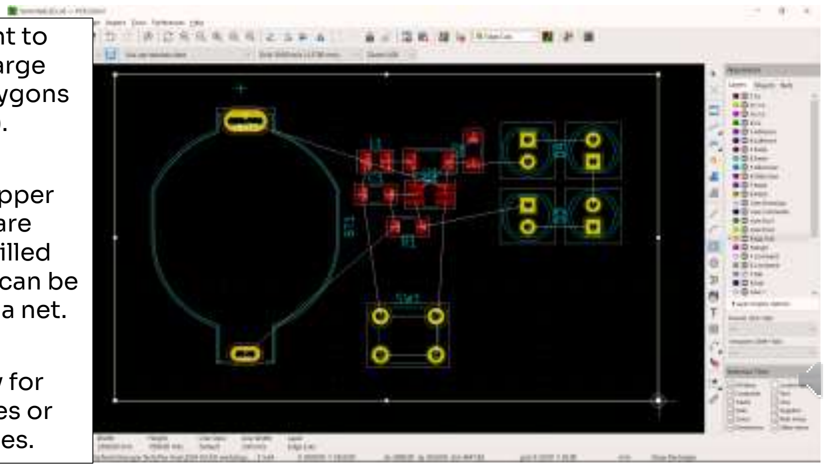


Layout

Next we want to define any large planes (or polygons or pours).

These are copper areas that are completely filled (flooded), and can be connected to a net.

Used widely for ground planes or power planes.



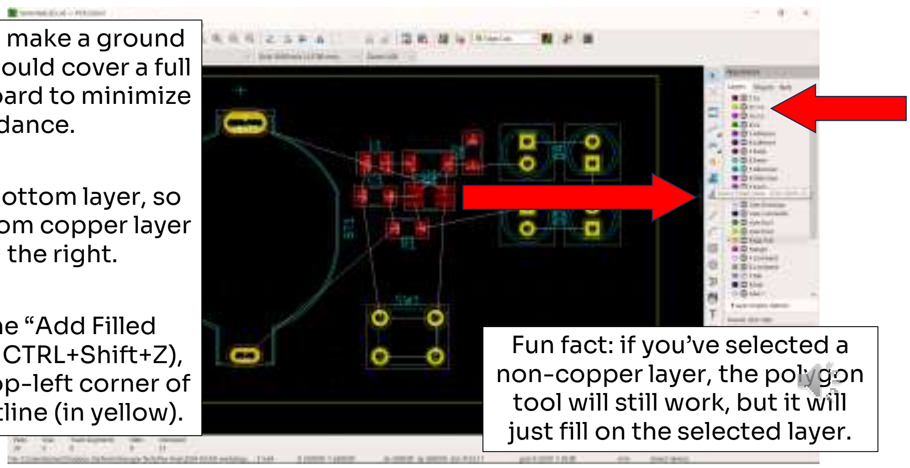
It’s safe to think of planes as arbitrarily-shaped traces. Larger traces have lower impedance, so full-board planes have the lowest possible impedance of any possible trace.

Layout

We're going to make a ground plane. These should cover a full layer of your board to minimize impedance.

We'll use the bottom layer, so select the bottom copper layer (B.Cu) on the right.

Then click the "Add Filled Zone" icon (or CTRL+Shift+Z), and click the top-left corner of your board outline (in yellow).



Fun fact: if you've selected a non-copper layer, the polygon tool will still work, but it will just fill on the selected layer.

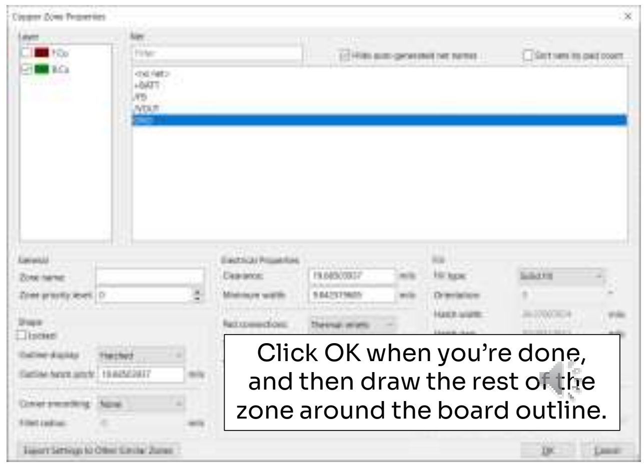
Layout

The "Copper Zone Properties" window defines our zone.

Make sure the layer is right (B.Cu), and set the net to GND (it's a ground plane).

"Zone priority" (left) will define the interactions between multiple overlapping zones.

"Clearance" (center) is critical if you're not getting soldermask, like if you fab this at The Hive.

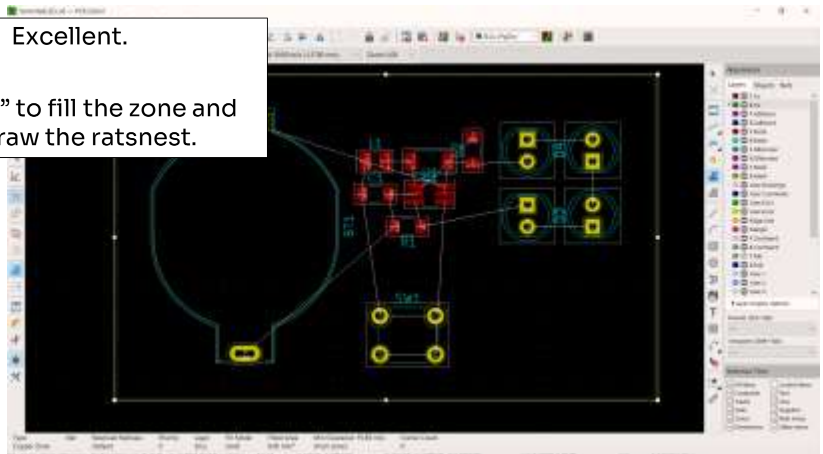


Click OK when you're done, and then draw the rest of the zone around the board outline.

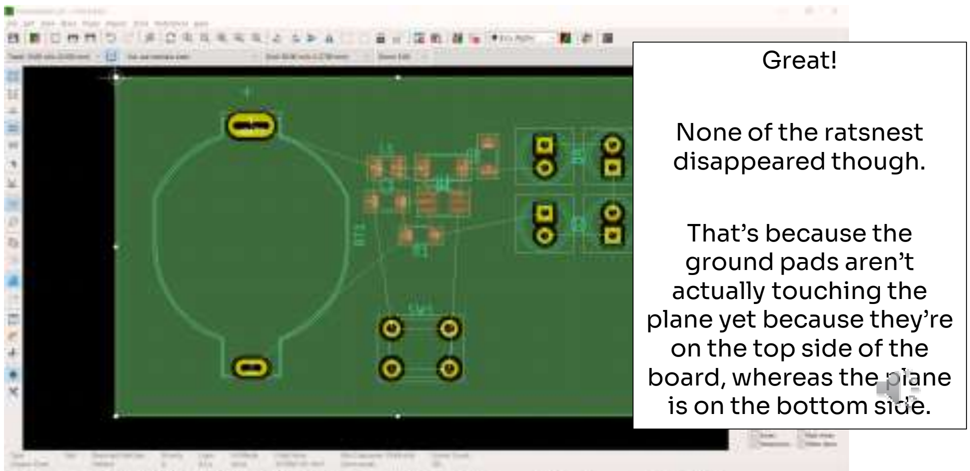
Layout

Excellent.

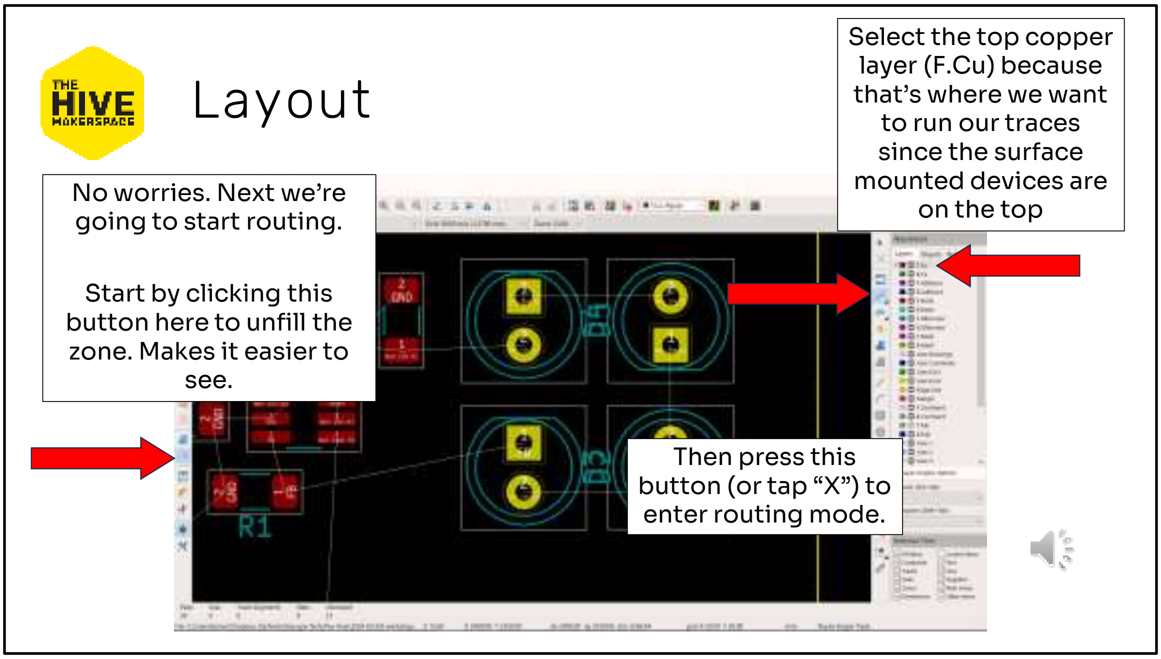
Press “B” to fill the zone and re-draw the ratsnest.



Layout



It’s also because I made a mistake when generating the zone and set it to “no net” rather than “GND”, so the GND pin on the battery should be connected but isn’t.



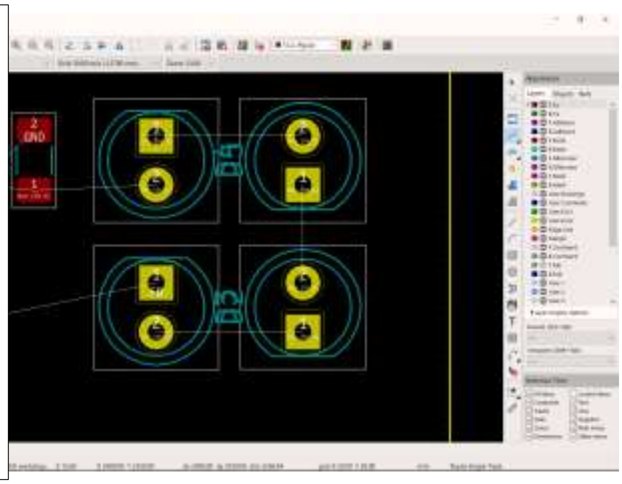
Hitting "X" will start a route where you mouse is on the canvas. Hitting the "route" icon will allow you to left-click somewhere to start the route.

Layout

I'm going to start with the LEDs because they're the simplest.

Just click (don't hold) a pad to start the route, then drag your mouse to the end of the route, and click again to end it.

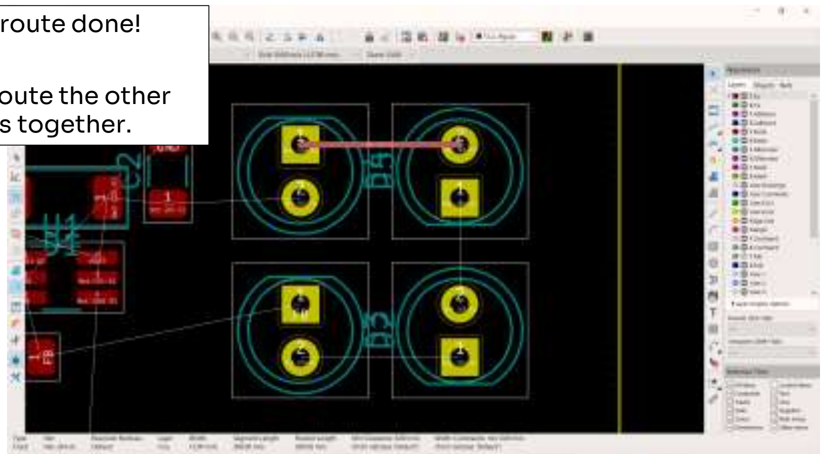
You don't need to follow the ratsnest exactly. That just shows what needs connecting, not always *how* to connect it.



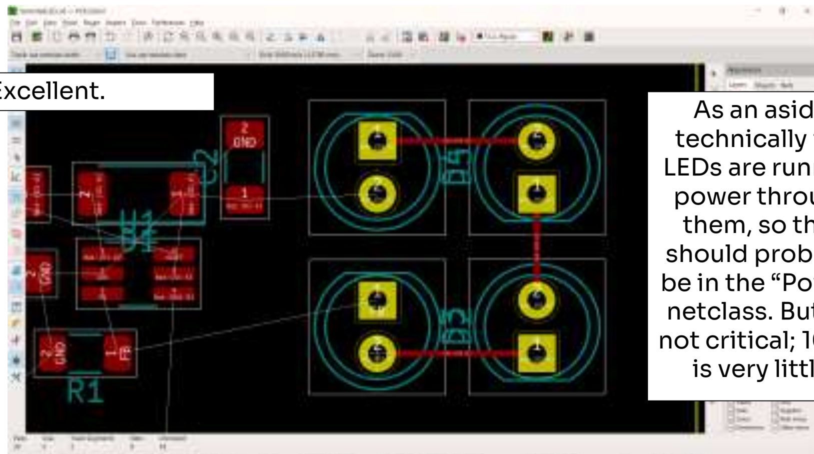
Layout

One route done!

Try to route the other LEDs together.



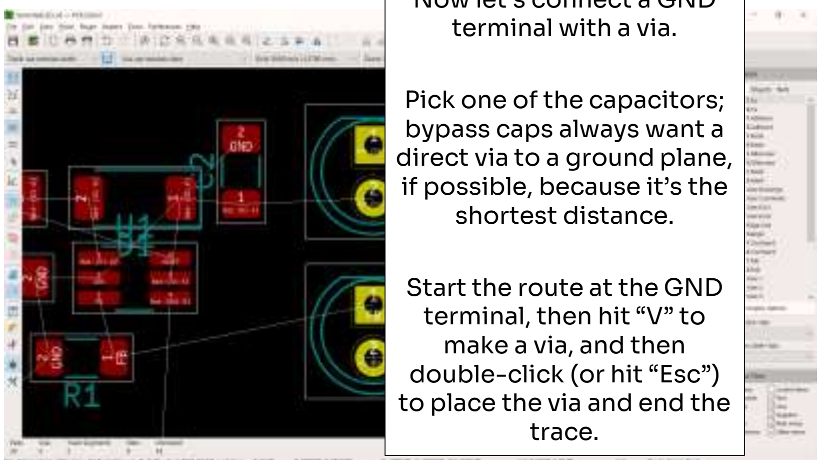
Layout



Excellent.

As an aside, technically the LEDs are running power through them, so they should probably be in the “Power” netclass. But it’s not critical; 10mA is very little.

Layout



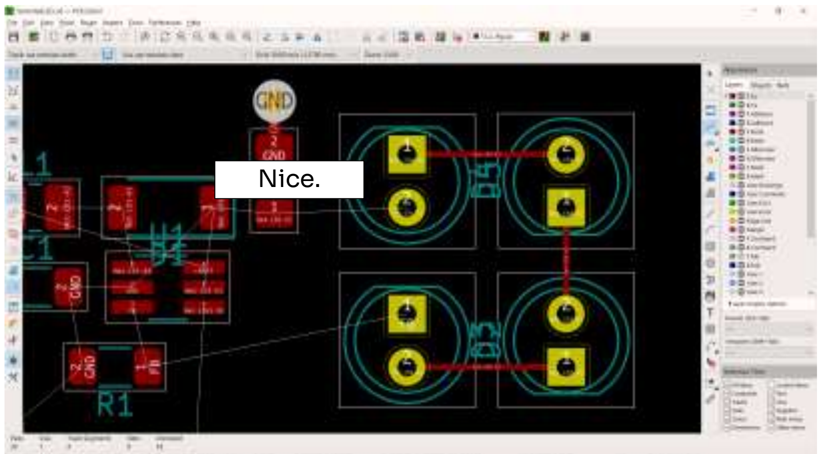
Now let's connect a GND terminal with a via.

Pick one of the capacitors; bypass caps always want a direct via to a ground plane, if possible, because it's the shortest distance.

Start the route at the GND terminal, then hit "V" to make a via, and then double-click (or hit "Esc") to place the via and end the trace.

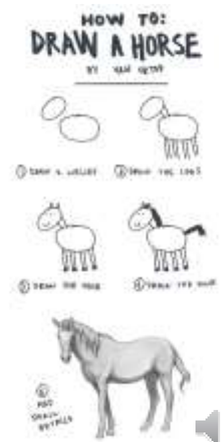
Not necessary to

Layout

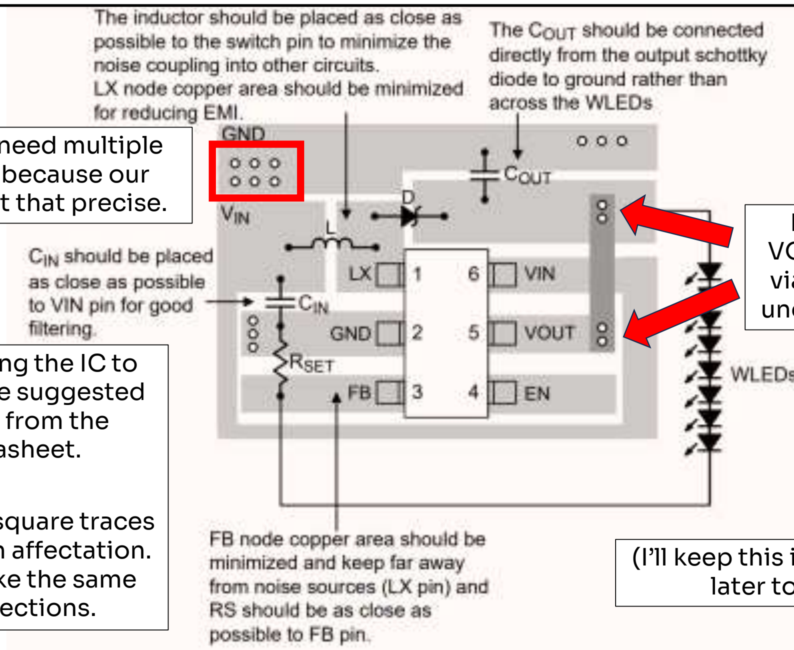


Layout

- So now it’s you’re turn to complete the layout.
- Hopefully it’s not too much like drawing a horse, but the following slides have a few tips.
- Plan to start with the IC, since that arrangement is the most specific.



You don't need multiple vias here because our design isn't that precise.



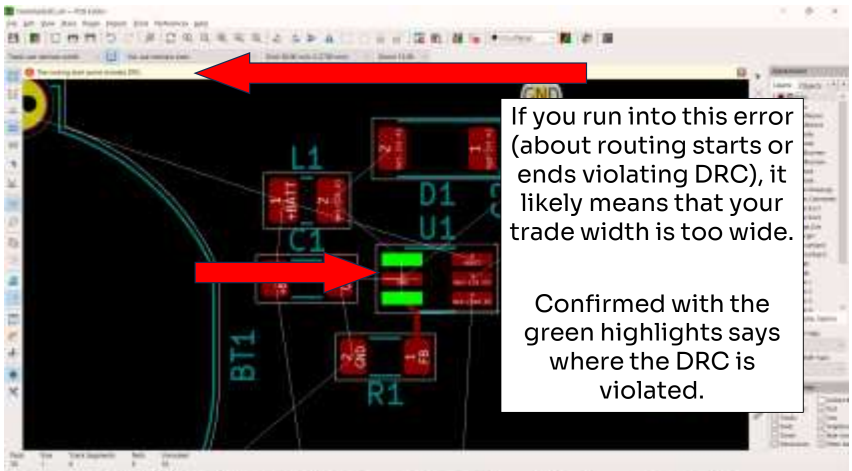
Notice that VOUT's trace is via-connected underneath VIN.

Try routing the IC to match the suggested layout from the datasheet.

Note the square traces are just an affectation. Just make the same connections.

(I'll keep this image up later too)

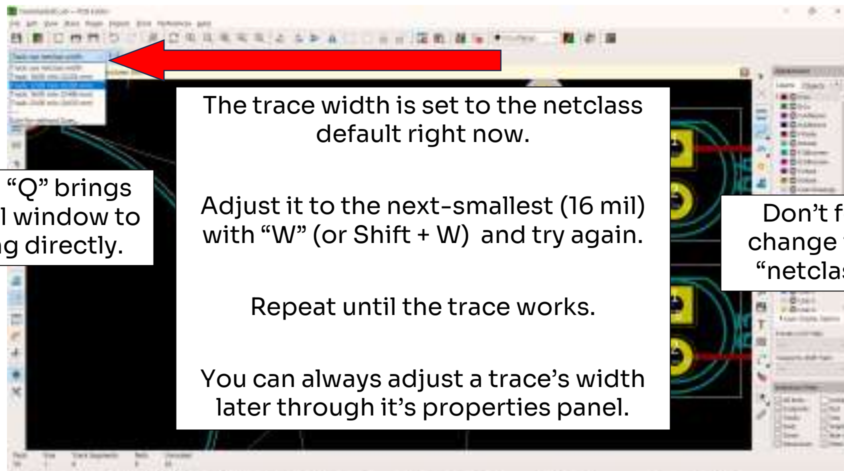
Layout



If you run into this error (about routing starts or ends violating DRC), it likely means that your trade width is too wide.

Confirmed with the green highlights says where the DRC is violated.

Layout



Pressing “Q” brings up a small window to set sizing directly.

The trace width is set to the netclass default right now.

Adjust it to the next-smallest (16 mil) with “W” (or Shift + W) and try again.

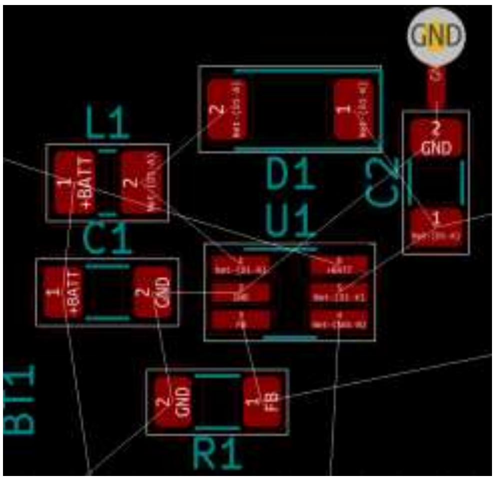
Repeat until the trace works.

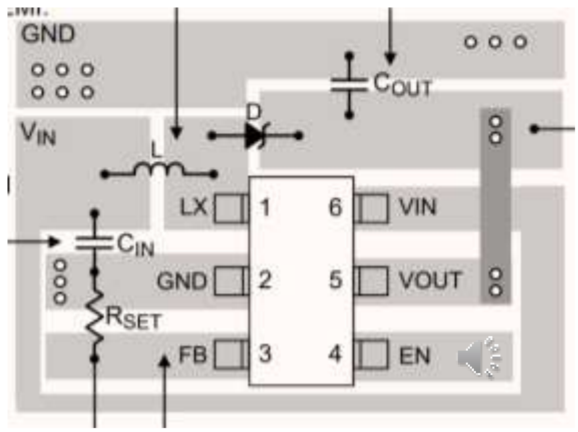
You can always adjust a trace’s width later through it’s properties panel.

Don’t forget to change it back to “netclass” after!

For reference

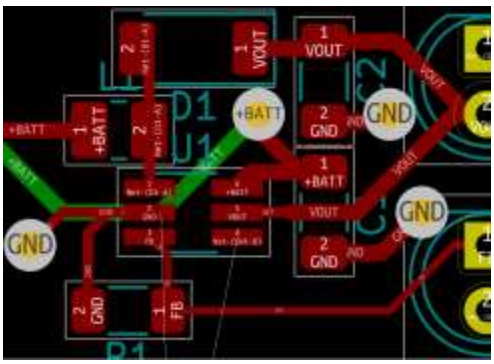
Press “/” to flip the trace’s bend angle.
“D” will drag a trace (better than “M”)

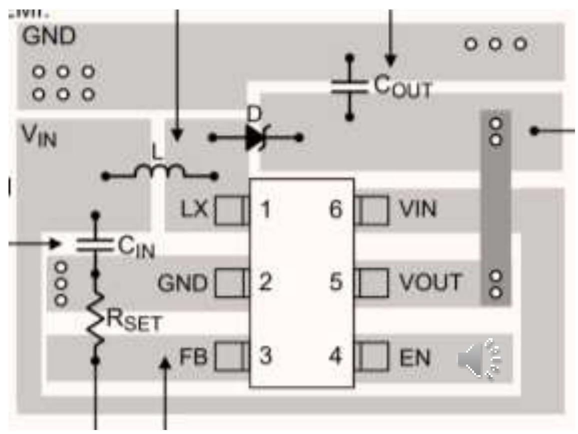




Okay, pause the video here and try to route the IC and its associated components, shown in my layout on the left, like the recommended layout on the right.

Routed!

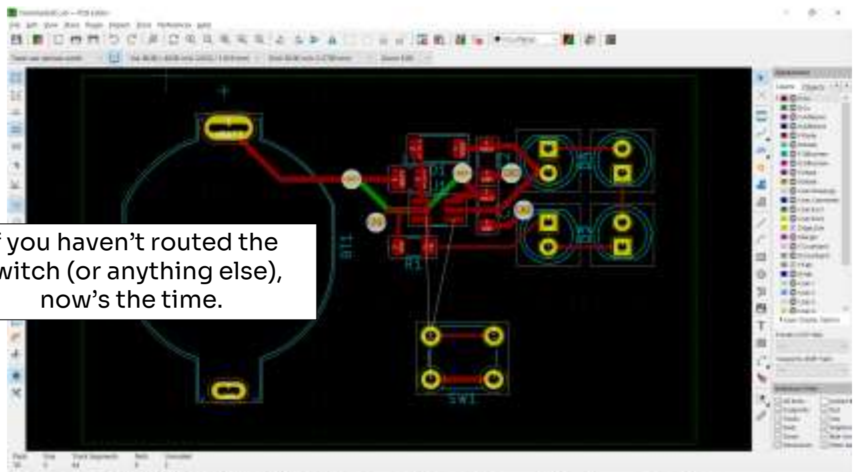




Not the prettiest routing job ever, but it'll probably work.

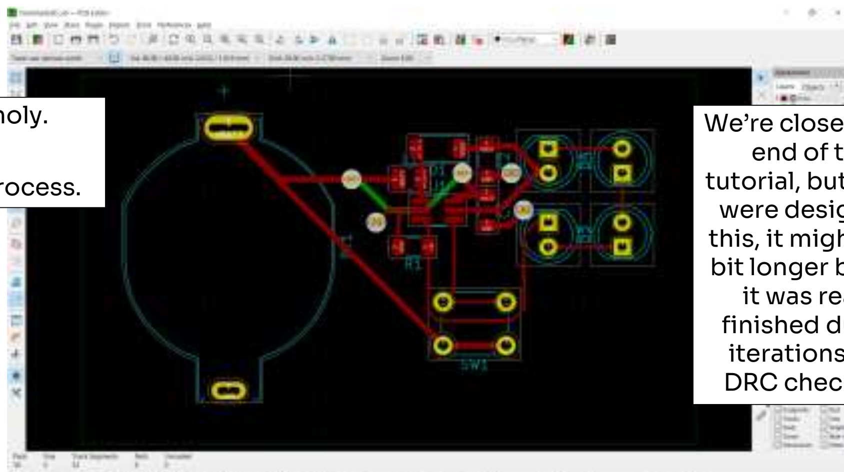
Part of the issue with matching the recommended layout is that our parts are much larger than the layout expectation, so they don't quite fit where the layout expects them.

Layout



If you haven’t routed the switch (or anything else), now’s the time.

Layout

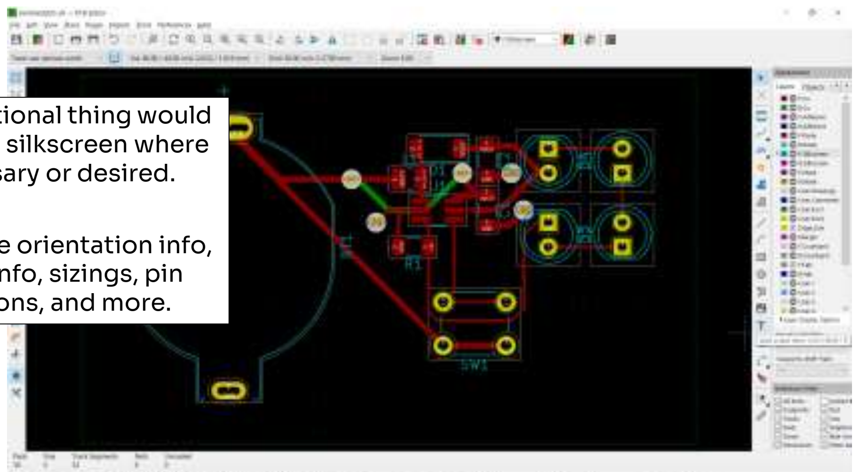


Holy moly.

What a process.

We're close to the end of the tutorial, but if you were designing this, it might be a bit longer before it was really finished due to iterations and DRC checking.

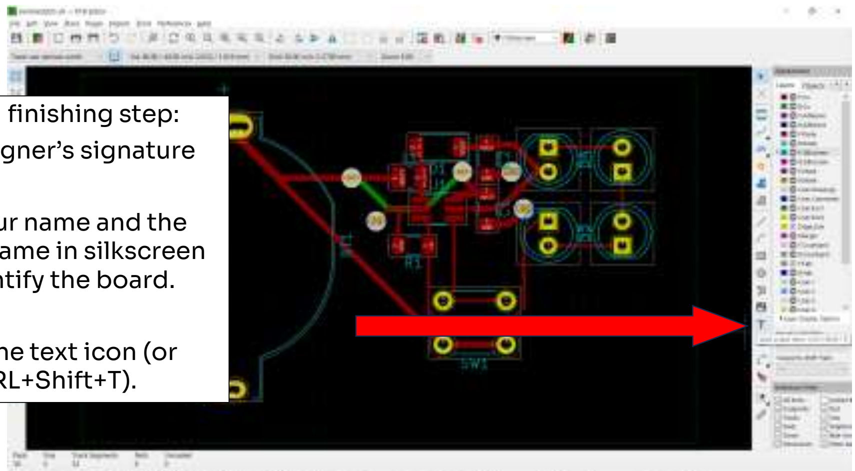
Layout



One additional thing would be to add silkscreen where necessary or desired.

Things like orientation info, usage info, sizings, pin functions, and more.

Layout



A small finishing step:
The designer's signature

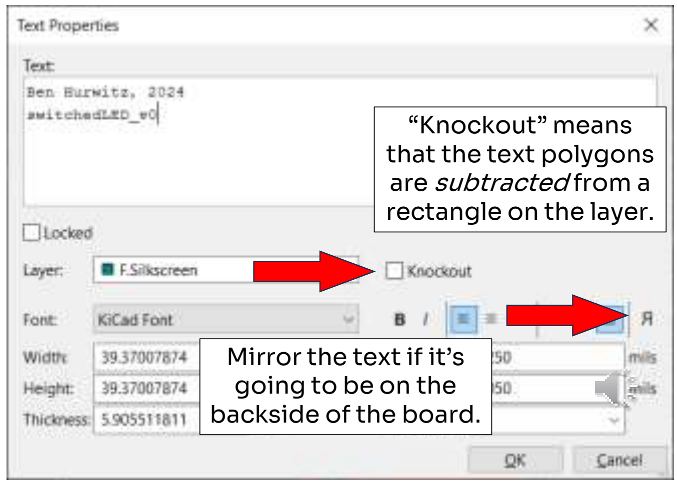
Add your name and the project name in silkscreen to identify the board.

Click the text icon (or CTRL+Shift+T).

Layout

Select the layer (typically silkscreen).

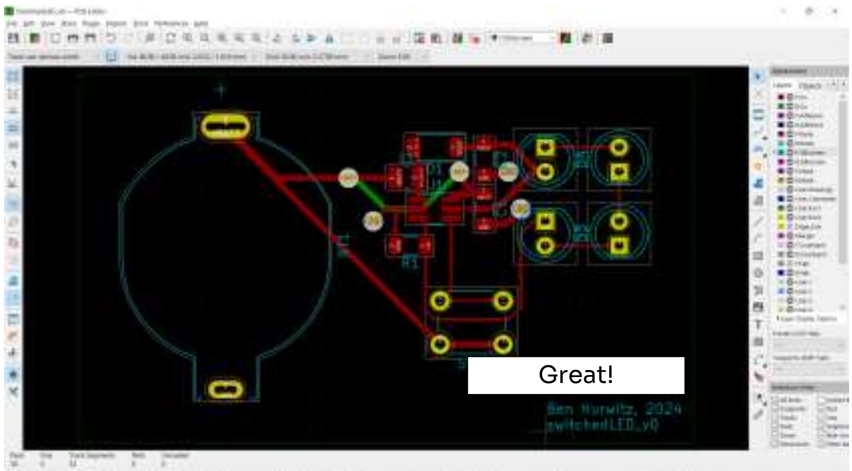
Adjust the font and size if you'd like.

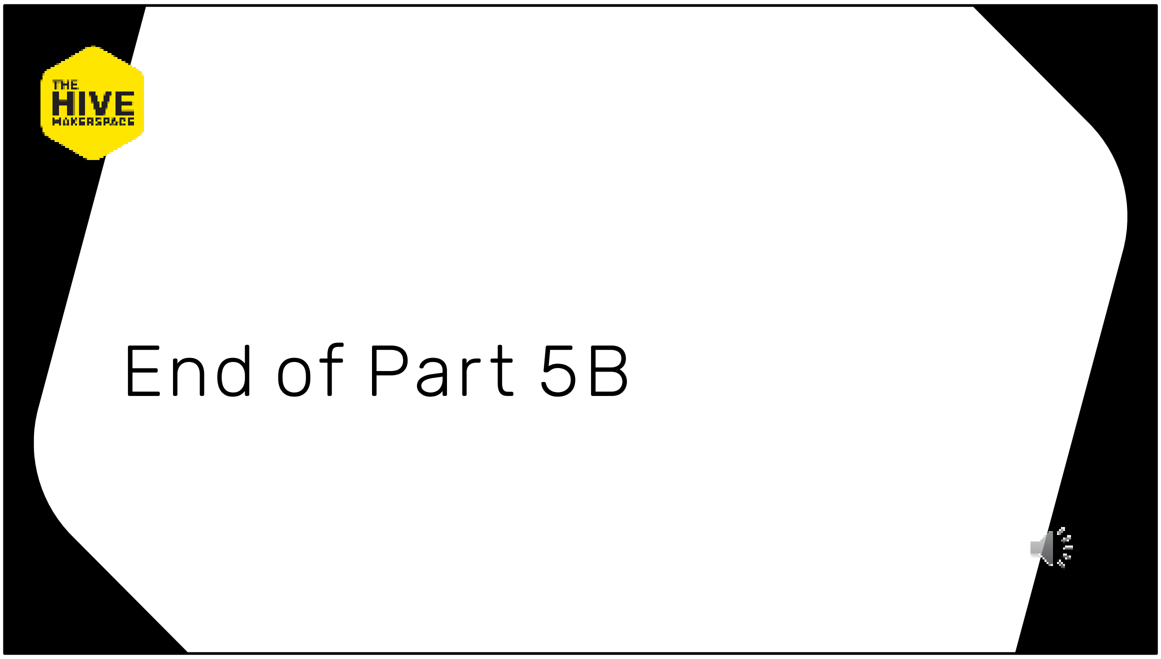


“Knockout” means that the text polygons are *subtracted* from a rectangle on the layer.

Mirror the text if it’s going to be on the backside of the board.

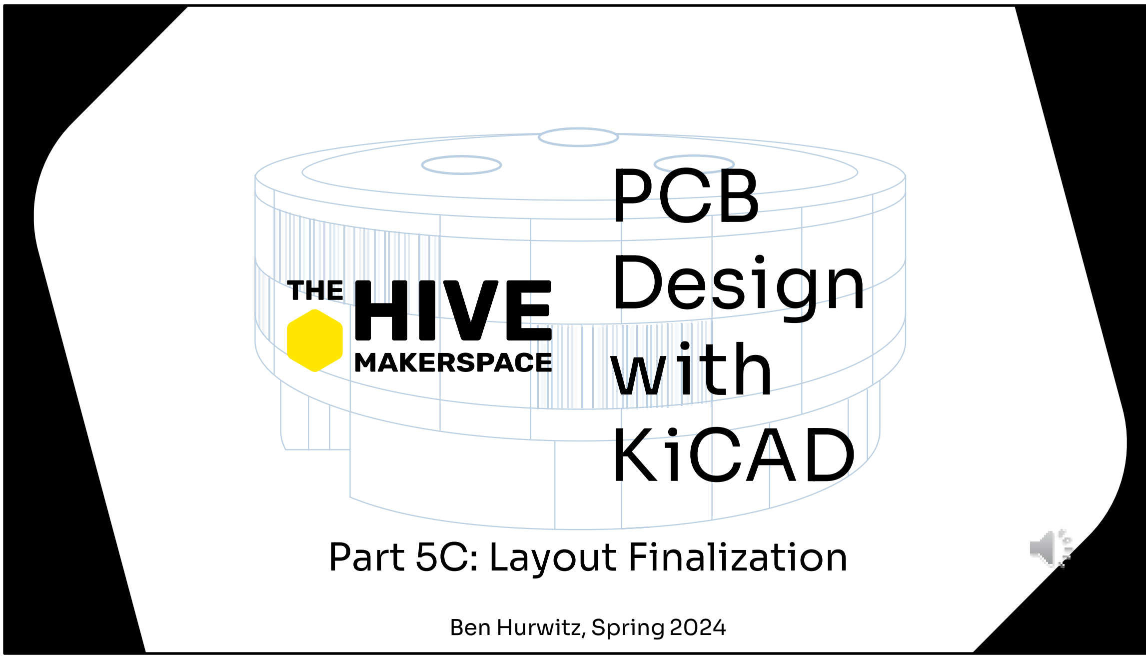
Layout





And with that, we end part 5B of our PCB design with KiCAD series with a nearly completed layout! All that's left is to check the mechanical dimensions of the various components, the DRC, and plot the gerbers. We'll talk about all that and more in part 5C. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

See you there.



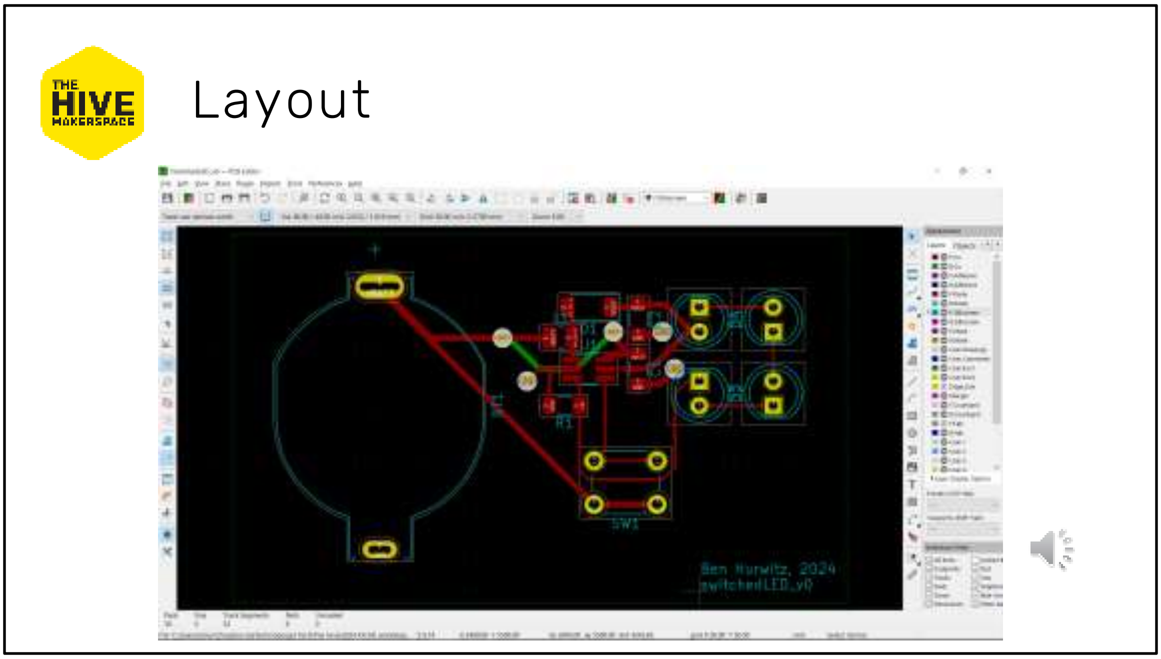
Hi all, welcome to The Hive's series on PCB Design with KiCAD. My name is Ben, and in this series, we've been walking through the PCB design process using KiCAD as our electronics design software.

Part 5 has been focused on the layout portion of the design.

In the last video, part 5B, we placed all the components and routed them together into a single cohesive layout.

In part 5C, we'll finish up the design process by confirming the board visually and through DRC, and then plotting our gerber files for fabrication and assembly.

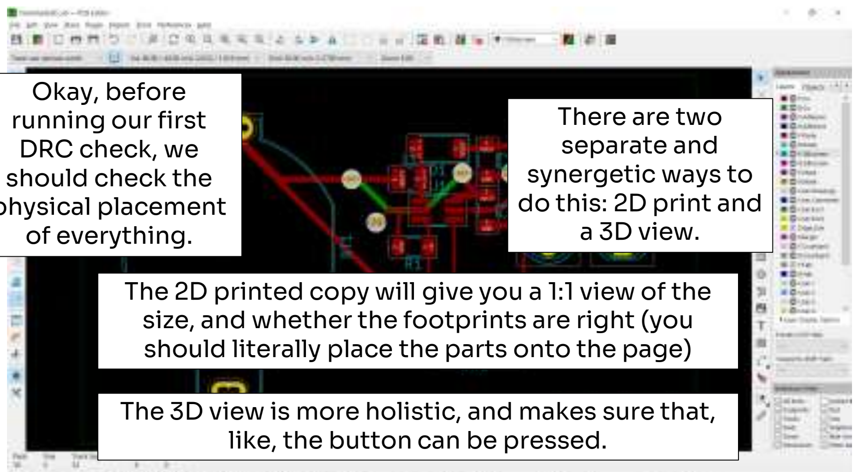
Let's get started.



My final layout looks like this. Yours may well look different, and that's okay. As long as everything is connected and there are no more air wires, those thin white ratsnest lines, then we can continue.

If you still have air wires in your design, pause the video here and finish routing first.

Print a copy/3D view



Okay, before running our first DRC check, we should check the physical placement of everything.

There are two separate and synergetic ways to do this: 2D print and a 3D view.

The 2D printed copy will give you a 1:1 view of the size, and whether the footprints are right (you should literally place the parts onto the page)

The 3D view is more holistic, and makes sure that, like, the button can be pressed.

Print a copy/3D view

Printing a 2D copy is easy.

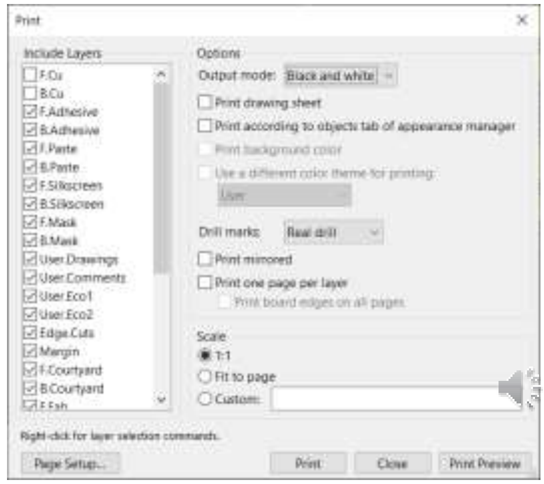
“File” > “Print”

Usually the default set of layers is good for mechanical scale.

Leave the drill marks as “Real drill” or the size will be wrong.

And keep the scale to 1:1.

Do a print preview first!



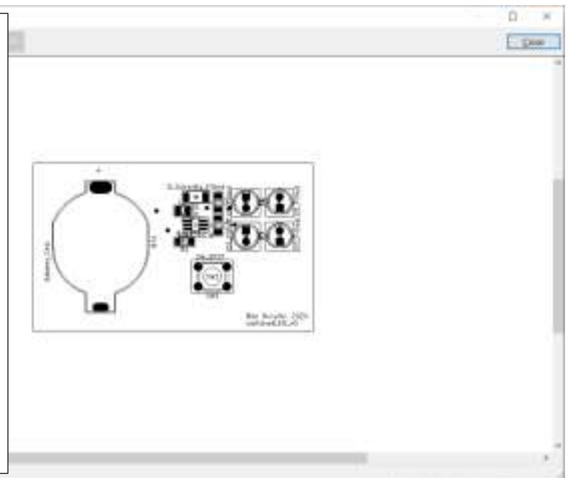
Print a copy/3D view

Here's the preview.

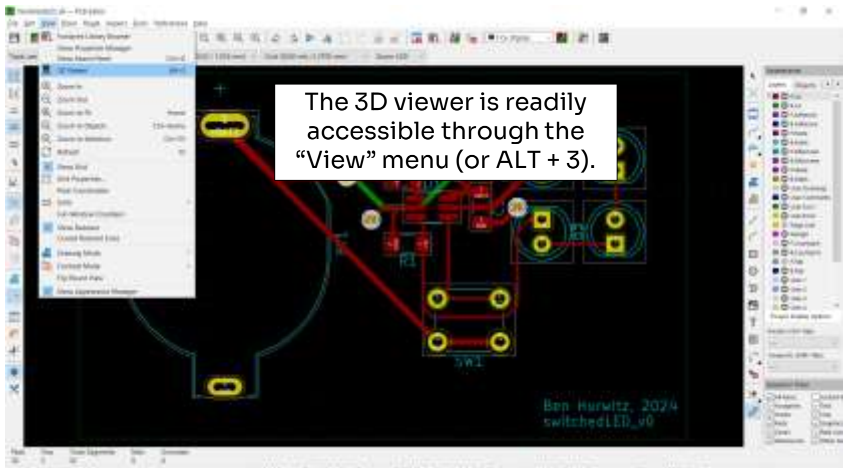
The routing isn't relevant because what we're looking at is the physical size and shape of the board and parts.

Physically place components on the board to confirm footprints.

Very helpful to avoid accidentally using the wrong footprint or a super tiny package.



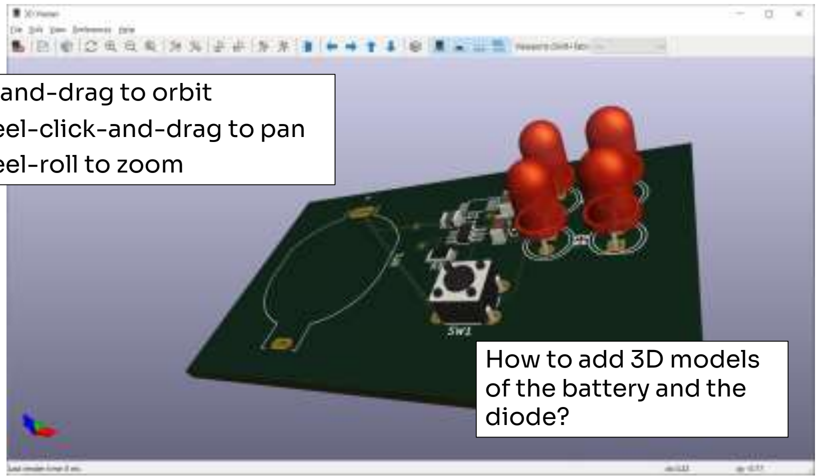
Print a copy/3D view



Print a copy/3D view

Print a copy/3D view

Left-click-and-drag to orbit
Scroll-wheel-click-and-drag to pan
Scroll-wheel-roll to zoom



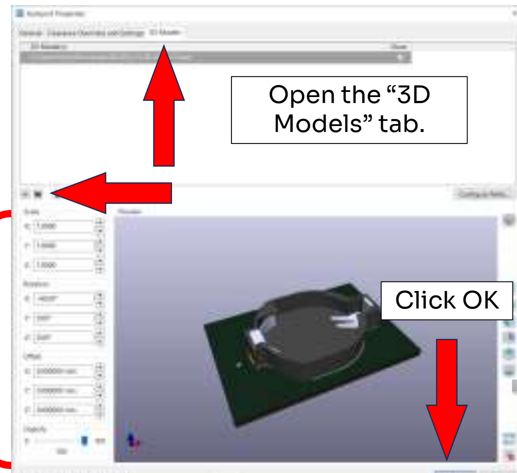
How to add 3D models of the battery and the diode?

