

**Design and Implementation  
of a Radio Frequency (RF)  
Printed Circuit Board (PCB)  
Using KiCad PCB Design Software**

Joe Cooney  
UC Davis  
EEC 134B  
6/10/2016

## Goal

This application note will seek to better inform users on how to use the PCB Design Software KiCad to create a RF PCB. The note will cover the basic process for creating schematics, footprints, board layouts, and channel traces. Also tips will be offered on certain areas where confusion or mistakes are likely to take place.

## Block Diagram

Before any work can begin in Kicad you must first be completely confident in your circuit block diagram and the components that will be used in said diagram. It is very important that the math in the block diagram adds up and that the circuit does not have any inherent errors. There are so many opportunities to make a mistake in PCB design that one simply can't afford any errors in the block diagram which the entire process will be based on. Simple errors not found at the beginning can be devastating when found right before printing, or worst case scenario, after printing.

### Radar Block Diagram

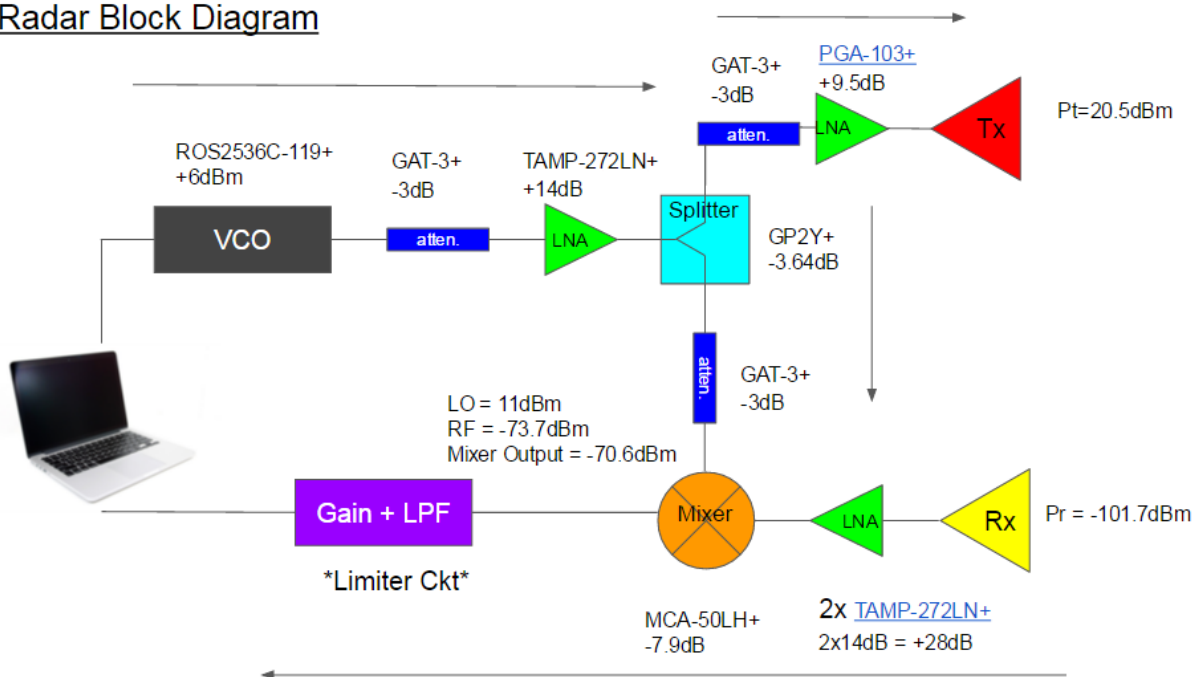


Figure #1: Block Diagram of RF PCB

Above is a good example of some of the key information the block diagram should contain, most importantly, the circuit component blocks, the model of each element, and the expected power at each junction in the circuit.

Once the block diagram is up to the team's standards, make sure you run it past someone who has experience with RF PCB circuit design and they can then double check the diagram for any more subtle errors. It is important that all of the major circuit elements have been selected and

the peripheral circuits that these components may require are fully understood. It is rare that one simply wires up single components for an RF circuit, for instance each amplifier will require some sort of biasing circuit in order to function as desired.

However, if you decide that you would like to minimize the amount of peripheral circuitry on the board, with careful part selection you may be able to find the majority of parts sold in a modular form factor where biasing circuitry is already contained within the IC. This was the method that our team decided to go with, and I am confident that it greatly reduced the time we spent finding errors while designing, and troubleshooting the finished board.

## Schematic

You can now begin to build your circuit schematic. Note that if desired you can first draw out the circuit with all of the components on paper or in some other circuit modeling simulator. However, since KiCad already supplies a circuit schematic editor, this was where we did all of the circuit layout. While many of the common circuit elements will be supplied (resistors, capacitors, supply pins), it is unlikely that they will have the exact schematic piece for each of your components. Not to worry though, it is quite easy to go into the **Schematic Library Editor** from the main KiCad menu and design your own. It doesn't need to be fancy, it just needs the appropriate number of input and output pins, good labeling of the pins, and good labeling of the component. Remember, you are likely going to be making many new components so it is important that they are labeled in an obvious and concise manner. Naming things like "Component 1", quickly gets out of hand and can be a massive headache once you have made multiple components. Remember to save the components in a new library and once back in the main editor you can search said library and select the component.

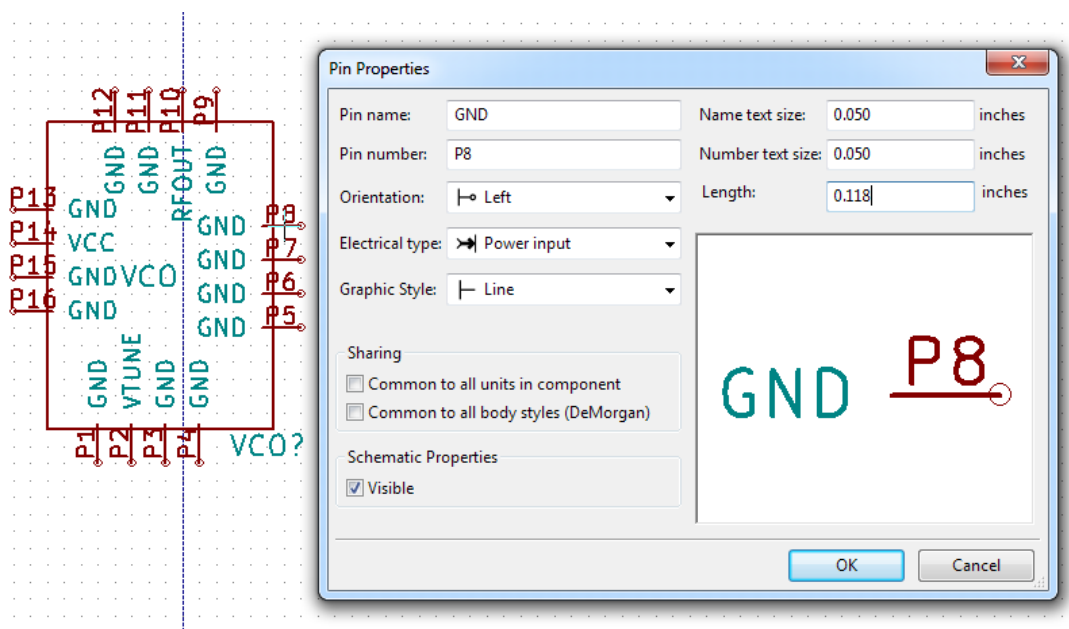
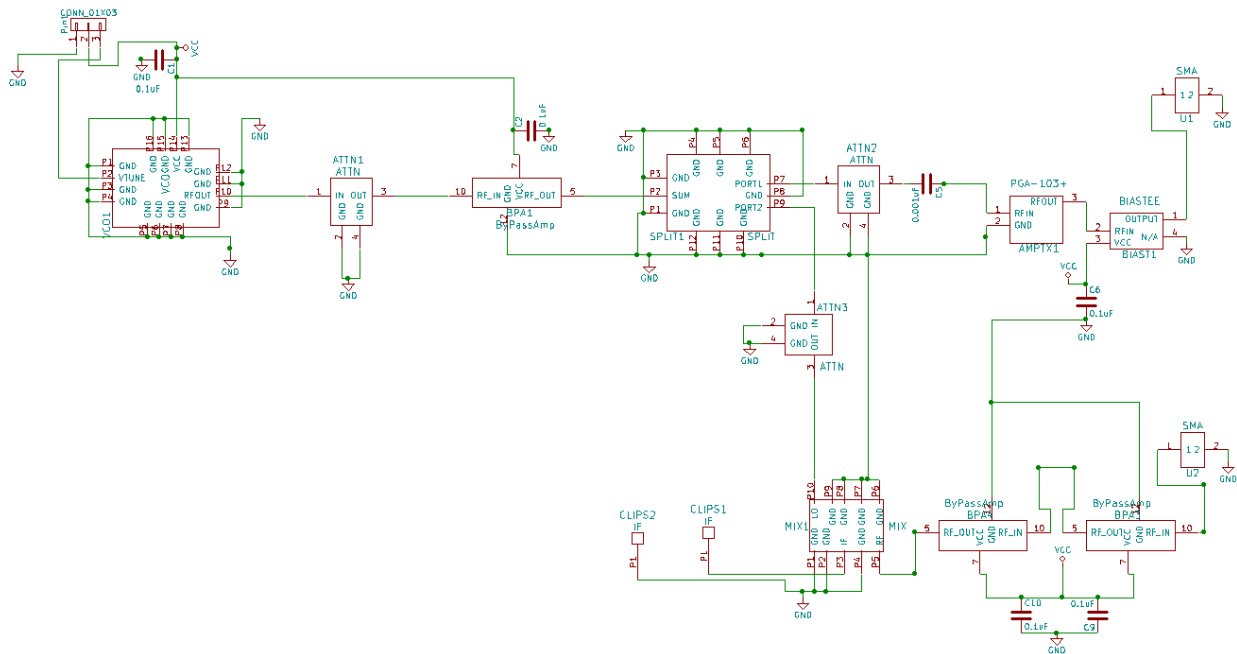