Print a copy/3D view

From the layout view, open the properties window of the footprint in question (right-click, “Properties”).

Open and locate an acceptable file type (STEP or IGES, mostly)

Adjust the position as necessary to sit flush on the board.

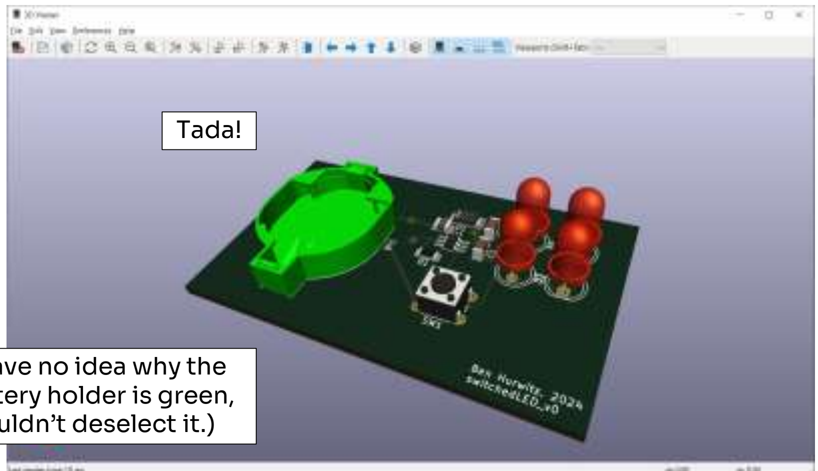


Open the “3D Models” tab.

Click OK

A well-designed 3D CAD model should have the origin aligned properly such that one or two rotations should have the part sitting flush, if that.

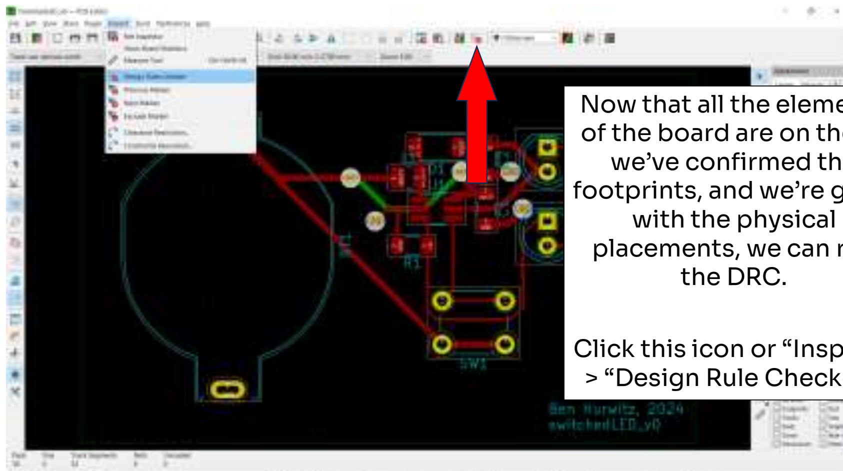
Print a copy/3D view



Tada!

(I have no idea why the battery holder is green, I couldn't deselect it.)

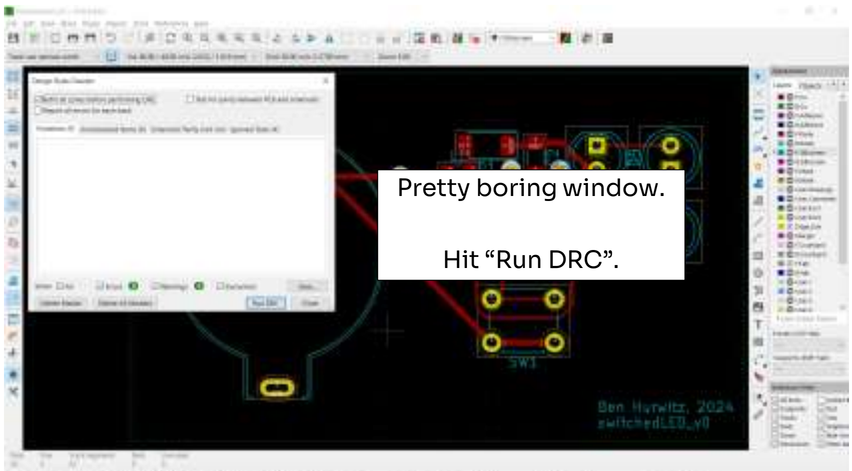
Layout



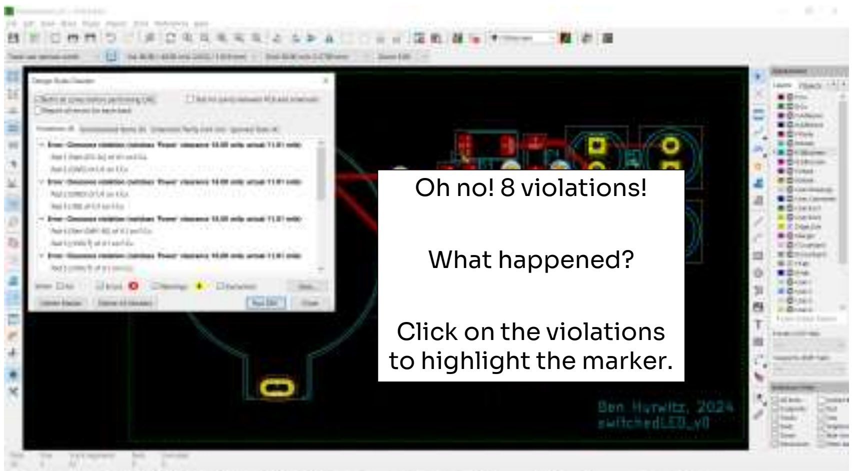
Now that all the elements of the board are on there, we've confirmed the footprints, and we're good with the physical placements, we can run the DRC.

Click this icon or "Inspect" > "Design Rule Checker"

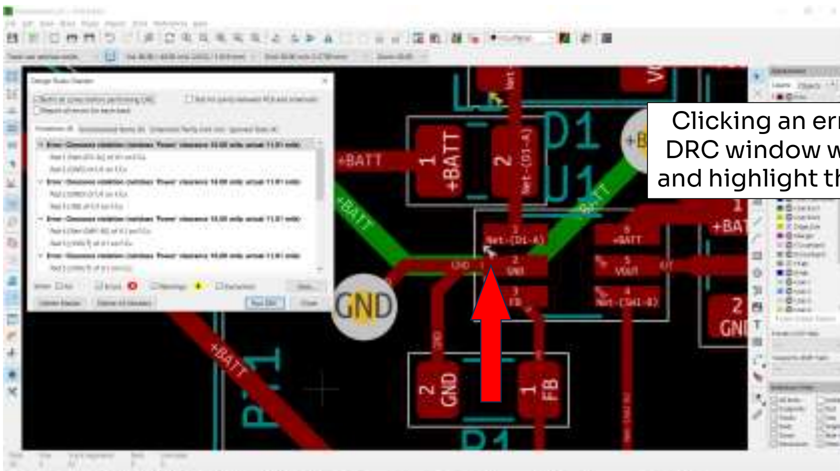
Layout


Pretty boring window.
Hit “Run DRC”.
Ben Hurwitz, 2024
switchedLED_v0
SW1

Layout

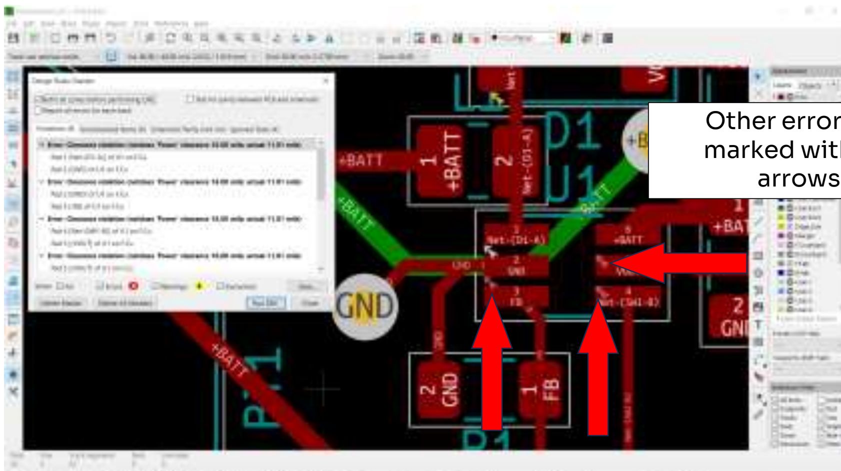

Oh no! 8 violations!
What happened?
Click on the violations to highlight the marker.
Ben Hurwitz, 2024
switchedLED_v0

Layout



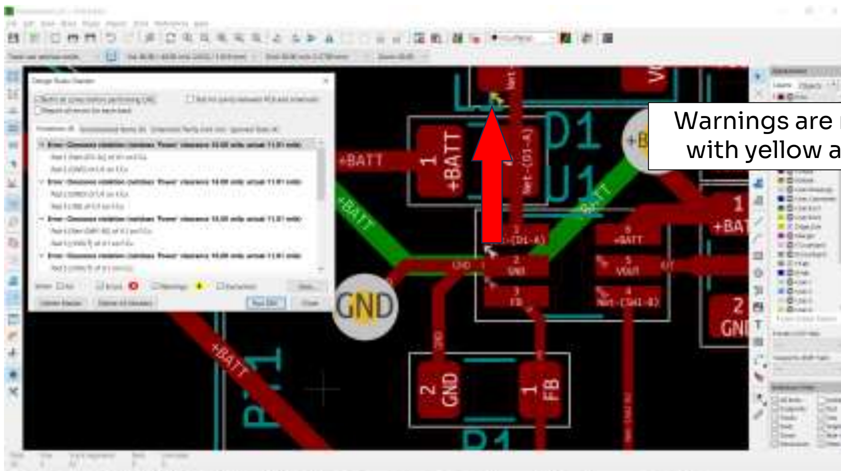
Clicking an error in the DRC window will center and highlight the marker.

Layout



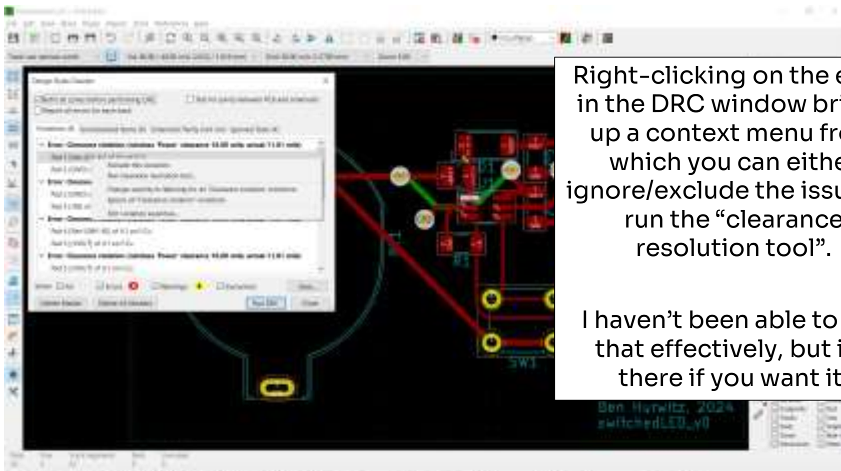
Other errors are marked with red arrows.

Layout



Warnings are marked with yellow arrows.

Layout



Right-clicking on the error in the DRC window brings up a context menu from which you can either ignore/exclude the issue, or run the “clearance resolution tool”.

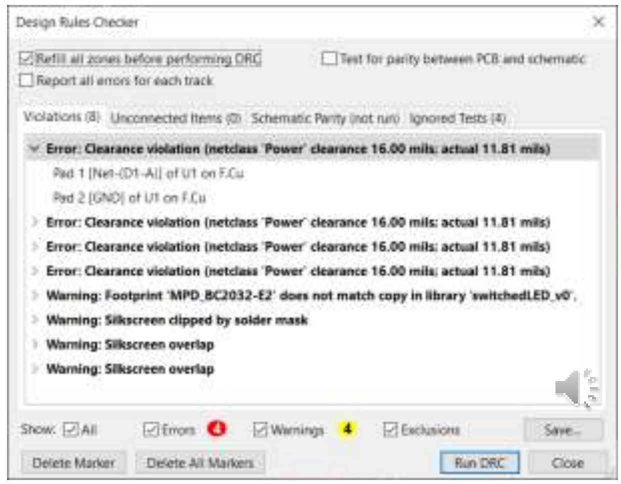
I haven’t been able to use that effectively, but it’s there if you want it.

Layout

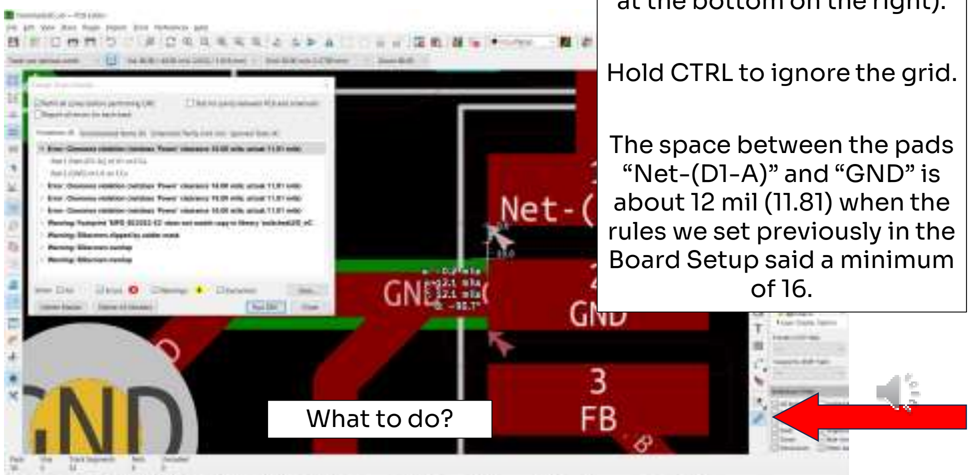
What are the errors?

The first four are "clearance violations" with things on the "Power" netclass, i.e. the clearance minimum was broken.

This means that something on a net within that netclass is too close to something not in that netclass.



Layout

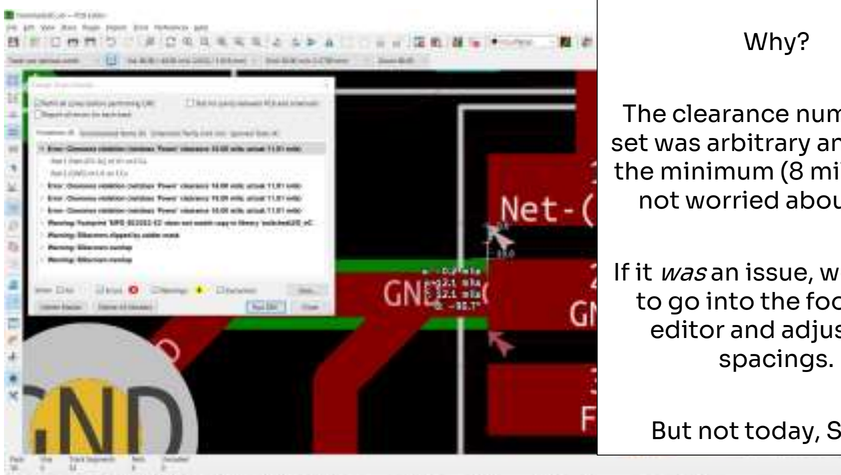


We see the issue by using the measure tool (CTRL+Shift+M, or the icon at the bottom on the right).

Hold CTRL to ignore the grid.

The space between the pads "Net-(D1-A)" and "GND" is about 12 mil (11.81) when the rules we set previously in the Board Setup said a minimum of 16.

Layout



Nothing – we can safely ignore this error.

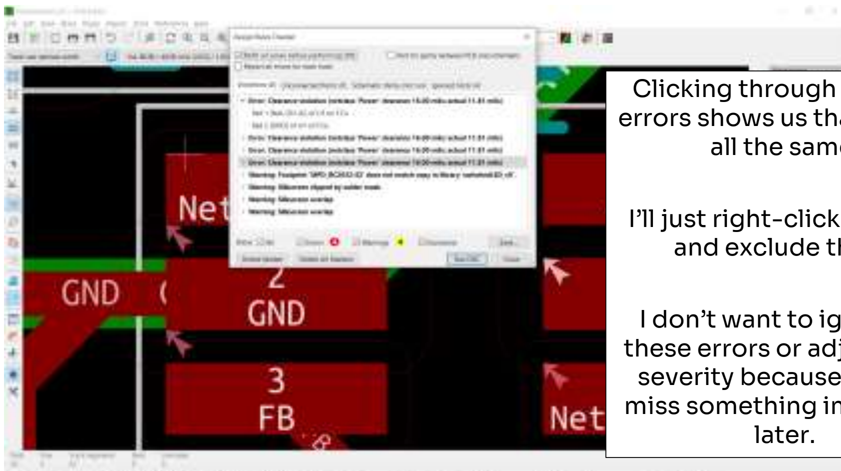
Why?

The clearance number we set was arbitrary and above the minimum (8 mil), so I'm not worried about this.

If it *was* an issue, we'd have to go into the footprint editor and adjust the spacings.

But not today, Satan.

Layout



Clicking through the four errors shows us that they're all the same.

I'll just right-click them all and exclude them.

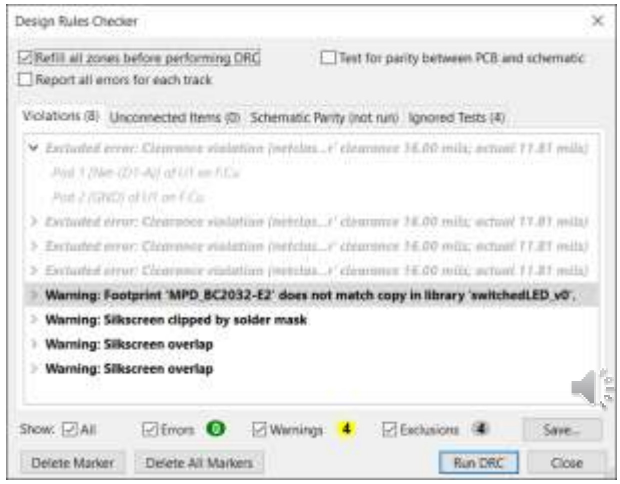
I don't want to ignore all these errors or adjust their severity because I might miss something important later.

Layout

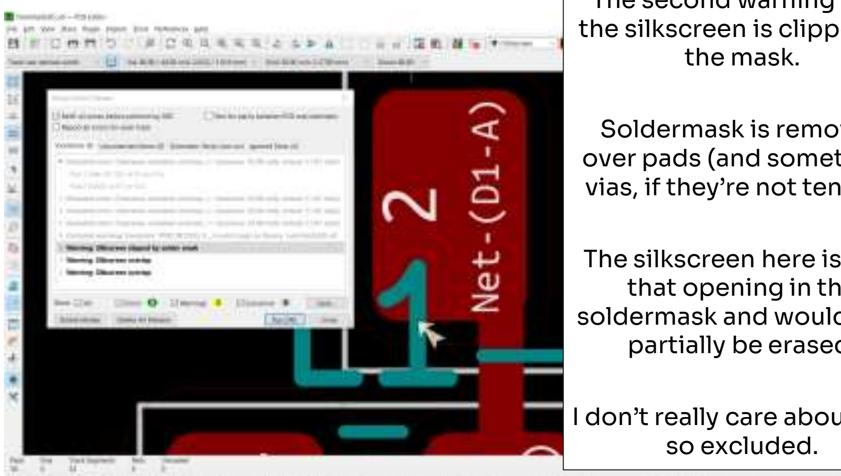
The first warning says that the battery footprint doesn't match what's in our library.

Well, of course it doesn't. We edited it.

Excluded.



Layout



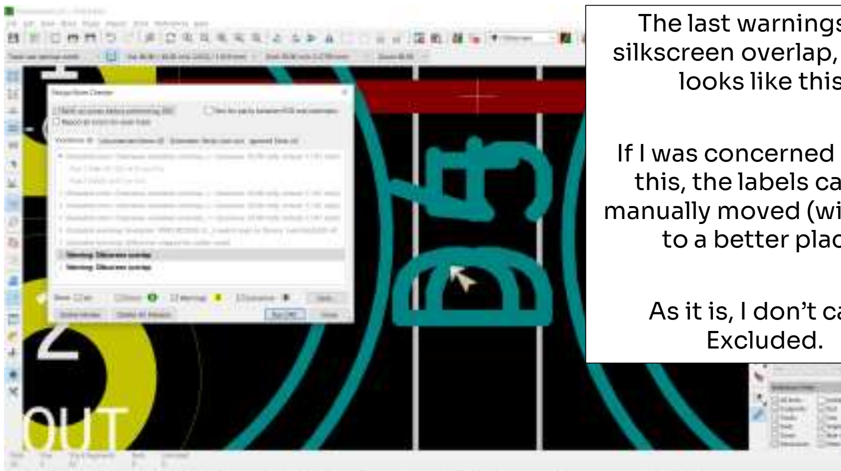
The second warning says the silkscreen is clipped by the mask.

Soldermask is removed over pads (and sometimes vias, if they're not tented).

The silkscreen here is over that opening in the soldermask and would thus partially be erased.

I don't really care about this, so excluded.

Layout

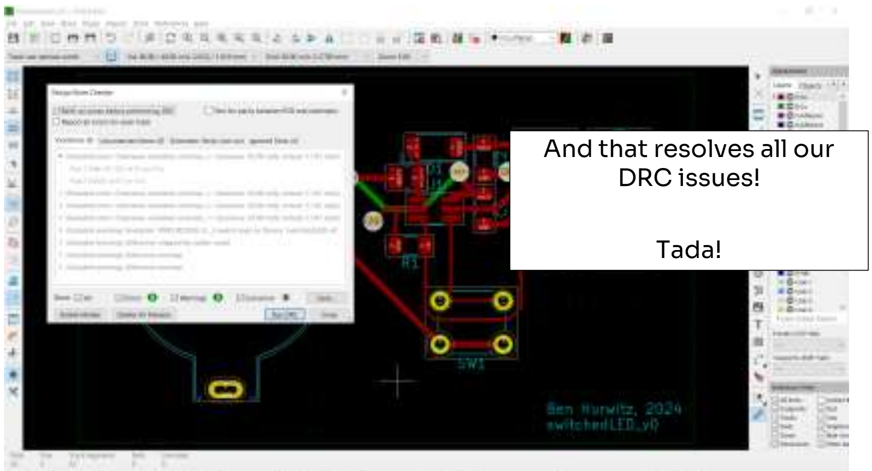


The last warnings are silkscreen overlap, which looks like this.

If I was concerned about this, the labels can be manually moved (with “M”) to a better place.

As it is, I don’t care. Excluded.

Layout


And that resolves all our DRC issues!
Tada!
R1
SW1
Ben Hurwitz, 2024
switchedLED_v0

The design is done!

Congratulations!

We have a fully designed
and checked board.

One final thing to do – plot the gerber files

Gerbers

Gerber files are special text files that define the polygons on each layer of your design.

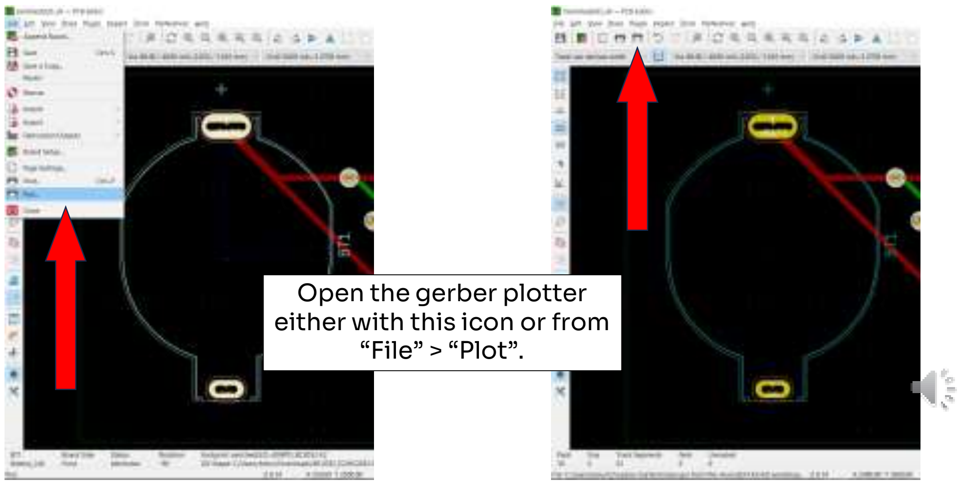
You typically need to sent them to the fab house for fabrication.

Some take KiCAD projects or board files directly these days.

Read the fab house's instructions for plotting these carefully!

It's easy to do something wrong and get a bad board.

Gerbers



Open the gerber plotter either with this icon or from "File" > "Plot".

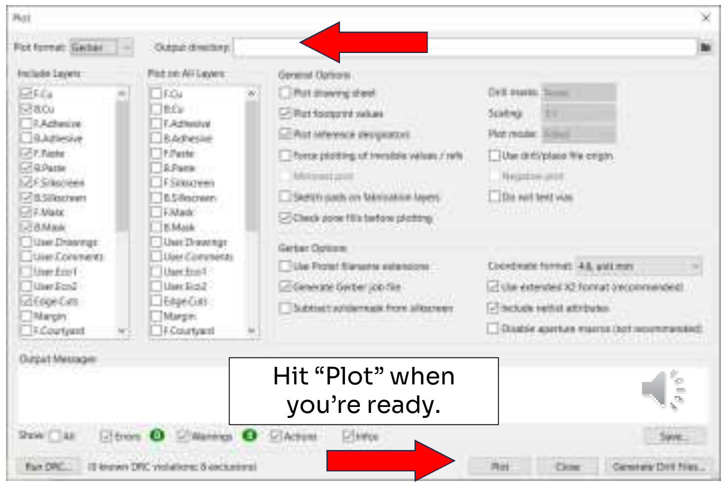
Layout

This is the plotting window.

Set an output directory for where to save the gerber files to.

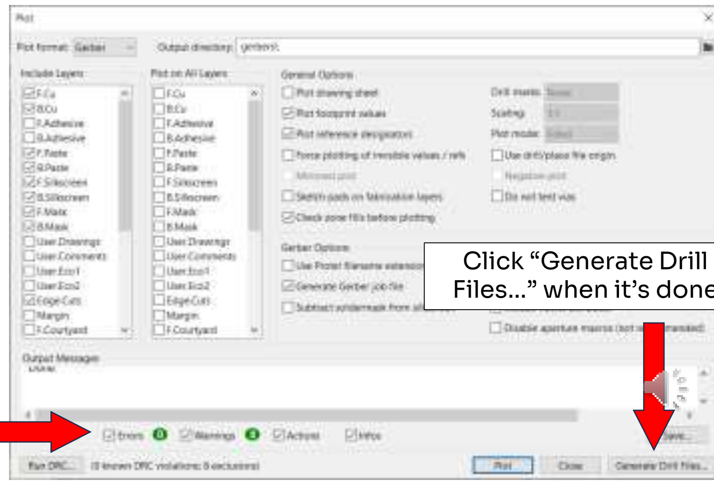
Read through your fab house's instructions carefully.

They will have details of what needs to be set in the middle.



Hit "Plot" when you're ready.

Layout



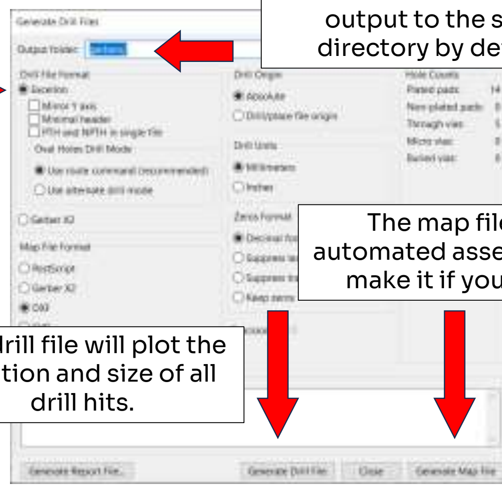
Click “Generate Drill Files…” when it’s done

Hopefully there are no errors.

Layout

Excellon is usually okay, but again, read the fab house instructions.

The rest of this will also be determined by your fab house.



The drill file or files will output to the same directory by default.

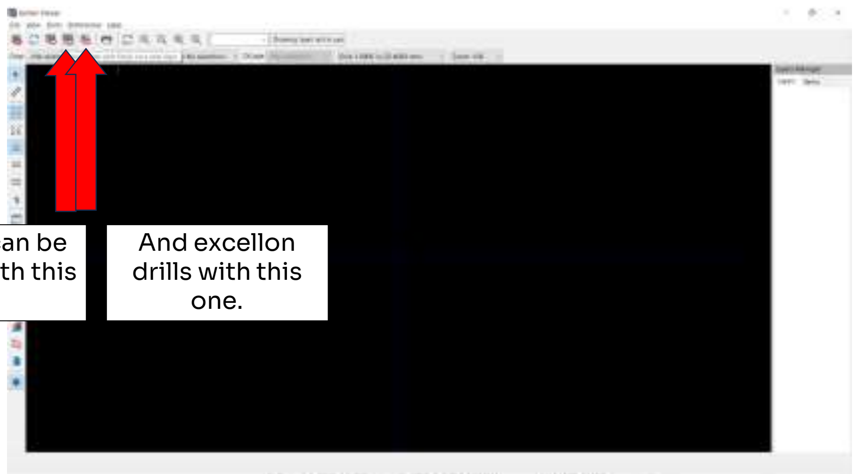
The map file is for automated assembly. Only make it if you need it.

The drill file will plot the position and size of all drill hits.

Gerber Viewer

Gerbers can be viewer
using the Gerber Viewer

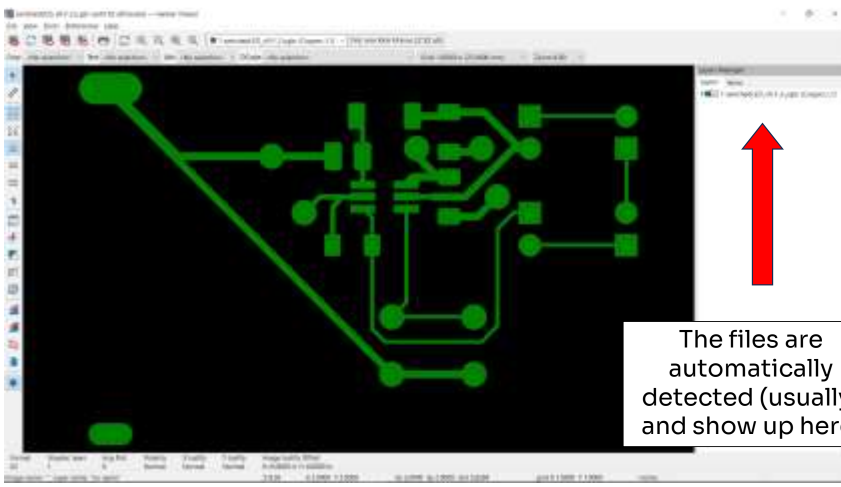
Gerber Viewer



Gerbers can be opened with this icon

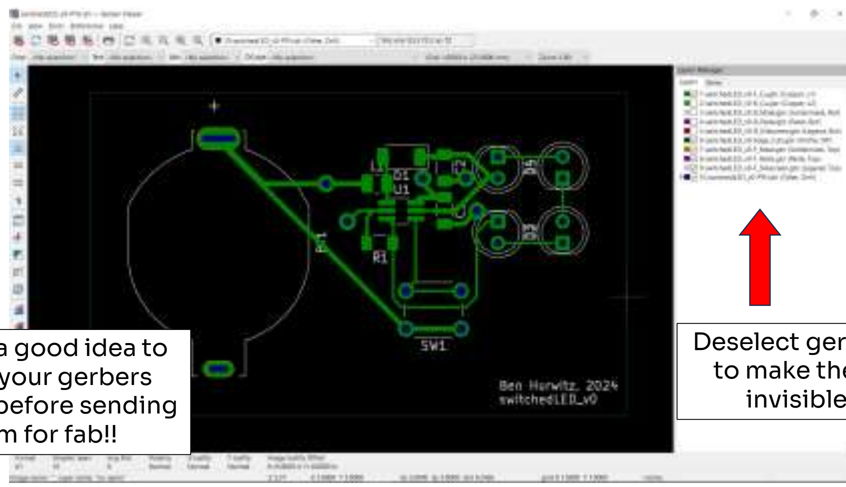
And excellon drills with this one.

Gerber Viewer



The files are automatically detected (usually) and show up here.

Gerber Viewer



Always a good idea to check your gerbers directly before sending them for fab!!

Deselect gerbers to make them invisible

Congratulations! You have a fully designed board that is ready to be sent for fabrication!

At this point, you could stop the video, but the net few slides I'll provide you with some additional rescourse and information for you to keep in your back pocket (or as a bookmark, if people still use those).

Next steps

- If you're interested in actually making this PCB:
  - You can fabricate the board here at The Hive yourself, or you can send the gerber files off to have it done for you
  - Email the Hive PCB group (hive-pcb@ece.gatech.edu) for the BOM, and we'll see if the Hive can purchase the components.
- Future revisions:
  - Consider re-selecting some parts for SMD only design, and then redo the layout using the backside for parts too.
  - The LEDs could be on a separate board that plugs in for a more point-and-shoot flashlight

Further Resources - KiCAD

- KiCAD documentation (https://docs.kicad.org/)
  - Everything about KiCAD and more, including “Getting Started”
- KiCAD library conventions (https://klc.kicad.org/)
  - The “Symbol” and “Footprint” subsections can help demystify the naming standards
- KiCAD forums (https://forum.kicad.info/) and resources
- KiCAD DRC templates (unofficial – GitHub)
- Plugin and Content Manager
  - Some are made by fab houses for official integration
  - Plenty of useful add-ons to the software

Further Resources - Design

- Design guidelines by David L. Jones (PDF) <- The real OG.
- Via guidelines (Cadence) and trace width requirements (Advanced Circuits)
- Phil's Lab (YouTube)
- SnapMagic has a desktop app with KiCAD integration
  - I just learned about this, but might be useful?
- Random, but The Hive's standard resistors are 7mm long with a body diameter of 2.5mm and a 0.6mm diameter leads.
  - For when you use those and need to select one of the many footprint options.

Further Resources – Fab

- Part sourcing: Digikey, Mouser, Octopart, Sparkfun, Adafruit, DFRobot, Amazon (YMMV), eBay (rare/hard to find)
- Fabrication houses: PCBShopper (price comparison), The Hive (local), OshPark (US, low cost), Advanced Circuits (US, mid-range), Epec (US, very fancy), Sunstone (US, mid-range), JLC PCB (CN), All PCB (CN)
  - Many have EDA-specific instructions

The Hive doesn't officially endorse any of these suppliers or fab houses; they all have plusses and minuses.

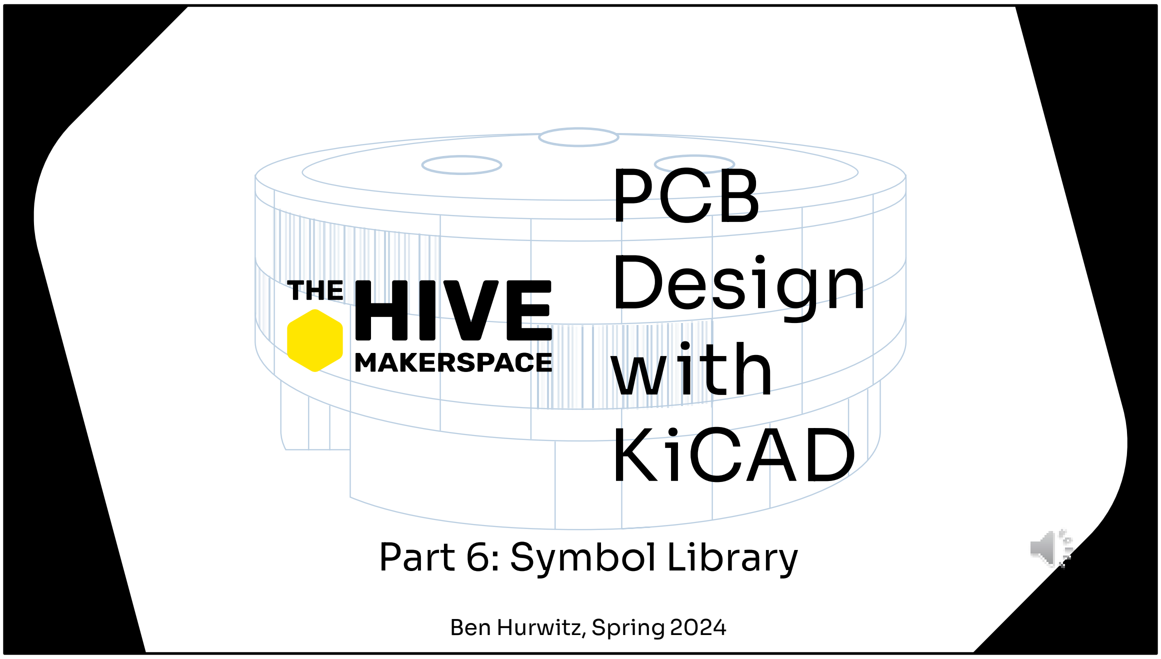
Further Resources – Misc

- Google.
  - Seriously, this is the thing you should get proficient with.
- Guides on design, KiCAD, and other EDA software are available on YouTube, Adafruit, Sparkfun, and many (many) more.
- Feel free to stop by The Hive to ask questions, as well. We're here to help with your design and fab!

And with that, we're done with part 5C and with the design! Congratulations. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

The remaining four videos cover library management and model creation. It's less exciting, but if you're thinking about doing more design, it's really valuable to understand how to keep your parts organized and ordered for later use and reuse. Part 6, which is next will look at symbol libraries, with some duplicate material from part 4. Part 7, which is split into three videos, will cover footprint libraries, and custom footprint generation.

Hope to see you there!



Hi all, welcome to The Hive's series on PCB Design with KiCAD. My name is Ben, and in this series, we've been walking through the PCB design process using KiCAD as our electronics design software.

The previous videos went through the design process all the way through, resulting in a complete PCB ready for fabrication. One thing that I mentioned during that process, and was featured in the original "EDA Design Flow" in part 2, was library management, and the idea of using only project-scope libraries, but when we actually did the design, I ignored this for simplicity and time-constraints.

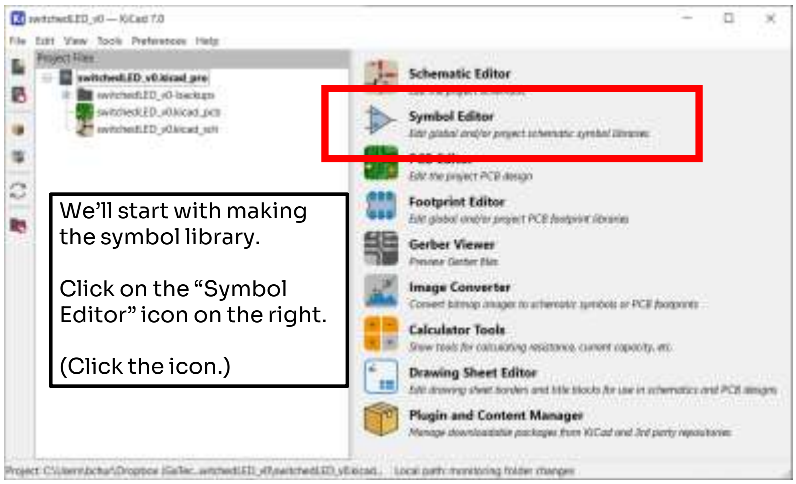
In this video, I will walk you through library selection and generating a single project-scoped symbol library to package with the rest of your project, and keep your work insulated from external changes.

This material is of course not required for a functional design, but it is good design practice, for KiCAD at least, to keep all your parts in a project-level library.

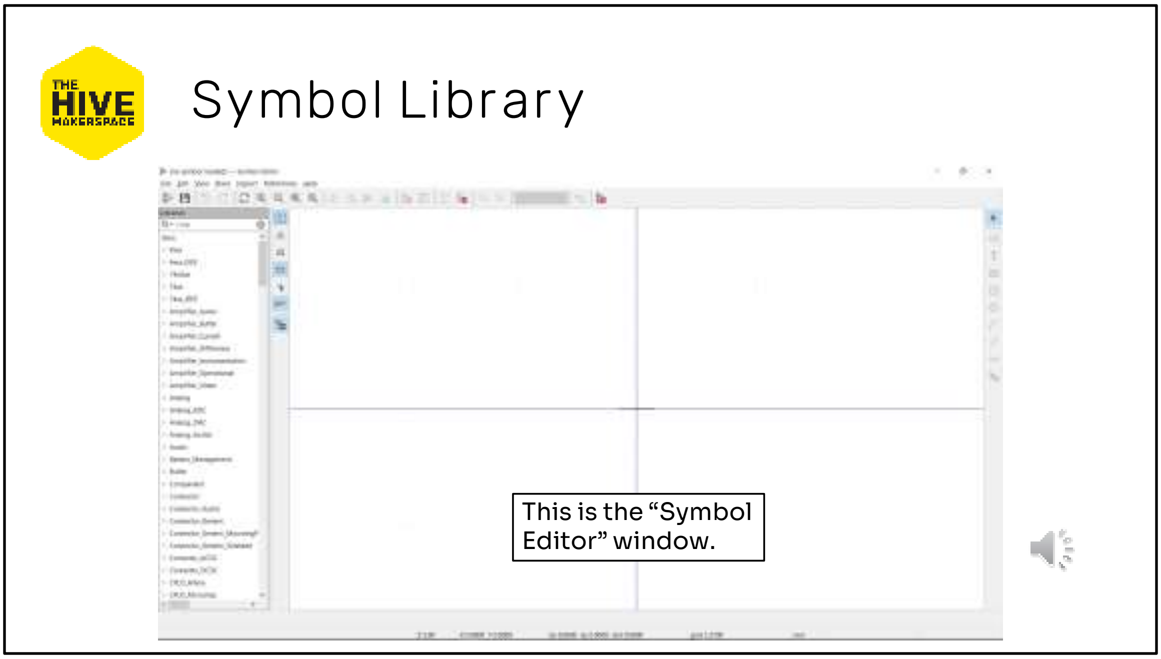
Because this is not related directly to the design flow of the previous videos, I'll make no assumptions about the state of your system or knowledge. So I apologize if some of this is repetition for some of you.

Let’s get started.

Creating the model libraries

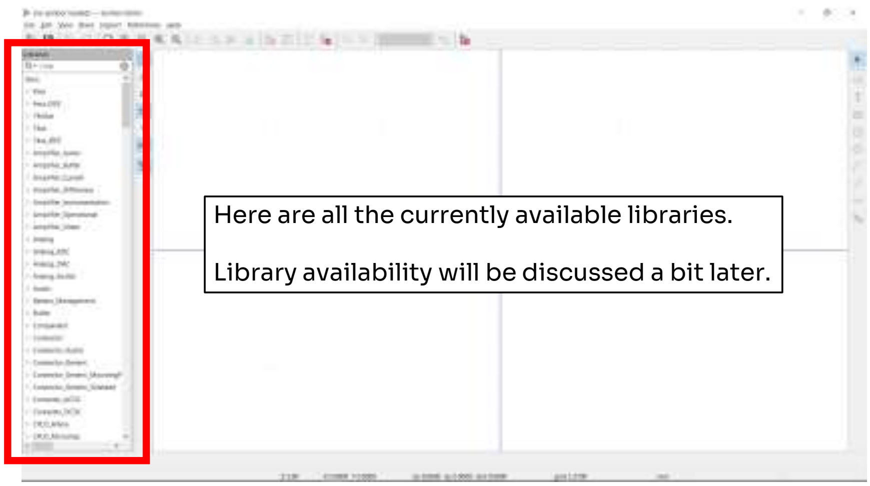


After creating the project, typically the first thing you'd want to do is create a single library where all of your components will live. This will be a living library, as in, components will be added to this library throughout the design iterations. Try not to remove any components since you never know what you'll need again. *



If you’re familiar with this window, the next few slides will be a review.

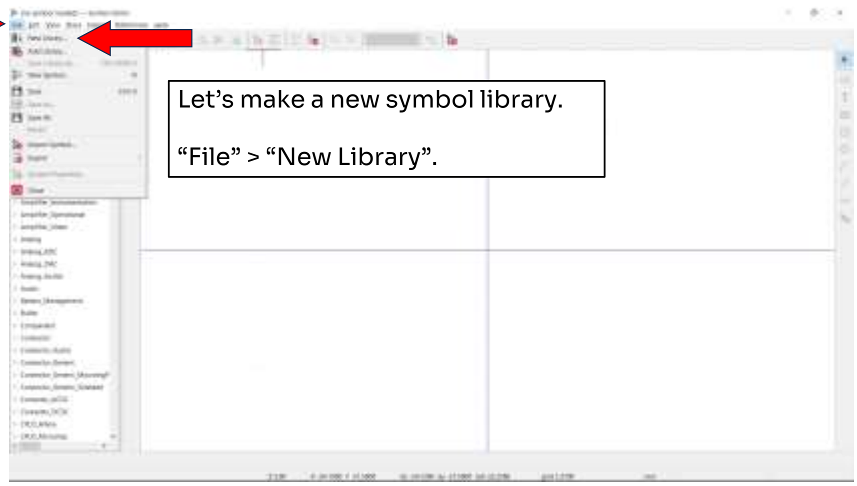
Symbol Library



Here are all the currently available libraries.

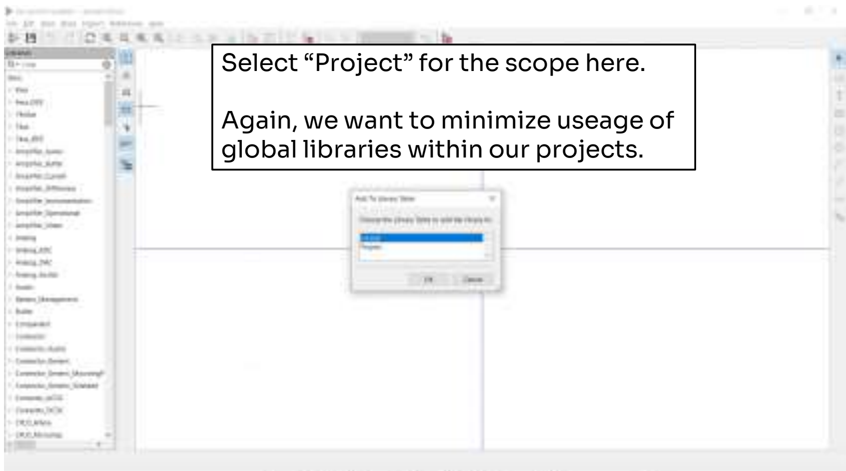
Library availability will be discussed a bit later.

Symbol Library



If you’ve already gone through the design process in the previous videos, don’t make a new library, since we’ll just be using the flashlight circuit we developed there.

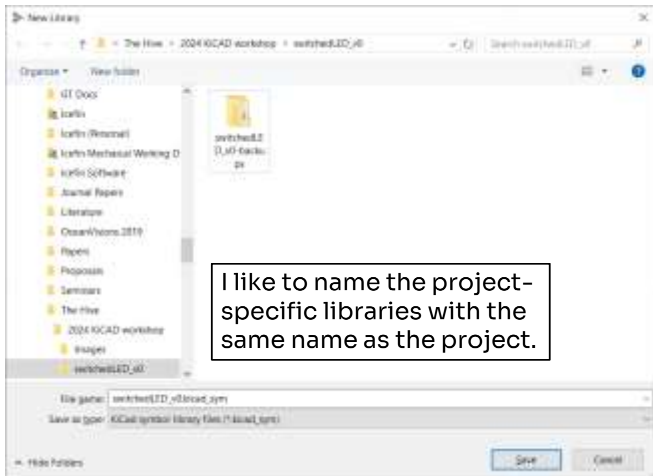
Symbol Library



Select “Project” for the scope here.

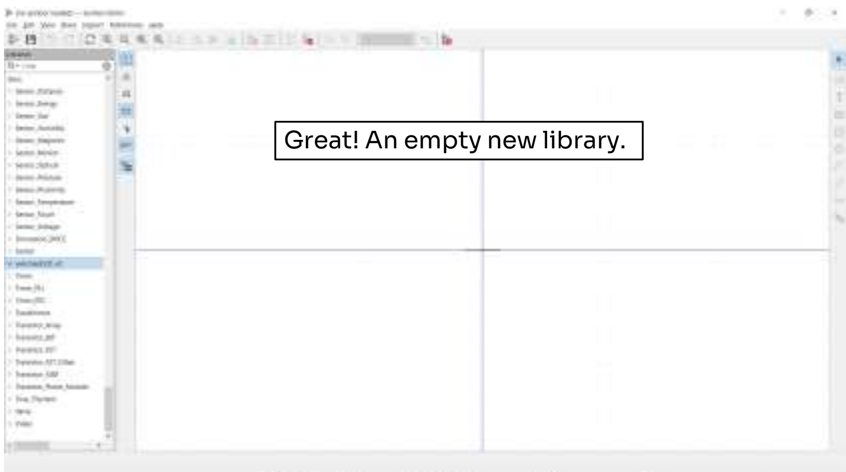
Again, we want to minimize useage of global libraries within our projects.

Symbol Library



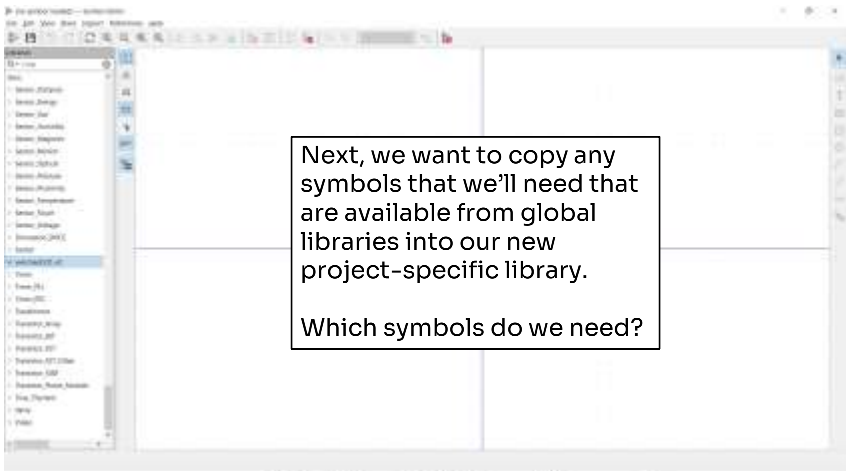
I like to name the project-specific libraries with the same name as the project.

Symbol Library



Again, if you've previously made a library for the flashlight circuit developed in videos 1-5C, move forward here with that library.

Symbol Library

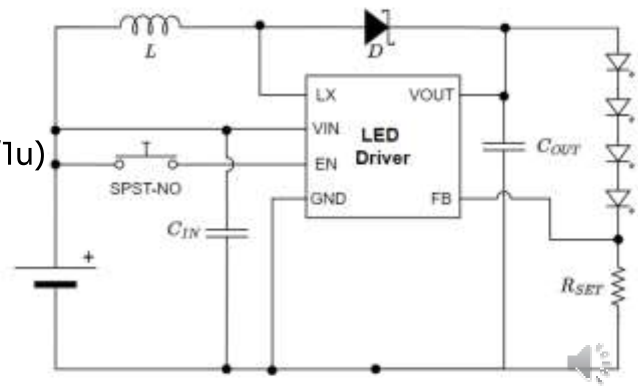


Next, we want to copy any symbols that we'll need that are available from global libraries into our new project-specific library.

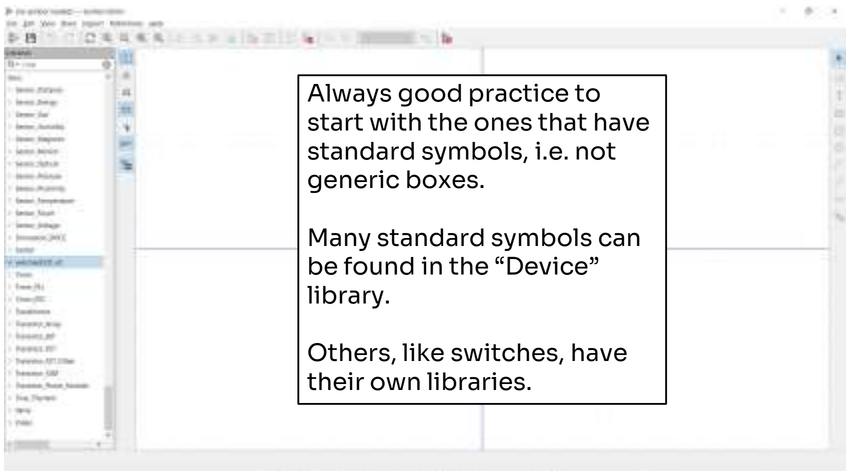
Which symbols do we need?

The flashlight circuit

- LED driver IC
- Battery + holder
- SPST-NO switch
- Cin, Cout (caps) (2.2u/1u)
- L (inductor) (22uH)
- D (Schottky diode)
- Rset (resistor) (30 Ω)
- 4x LEDs



Symbol Library

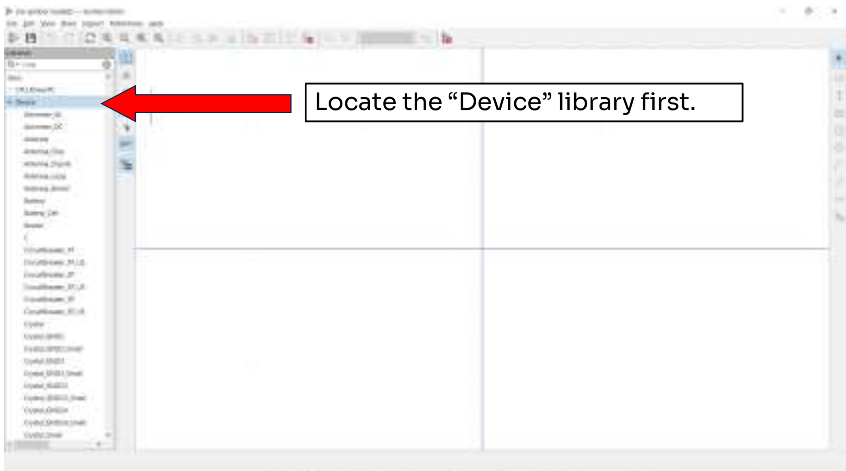


Always good practice to start with the ones that have standard symbols, i.e. not generic boxes.

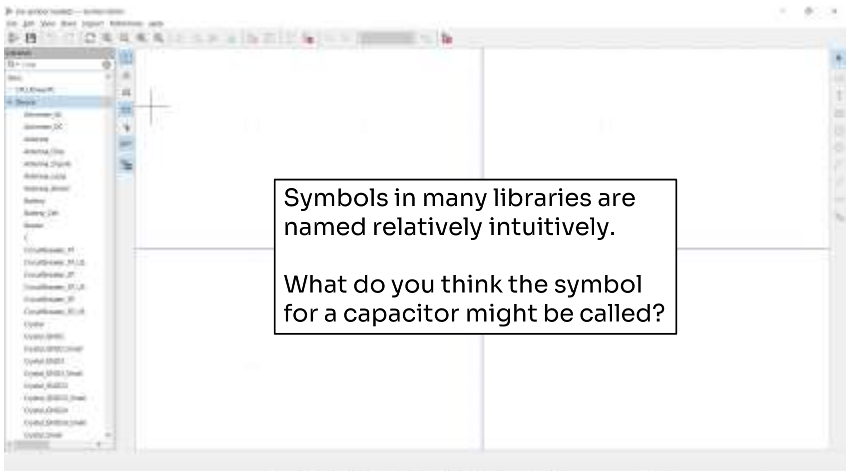
Many standard symbols can be found in the “Device” library.

Others, like switches, have their own libraries.

Symbol Library



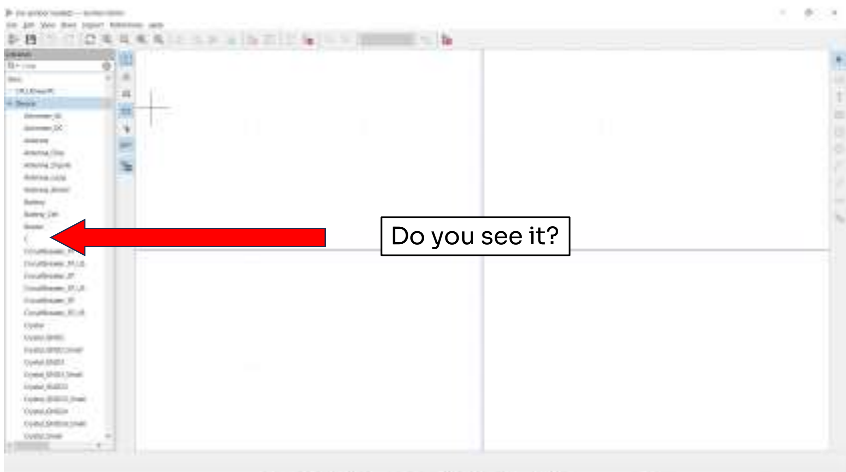
Symbol Library



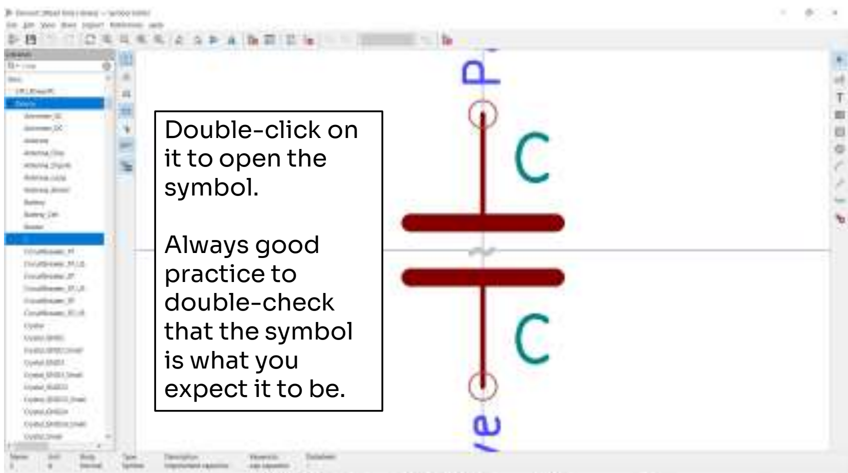
Symbols in many libraries are named relatively intuitively.

What do you think the symbol for a capacitor might be called?

Symbol Library



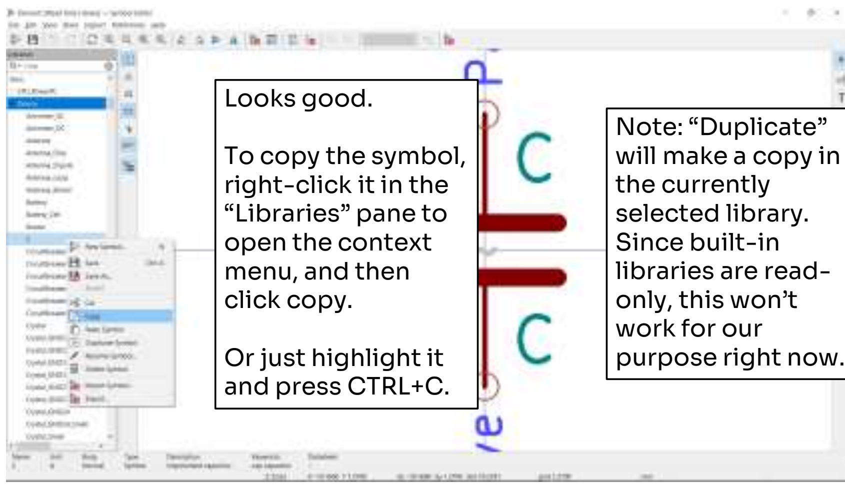
Symbol Library



Double-click on it to open the symbol.

Always good practice to double-check that the symbol is what you expect it to be.

Symbol Library



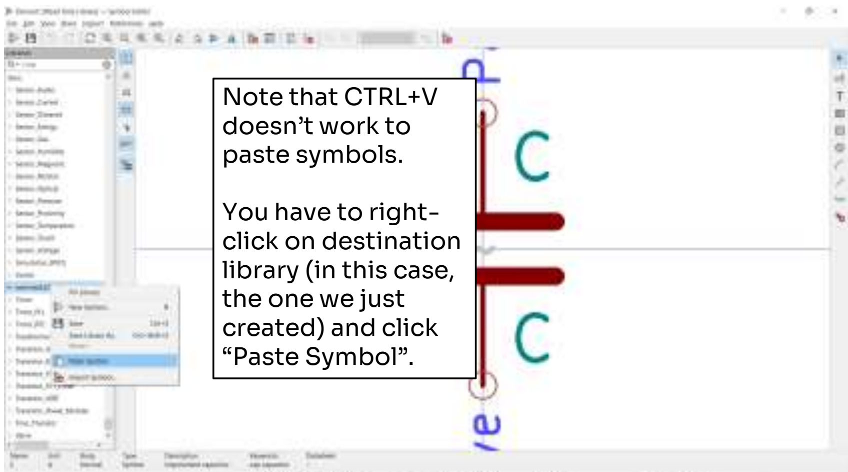
Looks good.

To copy the symbol, right-click it in the "Libraries" pane to open the context menu, and then click copy.

Or just highlight it and press CTRL+C.

Note: "Duplicate" will make a copy in the currently selected library. Since built-in libraries are read-only, this won't work for our purpose right now.

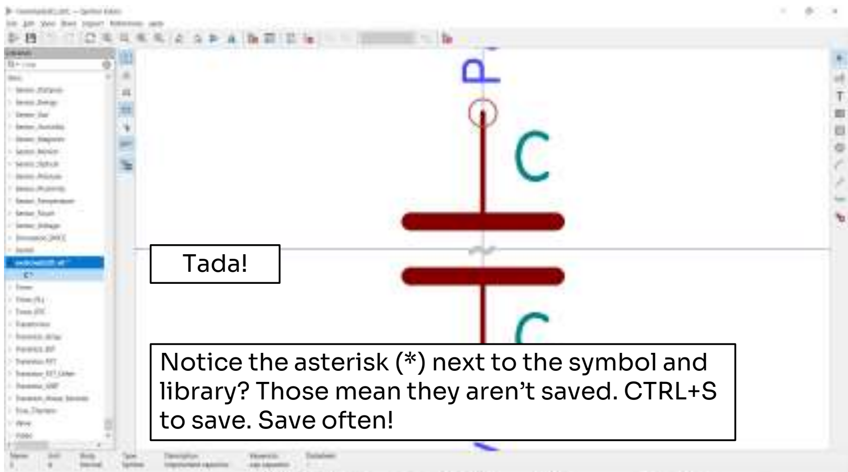
Symbol Library



Note that CTRL+V doesn't work to paste symbols.

You have to right-click on destination library (in this case, the one we just created) and click "Paste Symbol".

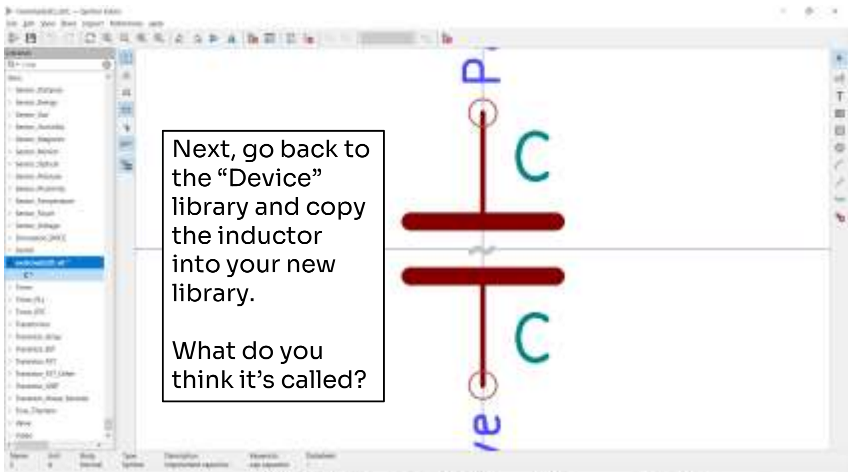
Symbol Library



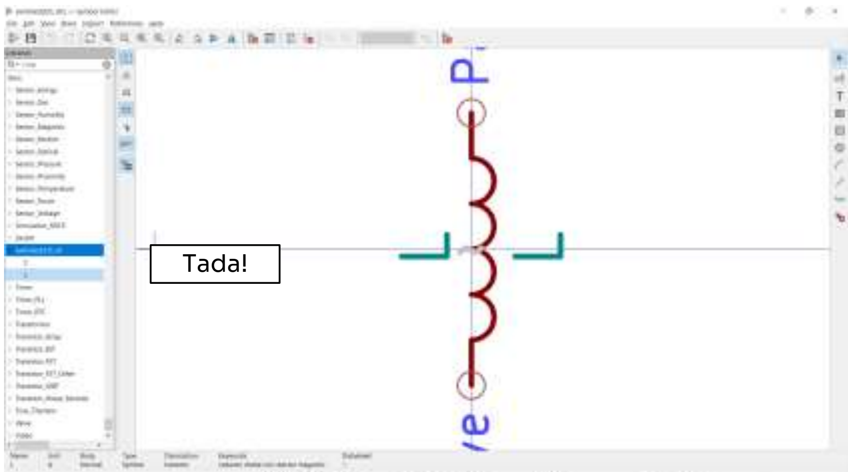
Tada!

Notice the asterisk (*) next to the symbol and library? Those mean they aren't saved. CTRL+S to save. Save often!

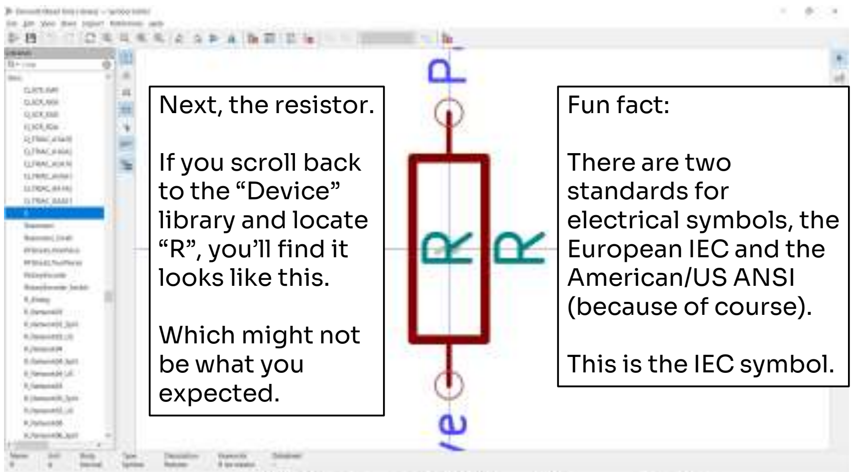
Symbol Library



Symbol Library



Symbol Library



Next, the resistor.

If you scroll back to the “Device” library and locate “R”, you’ll find it looks like this.

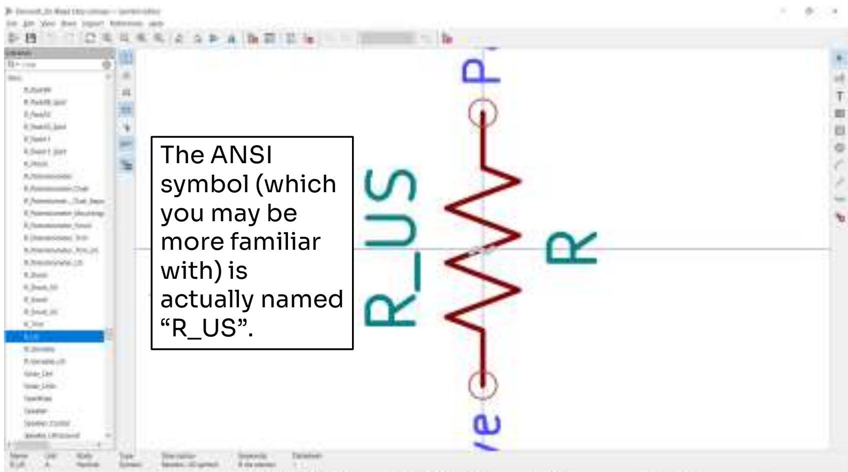
Which might not be what you expected.

Fun fact:

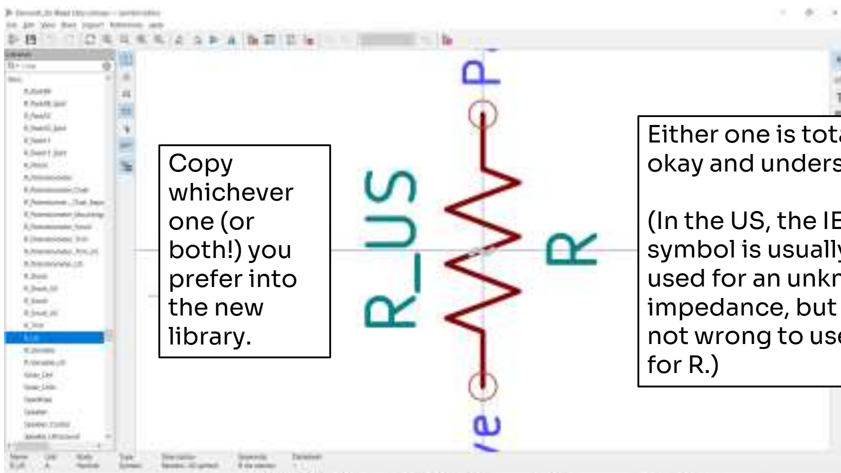
There are two standards for electrical symbols, the European IEC and the American/US ANSI (because of course).

This is the IEC symbol.

Symbol Library



Symbol Library

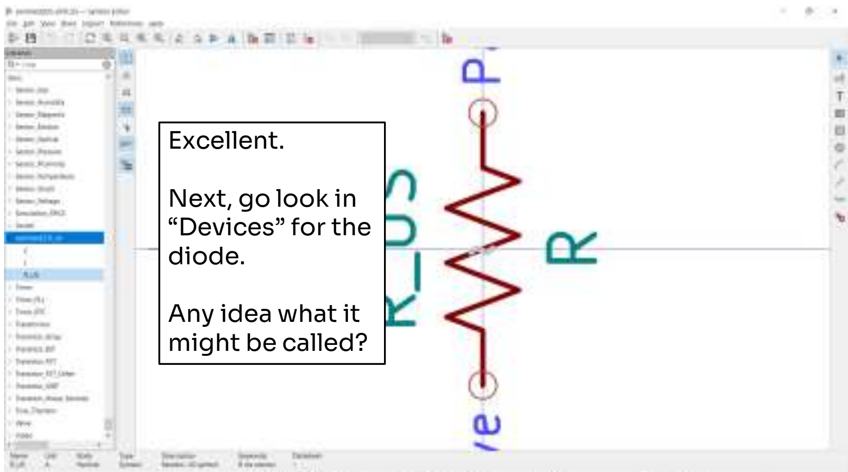


Copy whichever one (or both!) you prefer into the new library.

Either one is totally okay and understood.

(In the US, the IEC symbol is usually used for an unknown impedance, but it's not wrong to use it for R.)

Symbol Library

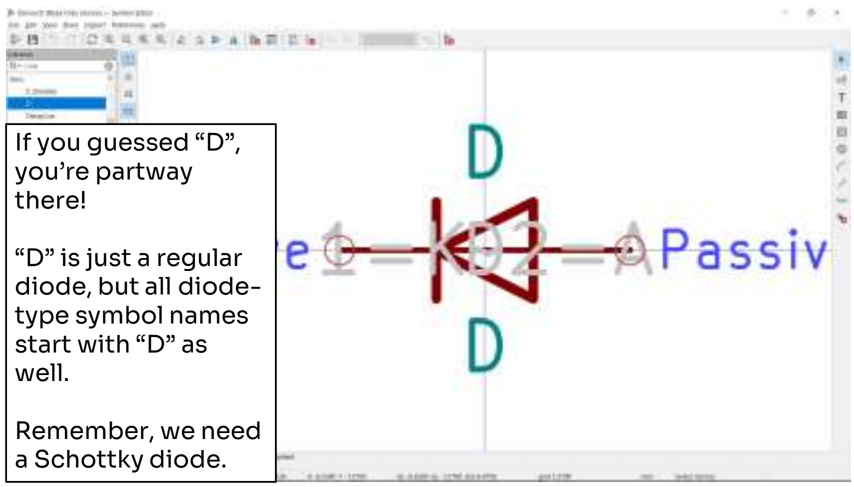


Excellent.

Next, go look in "Devices" for the diode.

Any idea what it might be called?

Symbol Library

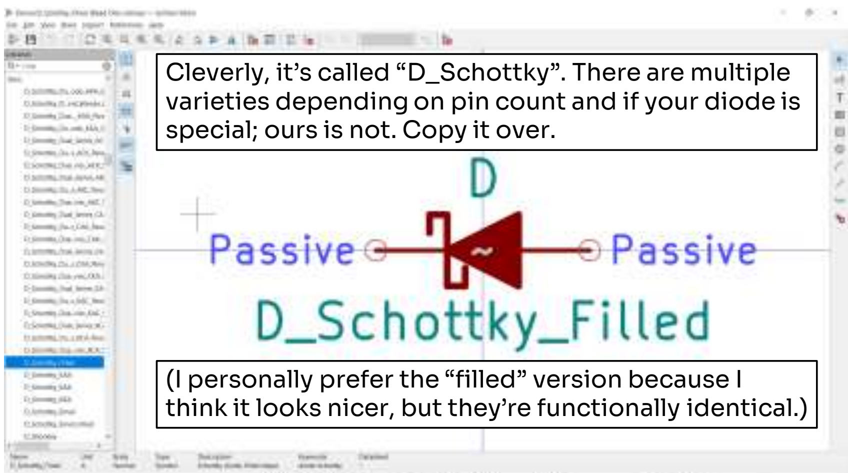


If you guessed “D”, you’re partway there!

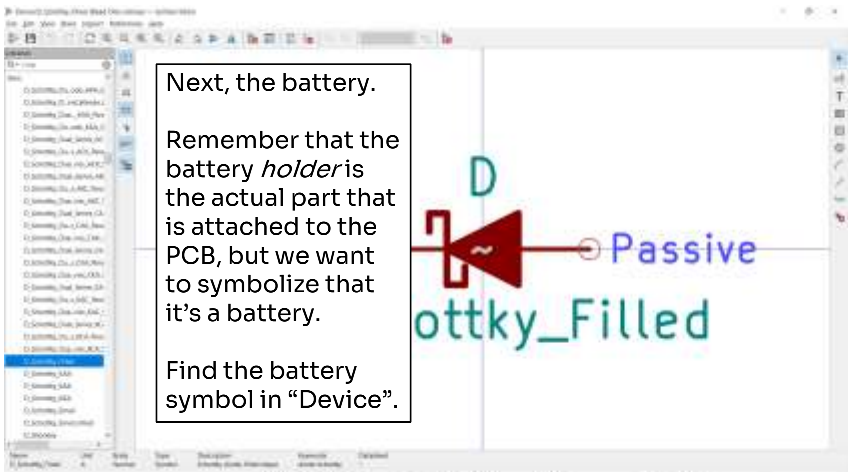
“D” is just a regular diode, but all diode-type symbol names start with “D” as well.

Remember, we need a Schottky diode.

Symbol Library



Symbol Library

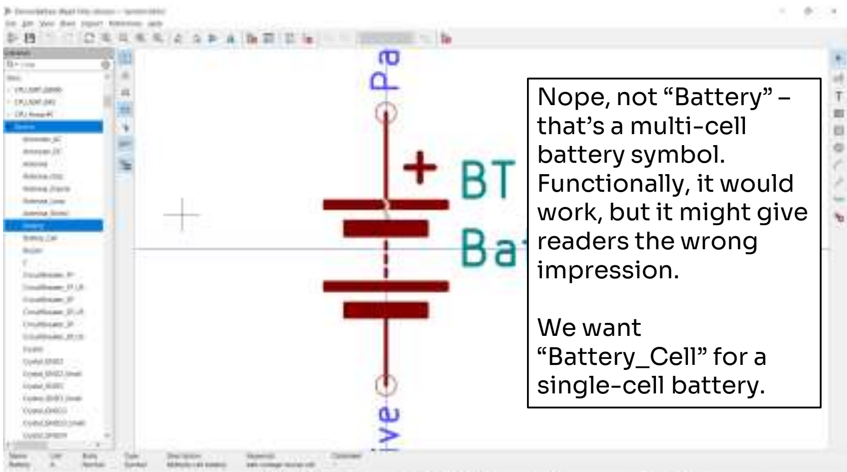


Next, the battery.

Remember that the battery *holder* is the actual part that is attached to the PCB, but we want to symbolize that it's a battery.

Find the battery symbol in “Device”.

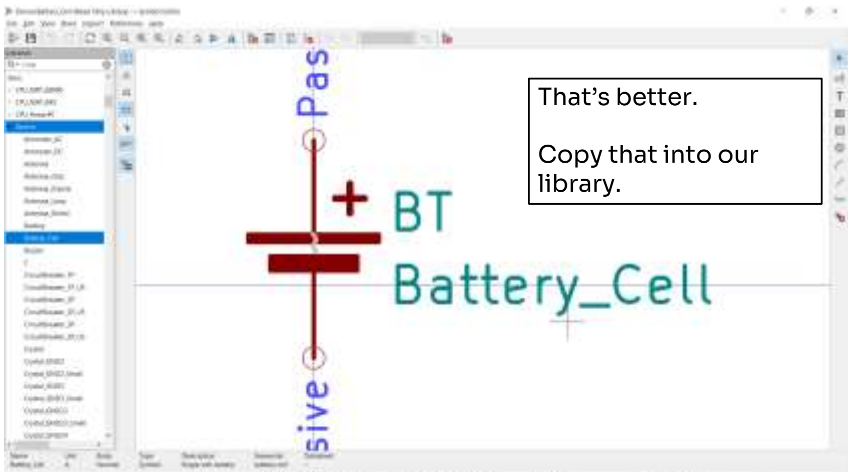
Symbol Library



Nope, not “Battery” – that’s a multi-cell battery symbol. Functionally, it would work, but it might give readers the wrong impression.

We want “Battery_Cell” for a single-cell battery.

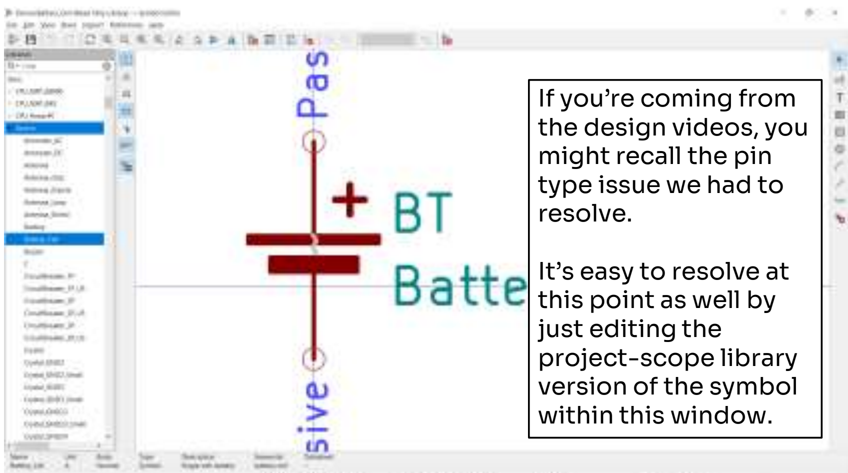
Symbol Library



That’s better.

Copy that into our library.

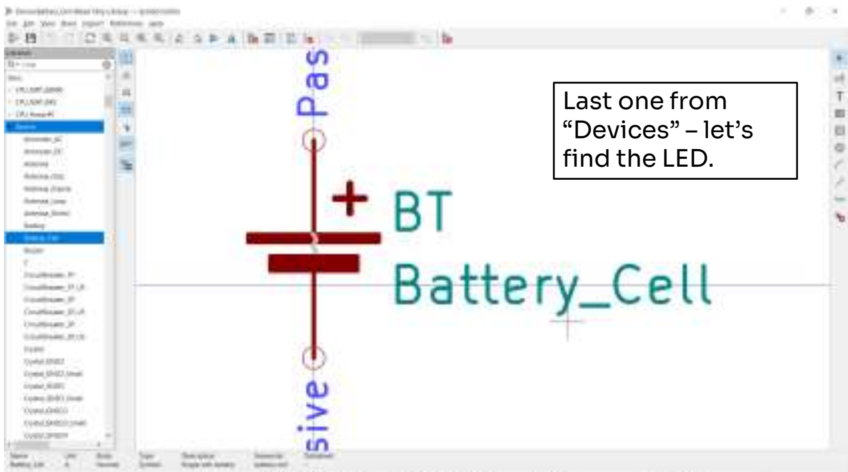
Symbol Library



If you're coming from the design videos, you might recall the pin type issue we had to resolve.

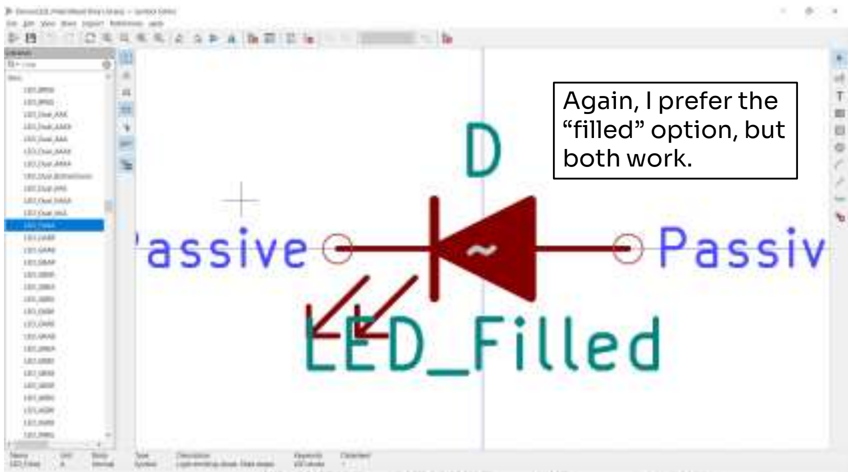
It's easy to resolve at this point as well by just editing the project-scope library version of the symbol within this window.

Symbol Library



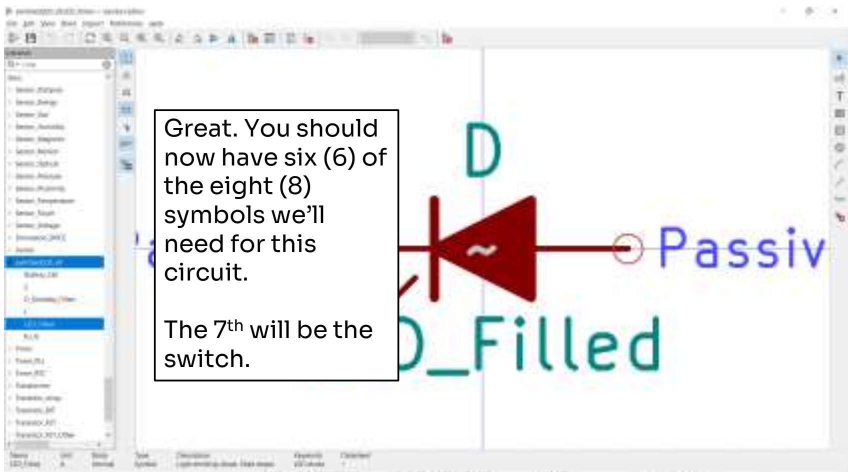
Last one from "Devices" – let's find the LED.

Symbol Library



Again, I prefer the “filled” option, but both work.

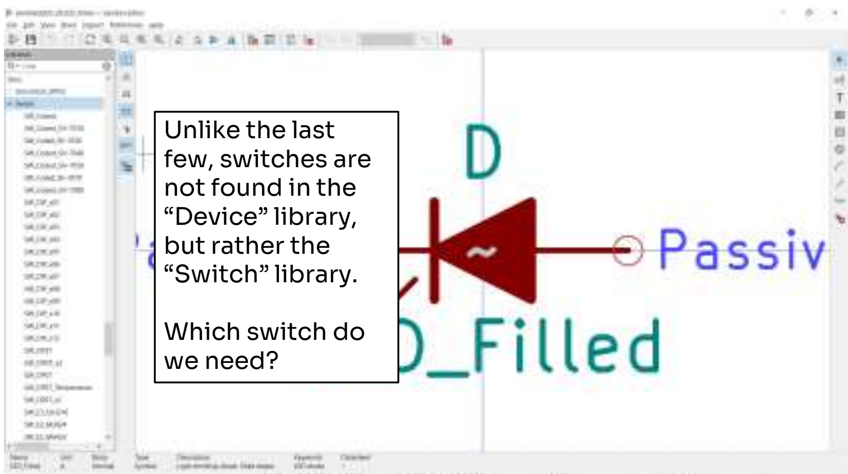
Symbol Library



Great. You should now have six (6) of the eight (8) symbols we'll need for this circuit.

The 7th will be the switch.

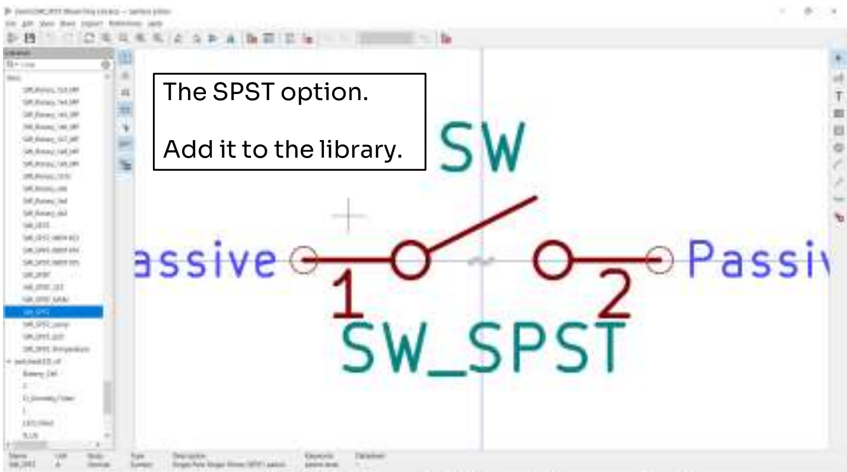
Symbol Library



Unlike the last few, switches are not found in the “Device” library, but rather the “Switch” library.

Which switch do we need?

Symbol Library



The SPST option.

Add it to the library.

If you don't remember the terminology, SPST stands for single-pole, single-throw, meaning it controls one circuit, or one pathway, with one output.

Symbol Library – Adding the IC

- The last component to add is the IC
- The IC is slightly different than the other components we have put down, for two reasons:
  1. The symbol is “unique”, or at least, a symbol with the same pinout would be difficult to locate in the built-in libraries
  2. The footprint may or may not be in the libraries (although it is a standard package, so it likely is… but we’ll pretend.)

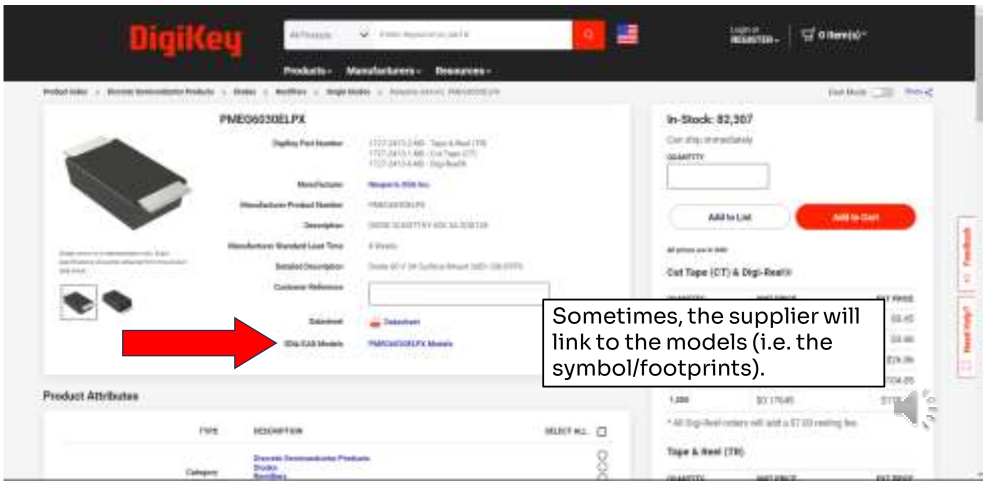
[With an empty slide] If you’ve already generated this symbol during the design videos, congrats! Skip to basically the end of this video, where I’ll show you how to manage which libraries are available to the project. Sorry I can’t be more specific.

For the rest of you…* [and continue with slide]

Symbol Library – Adding the IC

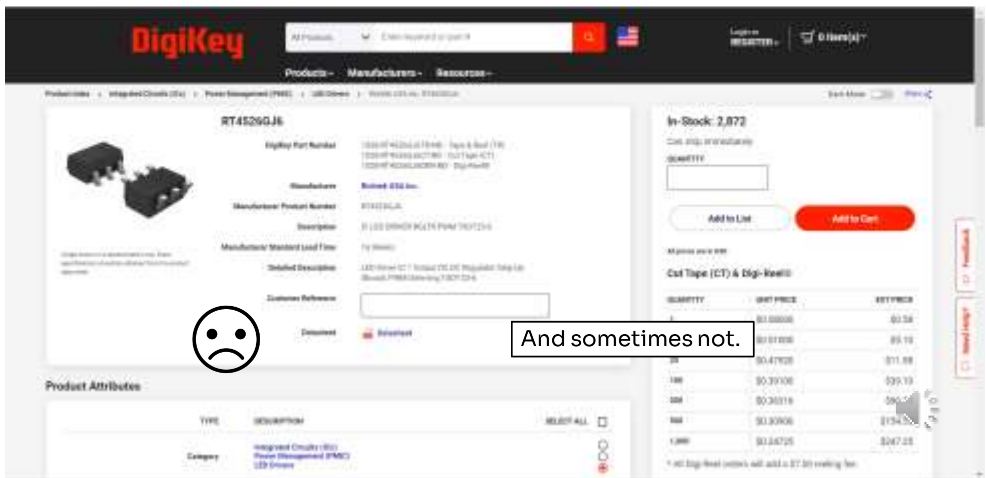
- The first thing to check is whether it's in one of the built-in libraries.
- We can use the filter to search for the RT4526.
- Nope.
  - This is pretty normal for most ICs and non-standard parts because, as I mentioned before, they're pseudo-unique.
- The next option is to see if someone else has already done the work of generating a symbol and footprint

Locating a model for the IC



Sometimes, the suppliers will link to models.

Locating a model for the IC



Sometimes not.

Locating a model for the IC

- There are plenty of places online who can generate (or may already have) these files.
- The two I use are UltraLibrarian and ~~SnapEDA~~ SnapMagic.
- UL does not require an account to download models, but does require one to request new models.
- SnapMagic ~~SnapEDA~~ requires an account to download or request, but you can also generate symbols in-browser.
- Both accounts are free to open.

If the supplier doesn’t link, we can go look for them manually.

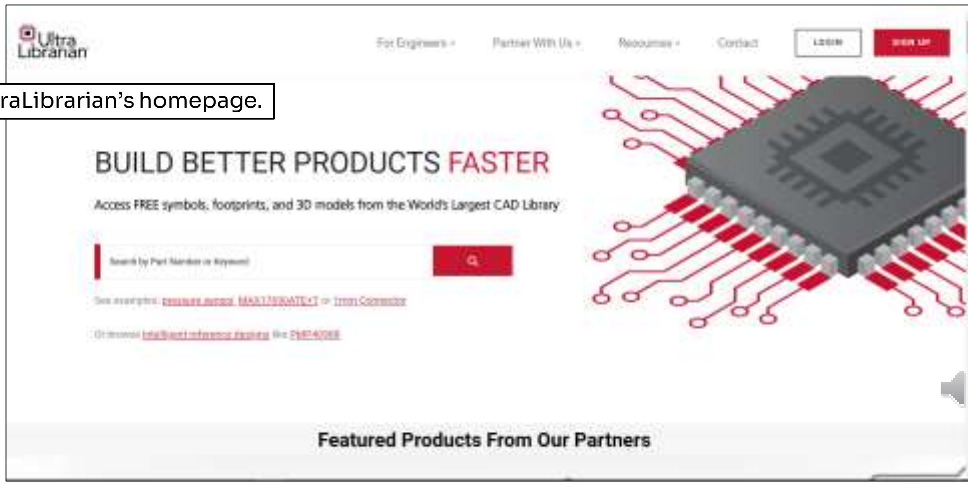
*The two places I have had good results with are Ultra Librarian and SnapMagic, which used to be called SnapEDA.

*Both require free accounts to request new models, but Ultra Librarian allows you to download pre-made ones without one.

*SnapMagic has a few additional tools that are available for registered accounts as well, like in-browser symbol and footprint generation.

**

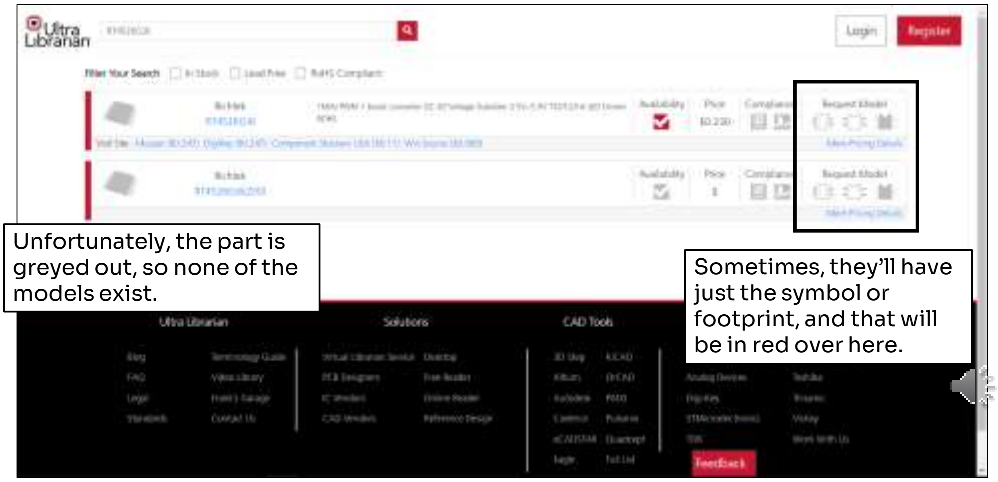
Locating a model for the IC



UltraLibrarian's homepage.

We'll start arbitrarily with Ultra Librarian. Search for the symbol you'd like a model for.

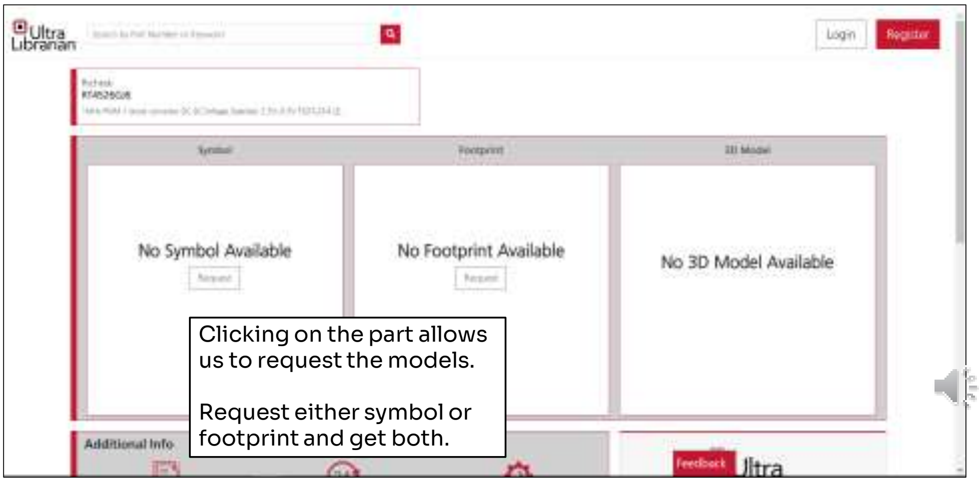
Locating a model for the IC



Unfortunately, the part is grayed, so no models exist.