Figure #2: Creating Schematic Components

The above **Figure #2** shows how the process of creating a schematic component looks in the editor. Remember that the Pin name will contain the information of how that pin is to be wired whereas the pin number is important because this will be what allows the user to match pins with each schematics respective footprint. If the footprint and the pin numbers don't match, you will receive multiple errors once you attempt to load the schematic into the PCB editor.

Designating the Electrical type is also a good idea to keep the design consistent, however, it is not always necessary as the type does not actually matter once layout begins in the PCB editor.



**Figure #3: Typical Schematic Layout**

**Figure #3** shows an example of what a completed circuit schematic could look like.

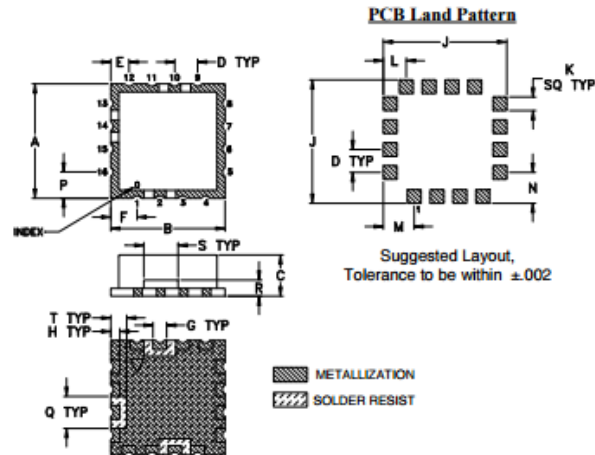
## Footprints

Once your schematic is complete it is a good time to start looking at the actual footprints you will be using on the board. A footprint contains the physical information that says where the pads go or where the holes are to be drilled, how large the actual IC, orientation information, etc. It will save you a lot of time if you can find the footprint you need online. That being said, you must be certain that the footprint you are downloading is in fact correct and matches the data sheet for your IC. There will likely be footprints you must make, so begin preparation by viewing the data sheet for your component. I have chosen to continue using the VCO (seen above in **Figure #2**) as an example component. This VCO's (the ROS-2536C-119 from Mini-Circuits) Datasheet can be found online in order to learn all of the physical layout information (**Figure #4**) needed for the PCB footprint.

### Pin Connections

RF OUT	10
VCC	14
V-TUNE	2
GROUND	1,3,4,5,6,7,8,9,11,12,13,15,16

### Outline Drawing



### Outline Dimensions (inch mm)

A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S	T	wt.
.500	.500	.220	.100	.080	.115	.060	.040	.540	.060	.100	.135	.135	.115	.140	.070	.150	.070	grams
12.70	12.70	5.59	2.54	2.03	2.92	1.52	1.02	13.72	1.52	2.54	3.43	3.43	2.92	3.56	1.78	3.81	1.78	1.2

Figure #4: Component Layout (ROS-2536C-119)

Remember to keep your units consistent. Datasheets often contain information in both metric (mm) and standard (inches/mils). Don't ever confuse mils with mm, they are completely different units with one mil being a thousandth of an inch, and one mm being a thousandth of a meter. We often used mils in our design since some of the specs were given to us in inches (i.e. make sure your board is less than 2x2 inches). The grid can be set on the far left of the PCB editor and the footprint editor. It is also very helpful (and this goes for all of KiCad) if you become familiar with hot keys that are available. In particular the space bar is very handy in the footprint editor as it sets the current location of the mouse to the (0,0) location of a graph. By then moving your mouse to a secondary location you can roughly measure the distance between the two spots. However it is a good idea to mostly rely on placing things in the appropriate position by changing the grid to fit your placement needs. For instance if you need two pads to be 100 mils apart, it is easiest to change the grid size in the upper left corner to 100 mils so that the pads will "snap" into place. It is imperative that you name your pads using the same convention that is in the datasheet and in your schematic version of the component. Notice how each pin in **Figure #5** (P1-P16) name matches with both the datasheet and with the schematic in **Figure #2**

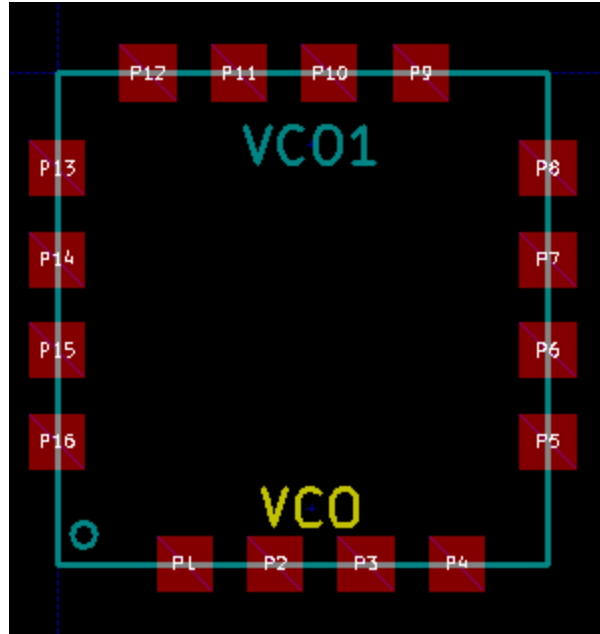


Figure #5: PCB Footprint (ROS-2536C-119)

Figure #5 shows the final layout for the VCO. Note: when creating footprints it is very nice to work with a partner. One person can read off of the datasheet while the other person enters the information into KiCad. With two people present a lot less errors occur, as everything can be double checked by the person with the datasheet.

### Netlist

Once all PCB footprints are done it's time to start actually laying out the board. Now you must create a netlist in the schematic editor and match up each schematic with its respective footprint that you have either made, or downloaded.

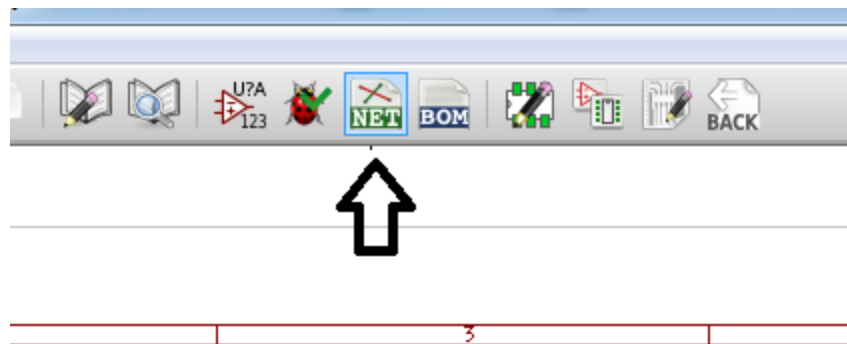
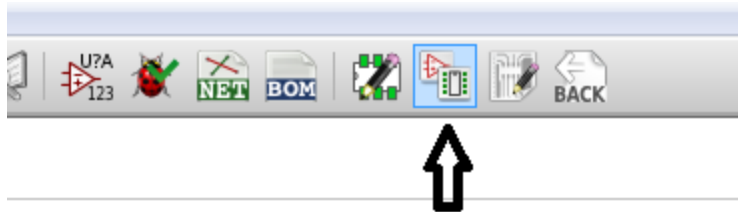


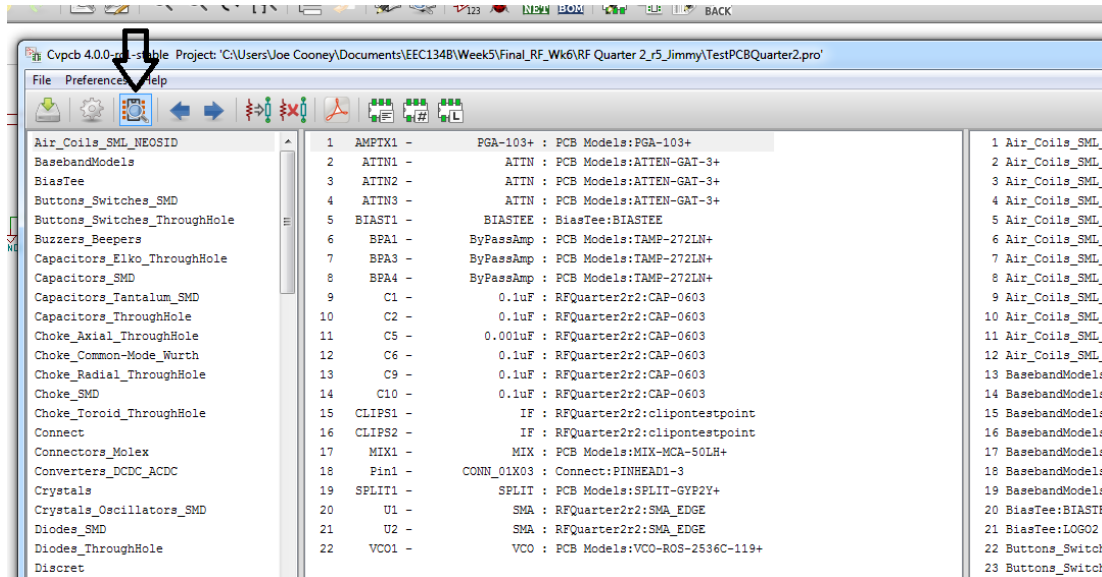
Figure #6: Generate Netlist

Start by hitting the “NET” button in the schematic editor (**Figure #6**). In the next window simply hit generate and make sure to save your file with a distinct name as you will need to access in order to begin using the PCB editor.



**Figure #7: Run CvPCB**

Next hit Run CvPCB (**Figure #7** also found in the schematic editor toolbar) in order to match components and footprints.



**Figure #8: Match Components to Footprints**

Here you can match each component to its footprint by selecting libraries in the left column and individual components in those libraries on the right. **Figure #8** has already had each component matched, but normally the right side in the middle column would be empty (until footprints are selected). If you are ever uncertain of the footprint you can always click the “View selected footprint) in the upper left corner to verify that you are selecting the appropriate footprint. Once completed matching components, be sure to save the netlist.

## PCB Editor

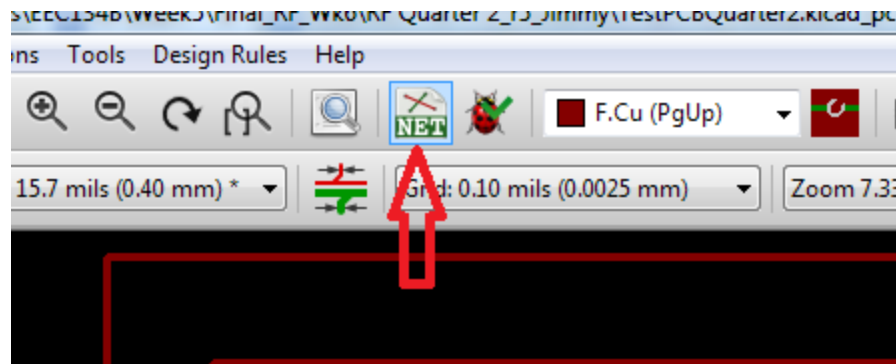


Figure #9: Read Netlist

Now the layout of the board can begin in the PCB Editor. Import the “Rat’s Nest” by clicking on the “Read Netlist” option in the toolbar. Make sure you select the netlist that you have been using and hit “Read Current Netlist” to import. You should now see a large pile of footprints with all of their interconnects (white lines).

Try to separate everything out using the hotkey “m” (for move). I highly recommend creating your edge cuts first (found on the far right of the editor). These lines will dictate the outer boundaries of your board and allow you to try and create the board in the space within. If you find yourself not being able to fit the components in the allotted space you can always easily redraw your edge cuts, but it’s a good idea to try and make the board as small as possible without suffering from performance issues. There are many sources available for how to lay out a RF board so I will not go into great detail on every aspect. I will later discuss some of the things I encountered while laying out RF boards as well as a few helpful hints.

## PCB Calculator

One important tip is to make sure that your RF trace is the appropriate width in order to achieve 50 Ohm impedance matching. This can be figured out by using the KiCad supplied “PCB Calculator”.



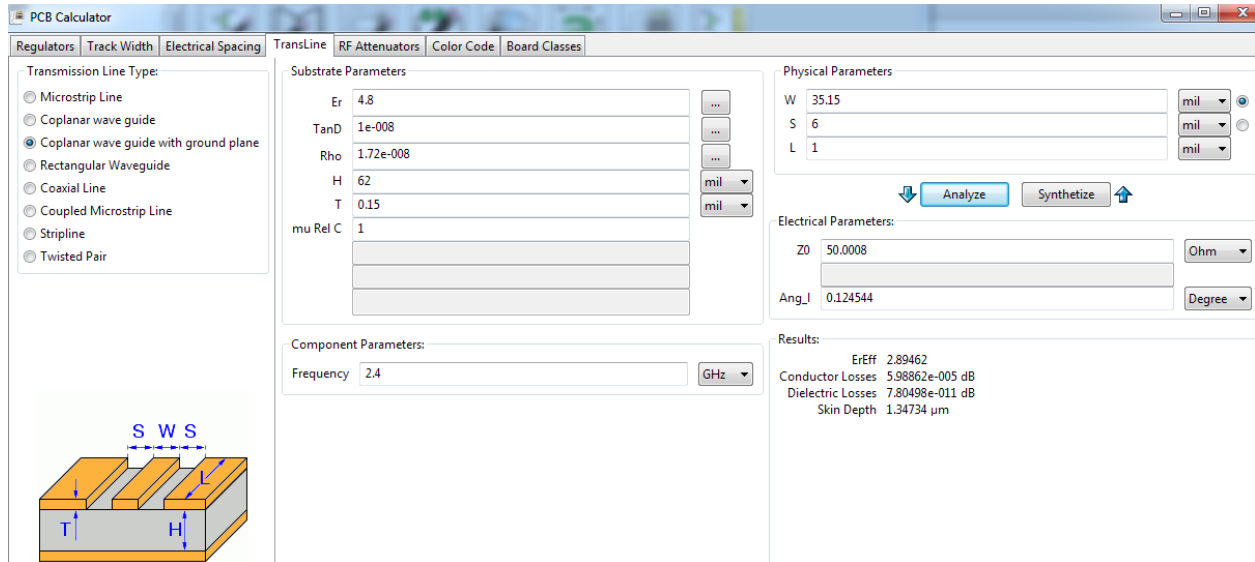


Figure #10: PCB Calculator

In our case we were using a coplanar waveguide with ground plane, with dielectric of 4.8, height of 62 mils and copper ground plane thickness of 0.15 mils, as well as set the frequency parameter to 2.4 GHz (our operating frequency). We choose a small spacing ( $S = 6$  mil) to allow us to use a more narrow transmission line ( $W$  of 35.15 mil). We did not worry too much about angle (or length), but just used the same transmission line thickness everywhere on the board. Be sure you understand the capabilities of the manufacture and what kind of material they will be using. Manufactures will have minimum width traces/gaps that they can print, so look into the company specifications before you settle on your trace dimensions.

Remember that when you go to do a ground fill on the the board that you should select the spacing to be whatever the spacing you have chosen here. Otherwise your trace impedance will not be 50 Ohm, a matching that most components depend on to function as expected.

### Set Track Width

It should be stated that the calculator function is **only** a calculator and does not affect anything on it's own within the editor. It is the user's job to take the numbers from the calculator and implement them within the editor. A new trace width can be added by selecting the "Design Rules" tab in the PCB editor, and then selecting the "Global Design Rules" tab. In the bottom right you can then enter the desired track width in inches (remember 1 mil is 0.001 inches).