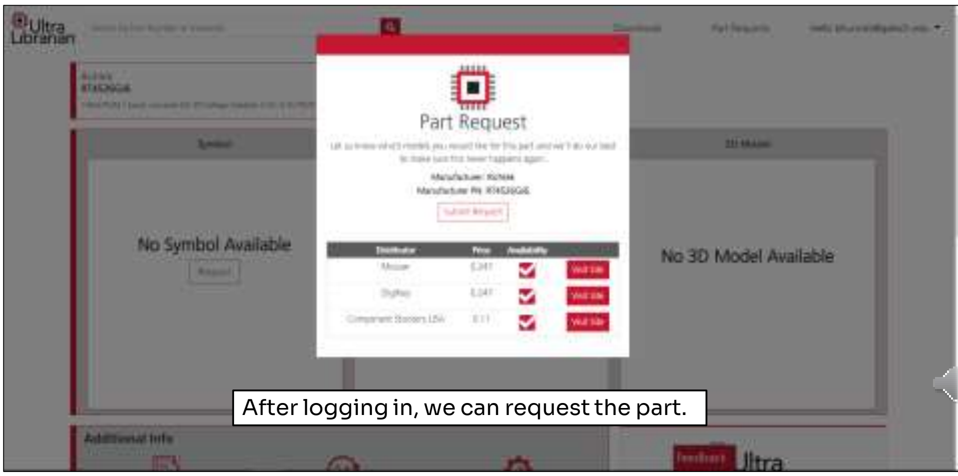
*Sometimes, the icons on the right will be filled in, indicating that they have a footprint or a symbol, but not today.

Locating a model for the IC



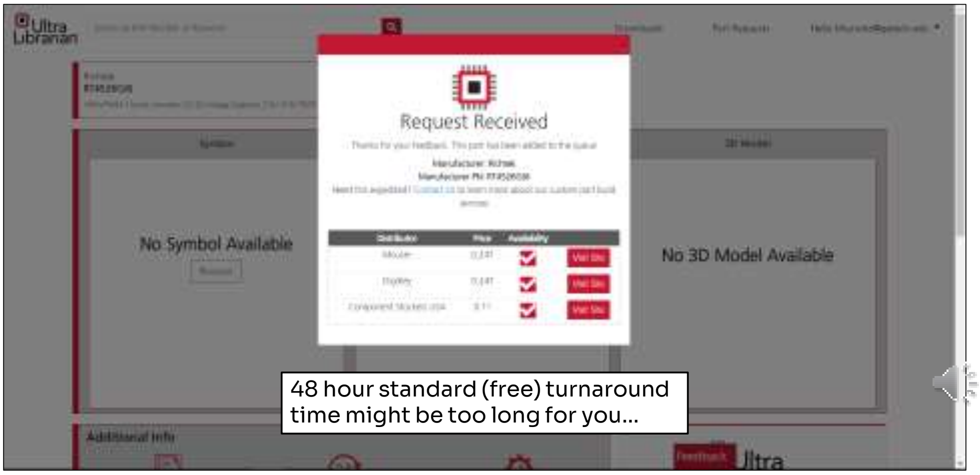
Clicking the part gets us to the part page, where we can ask them to create a model for us. Request either to get both.

Locating a model for the IC

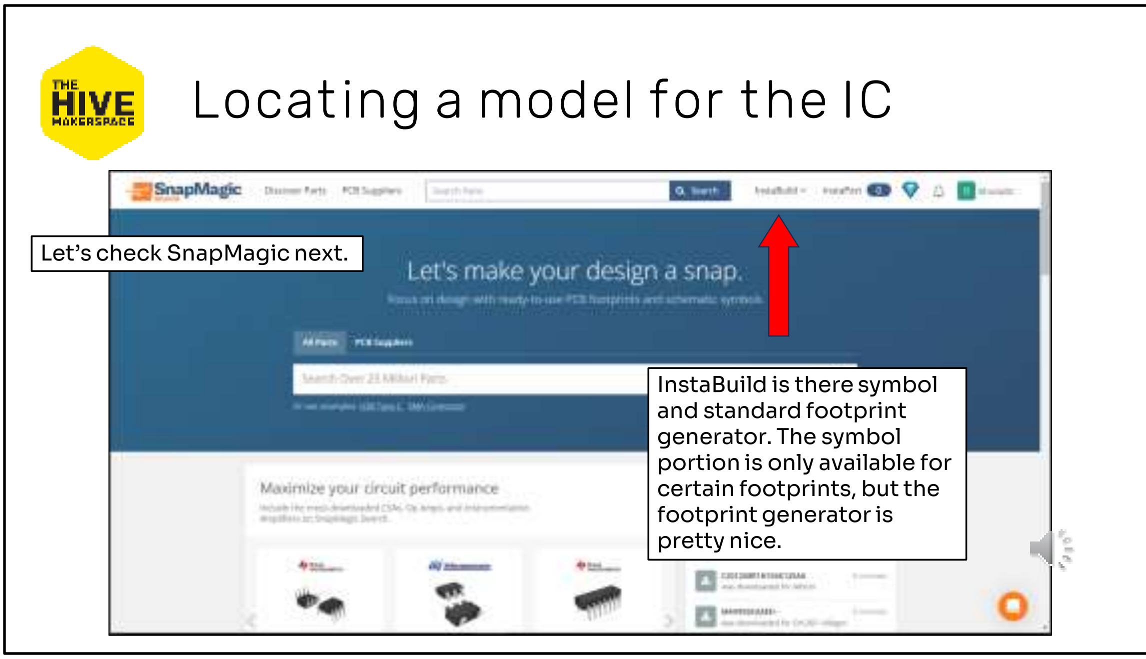


Looks like this

Locating a model for the IC

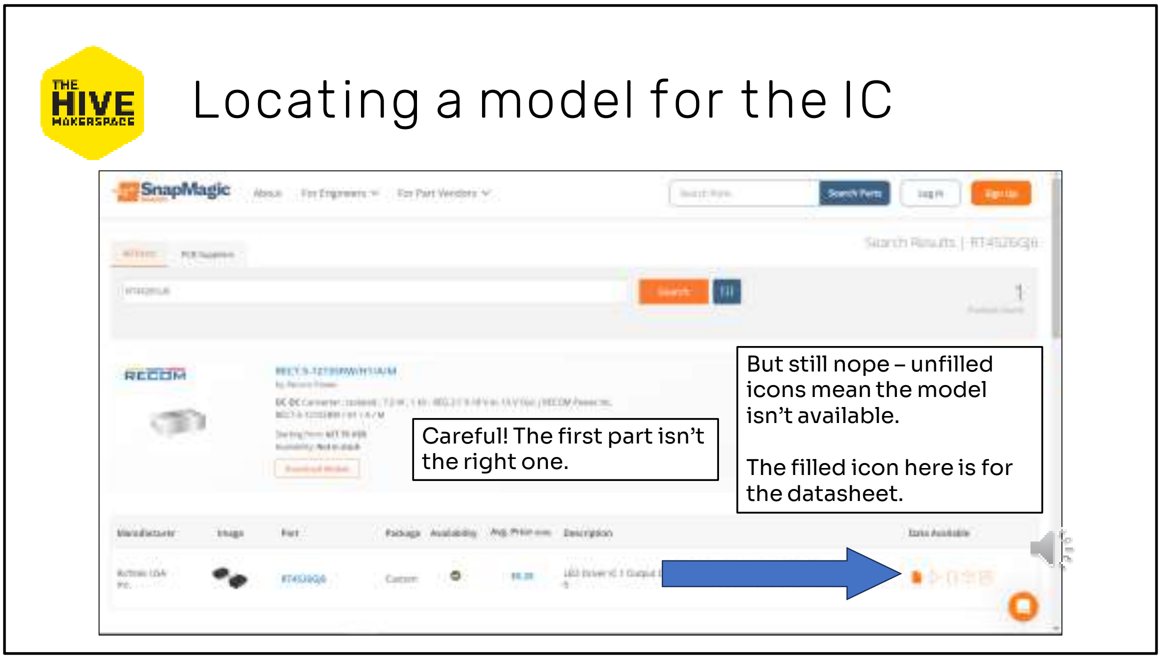


Standard turnaround guarantee is 48 hours, which may or may not be too long for you.



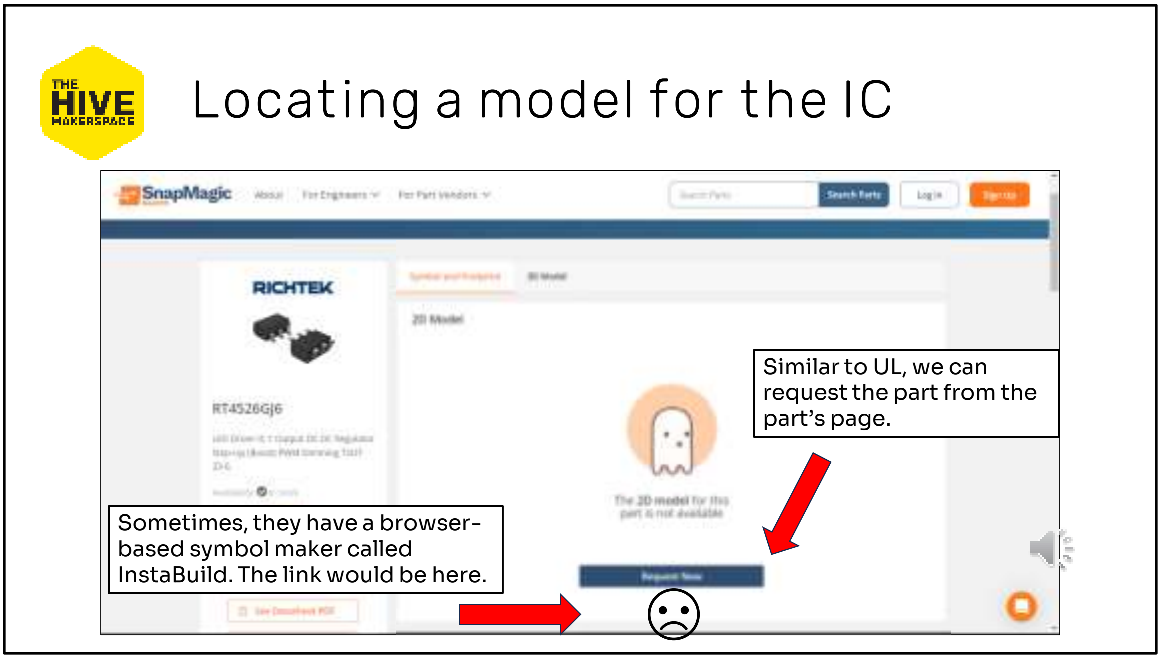
Let's check our other source, SnapMagic.

*SnapMagic's in-browser model generator is linked on their homepage, but while most standard IC footprints can be generated, the symbols are limited to select footprints only.



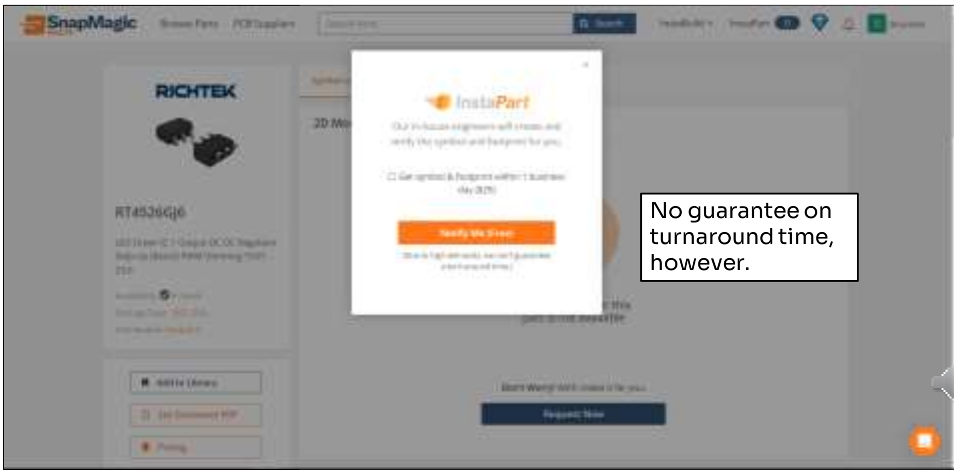
Searching for our part brings us to this page, *but be careful! Sometimes they recommend a part at the top that isn't right.

*Still, no models available – we can see this with the empty icons on the right.



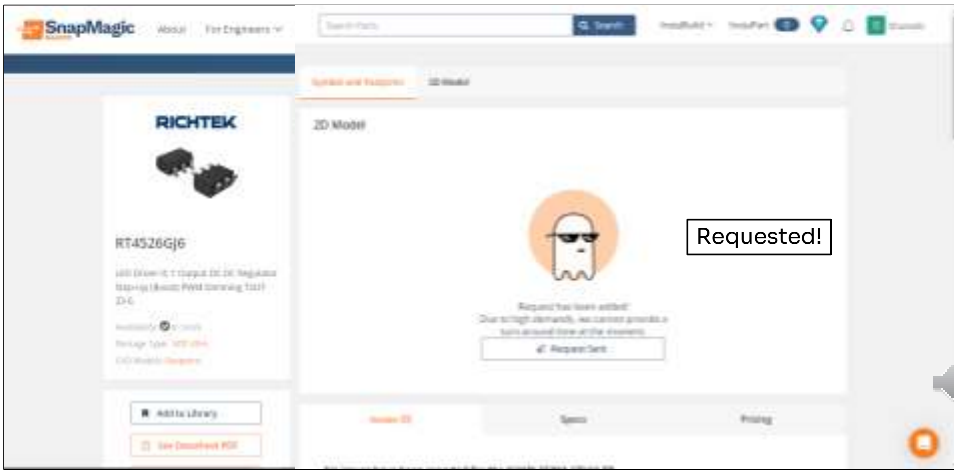
We can request the models from the parts page. *There would also be a link here if the in-brower symbol generator was available for this part, but it's not. Sad.

Locating a model for the IC



Unfortunately, SnapMagic doesn't have a guarantee on turnaround time.

Locating a model for the IC



But you get a cool ghost when you request the part.

~~Locating~~ Creating a model for the IC

- Struck out thrice. What to do?
- Can keep searching (someone *must* have made this before, right?), but returns may be diminishing.
- Because KiCAD stores symbols and footprints in separate libraries, we'll just go ahead and create our own symbol directly in KiCAD.
  - No need to create our own footprint (yet)

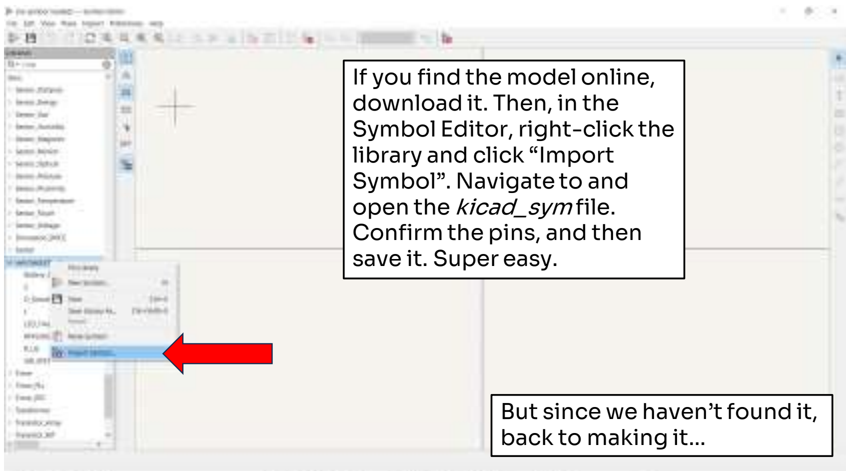
So three strikes. Are we out?

*We can keep searching randomly on the internet, but the returns will likely be diminishing, and the quality might be degraded.

*Instead, we can just go ahead and make the symbol in KiCAD directly.

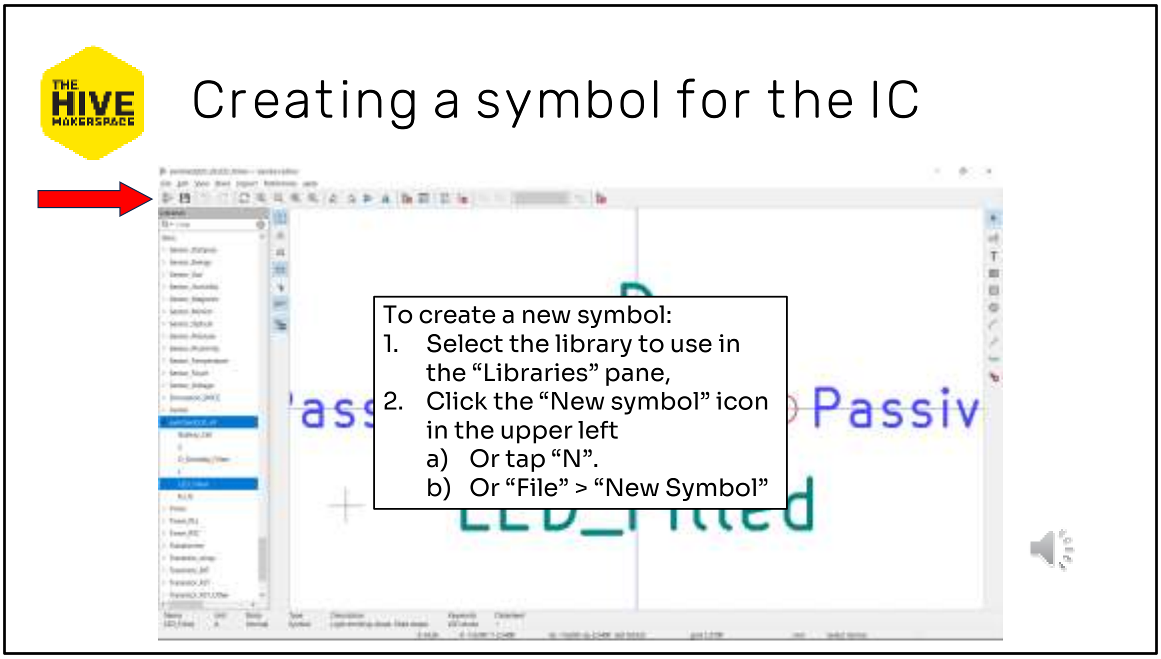
Symbols are relatively easy to make, and are less prone to errors than a custom footprint, though they can be tedious with many pins, so it's totally doable to make them without waiting for Ultra Librarian or Snap Magic to be done.

Aside: Importing a model



If you find the model online, download it. Then, in the Symbol Editor, right-click the library and click “Import Symbol”. Navigate to and open the *kicad_sym* file. Confirm the pins, and then save it. Super easy.

But since we haven’t found it, back to making it…



Next, we can create a new symbol just hitting the “N” key, the icon in the upper-left, or under the “File” menu.

Creating a symbol for the IC

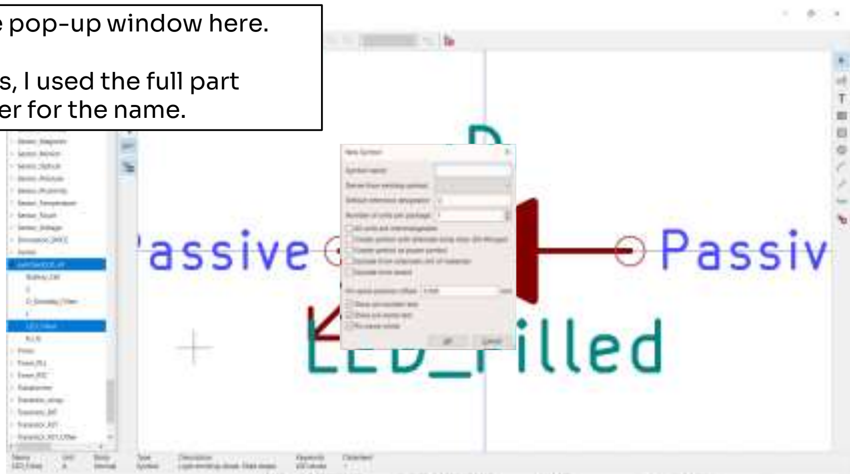
Fill in the pop-up window here.

Let’s fill in the window here.

Creating a symbol for the IC

Fill in the pop-up window here.

- For ICs, I used the full part number for the name.

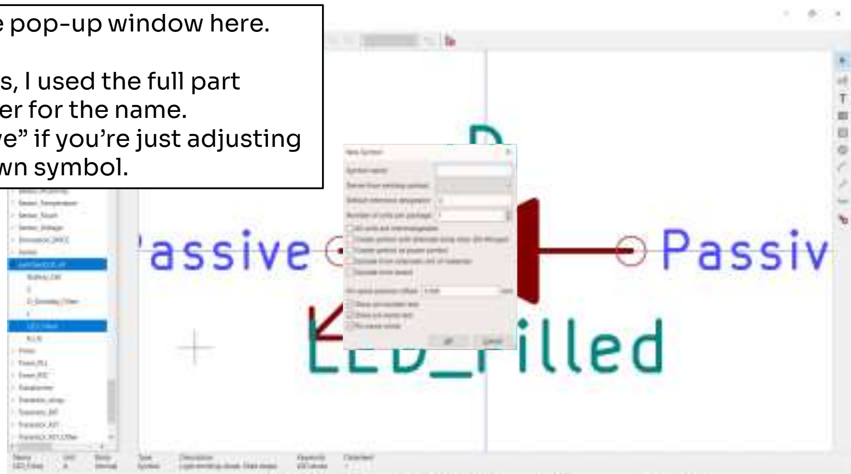


Generally, good policy is to name the symbol with the part number, which in this case would be RT4526. You can leave off the GJ6 in this case because, if you read the datasheet, those are the only options for the final three digits.

Creating a symbol for the IC

Fill in the pop-up window here.

- For ICs, I used the full part number for the name.
- “Derive” if you’re just adjusting a known symbol.

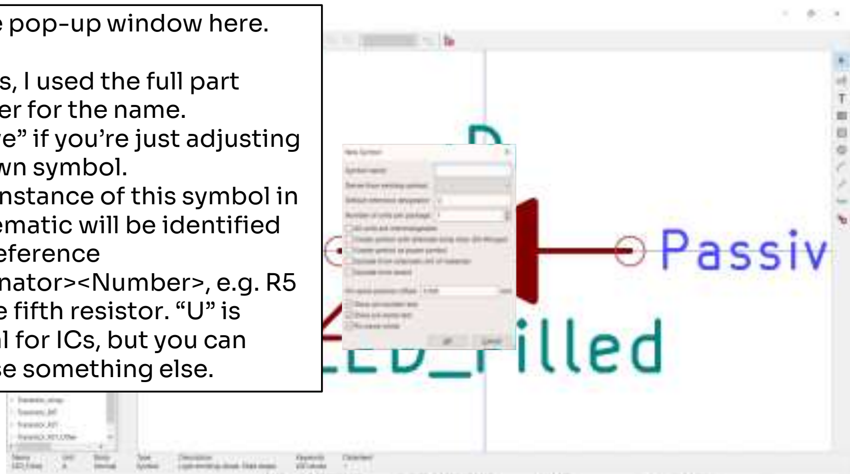


“Derive” refers to if you’re adjusting a known symbol, like creating a polarized capacitor by copying the basic capacitor.

Creating a symbol for the IC

Fill in the pop-up window here.

- For ICs, I used the full part number for the name.
- “Derive” if you’re just adjusting a known symbol.
- Each instance of this symbol in a schematic will be identified by <Reference Designator><Number>, e.g. R5 for the fifth resistor. “U” is typical for ICs, but you can choose something else.



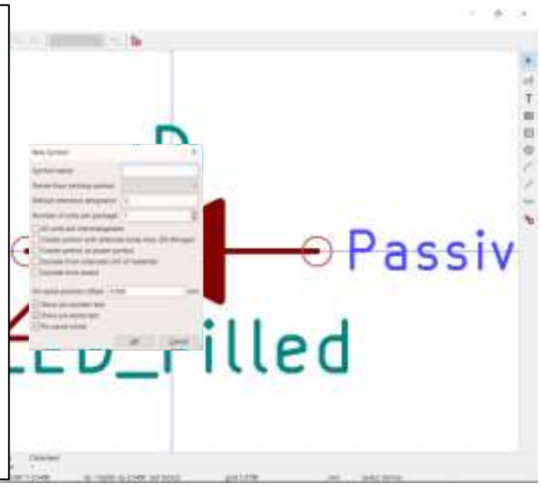
The reference designator is the letter that KiCAD uses to identify all instances of this symbol. Each instance will also get a number, so a resistor might be designated R5 for the fifth resistor of the schematic. “U” is very common for ICs, though you could choose something else if you’d like. It’s fine for multiple symbols to have the same designator character, like U or R; it just means they’re of the same type, so to speak.

Creating a symbol for the IC

Fill in the pop-up window here.

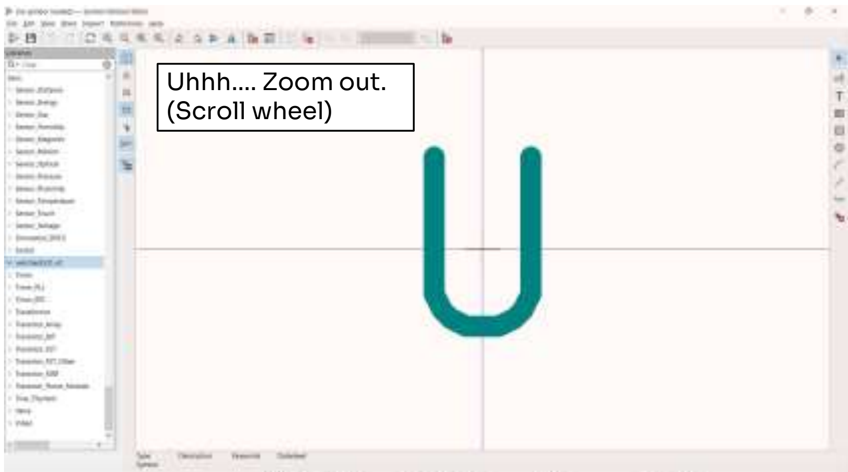
- For ICs, I used the full part number for the name.
- “Derive” if you’re just adjusting a known symbol.
- Each instance of this symbol in a schematic will be identified by <Reference Designator><Number>, e.g. R5 for the fifth resistor. “U” is typical for ICs, but you can choose something else.

The rest is not relevant now. Click OK to continue.



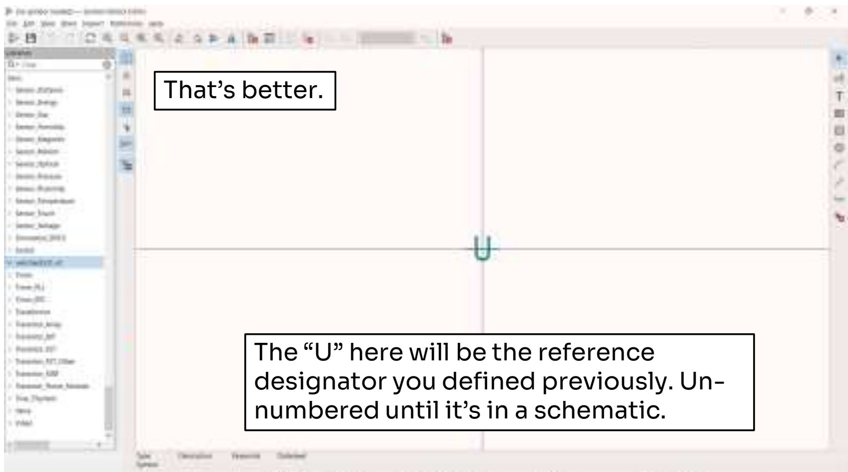
The rest doesn’t matter to us, so click “OK”.

Creating a symbol for the IC



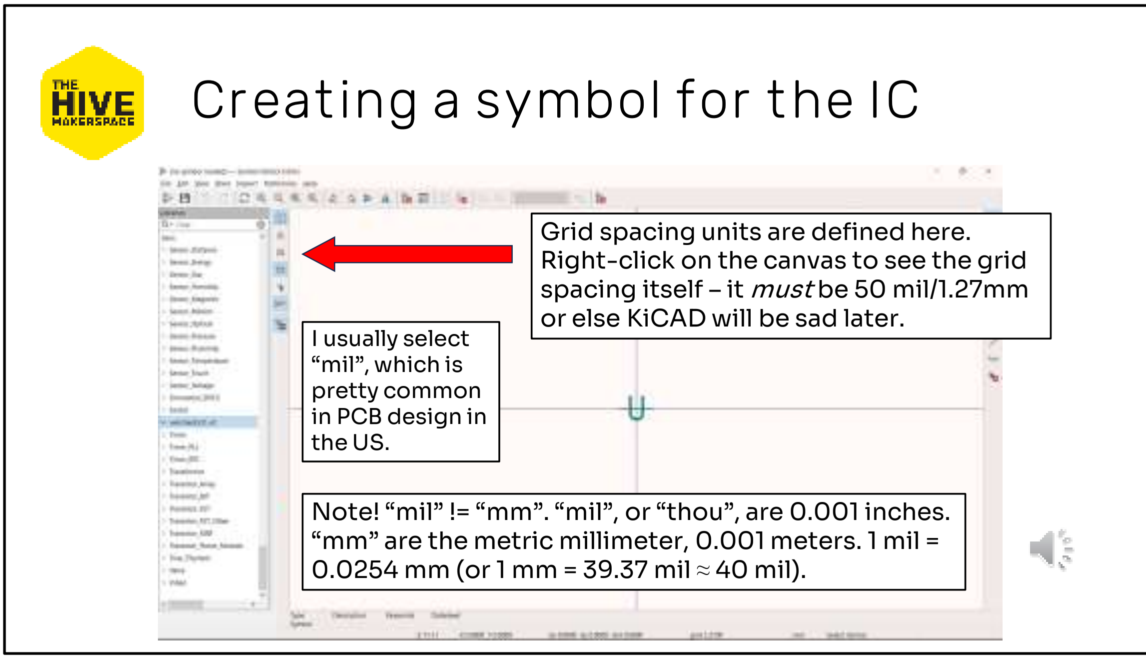
Since there’s nothing on the editor now, it will zoom automatically in to the reference designator.

Creating a symbol for the IC



(Apologies for the slight discontinuity here – I made this bit of slides prior to copying in the global symbols, so the library in these slides is empty. Oops!)

Something like this is better. The "U" is currently placed at the anchor point, which is the point at which the symbol will be attached to the mouse. We'll move those both later.

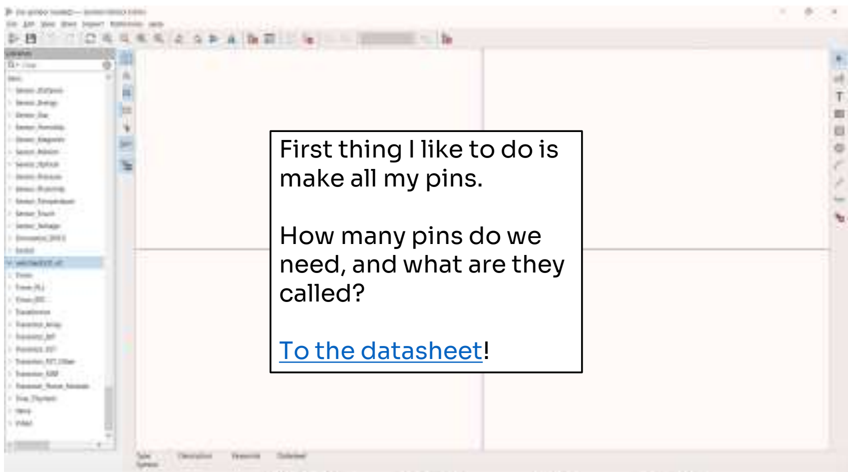


As usual, grid spacing and units are defined on the left. Check the grid by right-clicking. It should read 50 mil or 1.27 mm, or else KiCAD will not be able to attach wires to it and you’ll be sad later.

I usually select mils for my units because I’m used to thinking of hole sizes and trace widths in mils, and because I’m used to imperial units. But you can readily use metric units as well. KiCAD is based on metric values, after all.

Note that mils and millimeters are not the same. Mils, which are also known as thous, are a thousandth of an inch, or 0.001 inches. Millimeters are, of course, a thousandth of a meter. One millimeter is about forty mils.

Creating a symbol for the IC



First thing I like to do is make all my pins.

How many pins do we need, and what are they called?

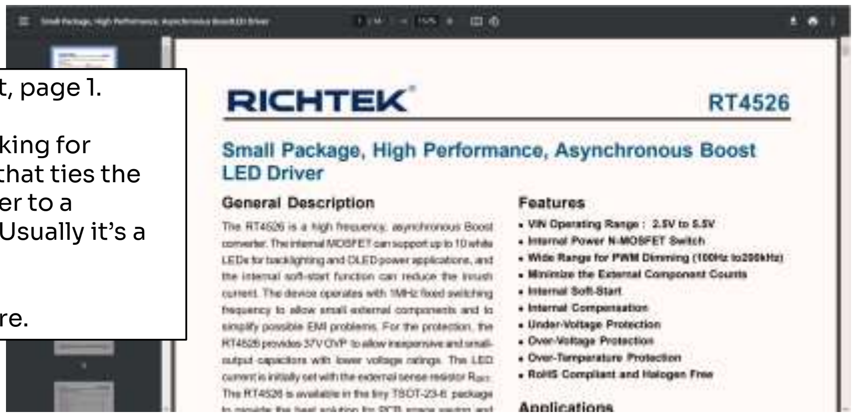
[To the datasheet]!

Creating a symbol for the IC

Datasheet, page 1.

We're looking for anything that ties the pin number to a function. Usually it's a table.

So not here.



RICHTEK

RT4526

Small Package, High Performance, Asynchronous Boost LED Driver

General Description

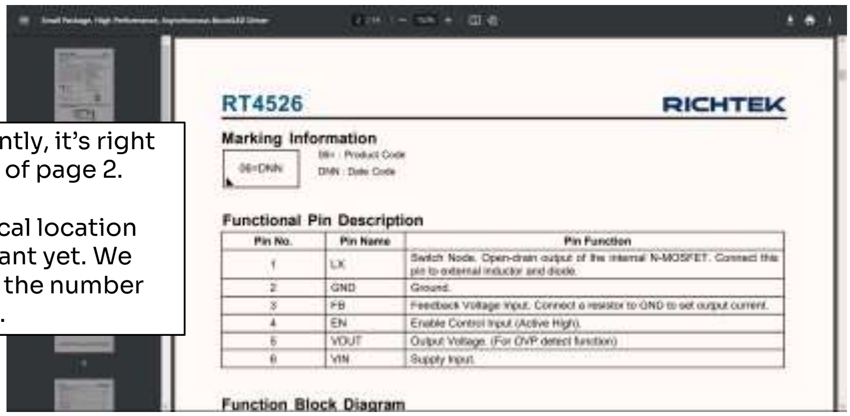
The RT4526 is a high frequency, asynchronous Boost converter. The internal MOSFET can support up to 10 white LEDs for backlighting and OLED power applications, and the internal soft-start function can reduce the inrush current. The device operates with 1MHz fixed switching frequency to allow small external components and to simplify possible EMI problems. For the protection, the RT4526 provides 37V OVP to allow inexpensive and small-output capacitors with lower voltage ratings. The LED current is initially set with the external sense resistor R_{SET}. The RT4526 is available in the tiny TSOT-23-6 package

Features

- VIN Operating Range : 2.5V to 5.5V
- Internal Power N-MOSFET Switch
- Wide Range for PWM Dimming (100Hz to200kHz)
- Minimize the External Component Counts
- Internal Soft-Start
- Internal Compensation
- Under-Voltage Protection
- Over-Voltage Protection
- Over-Temperature Protection
- RoHS Compliant and Halogen Free

Applications

Creating a symbol for the IC



RT4526 RICHTEK

Marking Information

0E=DNN

0E= : Product Code

DNN : Date Code

Functional Pin Description

Pin No.	Pin Name	Pin Function
1	LX	Switch Node. Open-drain output of the internal N-MOSFET. Connect this pin to external inductor and diode.
2	GND	Ground.
3	FB	Feedback Voltage Input. Connect a resistor to GND to set output current.
4	EN	Enable Control Input (Active High).
5	VOUT	Output Voltage. (For OVP detect function)
6	VIN	Supply Input.

Function Block Diagram

Conveniently, it's right at the top of page 2.

The physical location isn't relevant yet. We only want the number and name.

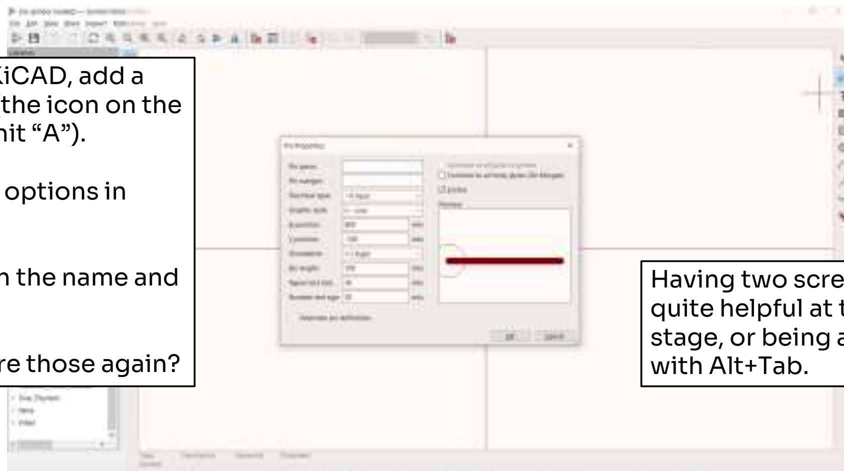
Creating a symbol for the IC

Back to KiCAD, add a new pin (the icon on the right, or hit "A").

Bunch of options in here.

Start with the name and number.

What were those again?



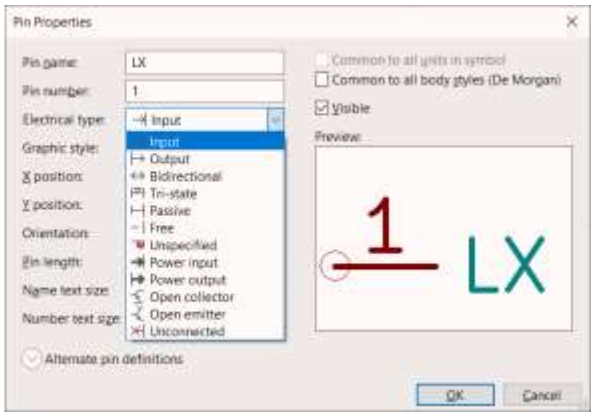
Having two screens is quite helpful at this stage, or being a wizard with Alt+Tab.

Creating a symbol for the IC

Start with pin 1.

By default, it's set as an "input". But is it?

Back to the pinout table!



Functional Pin Descripti

Pin No.	Pin Name
1	LX
2	GND
3	FB
4	EN
5	VOUT
6	VIN

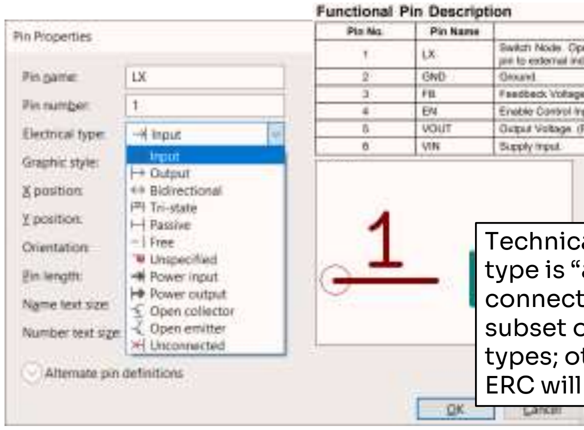
Creating a symbol for the IC

If you're not sure what it is, you can either set it to "Bidirectional" or "Unspecified" or "Passive".

The type won't cause the design to fail, but it might cause you headaches later with the ERC.

Functional Pin Description

Pin No.	Pin Name	Pin Function
1	LX	Switch Node. Open-drain output of the internal N-MOSFET. Connect this pin to external inductor and diode.
2	GND	Ground.
3	FB	Feedback Voltage Input. Connect a resistor to GND to set output current.
4	EN	Enable Control Input (Active High).
5	VOUT	Output Voltage. (For OVP detect function)
6	VIN	Supply Input.

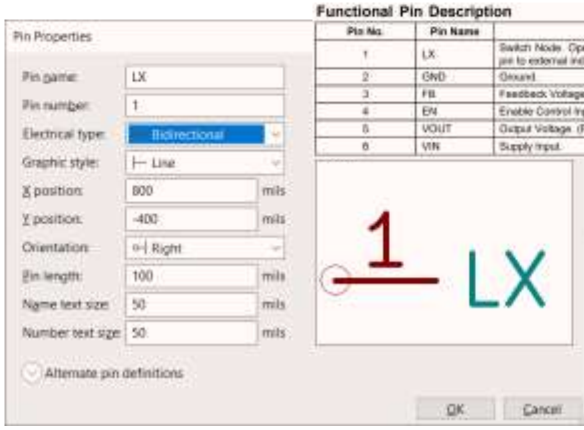


Technically, each pin type is "allowed" connected to only a subset of other pin-types; otherwise, the ERC will throw an error.

Creating a symbol for the IC

The rest of these are graphical choices, so we'll adjust them as needed later.

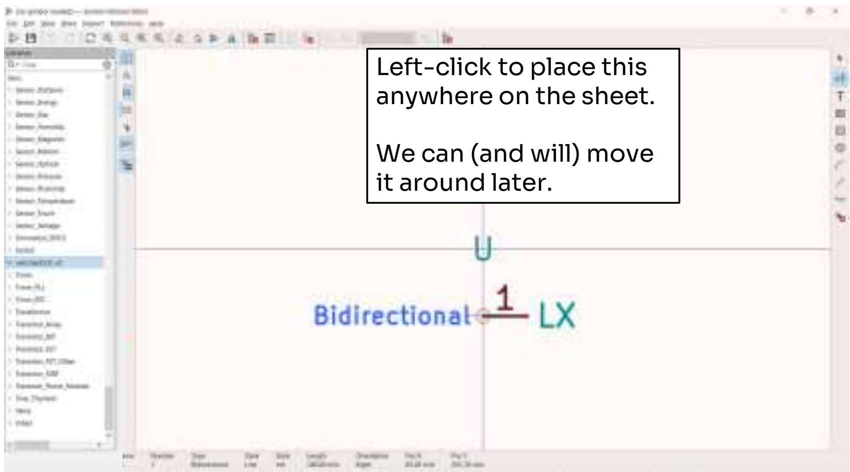
Click OK.



Functional Pin Description

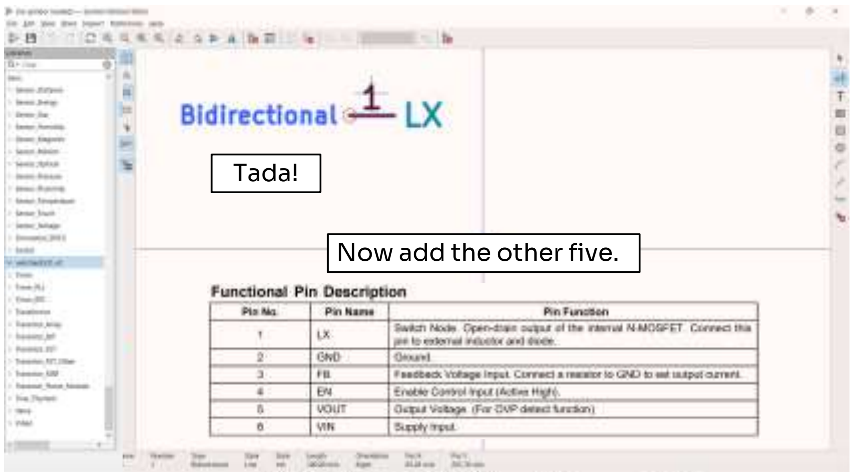
Pin No.	Pin Name	Pin Function
1	LX	Switch Node. Open-drain output of the internal N-MOSFET. Connect this pin to external inductor and diode.
2	GND	Ground.
3	FB	Feedback Voltage Input. Connect a resistor to GND to set output current.
4	EN	Enable Control Input (Active High).
5	VOUT	Output Voltage. (For OVP detect function)
6	VIN	Supply Input.

Creating a symbol for the IC



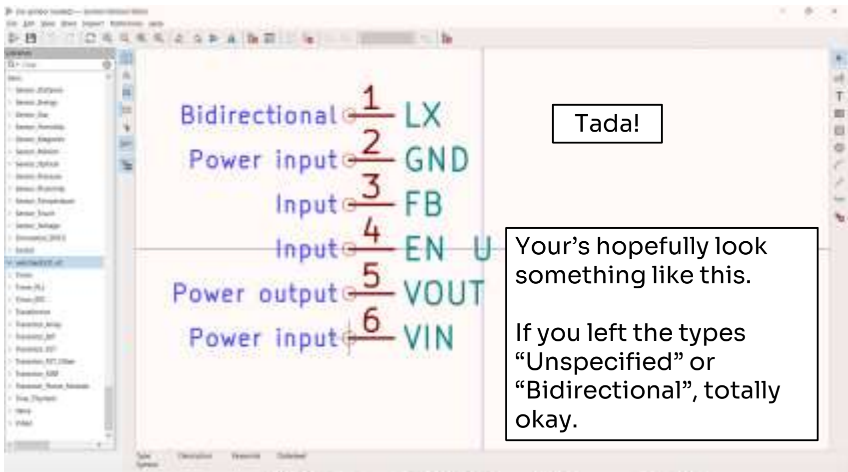
The pin type is in blue there.

Creating a symbol for the IC



Pin No.	Pin Name	Pin Function
1	LX	Switch Node. Open-drain output of the internal N-MOSFET. Connect this pin to external inductor and diode.
2	GND	Ground.
3	FB	Feedback Voltage Input. Connect a resistor to GND to set output current.
4	EN	Enable Control Input (Active High).
5	VOUT	Output Voltage. (For OVP detect function)
6	VIN	Supply Input.

Creating a symbol for the IC

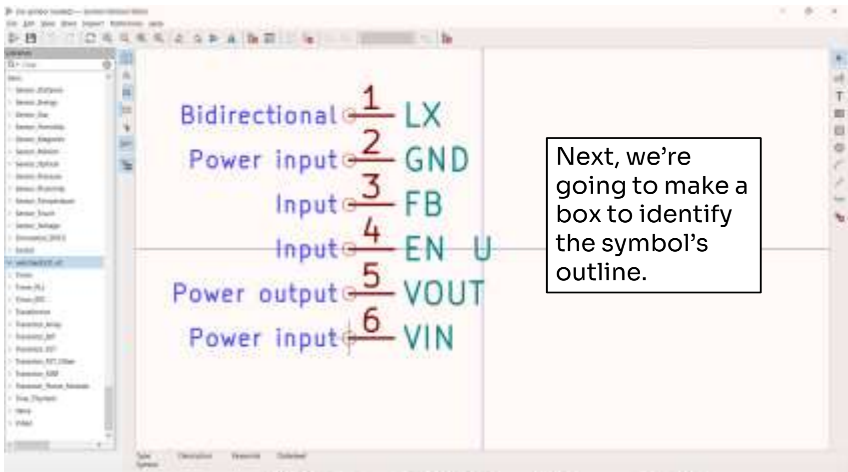


Tada!

Your's hopefully look something like this.

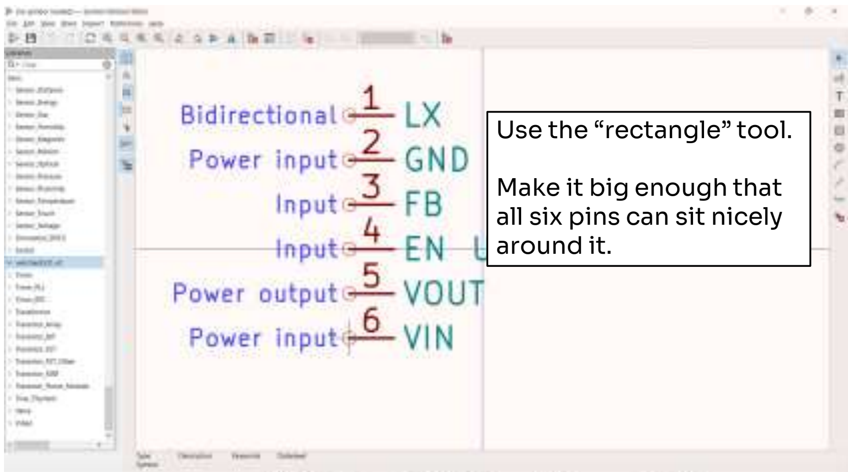
If you left the types "Unspecified" or "Bidirectional", totally okay.