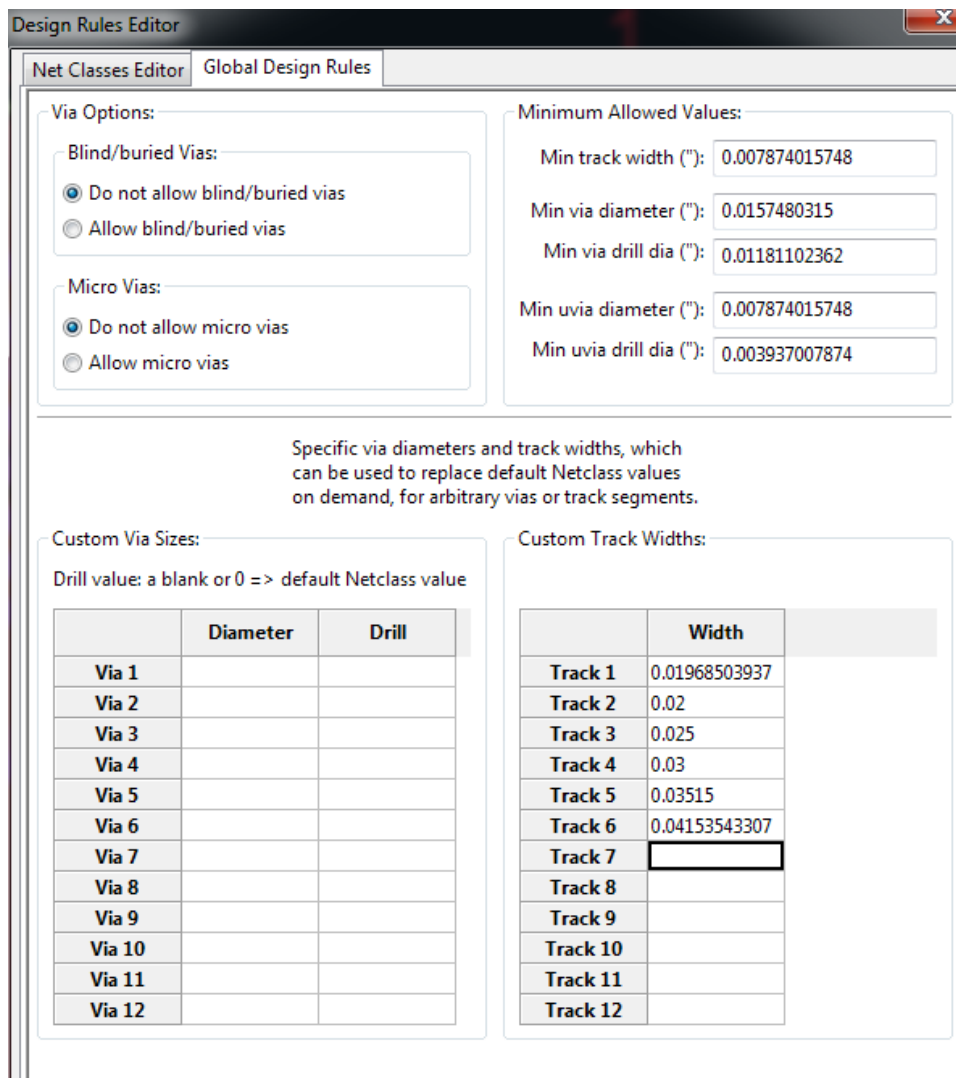


Figure #11: Custom Track Widths

### **Ground Fill**

Another important thing to do is creating a solid ground plane on your PCB. The ground plane function is to isolate signals on the board from each other and other electronics in the vicinity. The ground plane can be selected on the right of the PCB editor.

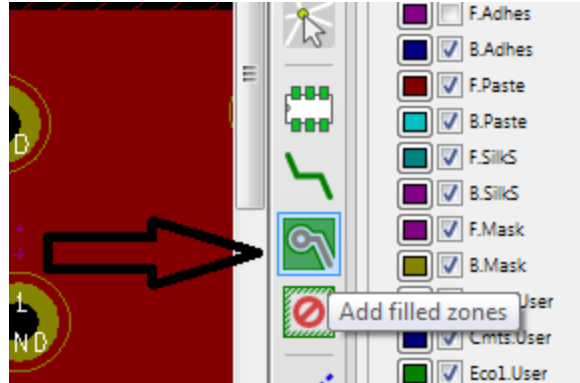


Figure #12: Ground Fill

Much like the edge cuts you can select an area that you would like to fill as a ground plane (ideally the entire front and back of the board). Remember to set the appropriate spacing determined from the calculator and then also to select the “Solid Fill” option. It is by default set to “Thermal Relief” which does not automatically connect grounded pads or vias to the ground plane. **Once your ground plane is in place you can always refresh or update the ground plane by hitting “Ctrl-B”.**

Last but not least, make sure that your top and bottom ground planes are sufficiently tied together. This is done by using multiple vias. It is most critical to have a strong ground plane directly next to the transmission lines, but it can also help to scatter the vias throughout the board as can be seen in later figures.

### Design Verification

Once you feel confident in your board design be sure to run the “Design Rules Check”. This will insure that you have made a board that meets minimum specifications (which can be determined by looking at the manufacturer's design specifications). It will also pick up on any unattached or open traces, or pads/holes that have not been connected correctly.

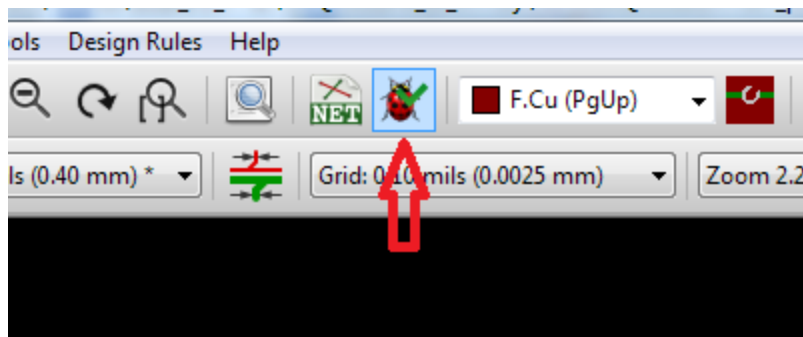


Figure #13: Design Rules Check

### Plot/Gerber Files

If it passes the Design Rules Check you can then create a zip folder containing the important layers that make up all the information needed to print your board. This can be done by selecting the “Plot” option underneath the “File Tab”. Make sure to maintain the “Gerber” format and then to select the critical layers: F/B Cu (The copper layers), F/B Silk (component/source identifier text),

F/B mask (protective layer), and Edge.Cuts. Also before hitting "Plot" first press the "Generate Drill File" button, as this is what will contain any drilling information (vias or thru hole). Finally hit Plot and make sure that all of the layers and drill file are contained in the folder you have selected. You can then .zip this file and upload it online to the companies DFM (Design For Manufacturability) Test. If the design passes, that same .zip folder can be sent to the manufacture to begin printing the board.

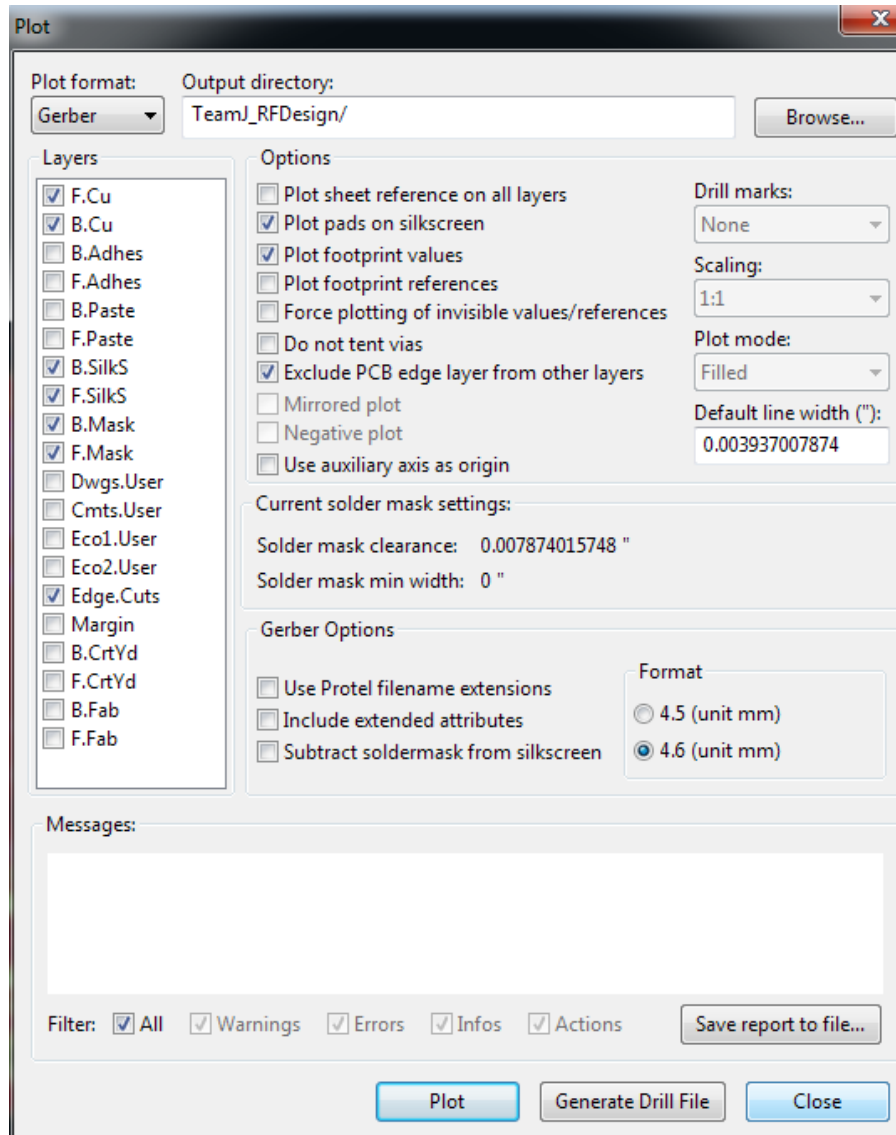


Figure #14: Plot/Gerber, and Drill Files

**Strong Warning:**

Just because the board passes a DFM test **does not mean** the board is without errors. Often the errors are not because there is a faulty connection or a misprinted copper layer. The errors will almost always come from a user design flaw. This is why it is absolutely critical that your board is reviewed by at minimum two other people. Every time I had someone else take a look at my designs, they were able to see the flaws that I could not. Added perspective is imperative,

it is very rare (especially for beginners) to have a board go without additional review and be problem free. Even with external review, problems will arise, but they can be greatly minimized by putting in extra hours beforehand meticulously scouring the board for errors.