Creating a symbol for the IC



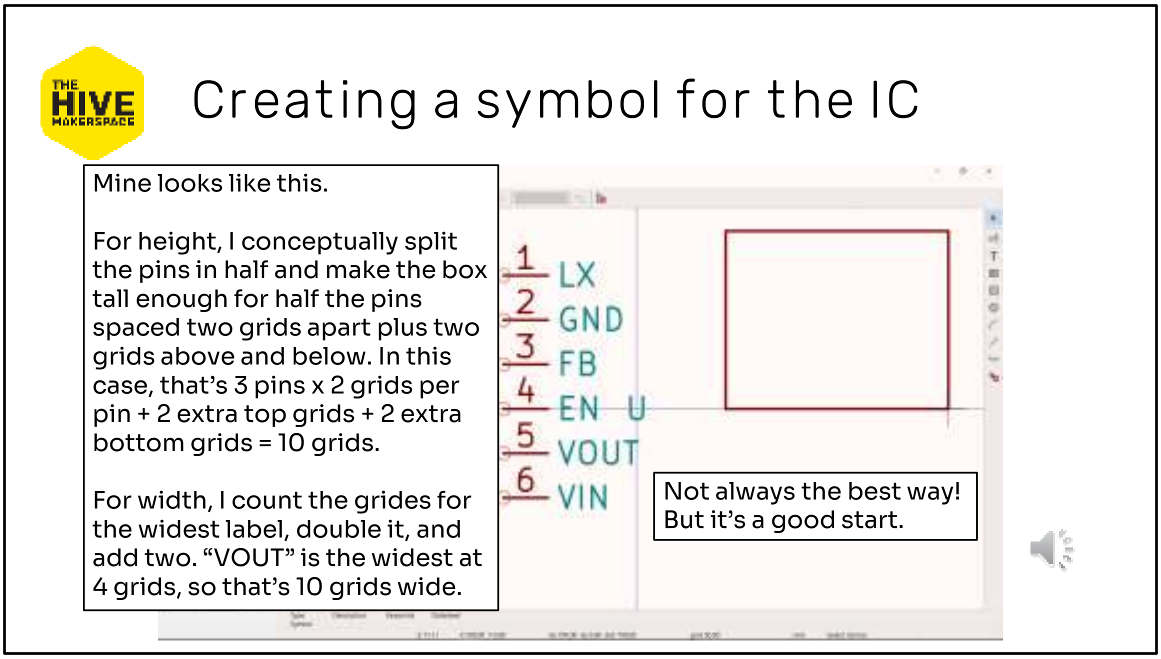
Next, we're going to make a box to identify the symbol's outline.

Creating a symbol for the IC



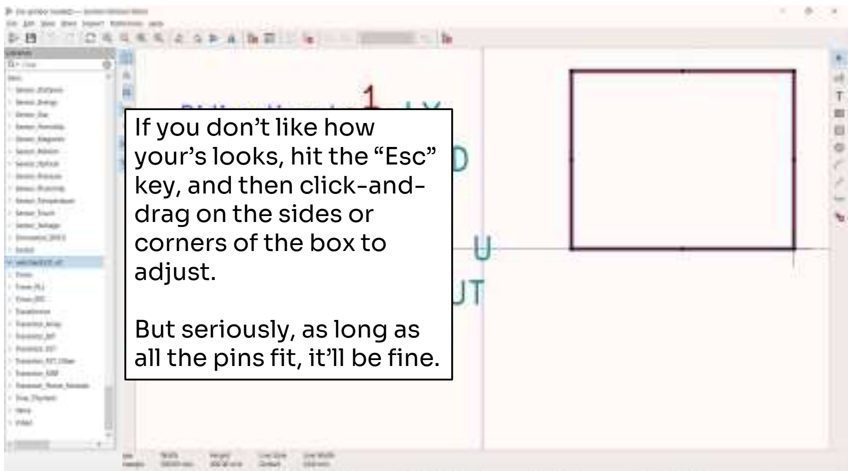
Use the “rectangle” tool.

Make it big enough that all six pins can sit nicely around it.



This is the algorithm by which I spec'd my box's size, but I abandoned that size basically in the next few slides, so you're safe to ignore that.

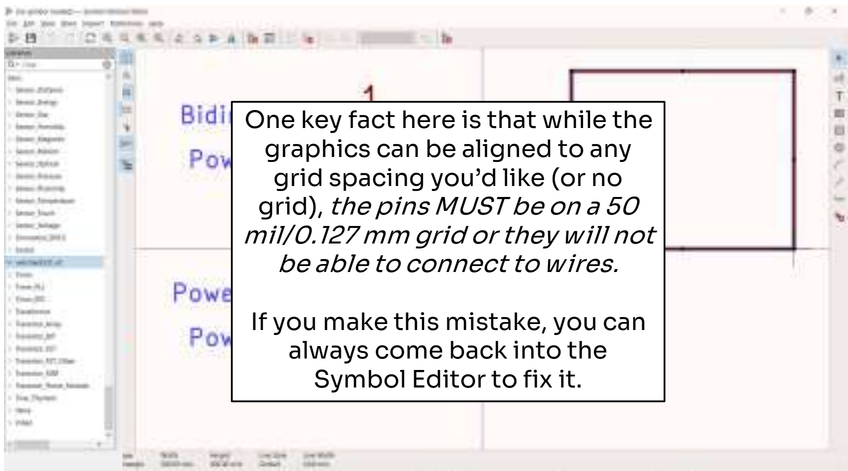
Creating a symbol for the IC



If you don't like how your's looks, hit the "Esc" key, and then click-and-drag on the sides or corners of the box to adjust.

But seriously, as long as all the pins fit, it'll be fine.

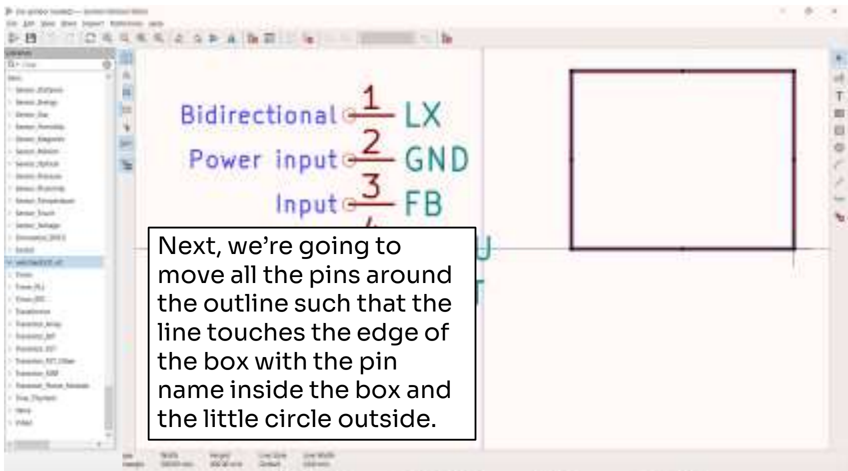
Creating a symbol for the IC



One key fact here is that while the graphics can be aligned to any grid spacing you'd like (or no grid), *the pins MUST be on a 50 mil/0.127 mm grid or they will not be able to connect to wires.*

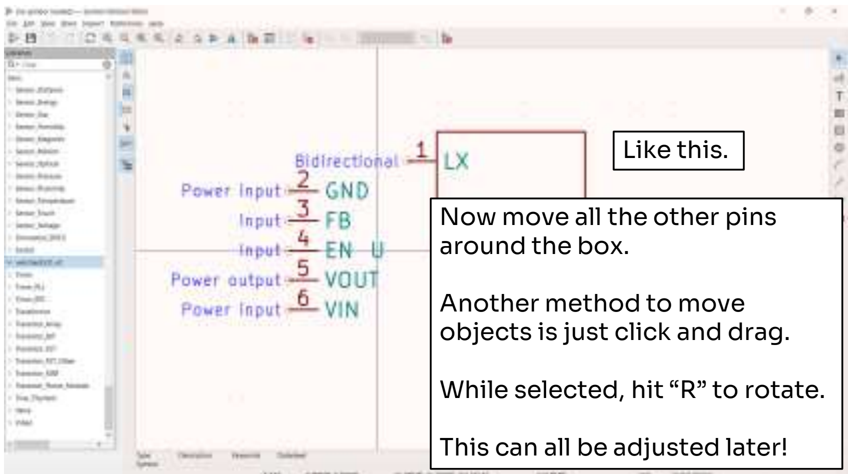
If you make this mistake, you can always come back into the Symbol Editor to fix it.

Creating a symbol for the IC



Next, we're going to move all the pins around the outline such that the line touches the edge of the box with the pin name inside the box and the little circle outside.

Creating a symbol for the IC



Like this.

Now move all the other pins around the box.

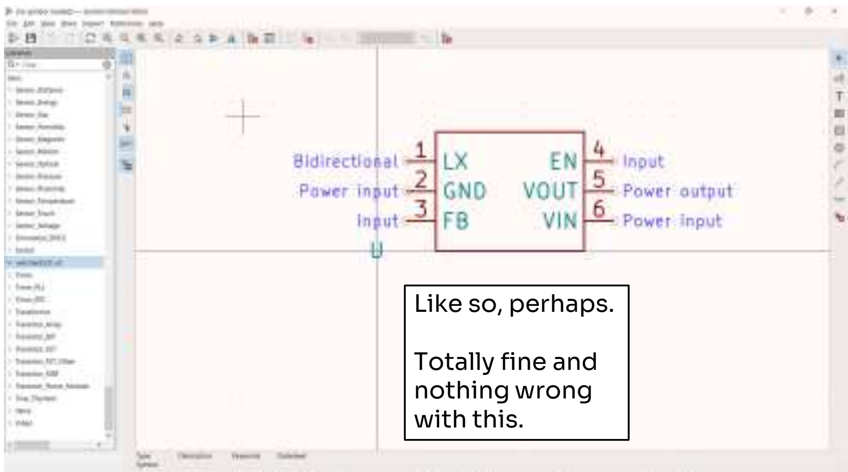
Another method to move objects is just click and drag.

While selected, hit “R” to rotate.

This can all be adjusted later!

You might consider pausing the video here while you place your pins before seeing what I did.

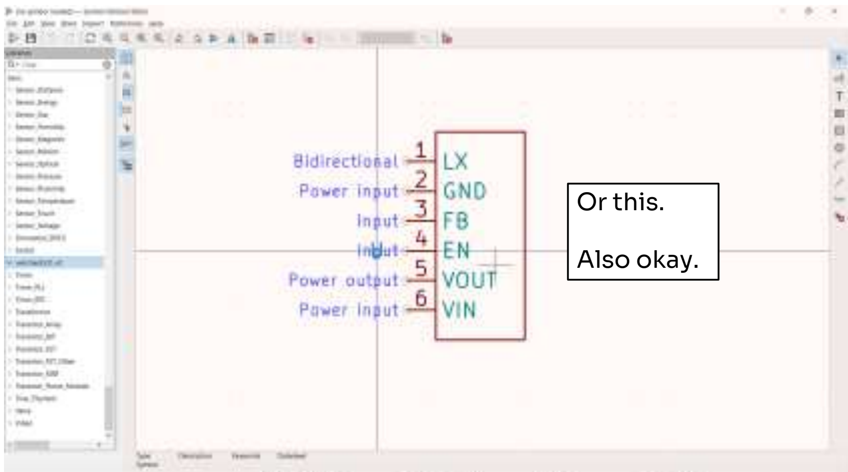
Creating a symbol for the IC



Like so, perhaps.

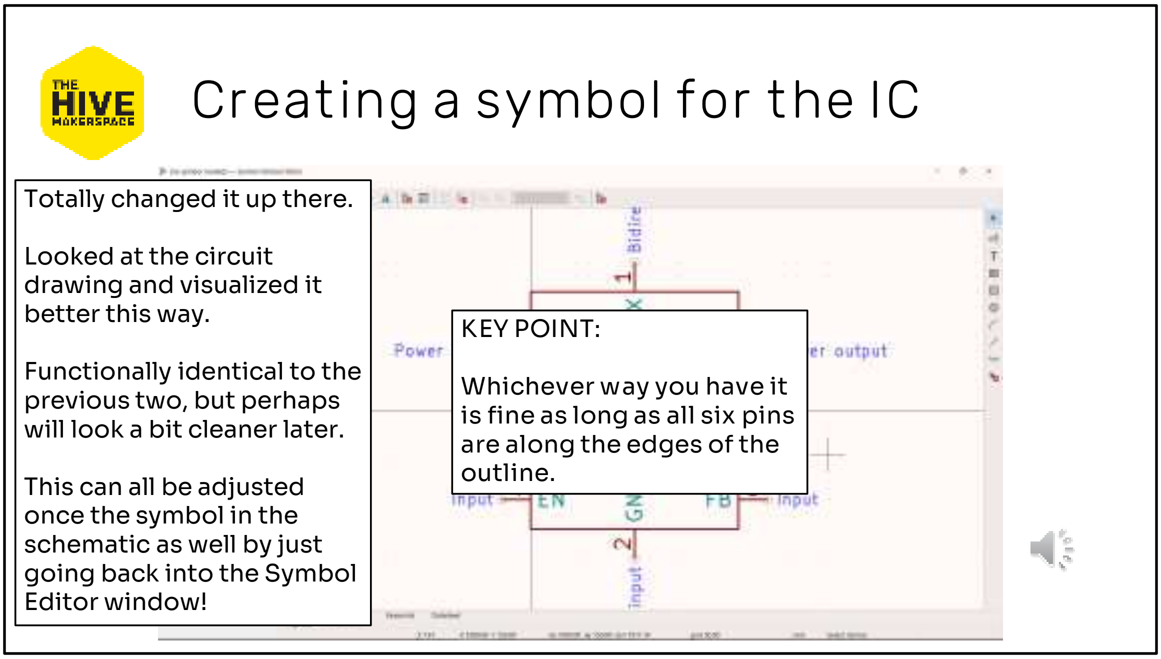
Totally fine and nothing wrong with this.

Creating a symbol for the IC



Or this.

Also okay.



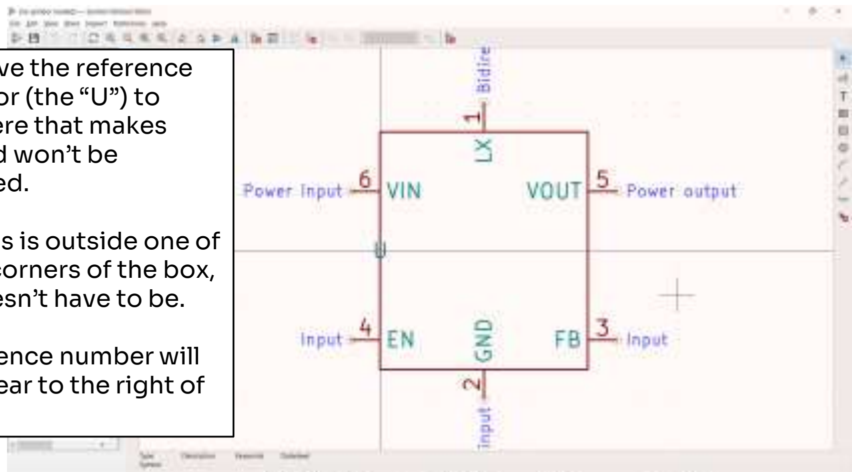
Additionally, when we learn about nets in the next video, you might understand why it doesn't matter how the symbol is laid out so much – pins can connect anywhere without a direct line.

Creating a symbol for the IC

Next, move the reference designator (the “U”) to somewhere that makes sense and won’t be obstructed.

Often, this is outside one of the four corners of the box, but it doesn’t have to be.

The reference number will later appear to the right of the “U”.



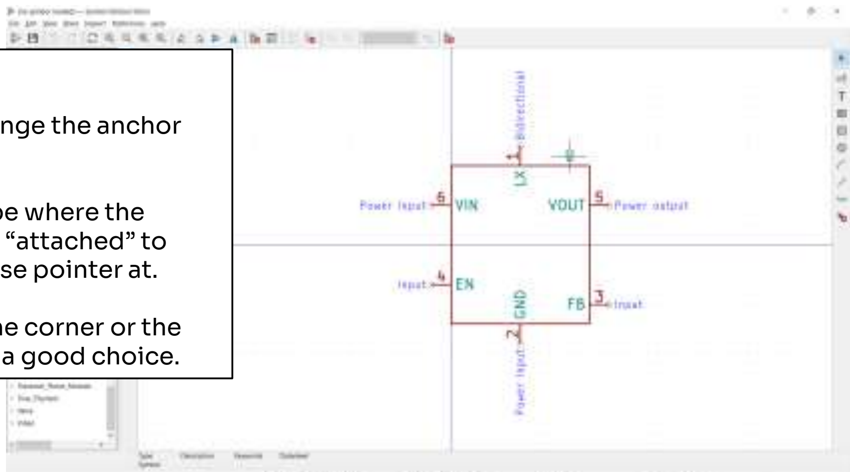
Creating a symbol for the IC

Great.

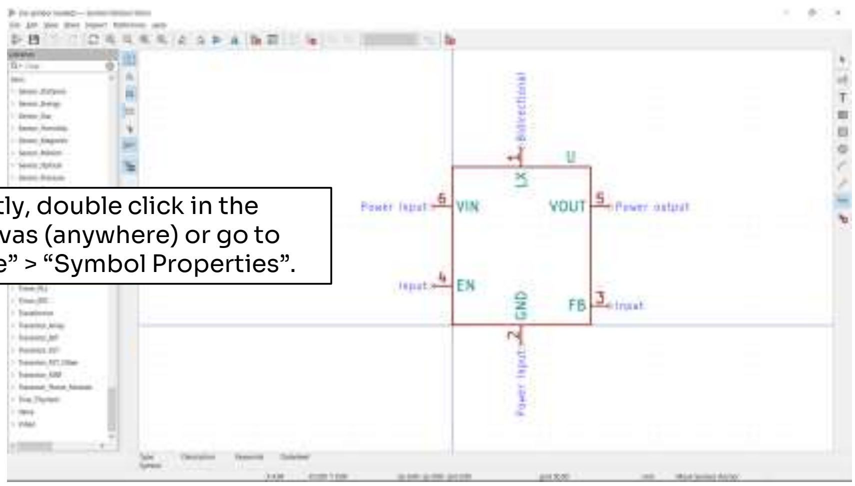
Next, change the anchor point.

This will be where the symbol is "attached" to your mouse pointer at.

Usually the corner or the middle is a good choice.



Creating a symbol for the IC



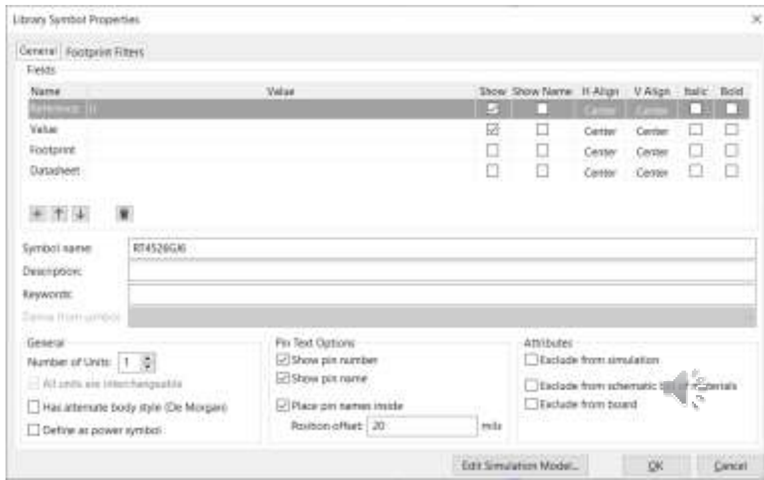
Lastly, double click in the canvas (anywhere) or go to “File” > “Symbol Properties”.

Creating a symbol for the IC

These windows often have a lot more options than we need right now, but they do offer a lot of flexibility.

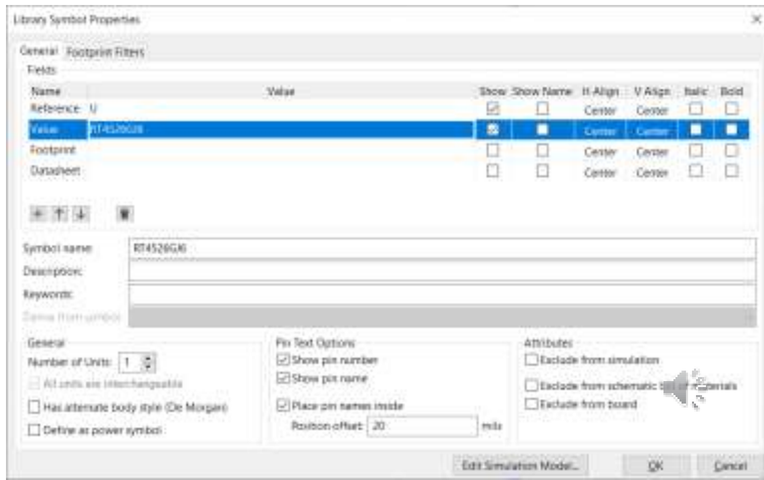
For now, just add the part number (RT4526GJ6) to the “Value” field’s value.

Adding a footprint here can be done if there’s a single footprint for this part. We don’t have a footprint yet, so we won’t add it, but later maybe.

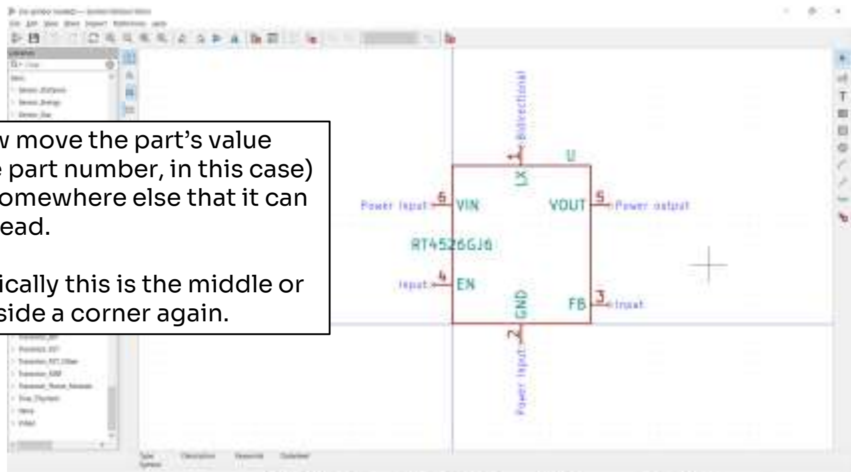


Creating a symbol for the IC

Click OK once your done.



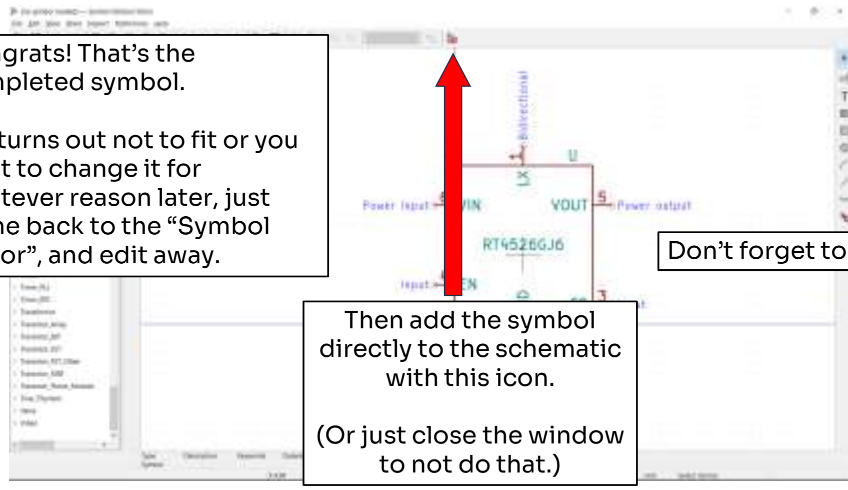
Creating a symbol for the IC



Now move the part's value (the part number, in this case) to somewhere else that it can be read.

Typically this is the middle or outside a corner again.

Creating a symbol for the IC



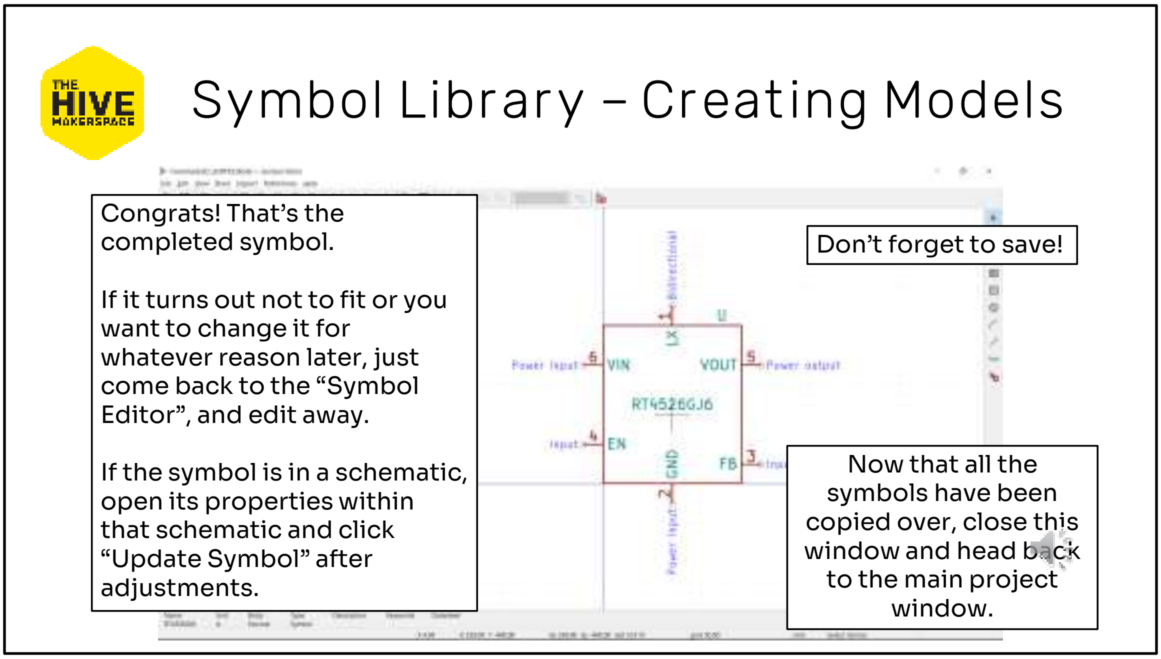
Congrats! That's the completed symbol.

If it turns out not to fit or you want to change it for whatever reason later, just come back to the "Symbol Editor", and edit away.

Then add the symbol directly to the schematic with this icon.

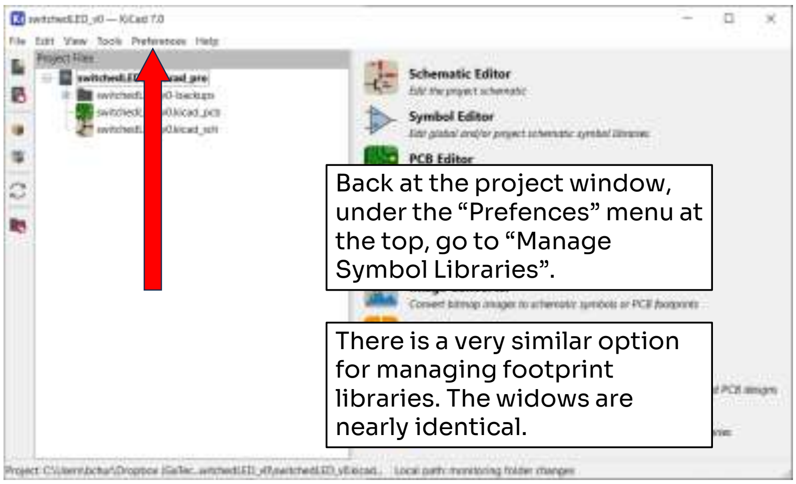
(Or just close the window to not do that.)

Don't forget to save!



Now that all the symbols have been added to the library, we can close this window. When adding symbols to your schematic, now just use the symbols in this library instead of in the built-in libraries.

Symbol Library – Creating Models



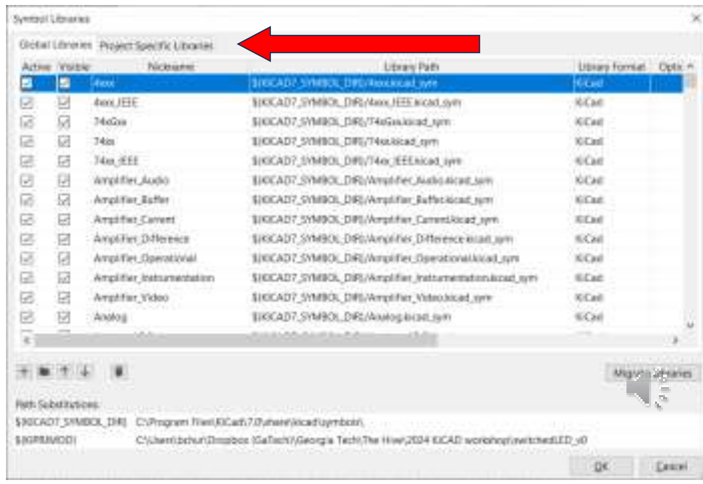
Back at the project window, under the “Prefences” menu at the top, go to “Manage Symbol Libraries”.

There is a very similar option for managing footprint libraries. The widows are nearly identical.

Symbol Library – Creating Models

The window that opens is where you can manage which libraries are active and visible.

I'm not sure if a library can be deactivated but still visible, but it can be active and invisible.

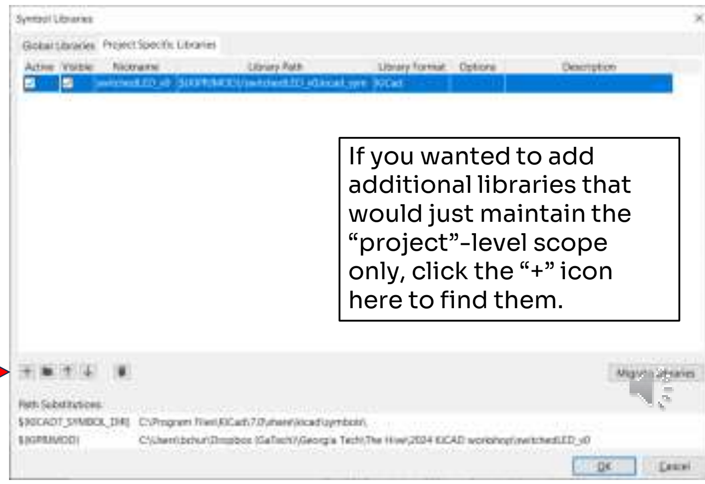


At the top, the two tabs let you switch between library scopes. Let's switch to "Project Specific Libraries".

Symbol Library – Creating Models

Conveniently, the library we just made is automatically added as both active and visible. Excellent – we can now use the symbols from within it.

Job well done.

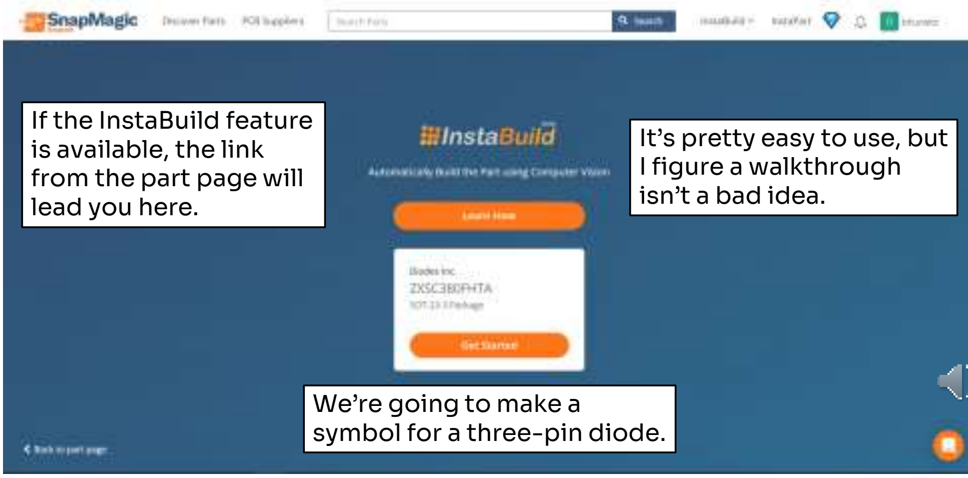


If you wanted to add additional libraries that would just maintain the "project"-level scope only, click the "+" icon here to find them.

SnapMagic's InstaBuild

- I'm going to spend the last few minutes demonstrating SnapMagic's InstaBuild symbol-generator.
- This is a browser-based "computer-vision-based" symbol generator.
- It's only available for symbols that specifically offer it on their parts page on the SnapMagic website.

Symbol Library – Creating Models

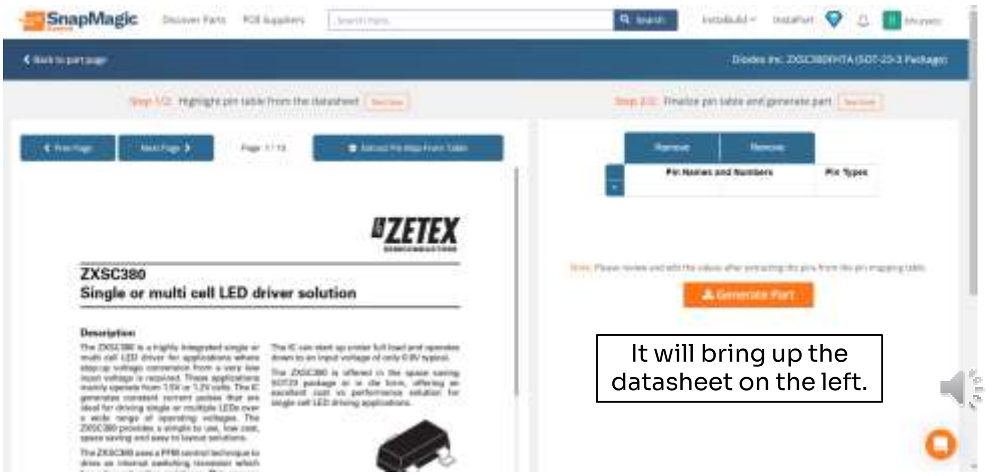


If the InstaBuild feature is available, the link from the part page will lead you here.

It’s pretty easy to use, but I figure a walkthrough isn’t a bad idea.

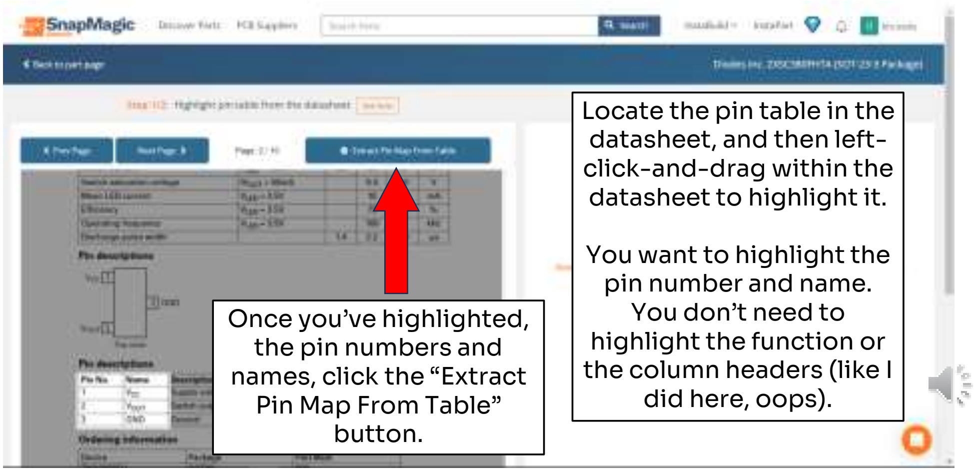
We’re going to make a symbol for a three-pin diode.

Symbol Library – Creating Models

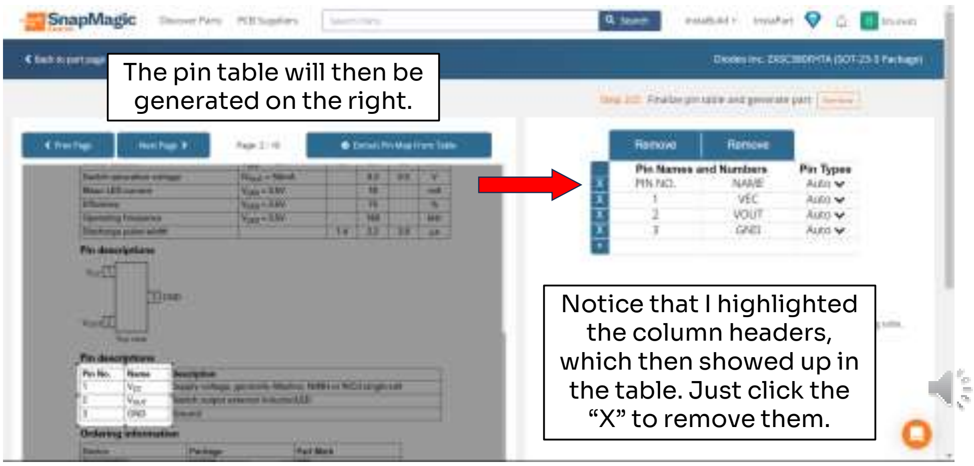


It will bring up the datasheet on the left.

Symbol Library – Creating Models



Symbol Library – Creating Models



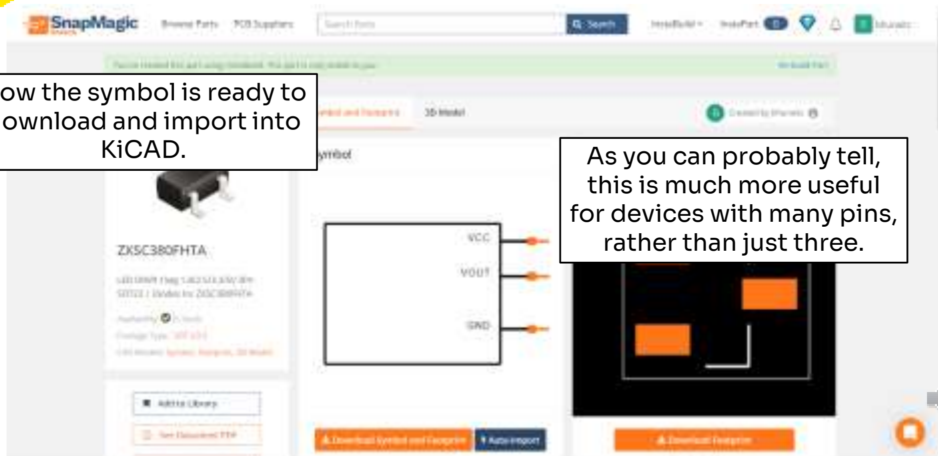
Symbol Library – Creating Models

Voilà!

Now just click “Generate Part” and the symbol will be generated.

Don’t expect anything fancy.

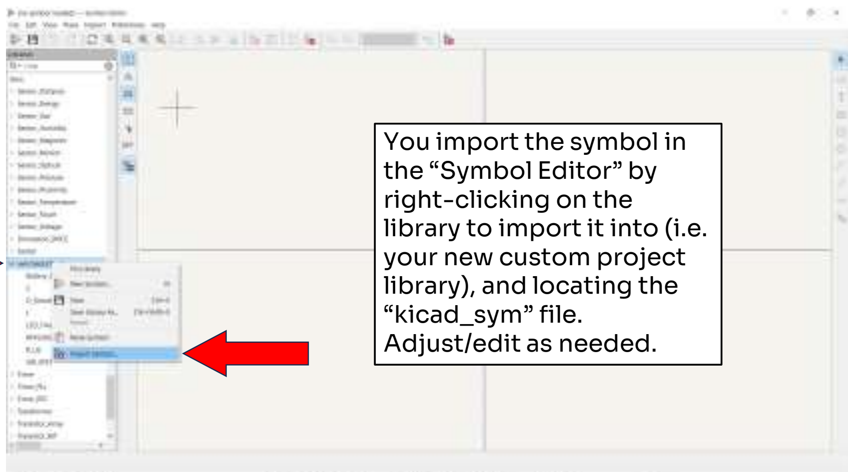
Symbol Library – Creating Models



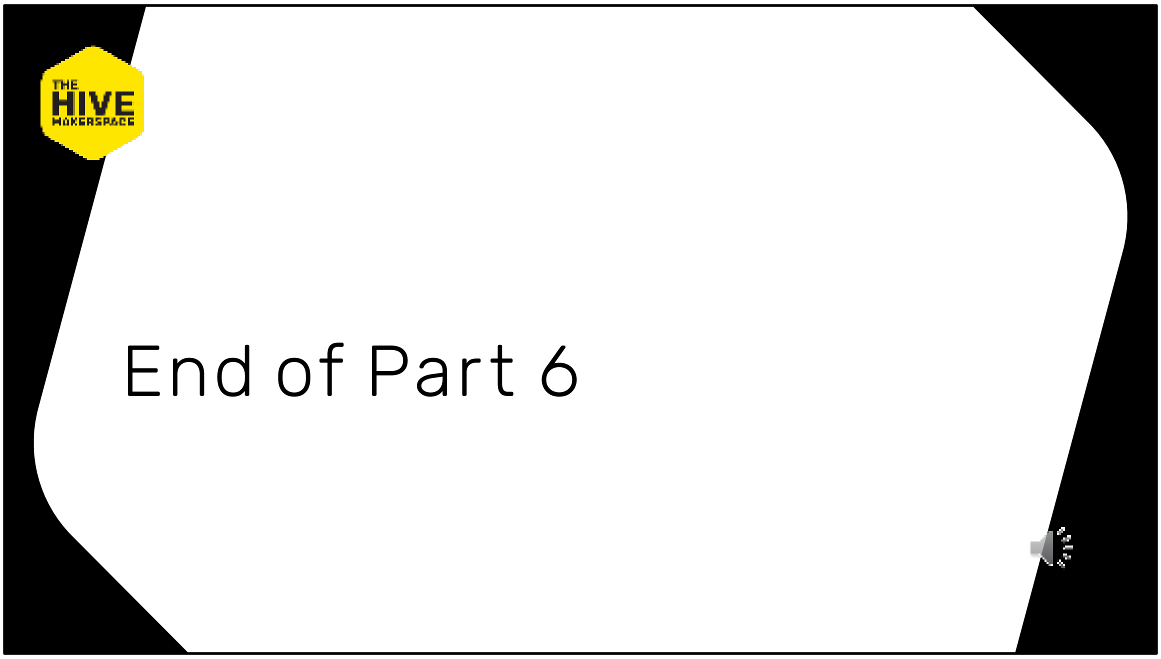
Now the symbol is ready to download and import into KiCAD.

As you can probably tell, this is much more useful for devices with many pins, rather than just three.

Symbol Library – Creating Models



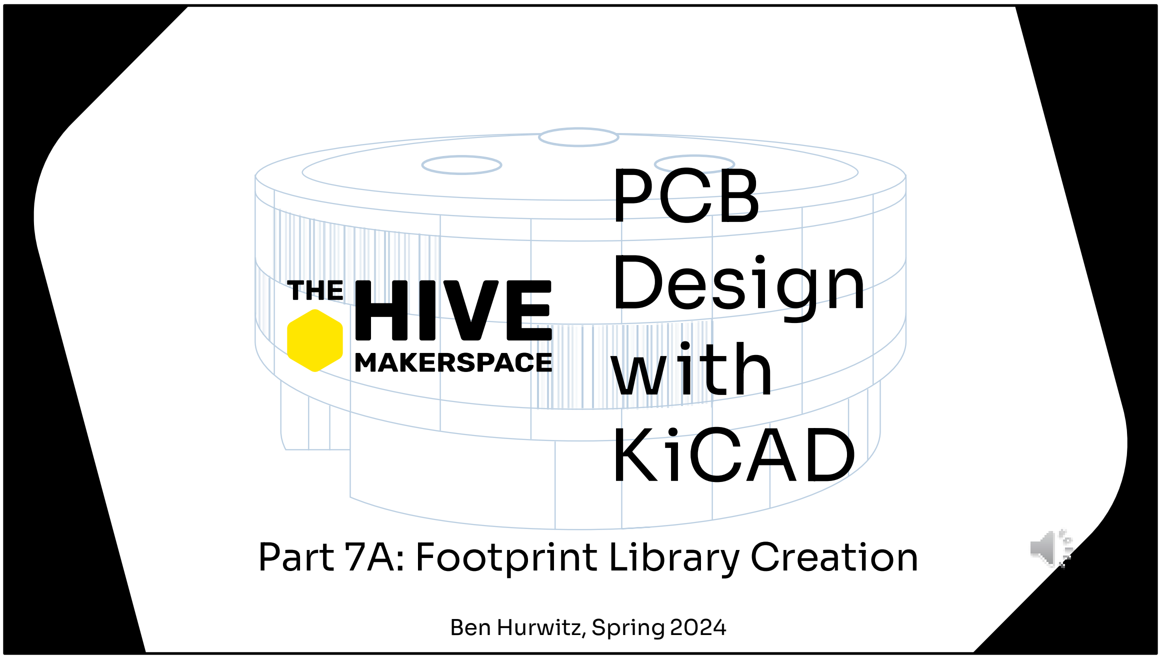
I explained this process earlier, but to import a downloaded symbol, right-click the library from within the Symbol Editor, and select “Import Symbol”. Then navigate to, and open, the “kicad_sym” symbol model file you downloaded. Edit and adjust as needed or wanted, and then save.



And that ends part 6 of this video series, in which we covered creating and filling our own project-specific library. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In the next video in the series, part 7A, we'll cover creating a footprint library and populating it with globally-available models from KiCAD's built-in libraries.

See you then!



Hi all, welcome to The Hive's series on PCB Design with KiCAD. My name is Ben, and in this series, we've been walking through the PCB design process using KiCAD as our electronics design software.

The previous videos went through the design process all the way through, resulting in a complete PCB ready for fabrication. One thing that I mentioned during that process, and was featured in the original "EDA Design Flow" in part 2, was library management, and the idea of using only project-scope libraries, but when we actually did the design, I ignored this for simplicity and time-constraints.

The last video, part 6, went through a project-specific symbol library.

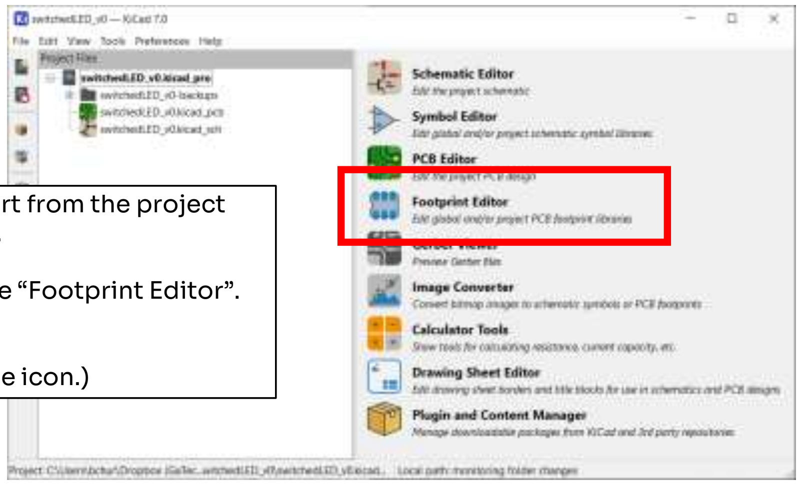
In this video, I will walk you through generating a single project-scoped footprint library to package with the rest of your project, and keep your work insulated from external changes, and then populating it with some of KiCAD's built-in models.

This material is of course not required for a functional design, but it is good design practice, for KiCAD at least, to keep all your parts in a project-level library.

Because this is not related directly to the design flow of the previous videos, I’ll make no assumptions about the state of your system or knowledge. So I apologize if some of this is repetition for some of you.

Let’s get started.