### Case Study: Designing an RF board for a Radar System

The following case study highlights some of the issues that I have personally faced while laying out a circuit in the PCB editor. Hopefully some of these issues will spare future designers from making similar mistakes.

This board in **Figure #15** was the first attempt at the PCB layout. When I first designed it I had the goal of having no bends in my transmission lines. This was mostly just out of a desire to keep the transmission line as close to something ideal as possible. With no bends or corners there is less to worry about with unintended parasitics from reflections and/or acid traps. For this reason the board ended up being extremely long, and as can be seen in **Figure #15**, wasting a lot of space. There is also some glaring design flaws aside from just the wasted space. Having the Tx and Rx so close to each other (on the far right of the board), and in parallel, increases the risk of coupling between the Rx and Tx. If we were say picking up only the Tx signal at our Rx this would make the entire board worthless as there would be no way to see a difference between our output and input signals. Also near the top right of the board there is a VCC line that runs underneath the Tx transmission line. This is also an undesirable feature as it too can cause interference with our RF trace. Finally at the bottom of the board there is another VCC trace that is dangerously close to the edge of the board. While this may clear a DFM report, it is very likely that the trace could be damaged when the edges of the board are cut, which would jeopardize the entire board as neither of the LNA's at the Rx would function (or possibly even the entire VCC rail could short out).

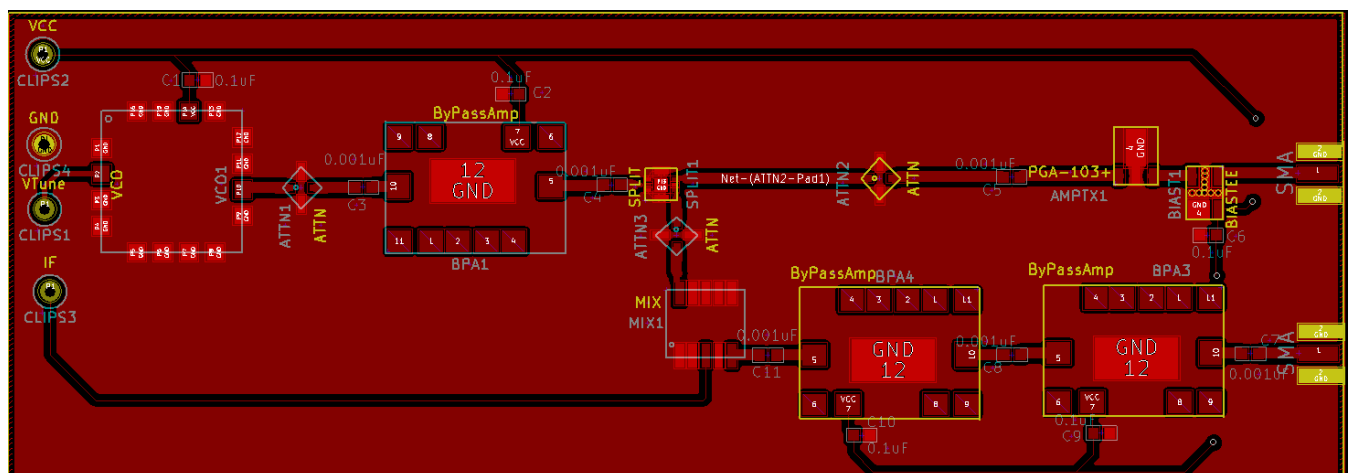


Figure #15: Initial PCB Layout

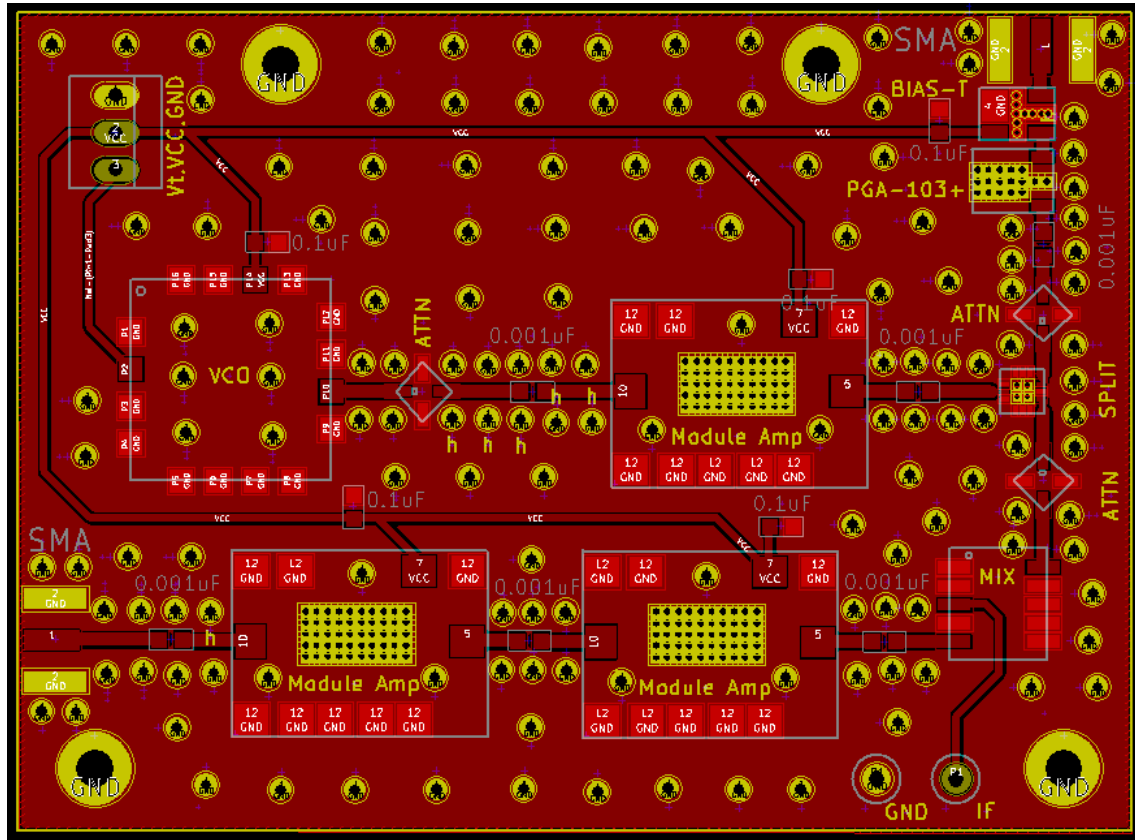


Figure #16: Revised PCB Layout

Our revised board is far superior to the original in many ways. It can be seen now that coupling between the input and output traces is very unlikely as they are now on opposite edges of the boards and perpendicular to each other. There is much less wasted space on the new design, while still maintaining straight transmission lines. Also in this board the VCC supply never crosses directly underneath a transmission line. This board also got the addition of multiple via's between the front and back ground planes to better isolate the traces and reduce coupling. Also there can be seen four larger via's which are for physically mounting the board using small rubber inserts with a sticky base. This allowed us to mount the board onto the plastic container used to house all of the circuitry for our radar. This board was eventually sent away to be printed. But even the above board had a serious design flaw that we would have to remedied.



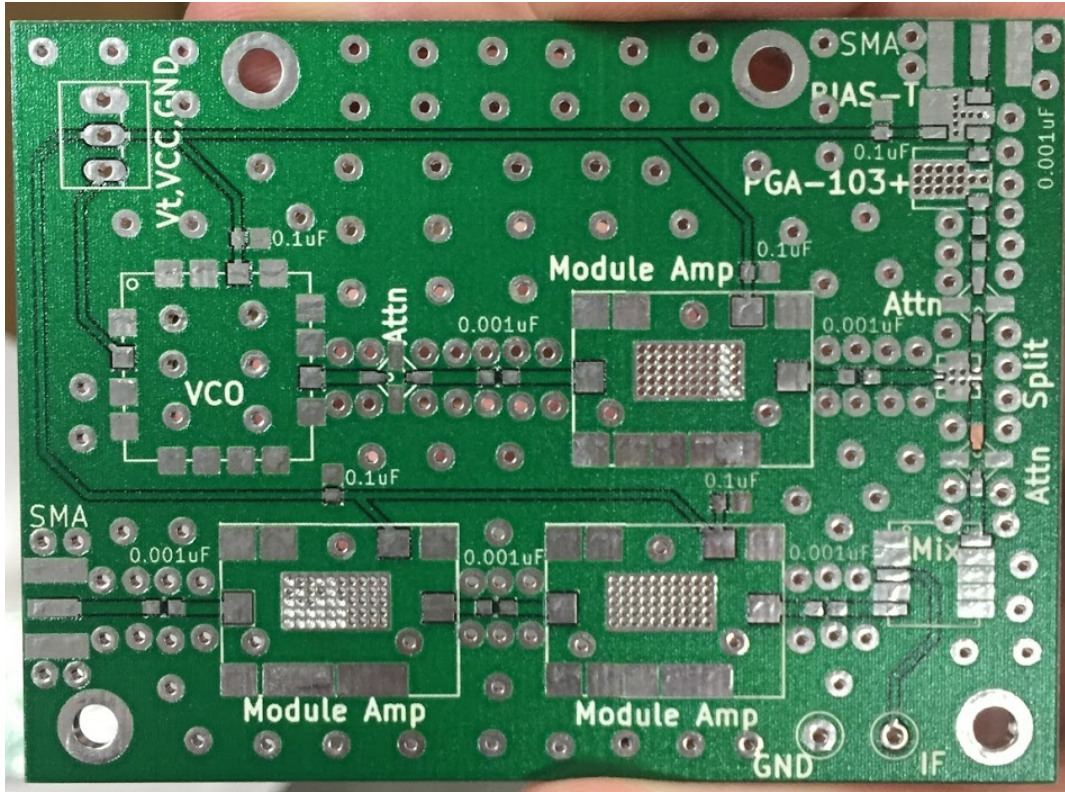


Figure #17: PCB Printing #1 (Board with Bias-T error)

In **Figure #17** we can see the printed board returned from the manufacture. The issue with this board can be found in the upper right corner where it says “BIAS-T”.

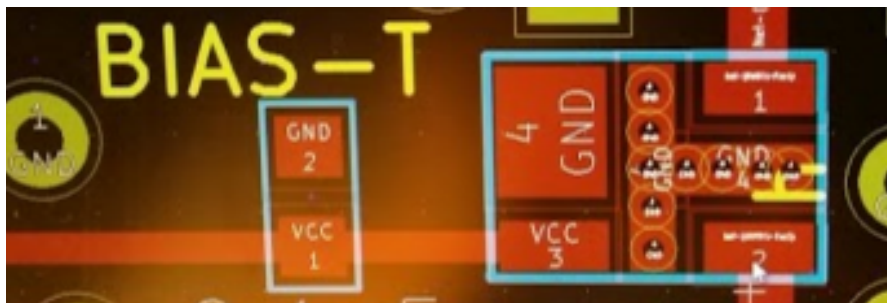
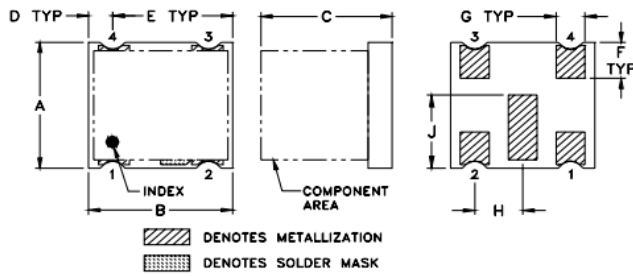


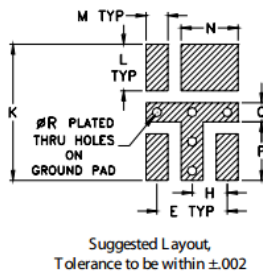
Figure #18: Close up of Bias-T error

The problem with the board is that we had looked at an inverted image of the Bias-T we were using (Note: this Bias-T is a bias circuit for a power amplifier). Instead of looking at the upper left diagram in the Outline Drawing (in the datasheet, **Figure #19**) I had looked at the top right diagram which is in fact a mirror image of the top down view. So instead of having the NOT USED pin tied to ground, it was now tied to VCC, and the input and output pins were swapped.

## Outline Drawing



## PCB Land Pattern



## Pad Terminations

RF	2
RF&DC	1
DC	3
NOT USED	4

Figure #19: Bias-T Layout from Data Sheet

It was tough to know how to proceed at this point, since we were getting very close to the testing day. We only had enough components to make two full boards and we knew that no matter how we rigged the board together it could not perform at the desired level. One option could have been to completely disregard the Bias-T and just place a 0 Ohm resistor that bridged the transmission lines together. Another option was to use a “dead bug” method, where we could place the IC upside down and used wires to tie the pads to their respective spots on the board. However, since we only had enough components for two boards, we opted to wait and not begin testing our board. This was mostly due to the difficulty of getting more parts, since all of our components were samples either obtained from mini-circuits website (where the number of samples of the same item is limited per month), or from a vendor of the company who came to UC Davis once. Since the supply was uncertain, we immediately fixed the error in KiCad, sent out for new boards and then proceeded to work on other aspects of our project until the boards returned.



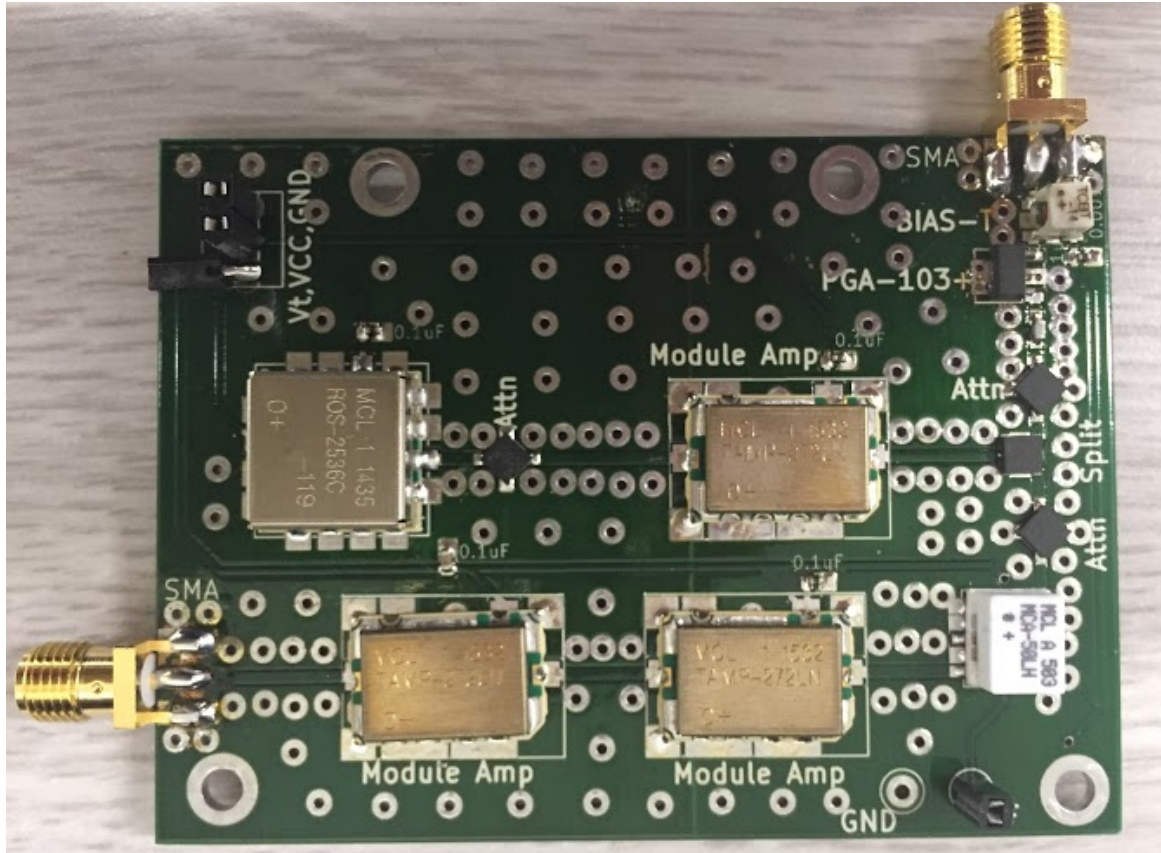


Figure #20: PCB Printing #2 (No Errors)

Above we can see the final board which actually took quite a bit more work to design as now the VCC line supplying the Bias-T with power had to be routed down underneath the mixer IC in the bottom right corner. This insured we did not directly cross our VCC signals and our high frequency transmission lines. We were very happy with this board, and it seemed to function well at around 30 meters. However there were some difficulties in the soldering of the board (you can see from this picture that some of the amps are a little crooked), and the board was actually resoldered quite a few times. **Every time you reheat the board you risk damaging anything that is already on there.** Since we had the extra parts, we resoldered a new board that only required one heating of the board. **This board far exceeded the first board in performance and allowed our system to see out to roughly 90 meters (about a 300% improvement in distance!).** We were very happy at this point that we had not wasted any parts on the flawed board, since we never would have seen this kind of performance without having enough parts to make a 2nd board. **I would recommend to anyone making a PCB to insure that you have at minimum enough components to make three boards, especially if you are new to soldering an RF board.**

I would like to conclude this application note with two more tips for RF Board design. The first being an issue I encountered many times, and the second, while not a critical skill, helped enhance the PCB design experience for me.