Footprint Library

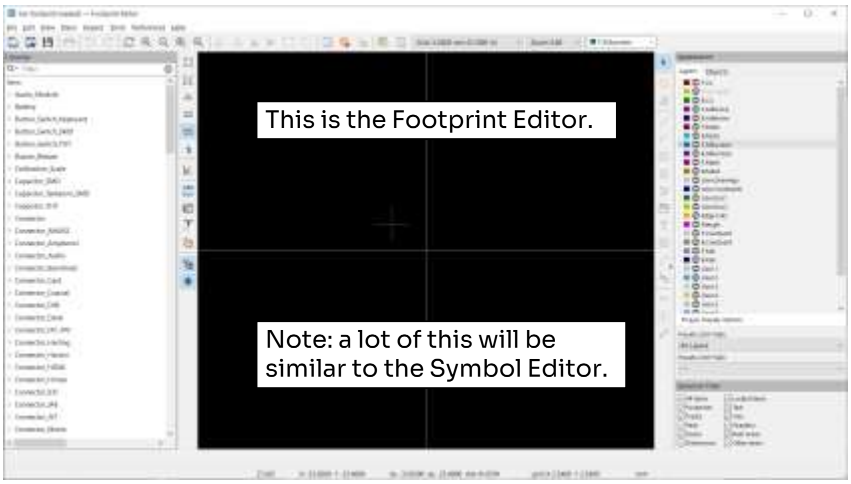


We'll start from the project window.

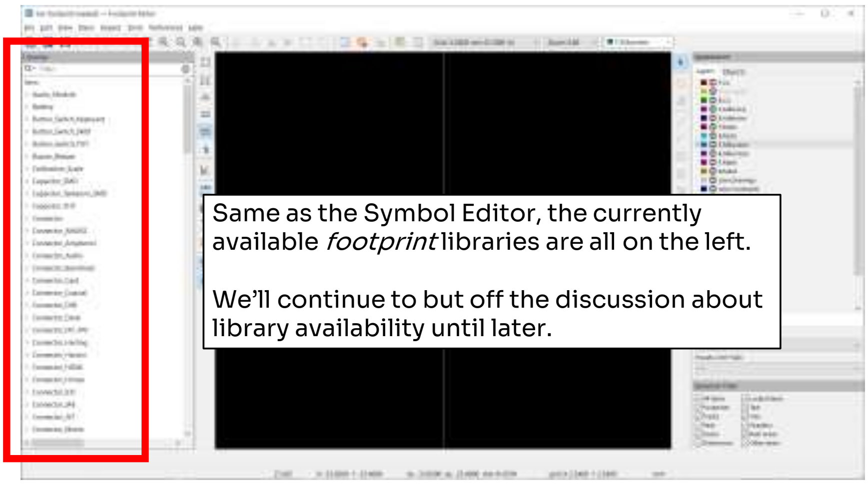
Open the "Footprint Editor".

(Click the icon.)

Footprint Library



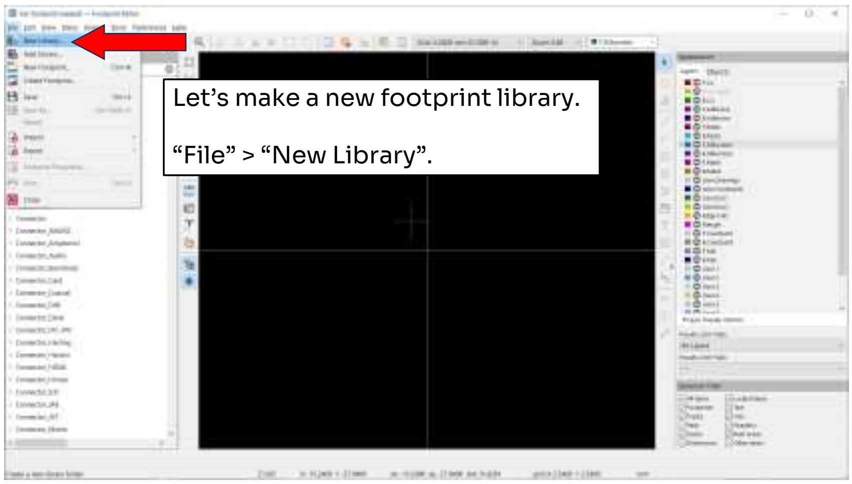
Footprint Library



Same as the Symbol Editor, the currently available *footprint* libraries are all on the left.

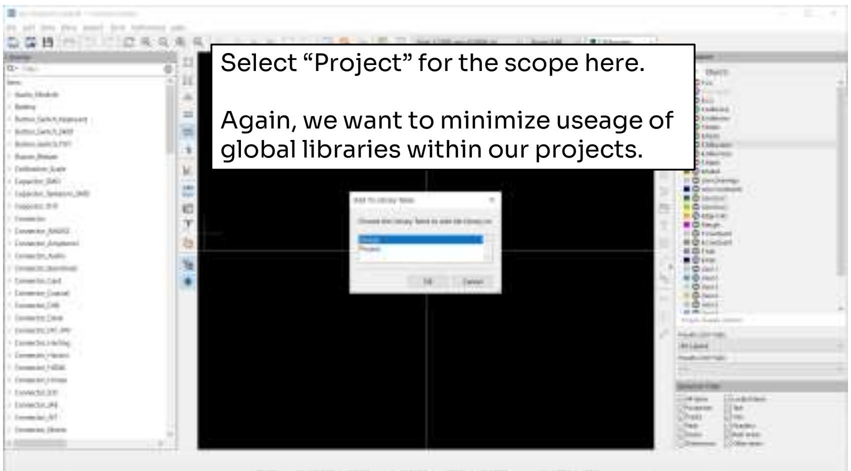
We'll continue to but off the discussion about library availability until later.

Footprint Library



We’ll be using the flashlight circuit that was developed in videos part 1-5 as our parts list to add, so if you’ve already made a footprint library during those videos, don’t bother to make another one.

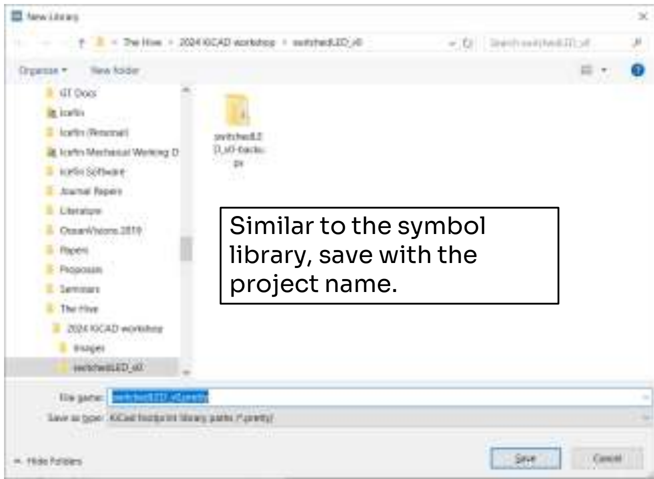
Footprint Library



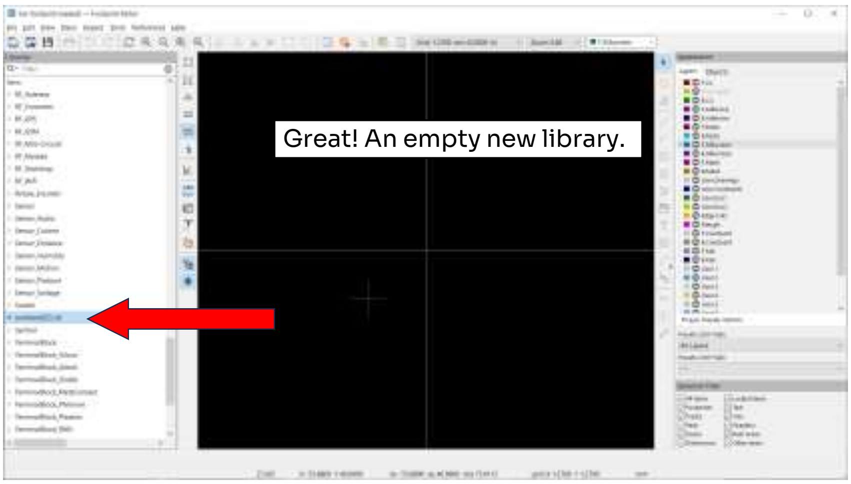
Select “Project” for the scope here.

Again, we want to minimize useage of global libraries within our projects.

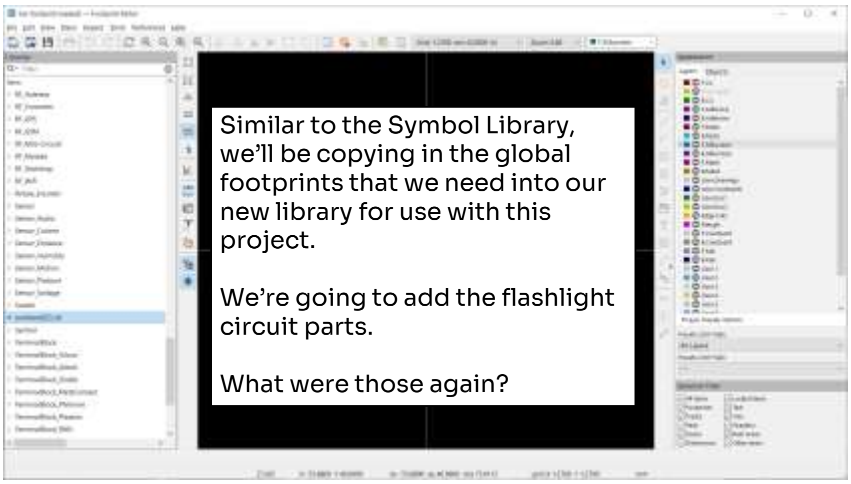
Footprint Library



Footprint Library



Footprint Library



Similar to the Symbol Library, we'll be copying in the global footprints that we need into our new library for use with this project.

We're going to add the flashlight circuit parts.

What were those again?

Parts List

Description	Part Num.	Mounting	Footprint
LED drive IC	RT4526GJ6	SMD	TSOT-23-6 (≤ 3.1 x 1.8 x 1 mm)
Battery holder	BC2032-E2	TH	Custom
Switch	TS02-66-70-BK-160-LCR-D	TH	4-TH 6mm x 6mm
Cin, 2.2uF	C3216X5R1C225KT	SMD	1206/3116 (3.1 x 1.6 x 0.55 mm)
Cout, 1uF	C3216X7R1C105KT	SMD	1206/3116 (3.1 x 1.6 x 0.55 mm)
L, 22uH	LBR2518T220M (22uH)	SMD	1008/2518 (2.5 x 1.8 x 1.8 mm)
D	PMEG6030ELPX	SMD	SOD-128 (4 x 2.7 x 1.1 mm)
Rset, 30 Ω	Unknown (from kit)	SMD	1206/3116 (3.1 x 1.6 x 0.55 mm)
LED	C512A-WNN-CZ0B0151	TH	5mm diam, 0.6mm lead holes

Don’t worry, you don’t have to memorize this.

Note! Blindly using global footprints can leave you exposed to potential issues if the parts aren’t actually standard.

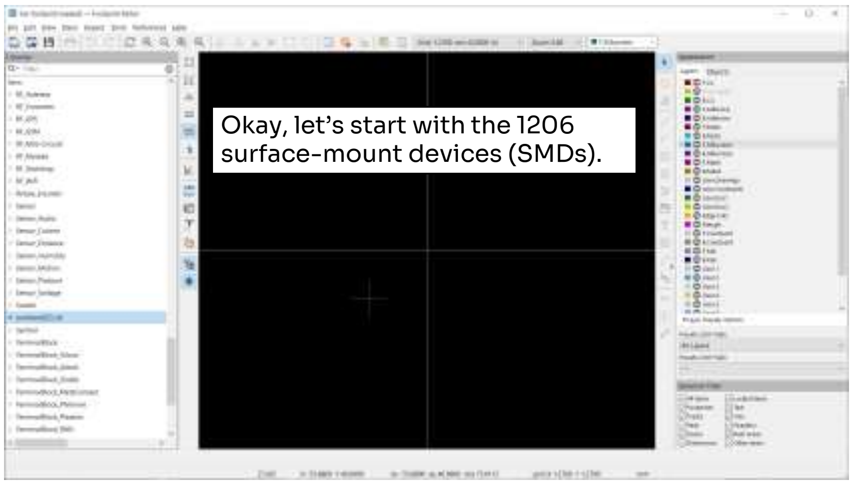
It’s up to you as the designer to confirm the dimensions of your parts and footprints.

Failure to do so is at your own risk.

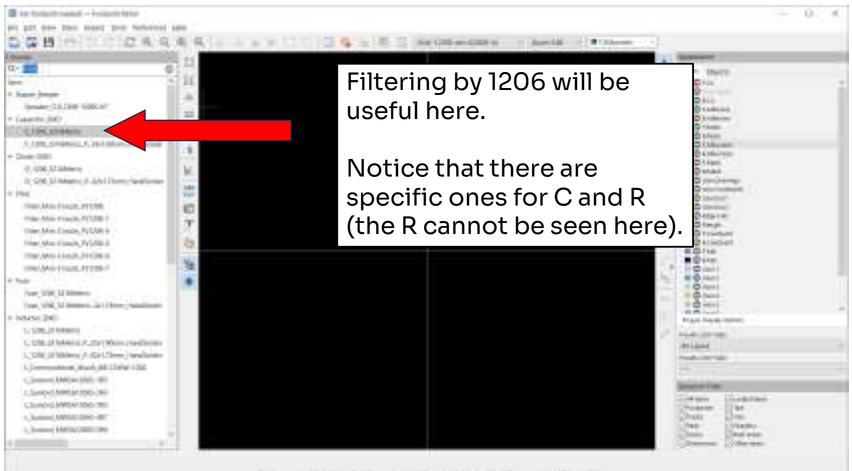
Assume, and make an ass out of you and me.

(I’ll leave that for an exercise for the reader.)

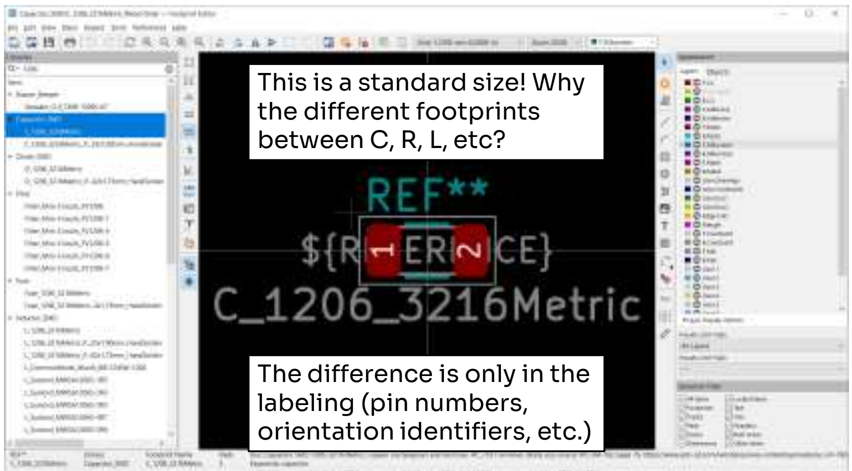
Footprint Library



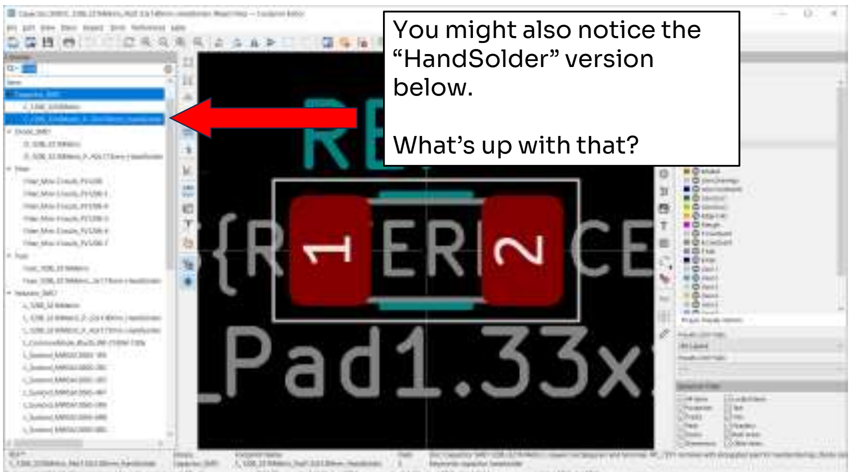
Footprint Library



Footprint Library



Footprint Library



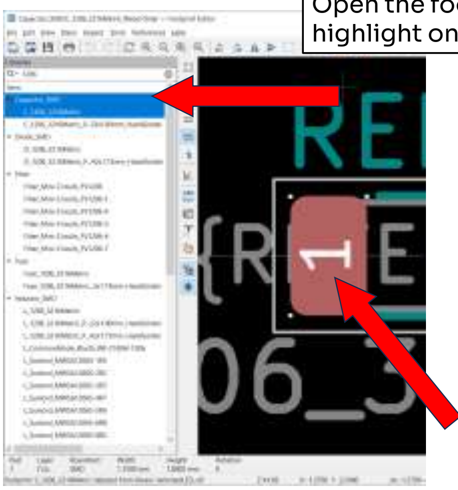
You might also notice the "HandSolder" version below.

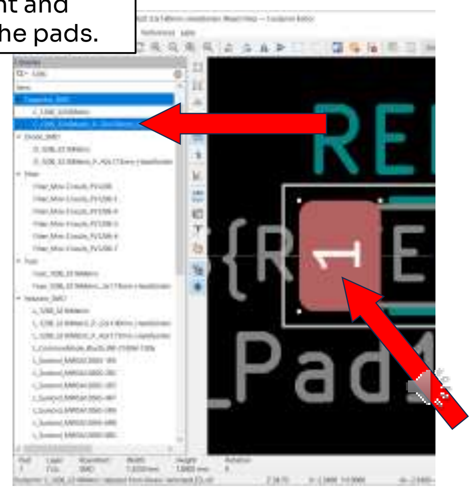
What's up with that?

(Answer on next slides)

Footprint Library

Open the footprint and highlight one of the pads.



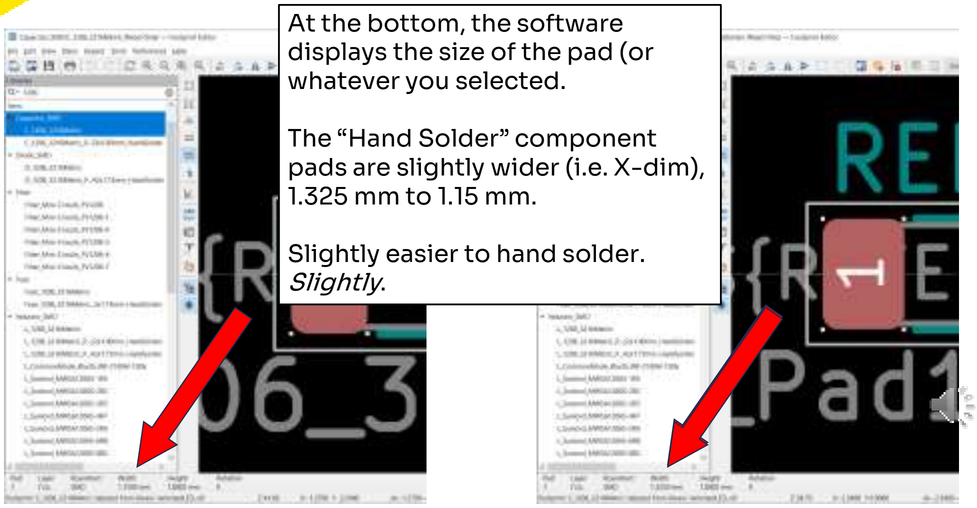


Footprint Library

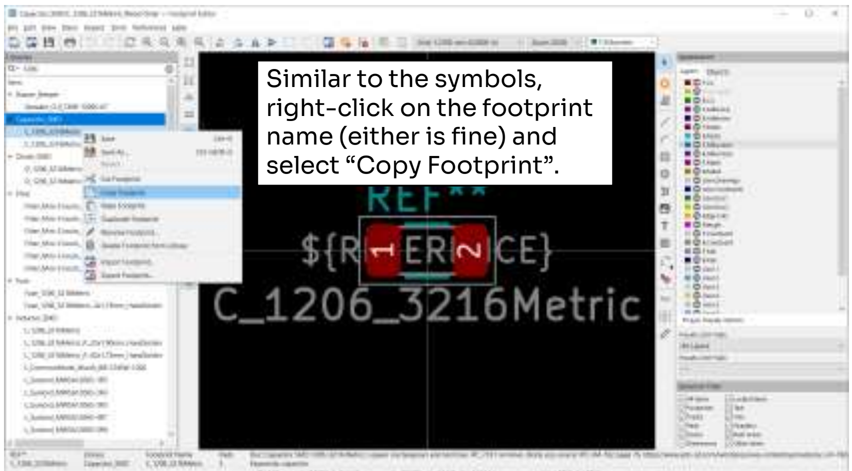
At the bottom, the software displays the size of the pad (or whatever you selected.

The “Hand Solder” component pads are slightly wider (i.e. X-dim), 1.325 mm to 1.15 mm.

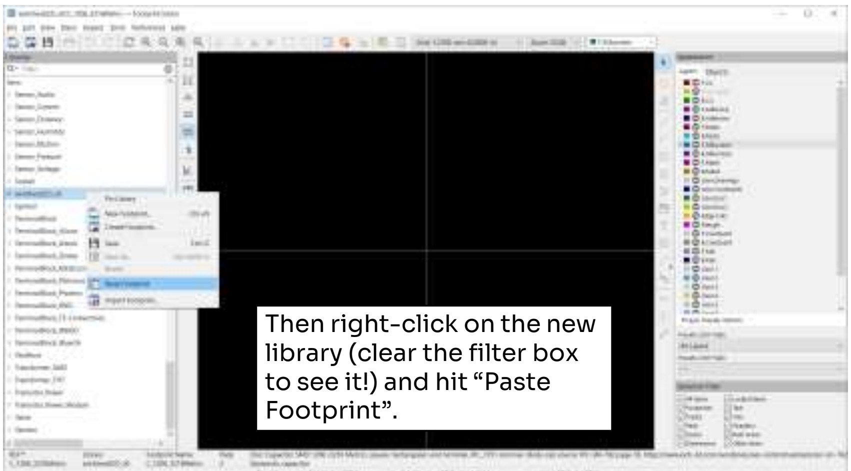
Slightly easier to hand solder. *Slightly*.



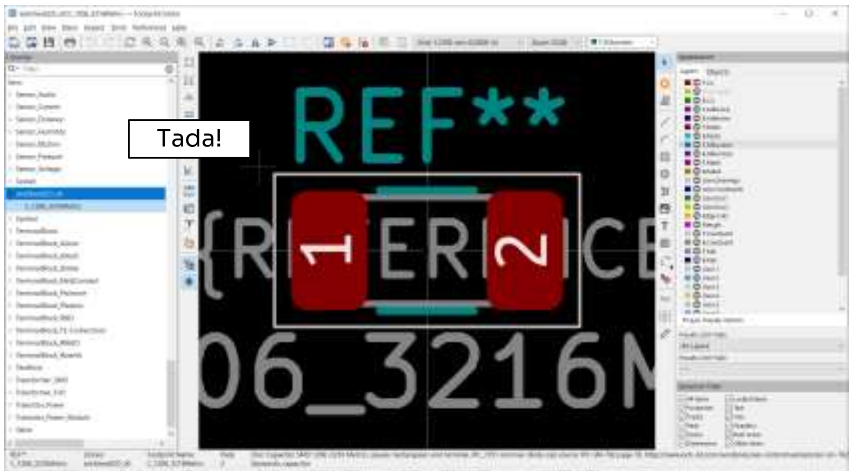
Footprint Library



Footprint Library

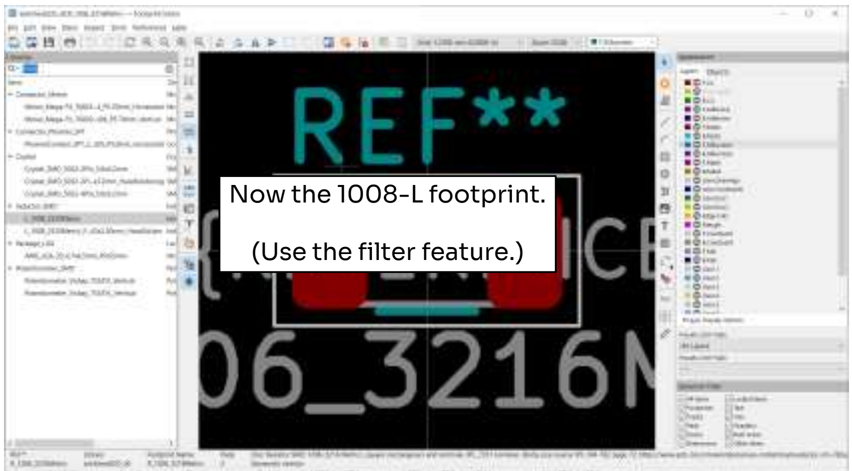


Footprint Library

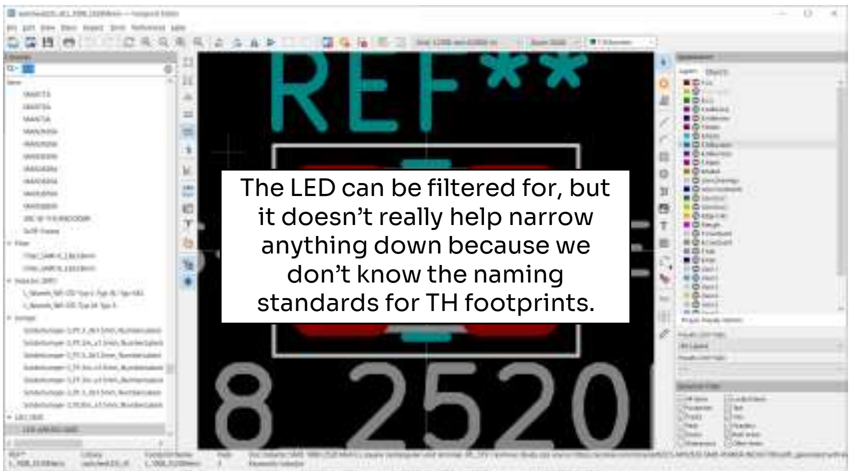


Footprint Library

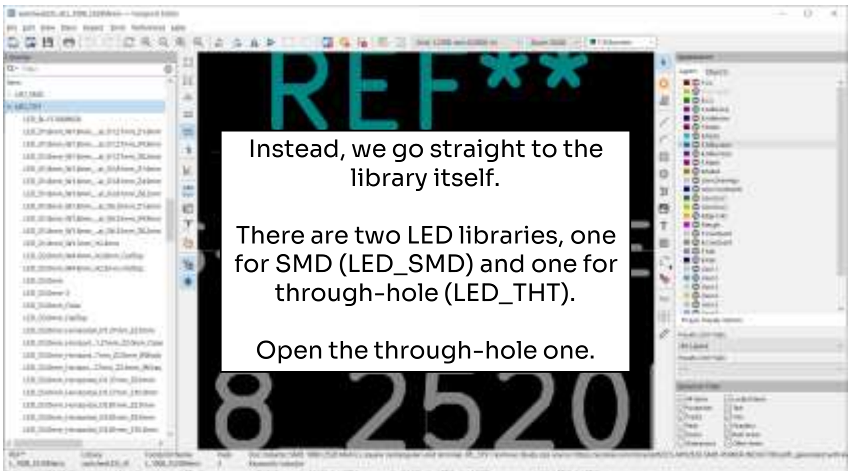
Footprint Library



Footprint Library



Footprint Library

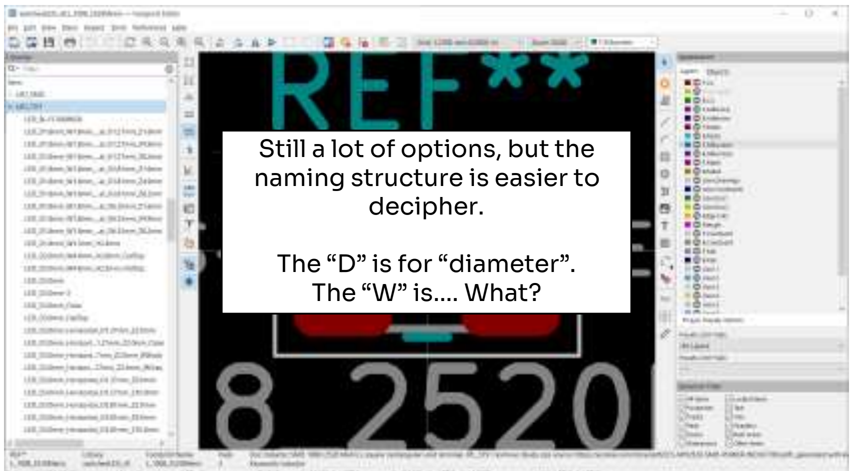


Instead, we go straight to the library itself.

There are two LED libraries, one for SMD (LED_SMD) and one for through-hole (LED_THT).

Open the through-hole one.

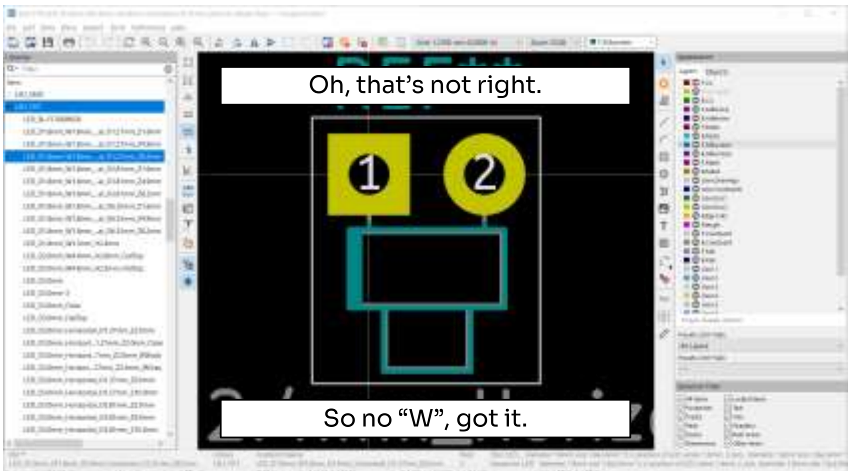
Footprint Library



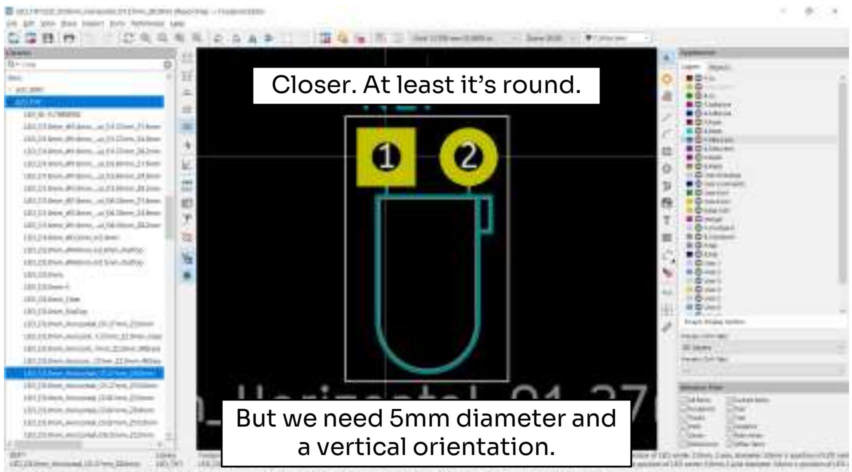
Still a lot of options, but the naming structure is easier to decipher.

The “D” is for “diameter”.
The “W” is…. What?

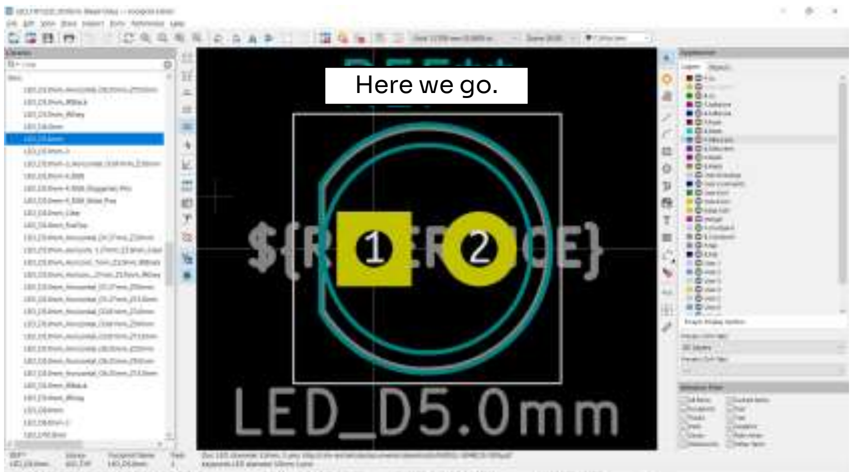
Footprint Library



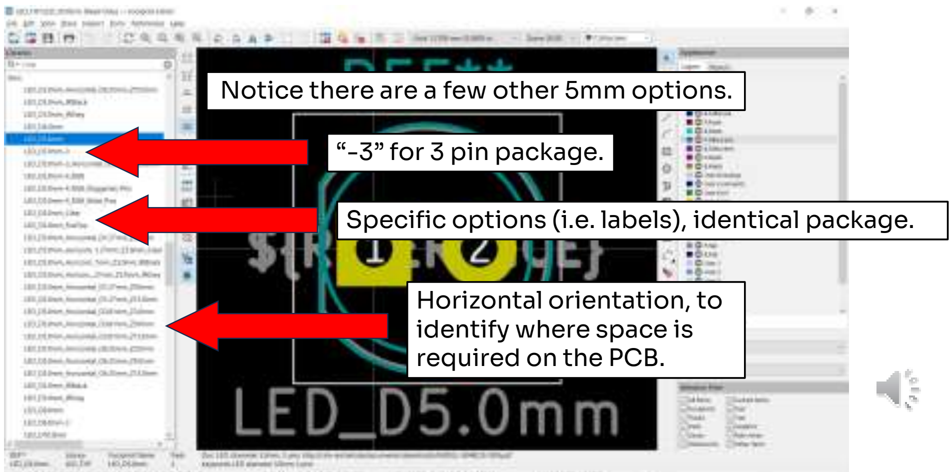
Footprint Library



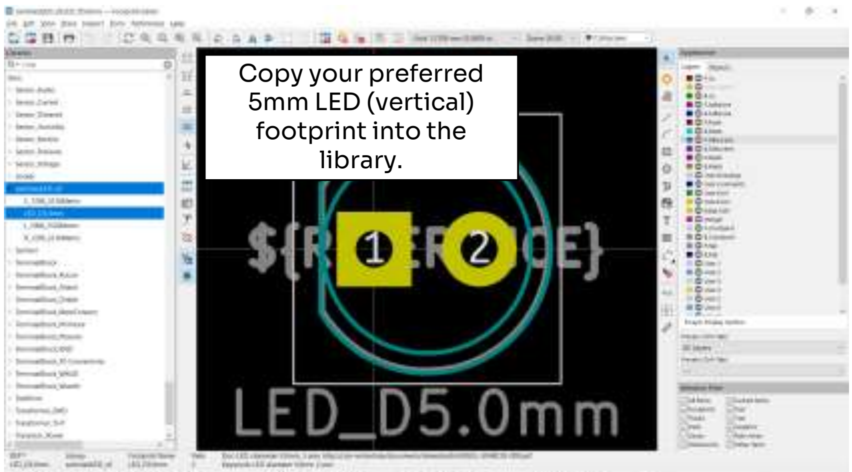
Footprint Library



Footprint Library



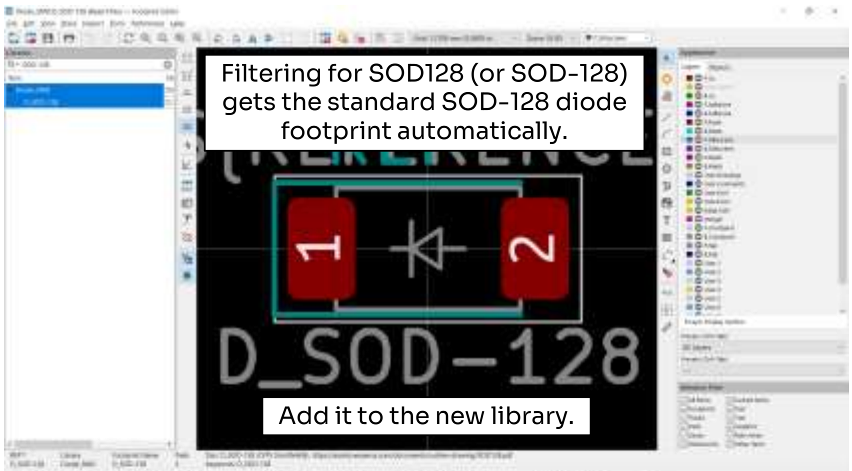
Footprint Library



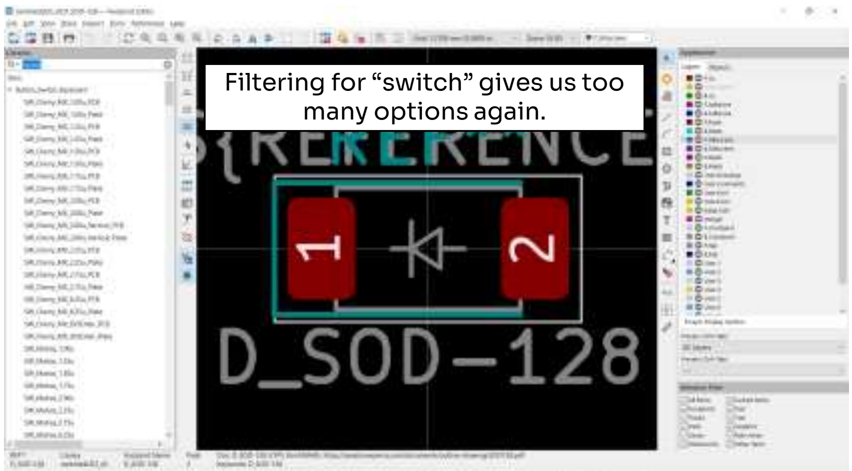
What's left?

	Description	Part Num.	Mounting	Footprint
	LED drive IC	RT4526GJ6	SMD	TSOT-23-6 (≤ 3.1 x 1.8 x 1 mm)
	Battery holder	BC2032-E2	TH	Custom
	Switch	TS02-66-70-BK-160-LCR-D	TH	4-TH 6mm x 6mm
✓	~~Cin, 2.2uF~~	~~C3216X5R1C225KT~~	~~SMD~~	~~1206/3116 (3.1 x 1.6 x 0.55 mm)~~
✓	~~Cout, 1uF~~	~~C3216X7R1C105KT~~	~~SMD~~	~~1206/3116 (3.1 x 1.6 x 0.55 mm)~~
✓	~~L, 22uH~~	~~LBR2518T220M (22uH)~~	~~SMD~~	~~1008/2518 (2.5 x 1.8 x 1.8 mm)~~
	D	PMEG6030ELPX	SMD	SOD-128 (4 x 2.7 x 1.1 mm)
✓	~~Rset, 30 Ω~~	~~Unknown (from kit)~~	~~SMD~~	~~1206/3116 (3.1 x 1.6 x 0.55 mm)~~
✓	~~LED~~	~~C512A-WNN-CZ0B0151~~	~~TH~~	~~5mm diam, 0.6mm lead holes~~

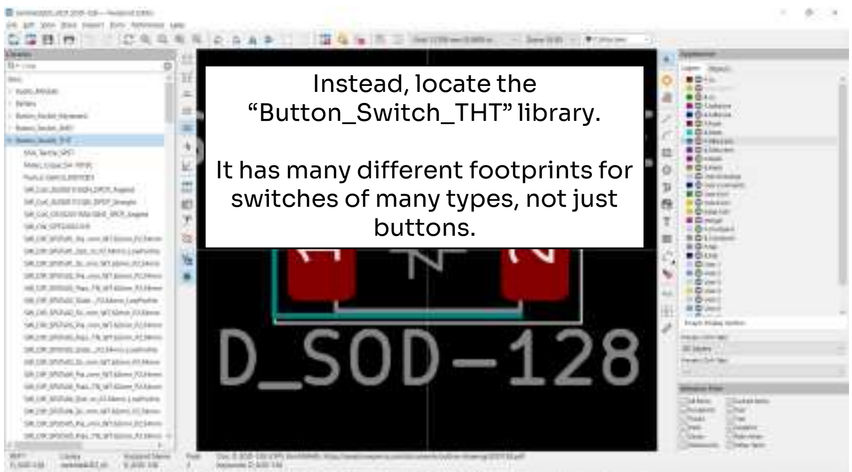
Footprint Library



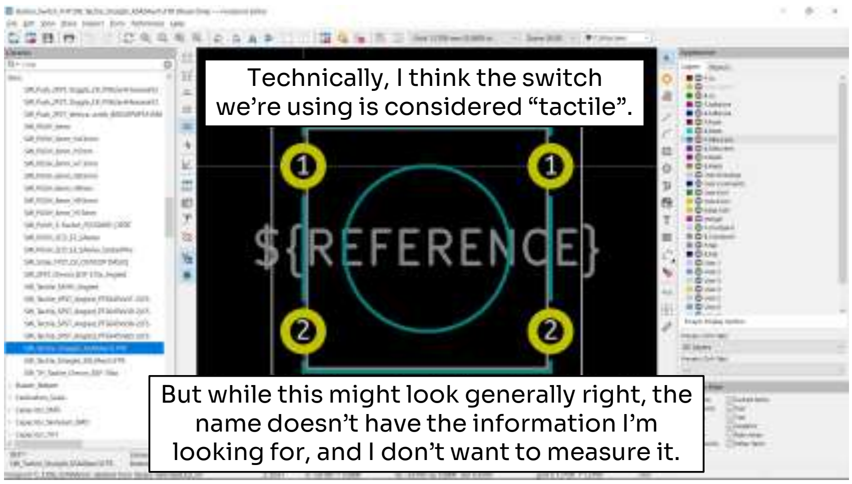
Footprint Library



Footprint Library

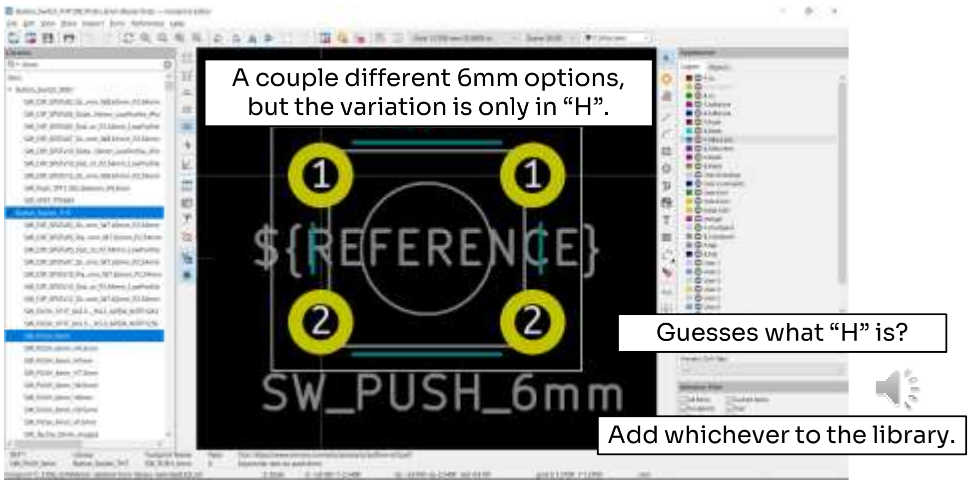


Footprint Library

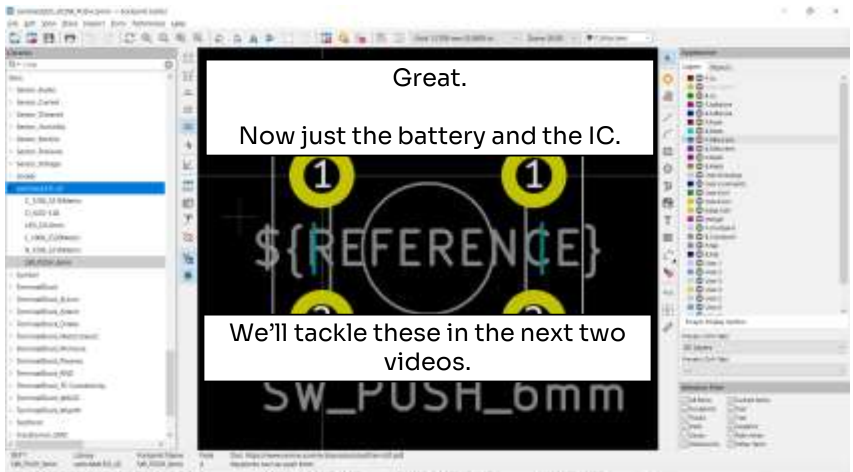


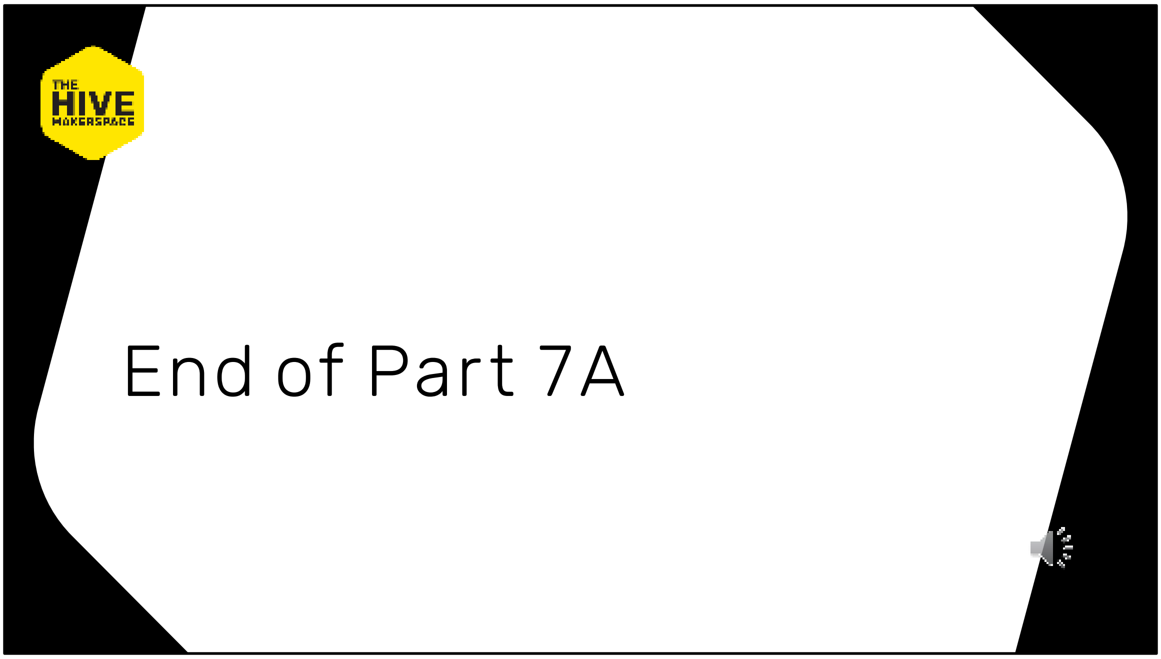
Footprint Library

Footprint Library



Footprint Library

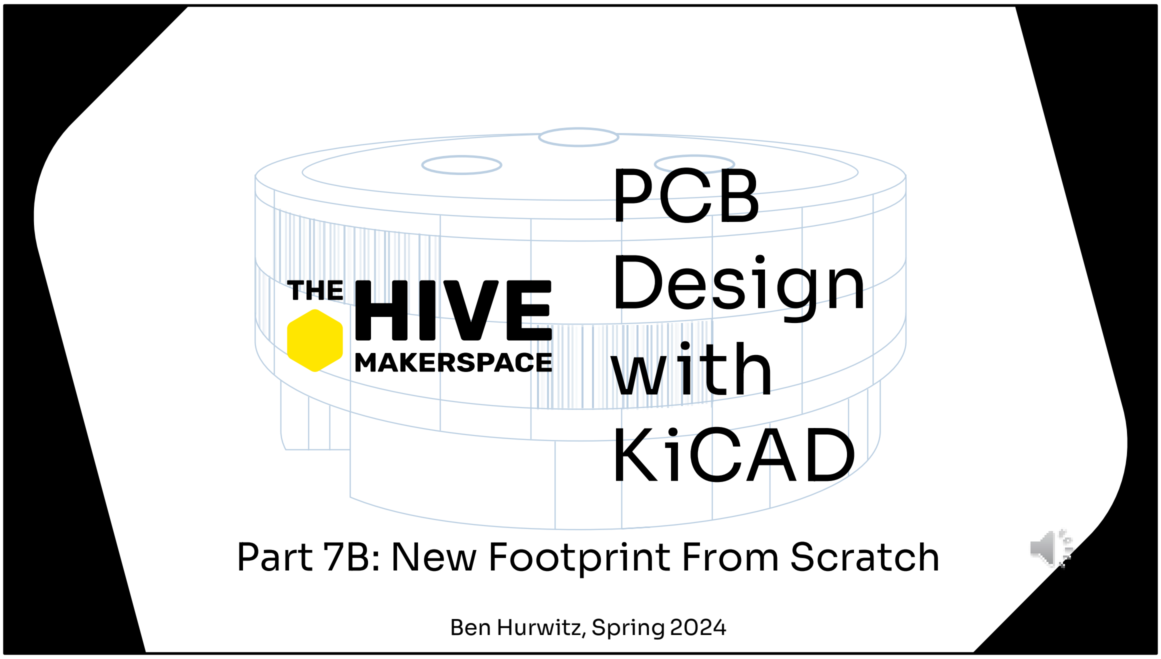




And that ends part 7A, in which we covered creating a new footprint library and copying in global components. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In the next video, I'll walk through the process of creating non-standard footprints, like the battery holder, from scratch in KiCAD, and then importing the footprint from the internet instead.