### Tip #1: Modifying Transmission Lines to Connect to Small Pads

A common issue that is encountered when making an RF board is dealing with large transmission line traces that cannot always physically make connections with small IC pads. At first this seems like it will force you to completely redesign the board, so that you can use smaller traces (i.e changing dielectric material, or spacing etc.). However, with a little trace creativity you can modify the transmission line so that a connection can be made to the smaller pad.

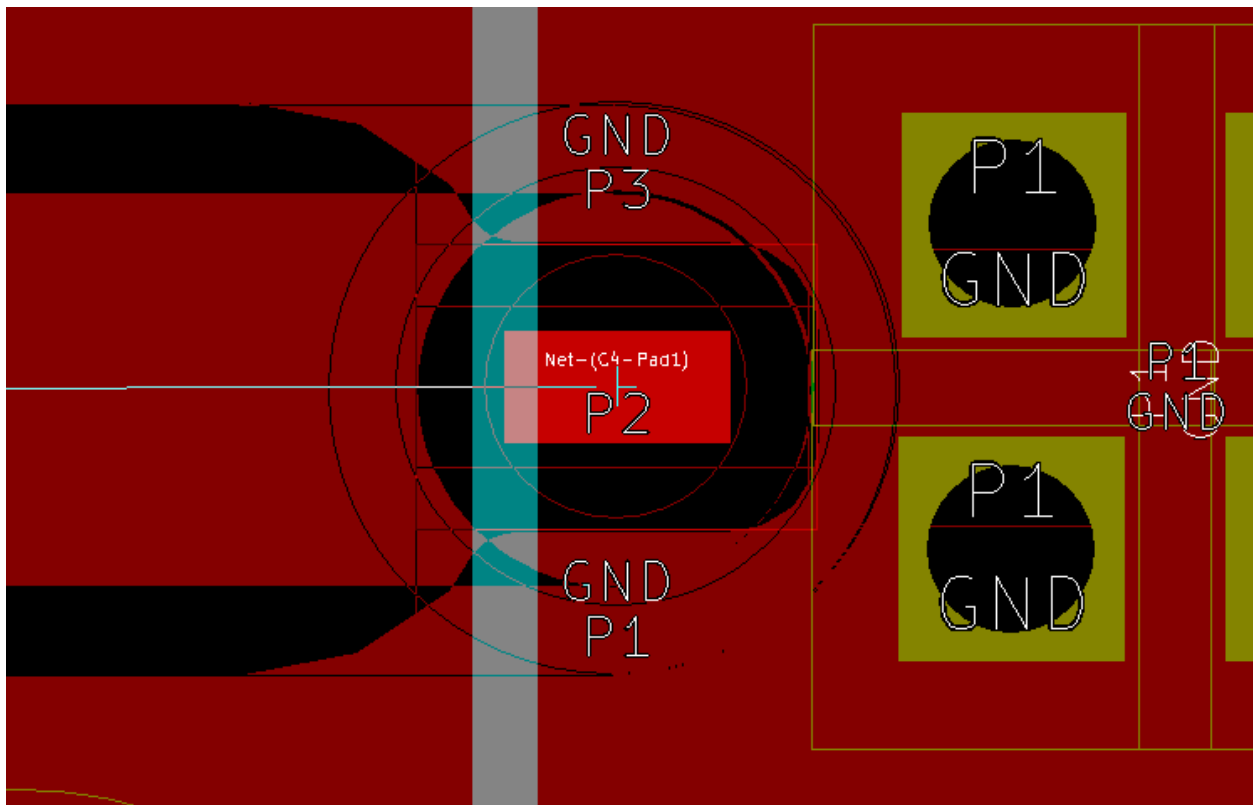


Figure #21: Large Trace Cannot Connect to Smaller Pad

This can be remedied by one doing their best to taper the trace down as it nears the pad. Note that this will change the impedance of the small section of the line, but as long as the tapering is kept as close as possible to the pad connection, the circuit should still perform as expected.

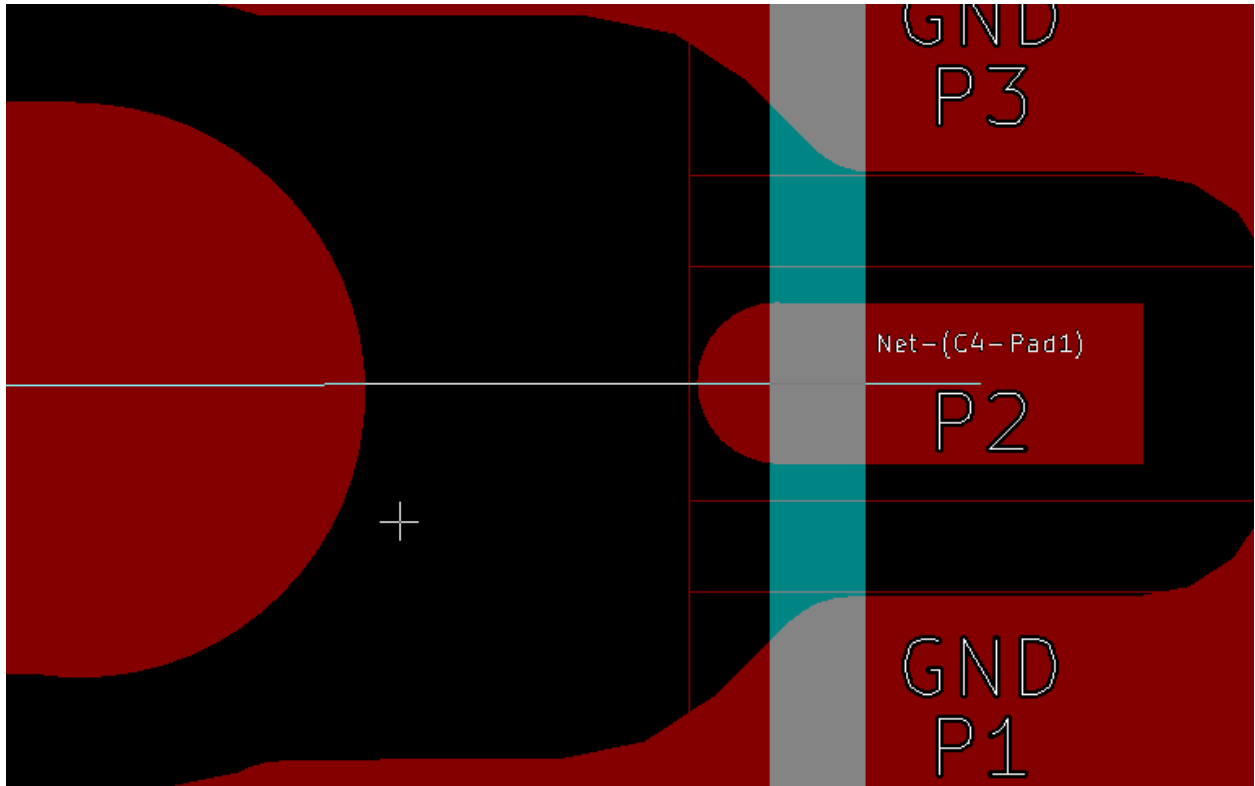


Figure #22: Tapering Tx Lines, Step 1

In the above **Figure #22** I stopped the large RF trace (at 35.1 mils width) trace about 30 mils from the pad and then with about a 10 mil trace extended the pad out a small distance.