See you then!



Hi all, welcome to The Hive's series on PCB Design with KiCAD. My name is Ben, and in this series, we've been walking through the PCB design process using KiCAD as our electronics design software.

The previous videos went through the design process all the way through, resulting in a complete PCB ready for fabrication. One thing that I mentioned during that process, and was featured in the original "EDA Design Flow" in part 2, was library management, and the idea of using only project-scope libraries, but when we actually did the design, I ignored this for simplicity and time-constraints.

The last video, part 7A, we made a new project-specific footprint library, and populated it with a bunch of globally-available footprints that we needed.

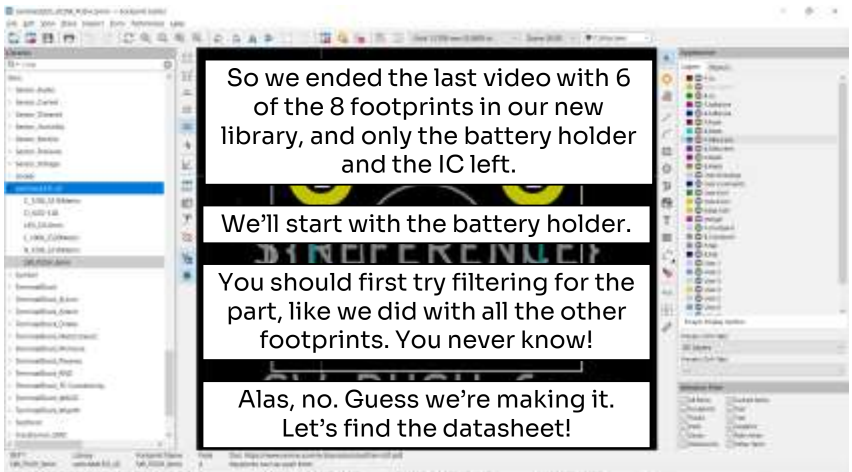
In this video, we'll look at generating a custom footprint from scratch, why you should never do it again, and then how to import an online model.

This material is of course not required for a functional design, but it is good design practice, for KiCAD at least, to keep all your parts in a project-level library.

Because this is not related directly to the design flow of the previous videos, I'll make no assumptions about the state of your system or knowledge other than that you watched part 7A. So I apologize if some of this is repetition for some of you.

Let's get started.

Footprint Library



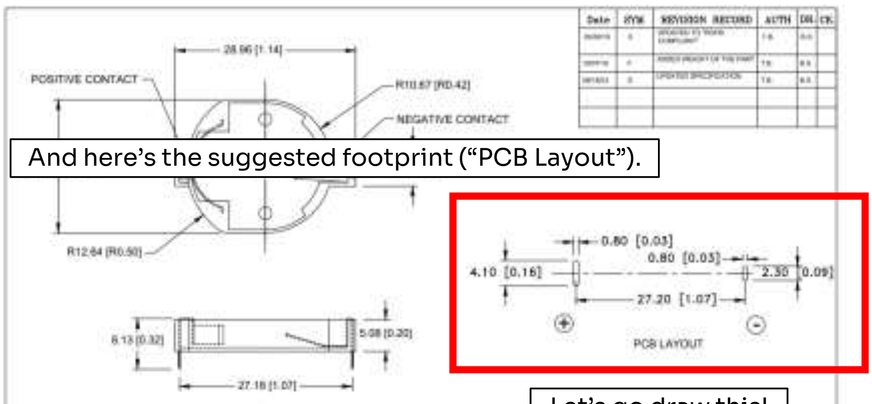
Footprint Library

I had to hunt down the technical drawing for the battery holder on the manufacturer's website because the datasheet link on Digikey gave me their catalogue.

Not helpful.

Footprint Library

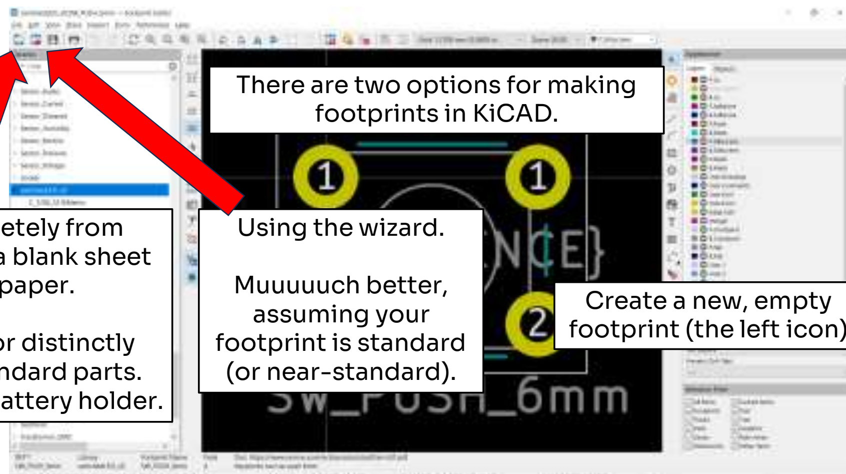
Here’s the technical drawing (the top half, at least).



And here’s the suggested footprint (“PCB Layout”).

Let’s go draw this!

Footprint Library



There are two options for making footprints in KiCAD.

Completely from scratch, a blank sheet of paper.

Good for distinctly non-standard parts. Like this battery holder.

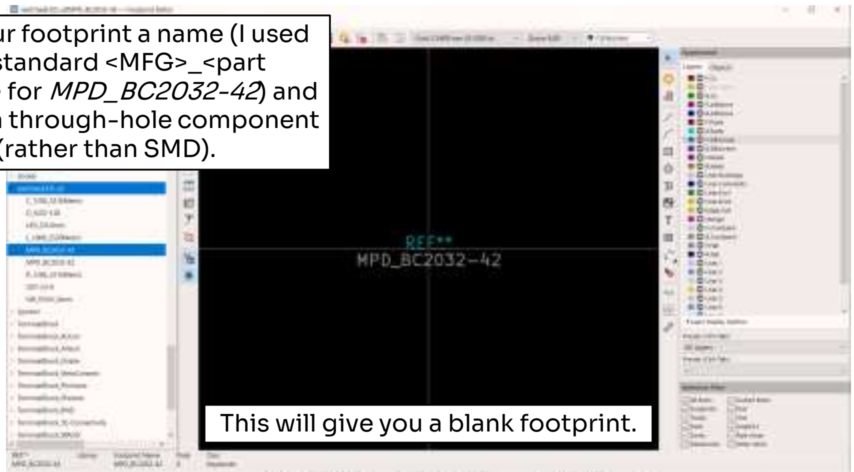
Using the wizard.

Muuuuuch better, assuming your footprint is standard (or near-standard).

Create a new, empty footprint (the left icon)

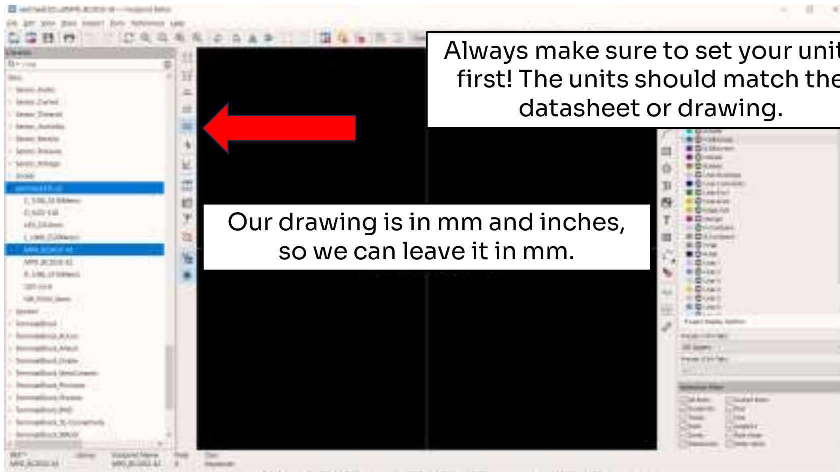
Footprint Library

Give your footprint a name (I used the standard <MFG>_<part number> for *MPD_BC2032-42*) and set it to a through-hole component (rather than SMD).



This will give you a blank footprint.

Footprint Library

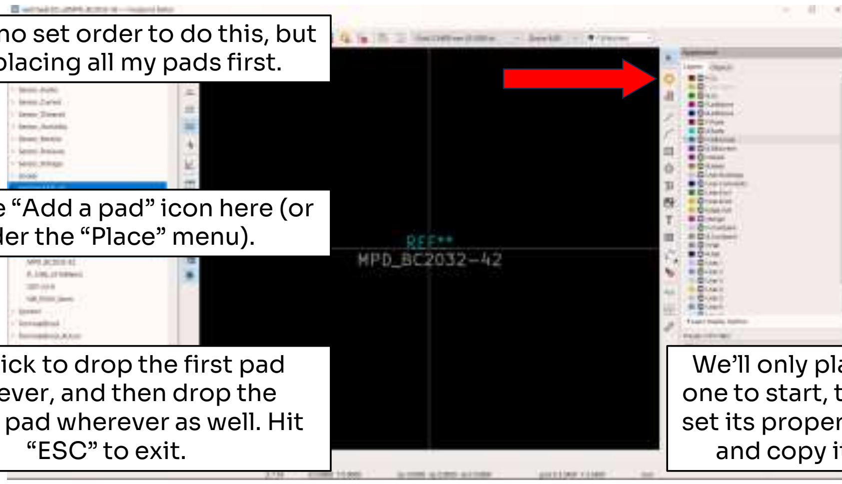


Always make sure to set your units first! The units should match the datasheet or drawing.

Our drawing is in mm and inches, so we can leave it in mm.

Footprint Library

There is no set order to do this, but I like placing all my pads first.



Click the “Add a pad” icon here (or under the “Place” menu).

Left click to drop the first pad wherever, and then drop the second pad wherever as well. Hit “ESC” to exit.

We’ll only place one to start, then set its properties and copy it.

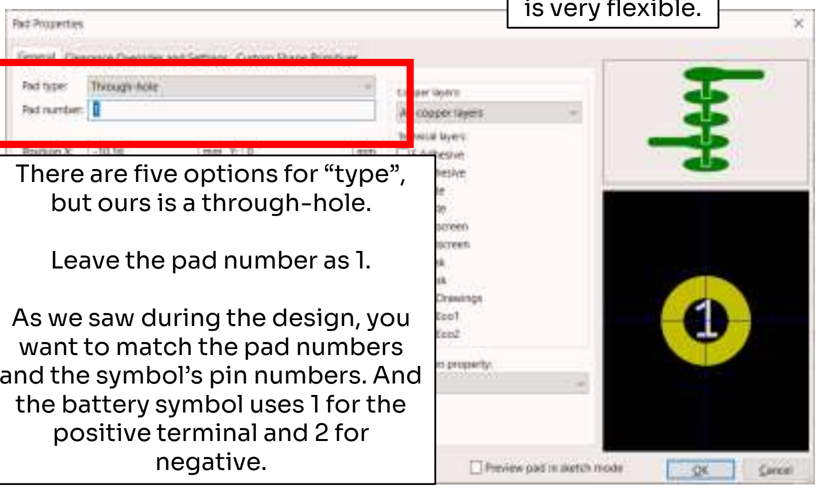
Footprint Library

Great.

Select the pad and hit “E” (or right-click and “Properties”) to open the “Pad Properties” window.

Footprint Library

Pad generation is very flexible.



There are five options for "type", but ours is a through-hole.

Leave the pad number as 1.

As we saw during the design, you want to match the pad numbers and the symbol's pin numbers. And the battery symbol uses 1 for the positive terminal and 2 for negative.

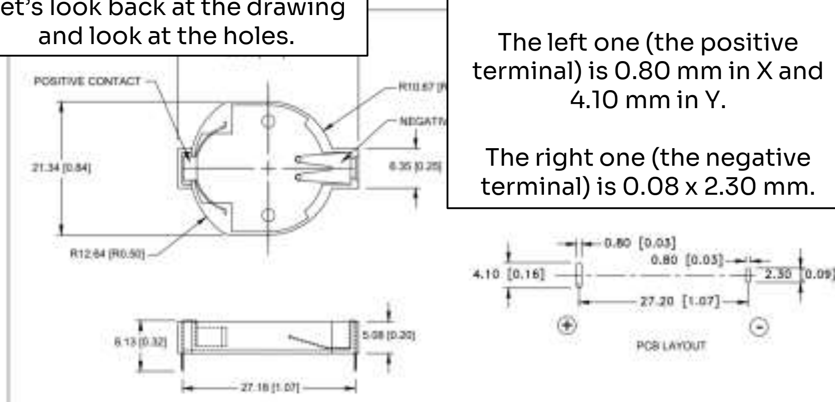
Footprint Library

Let’s look back at the drawing and look at the holes.

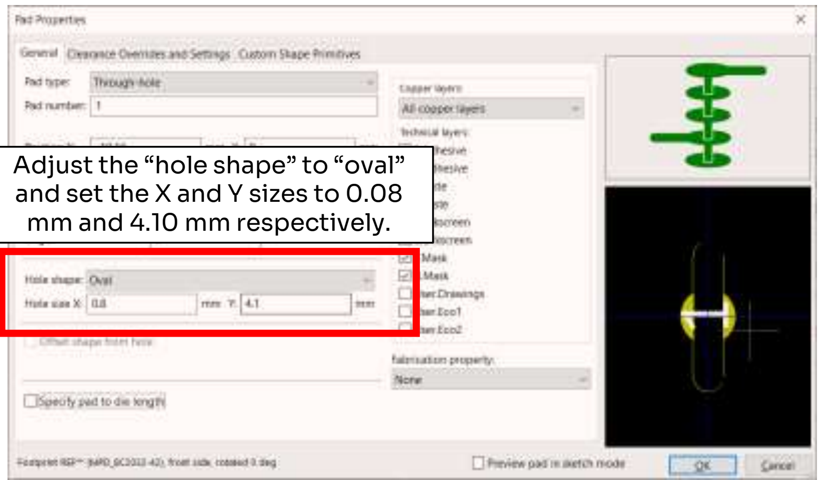
These are slots, so not circular. And they’re not the same!

The left one (the positive terminal) is 0.80 mm in X and 4.10 mm in Y.

The right one (the negative terminal) is 0.08 x 2.30 mm.

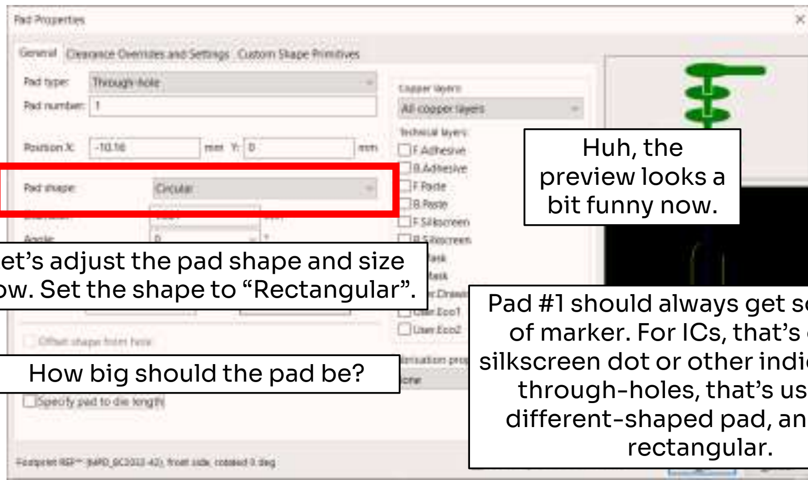


Footprint Library



Adjust the “hole shape” to “oval” and set the X and Y sizes to 0.08 mm and 4.10 mm respectively.

Footprint Library



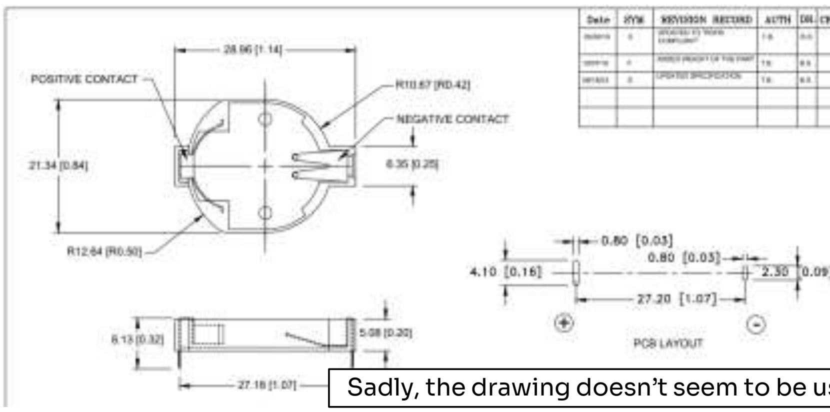
Huh, the preview looks a bit funny now.

Let's adjust the pad shape and size now. Set the shape to "Rectangular".

How big should the pad be?

Pad #1 should always get some sort of marker. For ICs, that's often a silkscreen dot or other indicator. For through-holes, that's usually a different-shaped pad, and often rectangular.

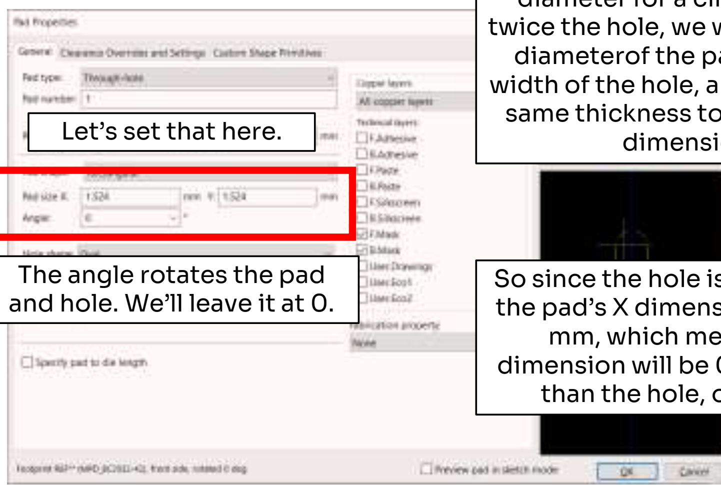
Footprint Library



Sadly, the drawing doesn’t seem to be useful here.

Footprint Library

Remembering that the pad diameter for a circular pad is twice the hole, we will make the X diameterof the pad twice the width of the hole, and the use the same thickness to add to the Y dimension.



Let’s set that here.

The angle rotates the pad and hole. We’ll leave it at 0.

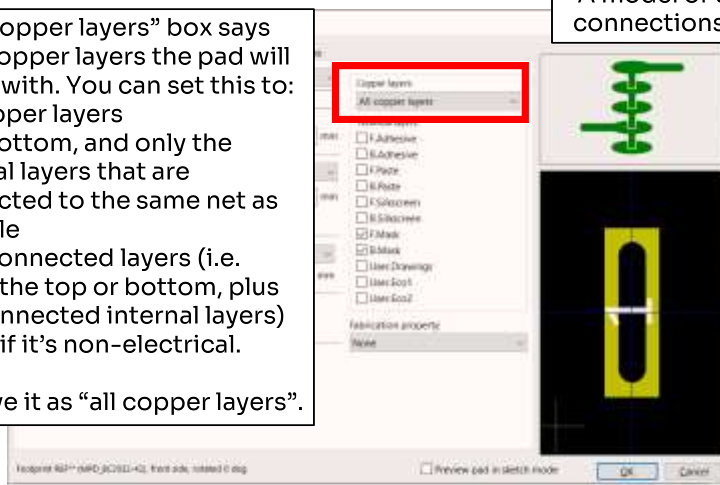
So since the hole is 0.8 x 4.1 mm, the pad’s X dimension will be 1.6 mm, which means the Y dimension will be 0.8 mm larger than the hole, or 4.9 mm.

Footprint Library

The "copper layers" box says which copper layers the pad will interact with. You can set this to:

- All copper layers
- Top, bottom, and only the internal layers that are connected to the same net as the hole
- Only connected layers (i.e. either the top or bottom, plus any connected internal layers)
- None, if it's non-electrical.

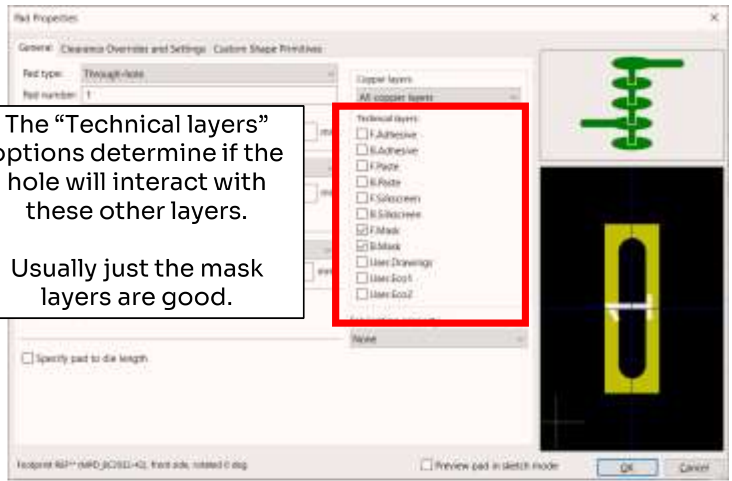
We'll leave it as "all copper layers".



A model of the hole connections is here.

Much better.

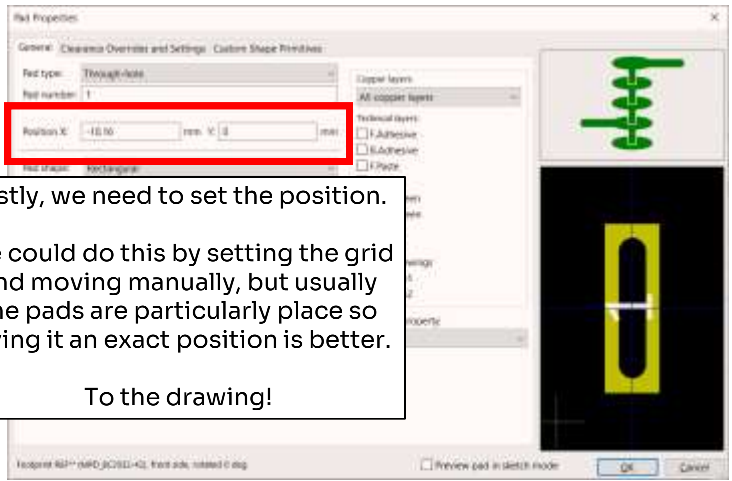
Footprint Library



The “Technical layers” options determine if the hole will interact with these other layers.

Usually just the mask layers are good.

Footprint Library



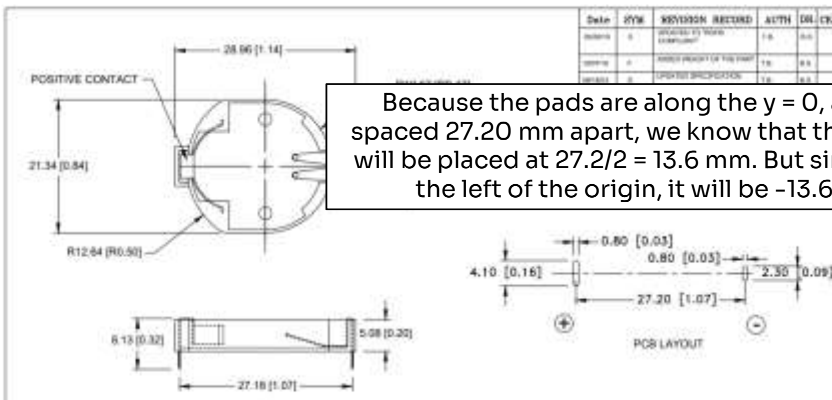
Lastly, we need to set the position.

We could do this by setting the grid and moving manually, but usually the pads are particularly place so giving it an exact position is better.

To the drawing!

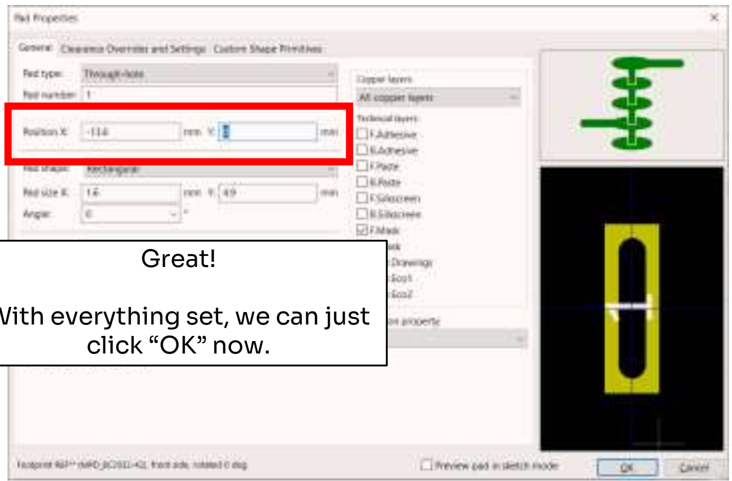
Footprint Library

It’s always a good idea to center the part at the origin.



Because the pads are along the y = 0, and are spaced 27.20 mm apart, we know that the left pad will be placed at 27.2/2 = 13.6 mm. But since it’s to the left of the origin, it will be -13.6mm.

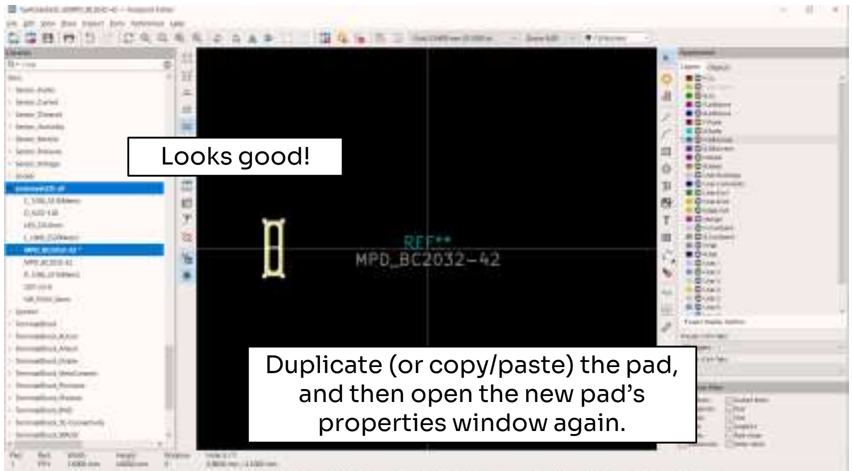
Footprint Library



Great!

With everything set, we can just click “OK” now.

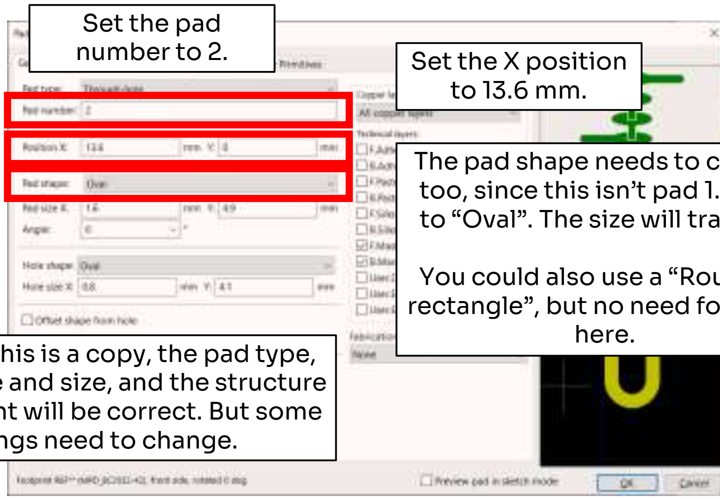
Footprint Library



Footprint Library

Set the pad number to 2.

Set the X position to 13.6 mm.

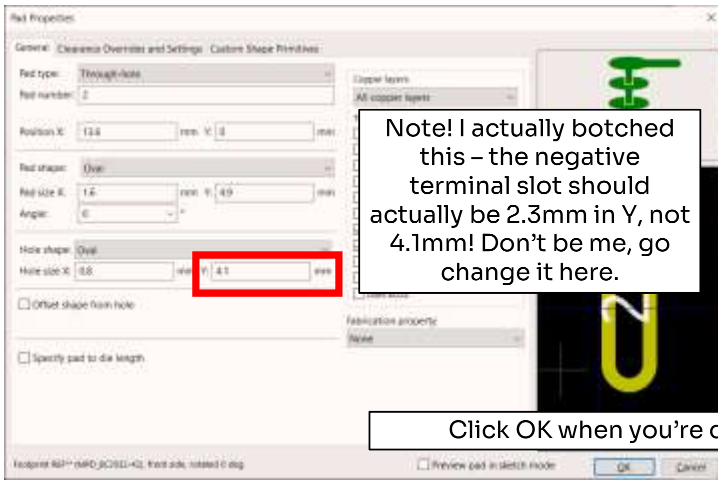


The pad shape needs to change too, since this isn't pad 1. Set it to "Oval". The size will transfer.

You could also use a "Rounded rectangle", but no need for fancy here.

Because this is a copy, the pad type, hole shape and size, and the structure on the right will be correct. But some things need to change.

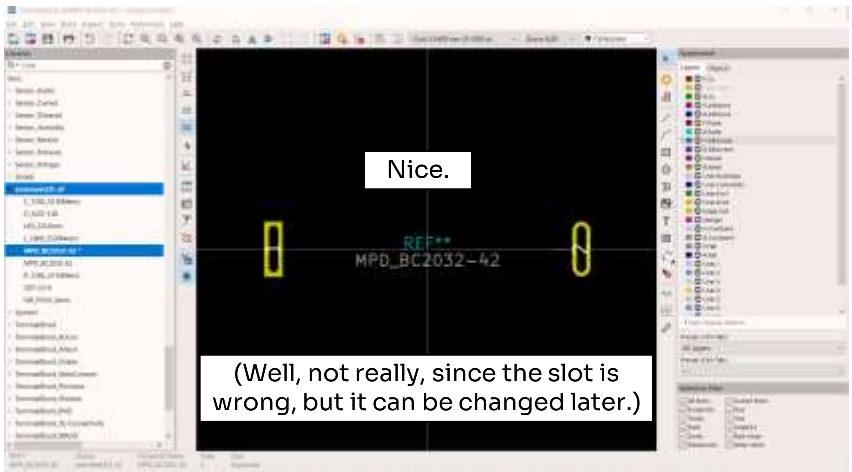
Footprint Library



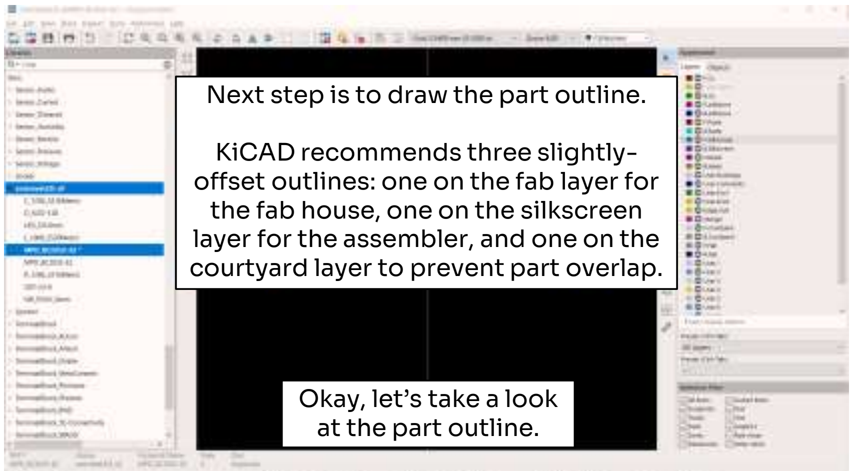
Note! I actually botched this – the negative terminal slot should actually be 2.3mm in Y, not 4.1mm! Don't be me, go change it here.

Click OK when you're done.

Footprint Library

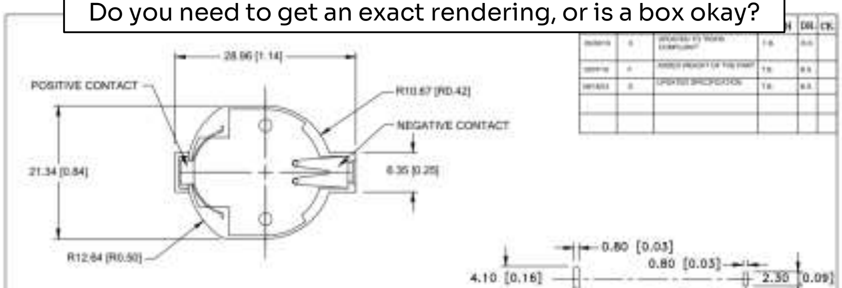


Footprint Library

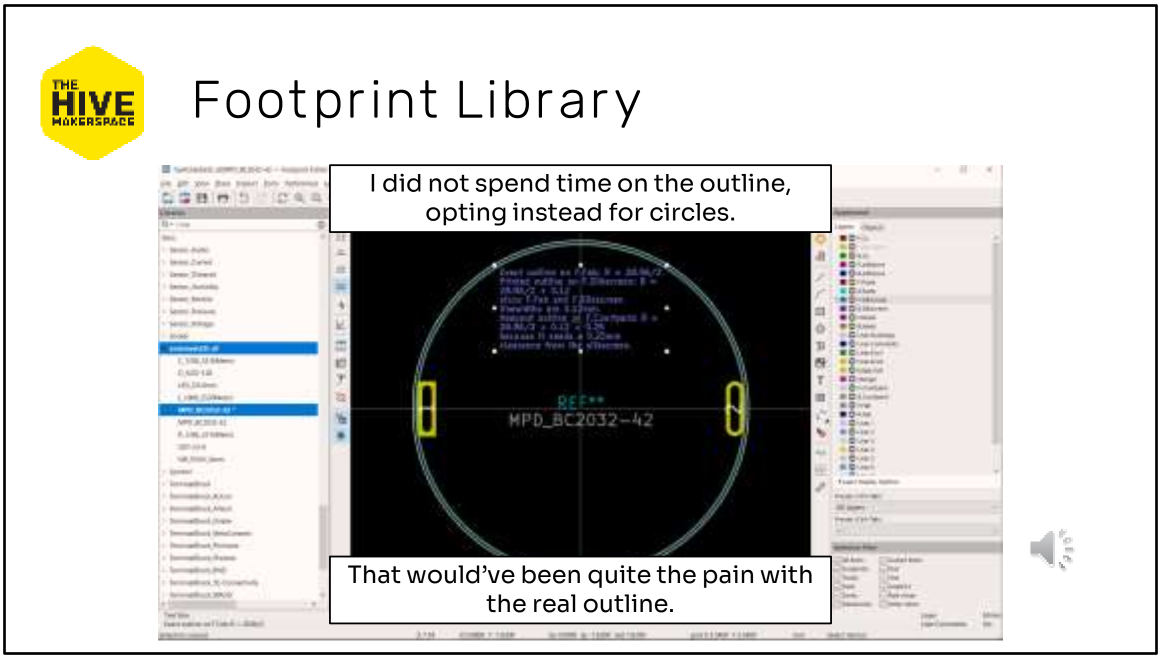


Footprint Library

How accurately you draw the outline is sortof up to you. Do you need to get an exact rendering, or is a box okay?



This footprint outline is going to be obnoxious to draw in KiCAD due to the non-primitive shape. We can draw arcs, but you'd basically specifying start and end coordinates and a radius. And there's not a wealth of info here.



I didn't bother with the exactly outline because, frankly, that would've been a pain and I don't care that much. So I made three circles.

The fab-layer circle approximates the part outline with a circle of diameter equal to the maximum X-dimension of 28.96 mm.

You don't want to overlap graphics because it's really hard to see them, so KiCAD recommends placing the silkscreen outline, which will be printed, immediately outside the fab outline. The circle is sized to be larger than the fab circle by half the linewidth of the fab layer plus half the linewidth of the silkscreen layer because the sizing is measured from the middle of a line. The linewidth can be found by looking at the properties of the circles drawn on those layers.

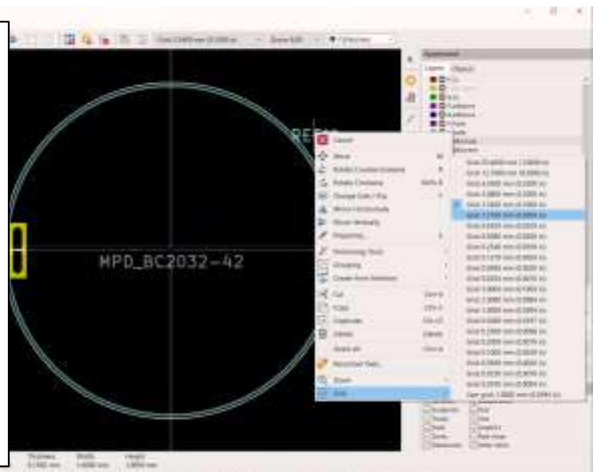
Lastly, the courtyard outline should be sized to have a 0.25mm clearance around the silkscreen outline, which is added to the radius.

These sizes are written in the image in purple on the User.Comments layer, so they won't show up anywhere that actually gets made.

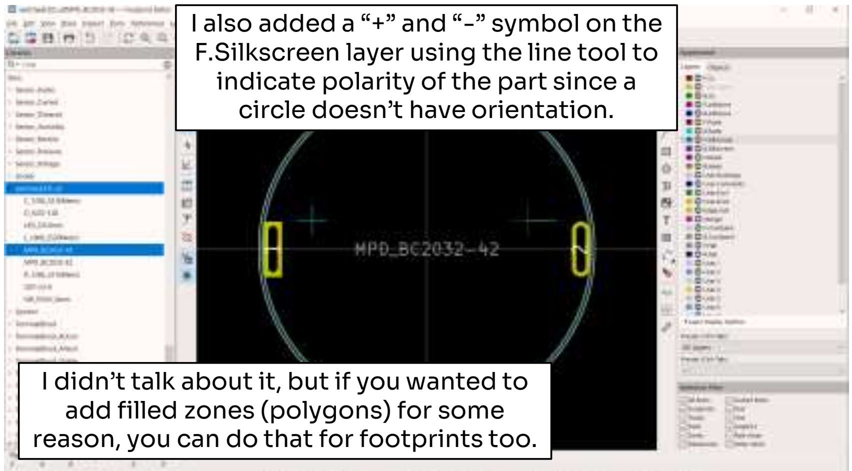
Footprint Library

We want to also move the reference designator, REF**, outside the outline so that the assembler can see it after the part is populated.

My grid was a bit too large, so while moving the REF** text, I right-clicked to open the context menu and adjust the grid through the Grid submenu.



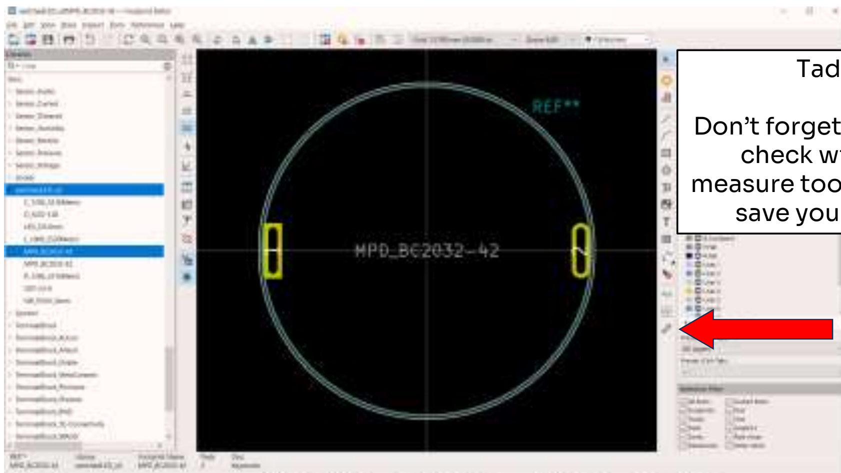
Footprint Library



I also added a "+" and "-" symbol on the F.Silkscreen layer using the line tool to indicate polarity of the part since a circle doesn't have orientation.

I didn't talk about it, but if you wanted to add filled zones (polygons) for some reason, you can do that for footprints too.

Footprint Library



Tada!

Don't forget to double check with the measure tool, and then save your work!

The measure tool icon is highlighted by the arrow.

Footprint Library

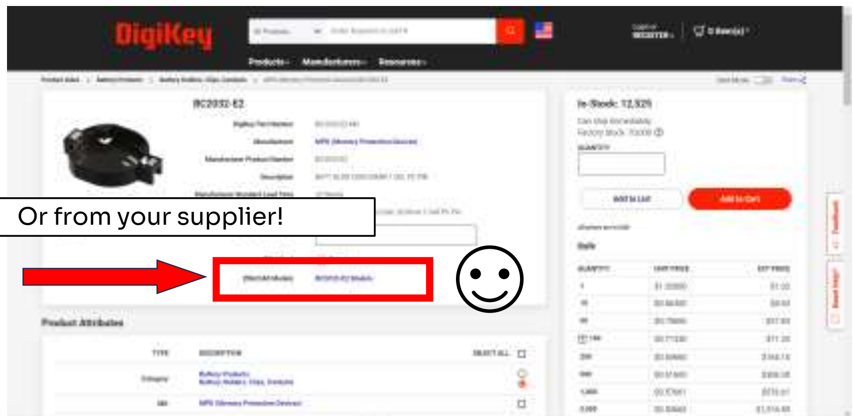
Holy moly.

So that wasn't so bad, but mostly because I opted to draw circles for the outline.

That won't always be possible, nor will datasheets always be any good, potentially making custom footprint generation nightmarish.

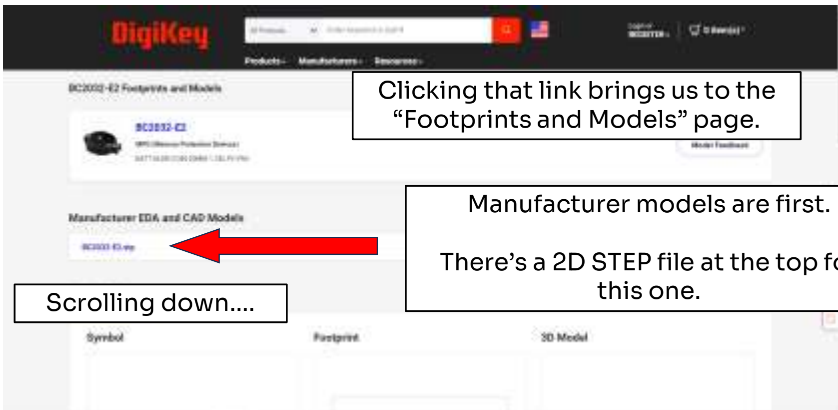
Thankfully, most of the time we don't have to do this because we can either locate or request the models from UltraLibrarian or SnapMagic.

Footprint Library



Or from your supplier!

Footprint Library



Clicking that link brings us to the “Footprints and Models” page.

Manufacturer models are first.

There’s a 2D STEP file at the top for this one.

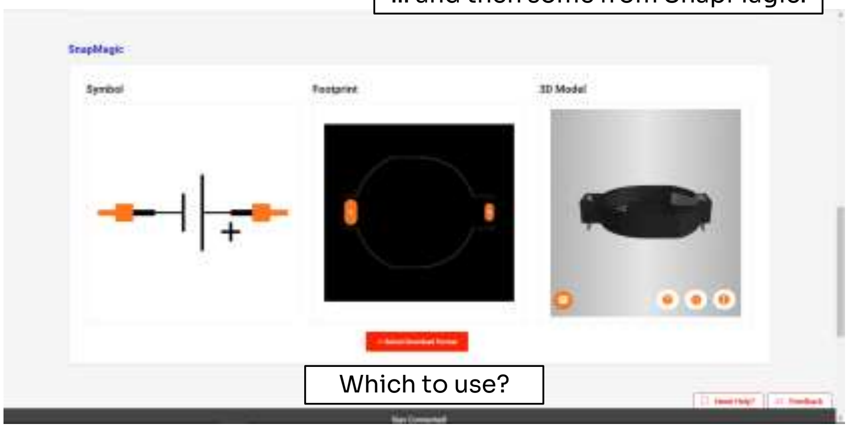
Scrolling down....

Footprint Library

... There's a set of models from UltraLibrarian...

Footprint Library

... and then some from SnapMagic.



Which to use?

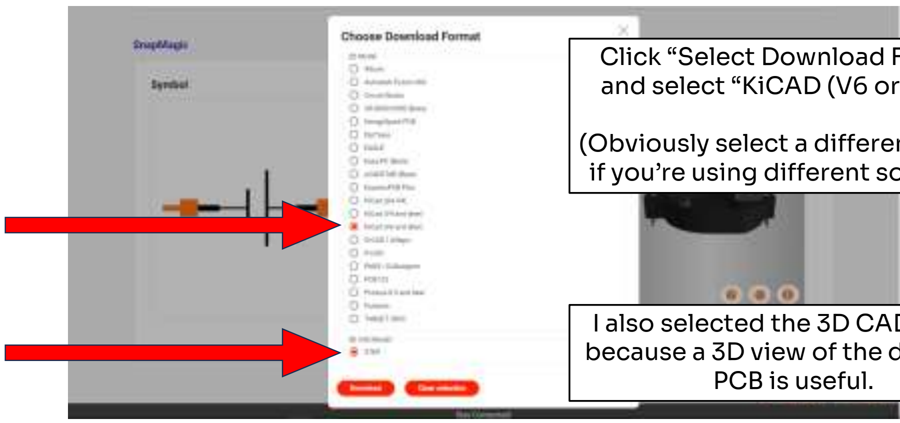
Footprint Library

There is no right or wrong answer here generally. Both models are extremely likely to be sufficient.

It’s on you to confirm that though!

I’m going to use SnapMagic here because I like the footprint better (and there’s a 3D model).

Footprint Library



Click “Select Download Format” and select “KiCAD (V6 or later)”.

(Obviously select a different format if you’re using different software.)

I also selected the 3D CAD model because a 3D view of the designed PCB is useful.

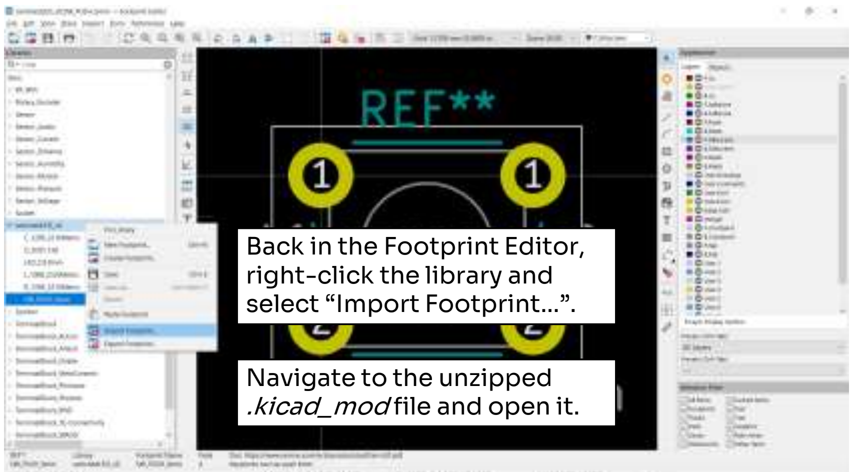
Footprint Library

Unzip the download somewhere accessible.

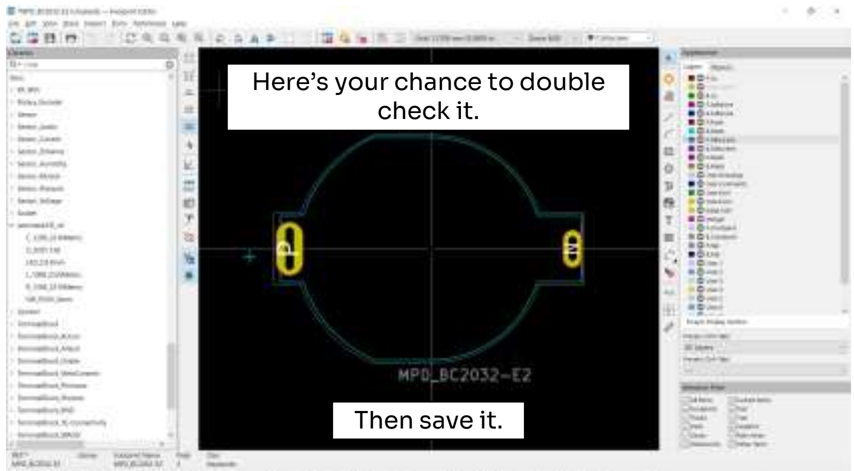
There will be five five files:

- a *.kicad_sym* symbol library
- a *.kicad_mod* footprint model library
- a *.step* 3D model
- a *.htm* link to a “How to import” webpage
- a license textfile

Footprint Library

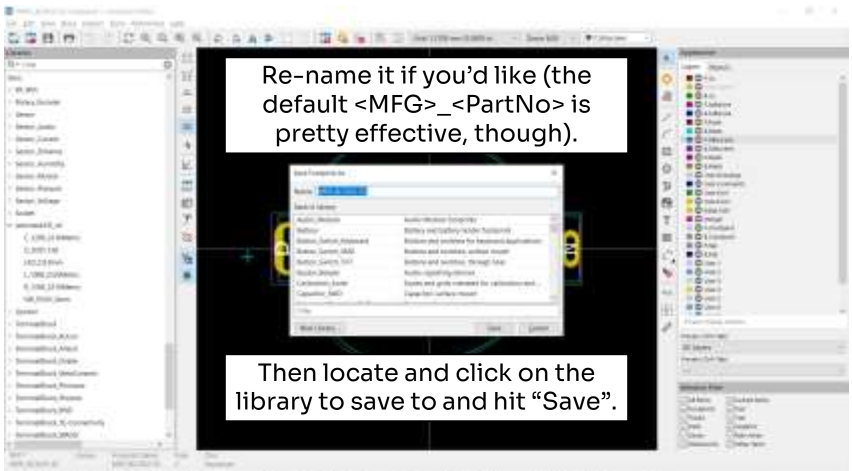


Footprint Library

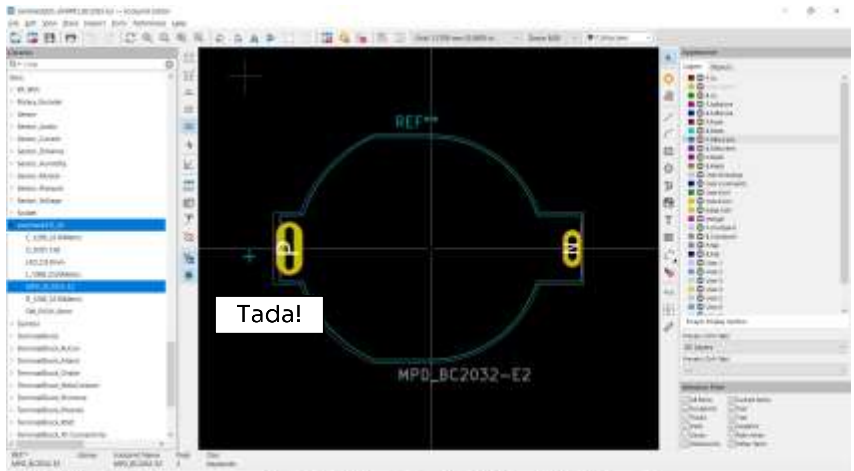


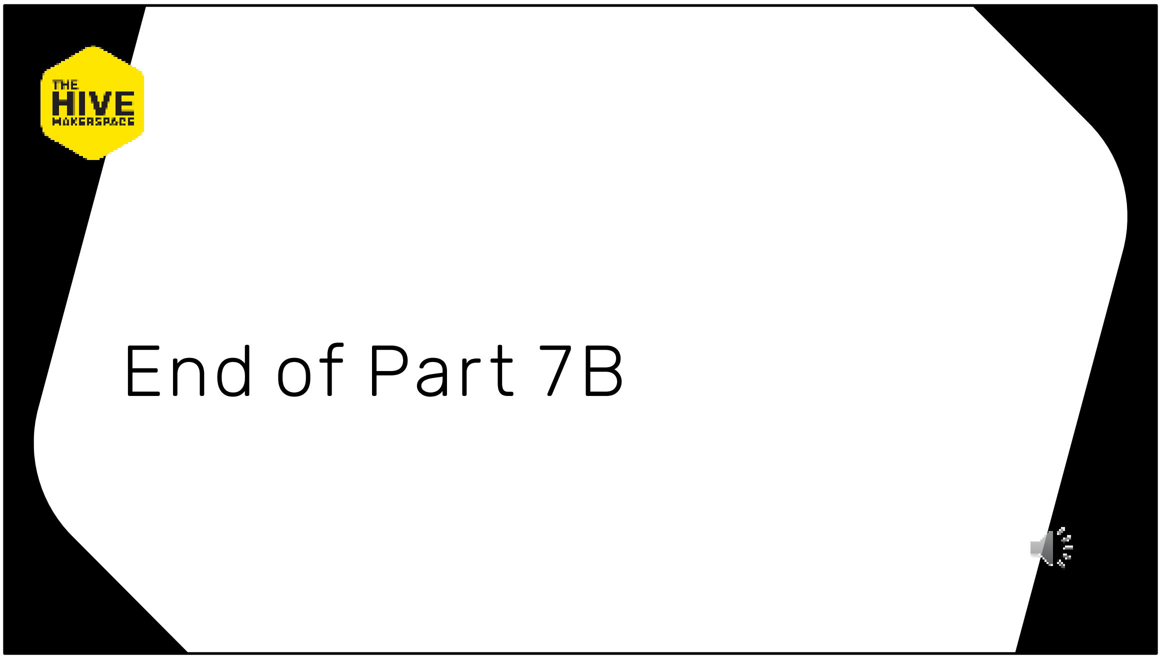
See how much nicer their looks? And notice the three outlines. Nice. They also intelligently put the positive indicator outside the part outline, meaning the user can see it after assembly.

Footprint Library



Footprint Library

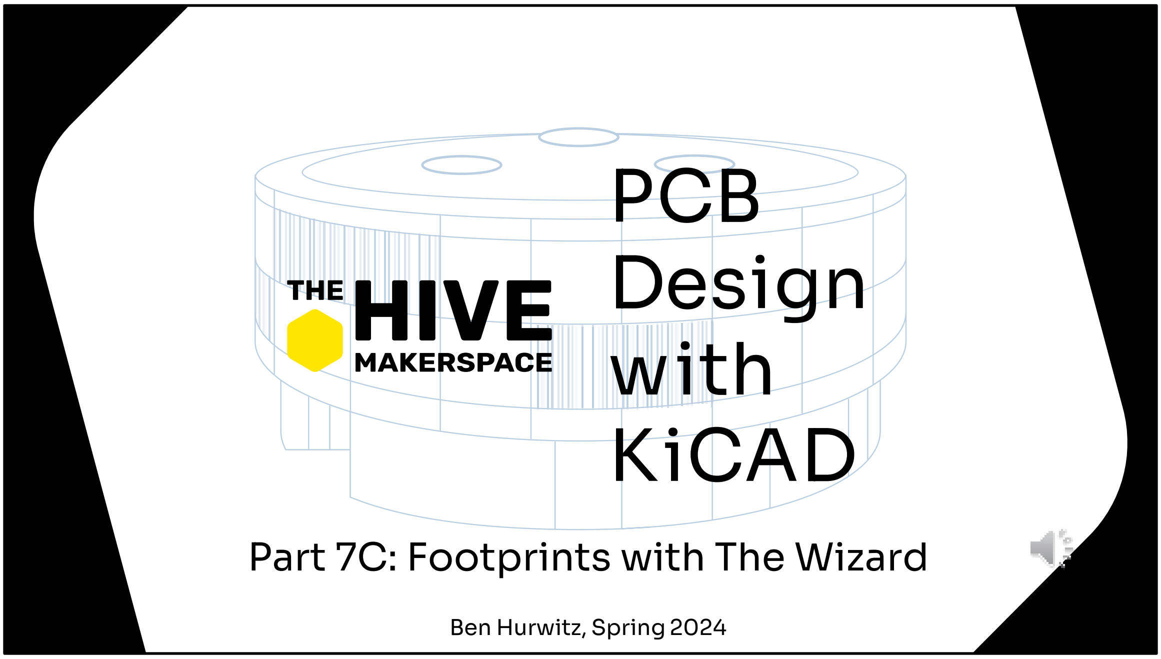




And that ends part 7B and our discussion of making footprints from scratch. As you've seen, that's not the best process, and it's really (really) error-prone if you're not super (super) careful, so it's highly advised to find the model online, or request it to be made by UltraLibrarian or SnapMagic. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In the next video, we'll cover using the footprint wizard, which can make generating standard or pseudo-standard footprints easier. Or more difficult. Guess we'll find out!

See you there!



Hi all, welcome to The Hive's series on PCB Design with KiCAD. My name is Ben, and in this series, we've been walking through the PCB design process using KiCAD as our electronics design software.

The previous videos went through the design process all the way through, resulting in a complete PCB ready for fabrication. One thing that I mentioned during that process, and was featured in the original "EDA Design Flow" in part 2, was library management, and the idea of using only project-scope libraries, but when we actually did the design, I ignored this for simplicity and time-constraints.

Part 6 went through a project-specific symbol library.

In parts 7A and B, we've made a custom library and a custom footprint using the "blank slate" method.

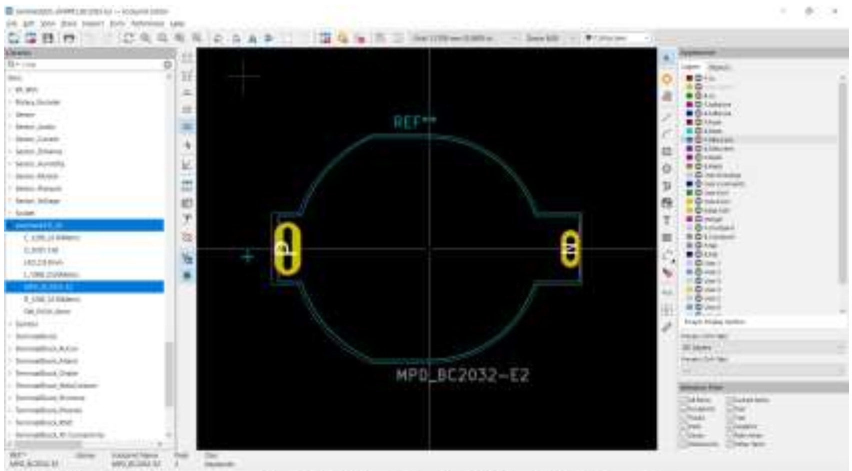
In this video, we'll use the wizard to make a footprint for our IC, then decide never to repeat that if possible.

This material is of course not required for a functional design, but it is good design practice, for KiCAD at least, to keep all your parts in a project-level library.

Because this is not related directly to the design flow of the previous videos, I'll make no assumptions about the state of your system or knowledge. So I apologize if some of this is repetition for some of you.

Let's get started.

Footprint Library



As a review, this is where we left the last video, with seven of the required eight footprints in our project-specific library. Obviously, this battery holder was downloaded, not made by me, so if you saved your personal copy rather than the downloaded one, first, kudos to you, but second, it'll obviously look different unless you spent your precious time making this, in which case, your employer will not be happy. But I digress.

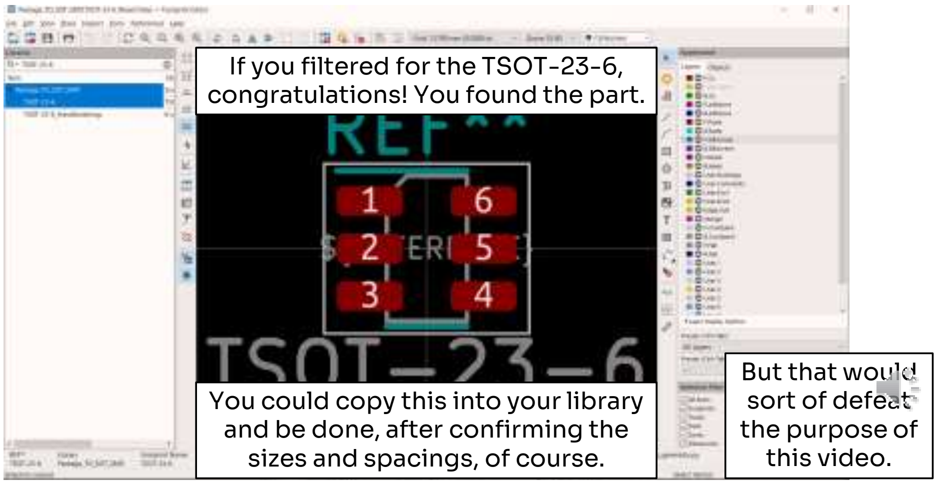
Footprint Library

Finally, onto the IC.

ICs are usually either a standard package with a known footprint, or based off a standard package, *but always confirm the supplier-provided package with the datasheet!*

KiCAD has a large selection of standard footprints that *should* work. So the first place to check for the footprint is in KiCAD itself with the filter box in the Footprint Editor. Filter for the part number and the package name.

Footprint Library



Footprint Library

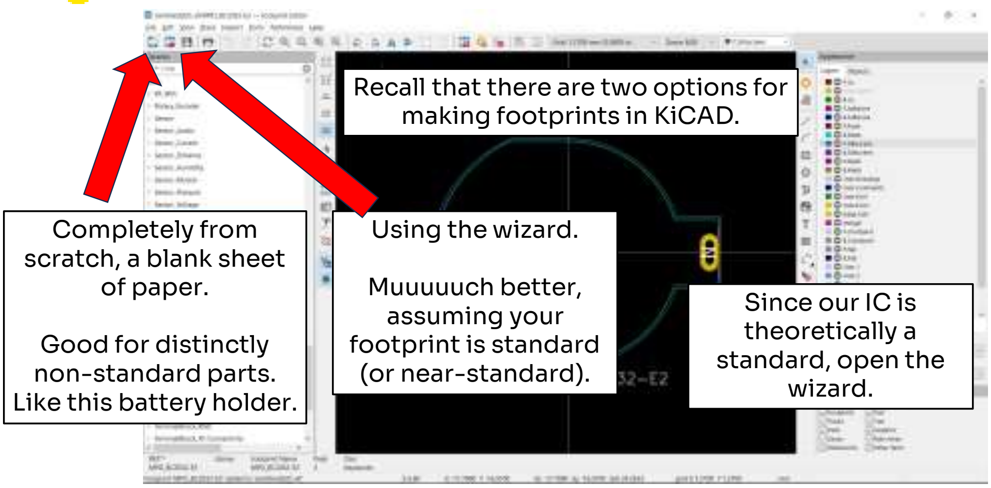
If we didn't find the *correct* package, we would next search online, either from your supplier, the manufacture, Ultra Librarian, or SnapMagic. Then it could be imported like we demonstrated with the battery holder.

If not, we can request it and wait.

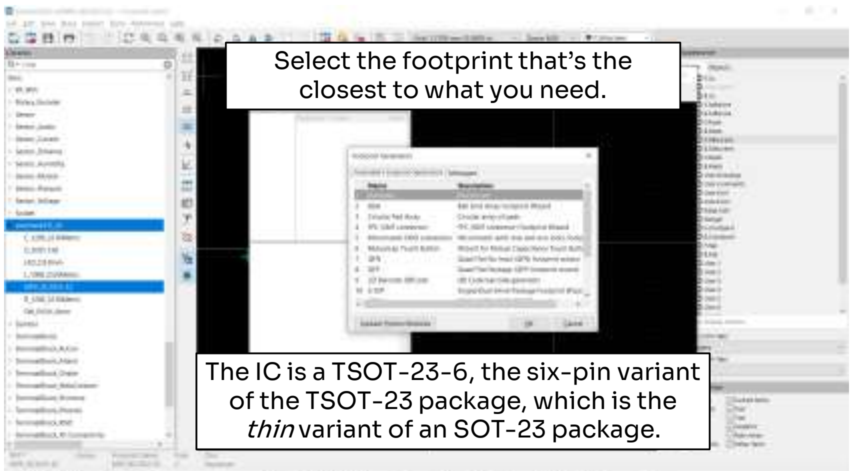
The last recourse is to make it, as we saw with the battery holder.

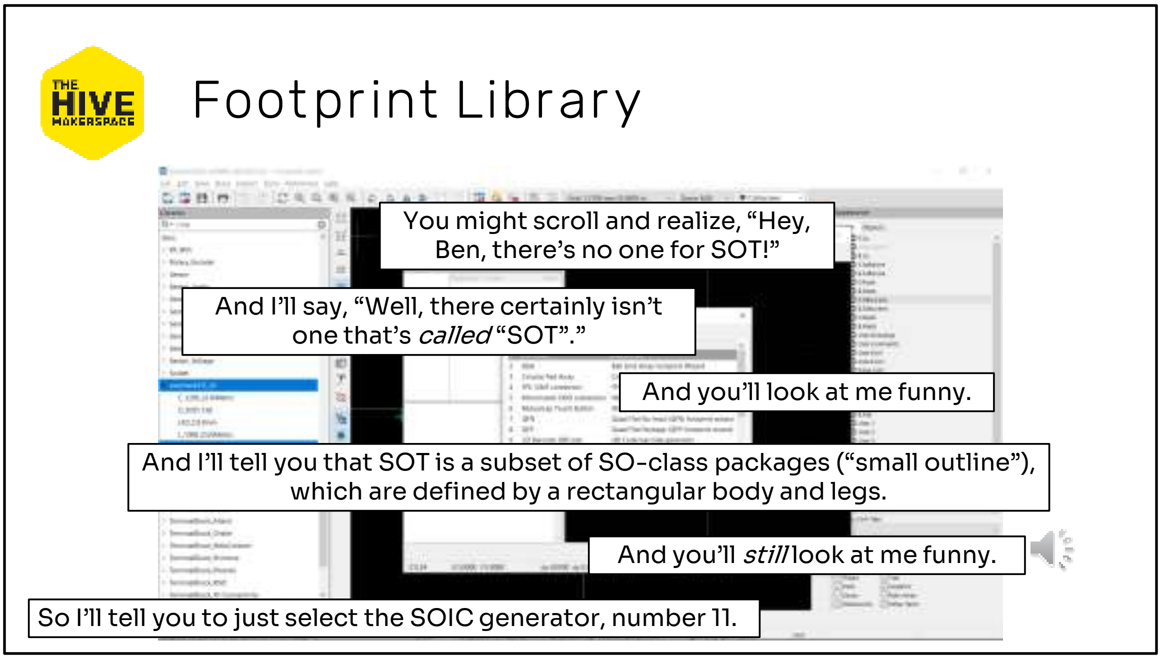
But because this is a tutorial about KiCAD, we'll do that.

Footprint Library



Footprint Library





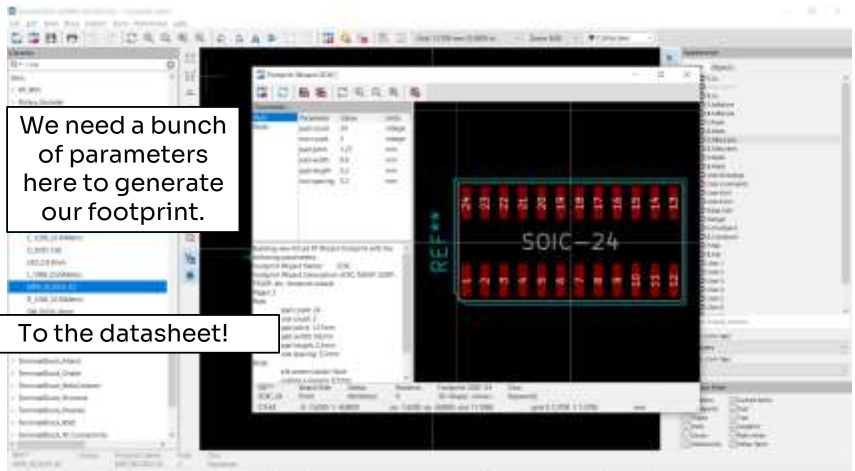
Because SOIC stands for “Small Outline Integrated Circuit” and is the overarching family that includes SOT package types, among others.

Aside: Package nomenclature

- Package nomenclature is *the worst*, and it's something you'll get familiar with as you design boards.
- There are so many standards, and the variation between some are so tiny. It can be really painful.
- The datasheet's mechanical drawings are the *one and only truth* for package sizes.
- Use caution!

This is a very (very) small subset of package families.

Footprint Library

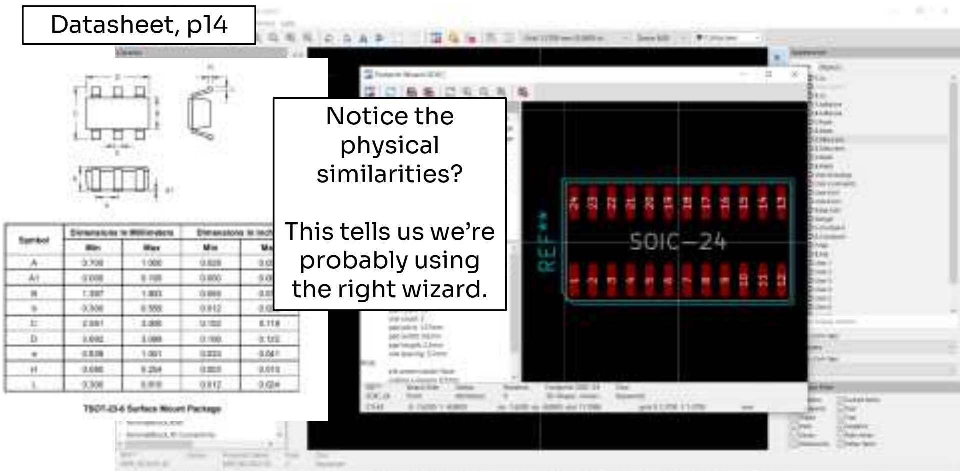


We need a bunch of parameters here to generate our footprint.

To the datasheet!

Footprint Library

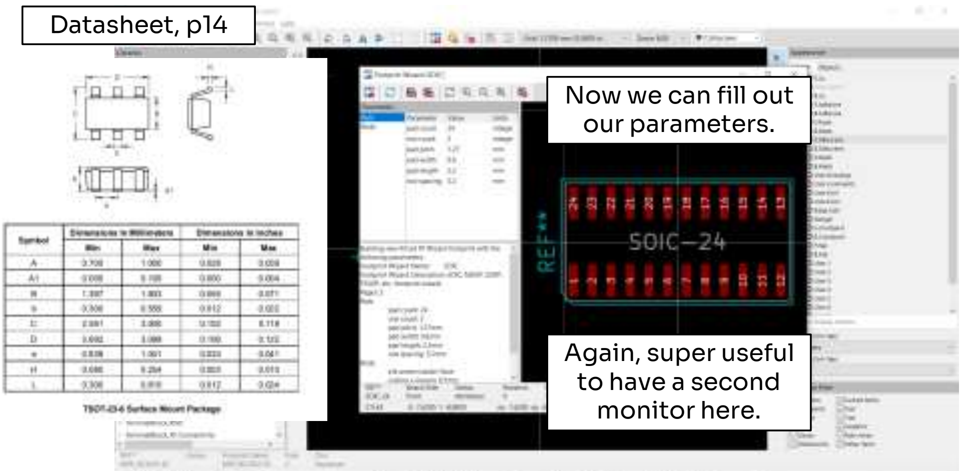
Datasheet, p14



Notice the physical similarities?

This tells us we're probably using the right wizard.

Footprint Library



So what the heck are all those parameters, anyway?

A – Package height

A1 – Seating plane height

B – Body width

b – Leg/pin width

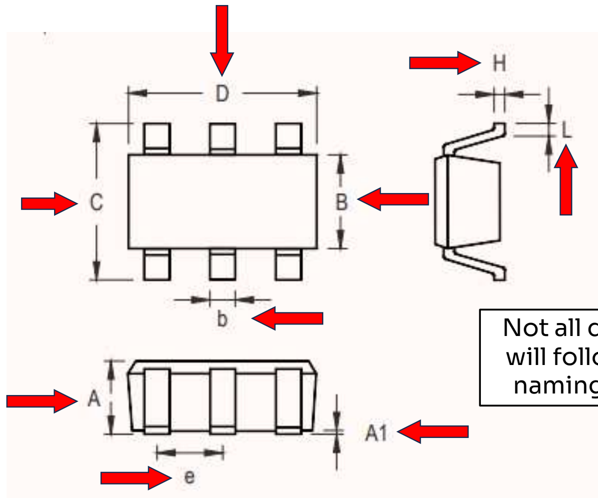
C – Package width

D – Package length

e – Leg/pin pitch

H – Leg/pin thickness

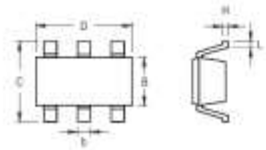
L – Leg/pin length



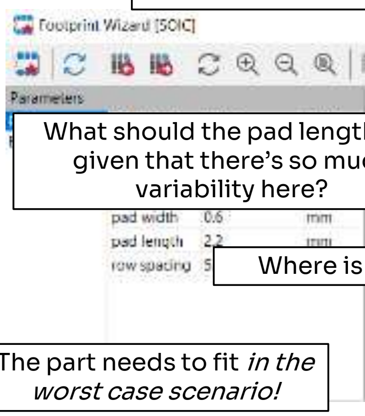
Not all datasheets will following this naming schema!

A few of thes requested parameters are readily available – pad count (6), row count (2), pad pitch (e), and pad width (b).

But there are a few tricky bits.



Should we be using min, max, or something else?



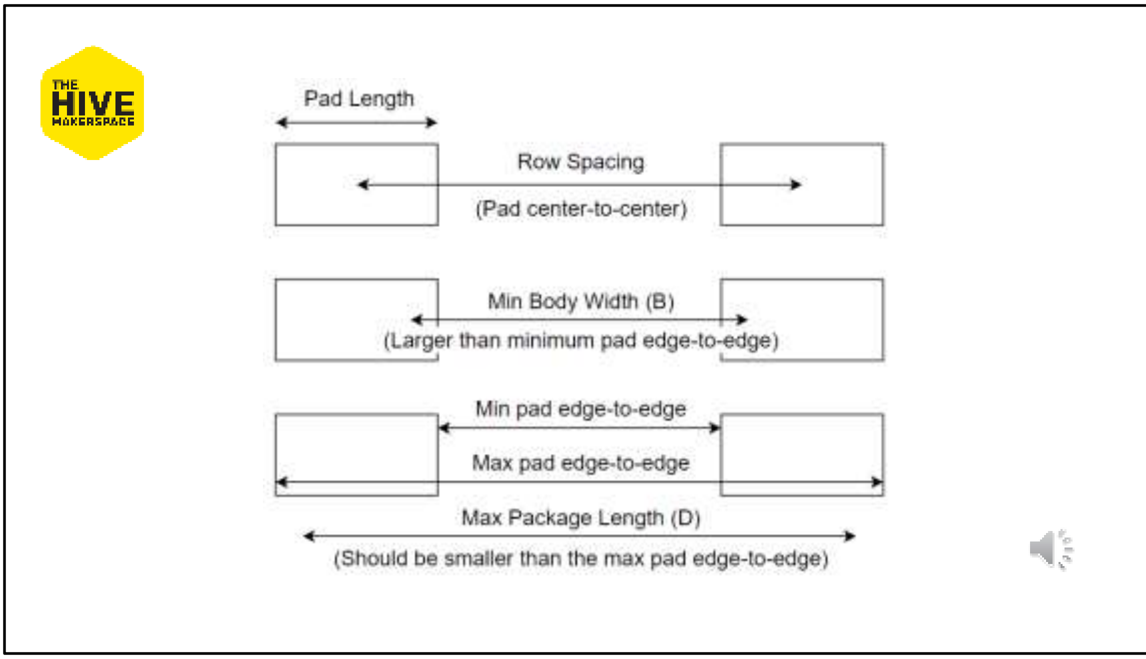
What should the pad length be, given that there's so much variability here?

Where is "row spacing" measured from?

Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min	Max	Min	Max
A	0.700	1.000	0.028	0.039
A1	0.000	0.100	0.000	0.004
B	1.397	1.803	0.055	0.071
b	0.300	0.559	0.012	0.022
[illegible]	[illegible]	[illegible]000	0.102	0.118
[illegible]	[illegible]	[illegible]099	0.106	0.122
e	0.838	1.041	0.033	0.041
H	0.080	0.254	0.003	0.010

The part needs to fit *in the worst case scenario!*

How much bigger should the pads be than the legs themselves?

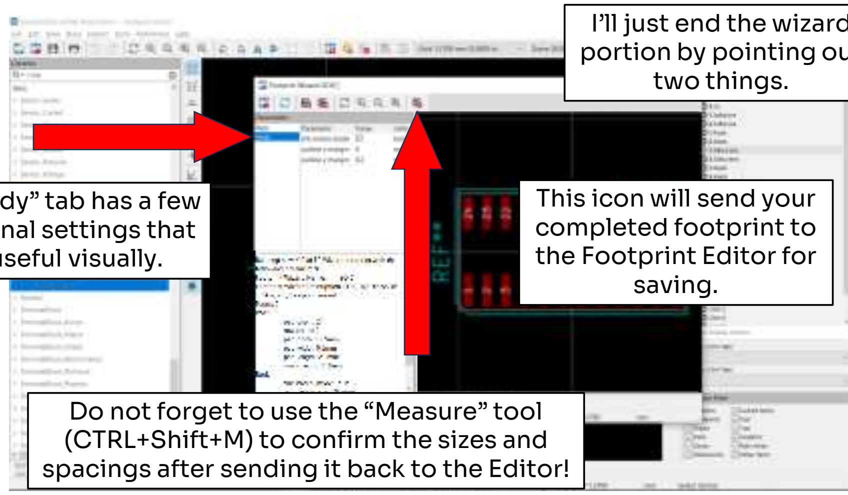


This is a mockup of what the footprint parameters are. The row-spacing is measured center-to-center, but the pad interior edge-to-edge is smaller than the minimum body width, and the maximum package length is shorter than the pad exterior edge-to-edge.

Still doesn't answer the key question of what the pad length should be.

- Some datasheets are better about giving useful dimensions, or even better, a recommended footprint (like the battery holder did).
- Standard packages that are *not* in KiCAD still have a standard footprint that you can search for and use.
- Otherwise, you'll have to make your best judgement.
- You might be getting a sense for why companies should do this for you.
- Given all this, we're not actually going to make this part. It's not worth our time.

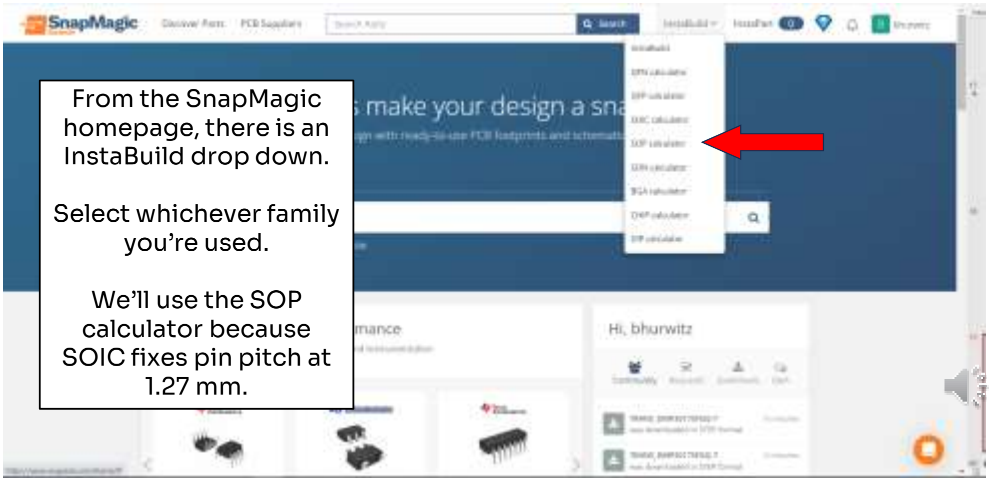
Footprint Library



SnapMagic Footprint Generator

- Okay, so now we know multiple methods to make a footprint:
  - Make it (from scratch or through wizard)
  - Request/locate it (Ultra Librarian, SnapMagic, manufacture, or supplier) and import it
  - Use the built-in version, if available
- The last method I'll show you as an option is the SnapMagic InstaBuild footprint generator

SnapMagic Footprint Generator

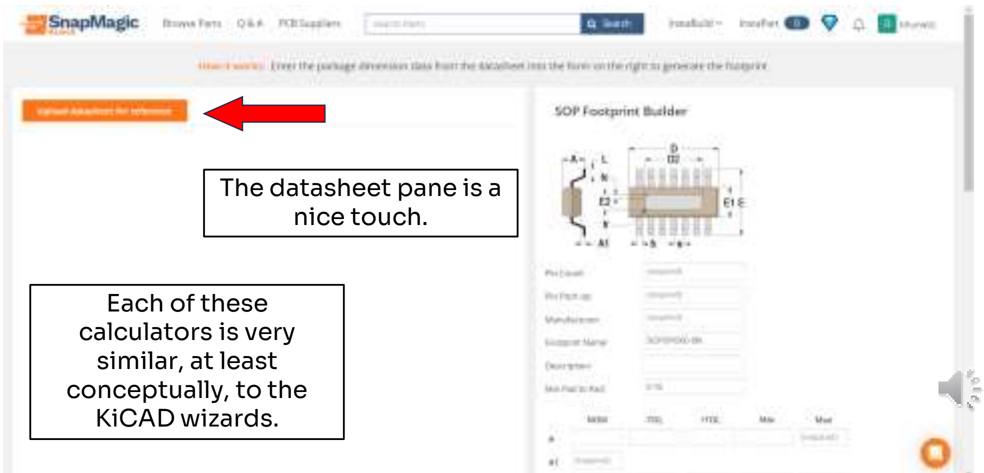


From the SnapMagic homepage, there is an InstaBuild drop down.

Select whichever family you're used.

We'll use the SOP calculator because SOIC fixes pin pitch at 1.27 mm.

SnapMagic Footprint Generator

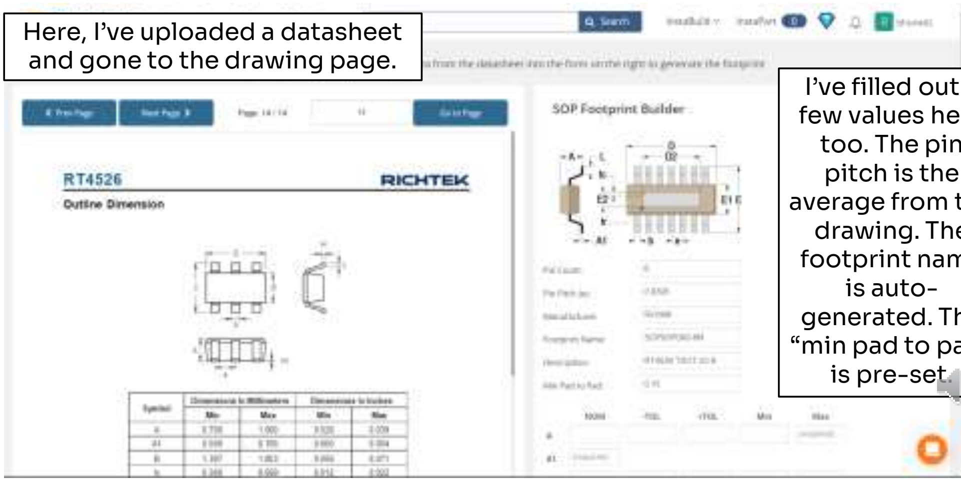


The datasheet pane is a nice touch.

Each of these calculators is very similar, at least conceptually, to the KiCAD wizards.

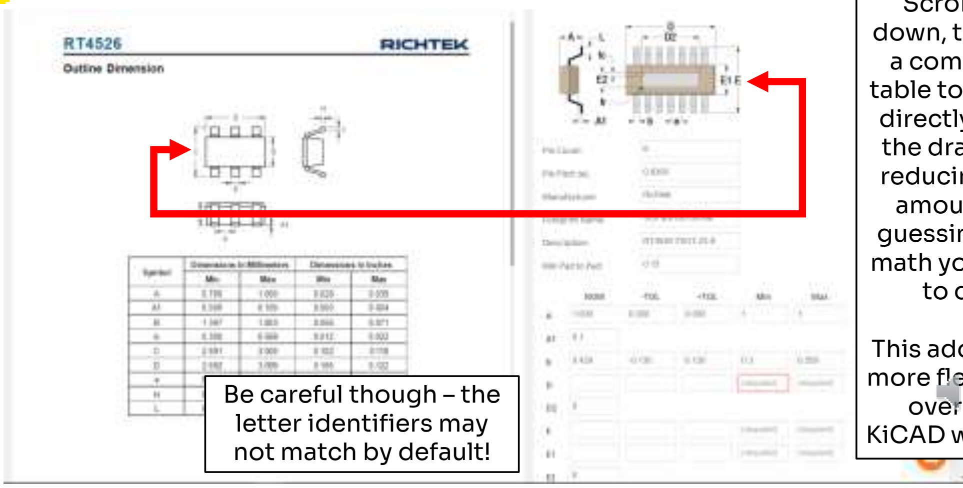
SnapMagic Footprint Generator

Here, I've uploaded a datasheet and gone to the drawing page.



I've filled out a few values here too. The pin pitch is the average from the drawing. The footprint name is auto-generated. The "min pad to pad" is pre-set.

SnapMagic Footprint Generator

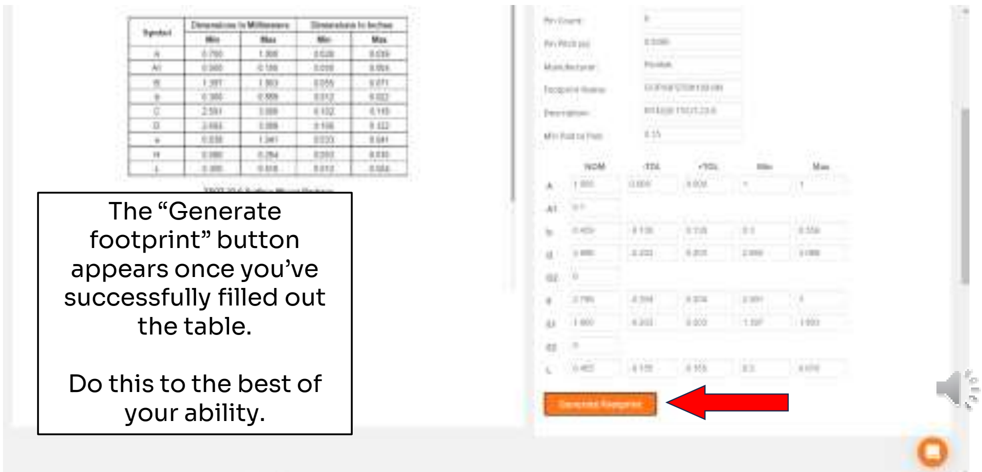


Be careful though – the letter identifiers may not match by default!

Scrolling down, there is a complete table to fill out directly from the drawing, reducing the amount of guessing and math you have to do.

This adds a lot more flexibility over the KiCAD wizards.

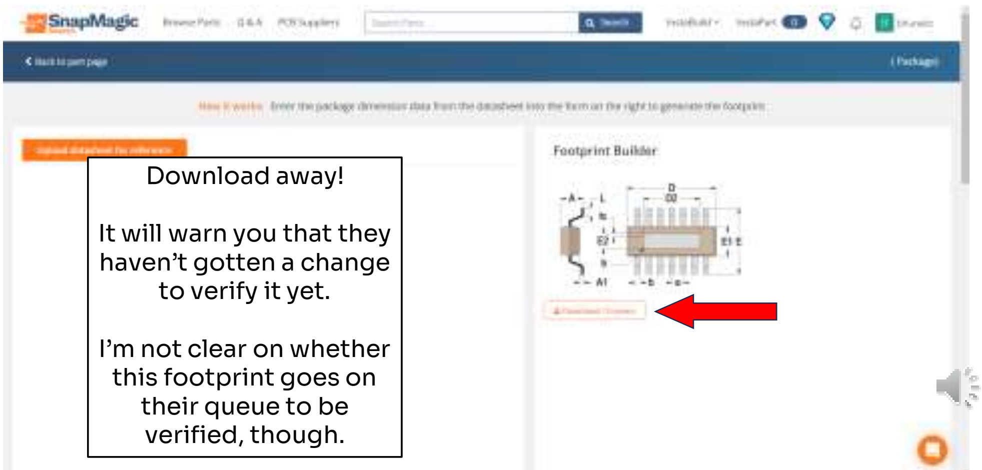
SnapMagic Footprint Generator



The "Generate footprint" button appears once you've successfully filled out the table.

Do this to the best of your ability.

SnapMagic Footprint Generator



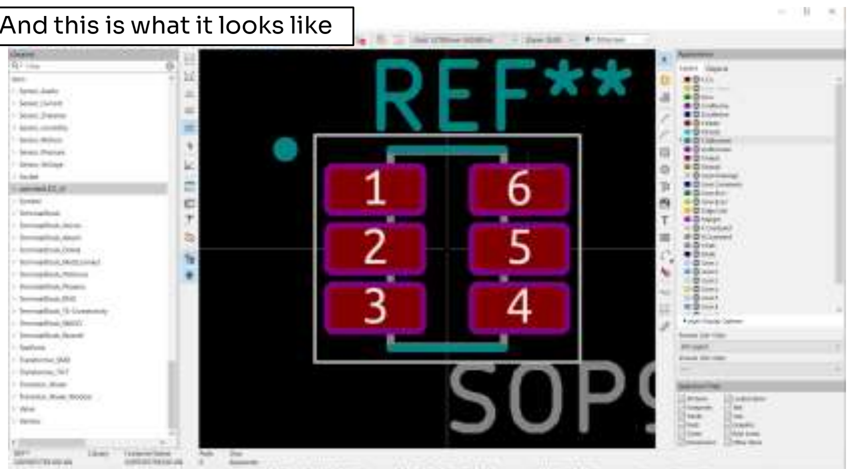
Download away!

It will warn you that they haven't gotten a change to verify it yet.

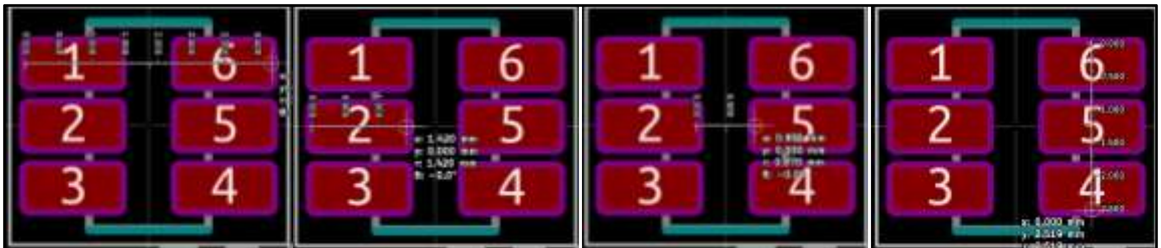
I'm not clear on whether this footprint goes on their queue to be verified, though.

SnapMagic Footprint Generator

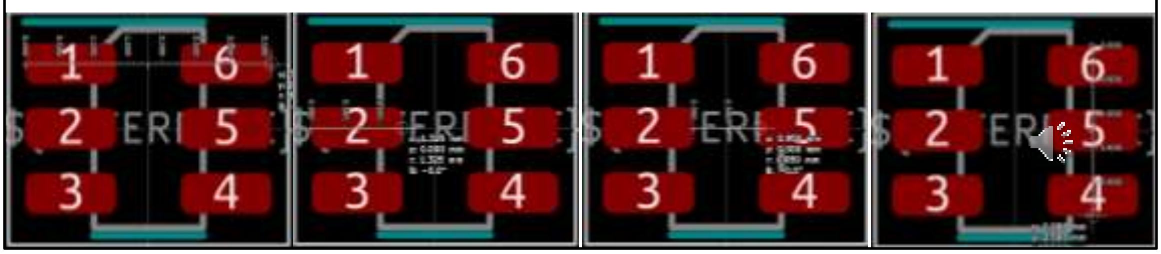
And this is what it looks like



The pink is the mask layer, which is used for defining when soldermask openings are on the board. The other parts have those polygons too, but they're just the same size as the copper pad, which takes visual priority.

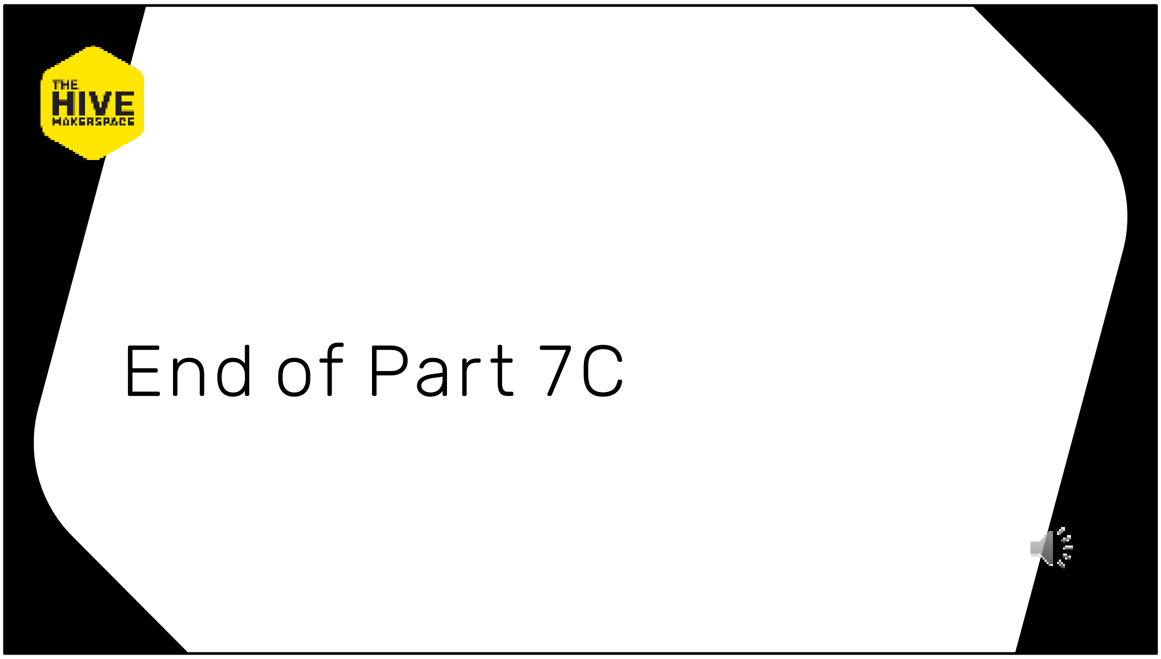


	Model Width	Pad Width	Int. Gap	Pad-Col Height
SnapMagic	3.71	1.42	0.87	2.519
KiCAD	3.6	1.325	0.95	2.5



Here's a comparison between some dimensions between the SnapMagic model, on top, and the KiCAD model, on the bottom. It's close, but they're not not quite the same. Will one not work? We'd have to put them on a board and print a 1-to-1 copy of the board to find out. But the differences are small, so hopefully.

Either way, this is a great example of how error-prone and variable the footprint generation process can be! Especially given the variation between datasheets. It's critical to confirm your parts fit before sending your designs off for fab.



And with that, we conclude part 7C, and, as of spring 2024, the end of the PCB Design with KiCAD tutorial series.

If you have further design questions and you're on the Georgia Tech campus, feel free to stop by The Hive during open hours, normally during the semester from 11-6. There is usually a PI, MPI, or staff member available to help you, even if they don't know KiCAD. Design questions transcend software choices.

If you're not on campus, the internet is your friend here. For KICAD-specific questions, there are probably hundred of tutorials on KiCAD, especially the basics we've covered in this series, the forums are quite active, and the documentation is pretty good (though not complete). For design-specific questions, well, you'll probably be doing a lot of trial and error. Make some small boards to demo the concepts you're working on, read a lot, and decide how much error is tolerable to your system.

Design is a never-ending topic, and there is always more to learn.

Thanks for watching, and good luck!