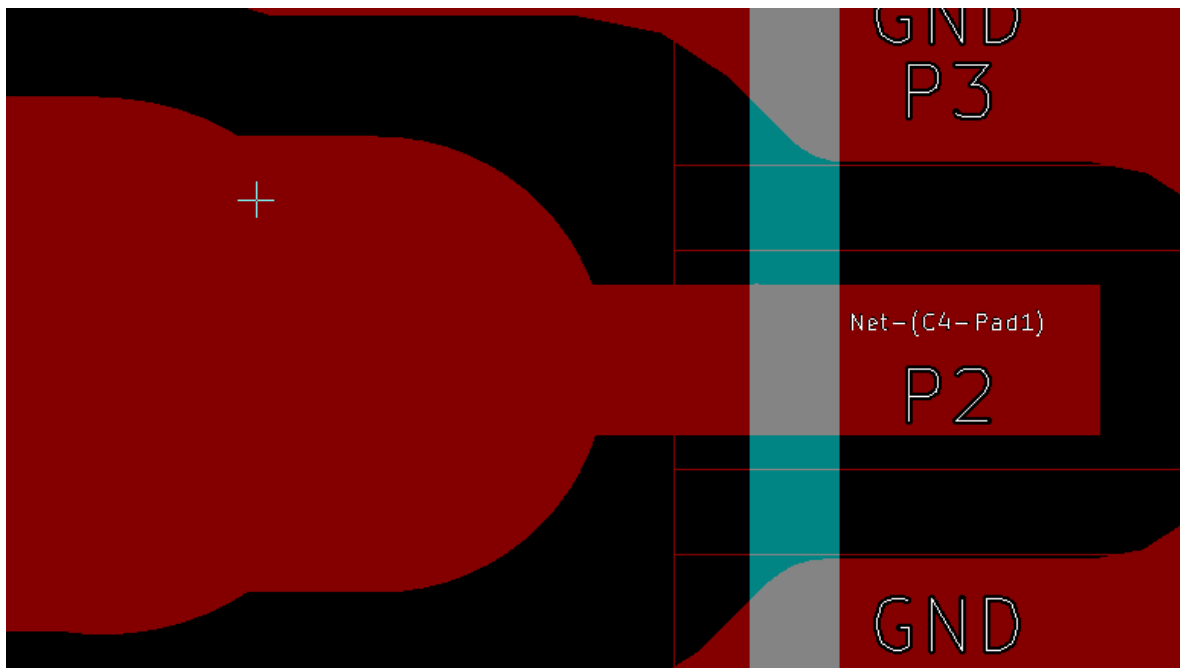


Figure #23: Tapering Tx Lines, Step 2

In the next step (**Figure #23**) I used another width of about 30 mils to taper down the trace from the 35 mil width. While also from the left making sure my 10 mil trace makes an actual connection. Note: If your traces do not actually make a connection the white line from the rat's nest will not go away and it will not pass KiCad's connection test.

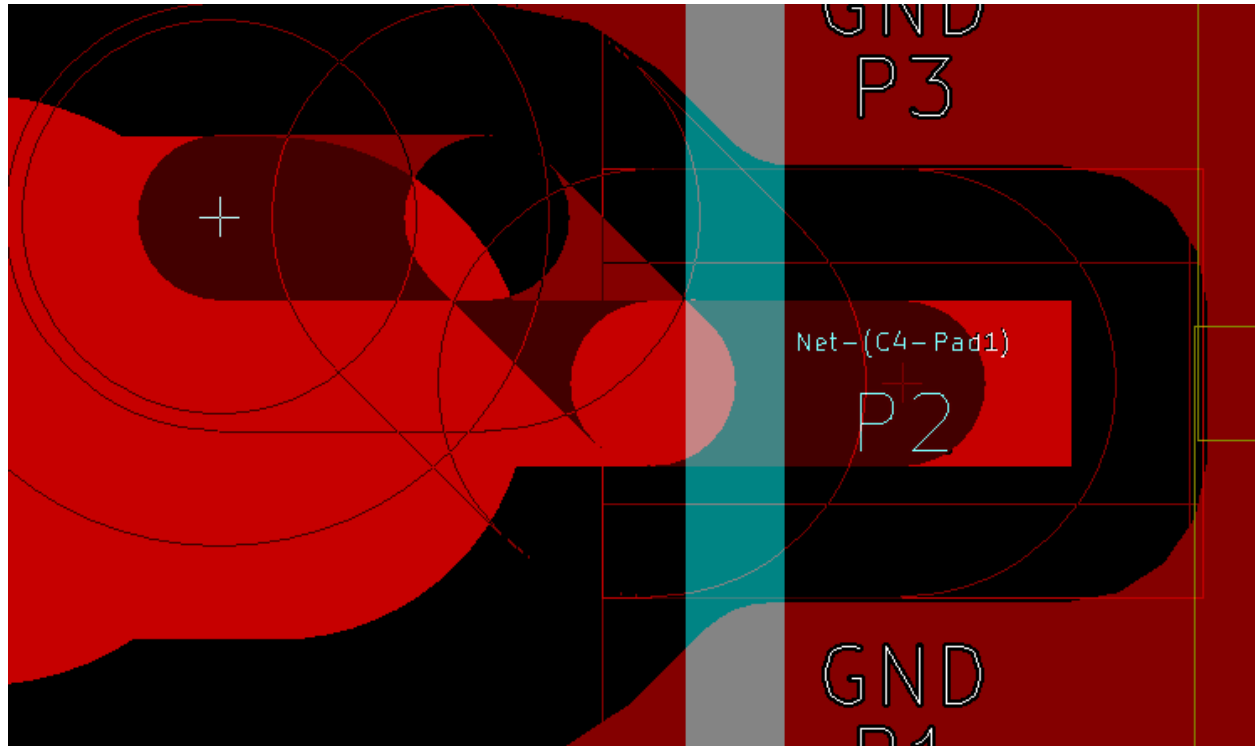


Figure #24: Tapering Tx Lines, Step 3

Here is a technique I used to help create the final taper into the pad by coming straight out of the pad and then making a 45 degree angle and clicking with your mouse, you can achieve a diagonal edge to the trace and the line it up precisely with the edge of the 30 mil trace.

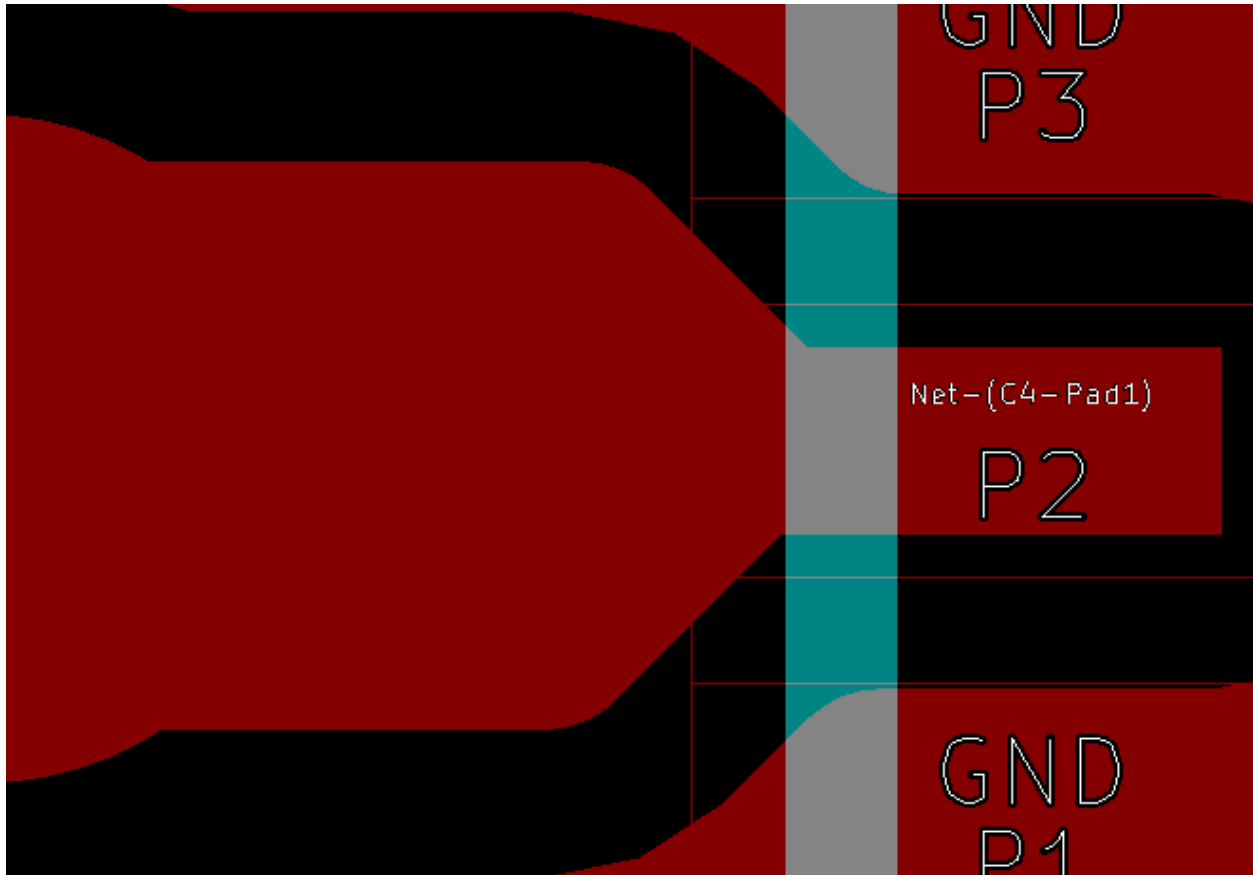


Figure #25: Tapering Tx Lines, Finished Product

The final result gives a good connection with the pad while reducing the trace width as little as possible. The chief goal is to maintain the transmission line width as much as possible and then to taper in a very short amount of space.

**Tip #2: Don't Forget to Have Fun**

I found that one of the ways to really add the final personal touch to a board was to add some sort of distinguishing logo or picture to the PCB. This section will walk through the basic steps to create a single color logo for the back of the PCB.

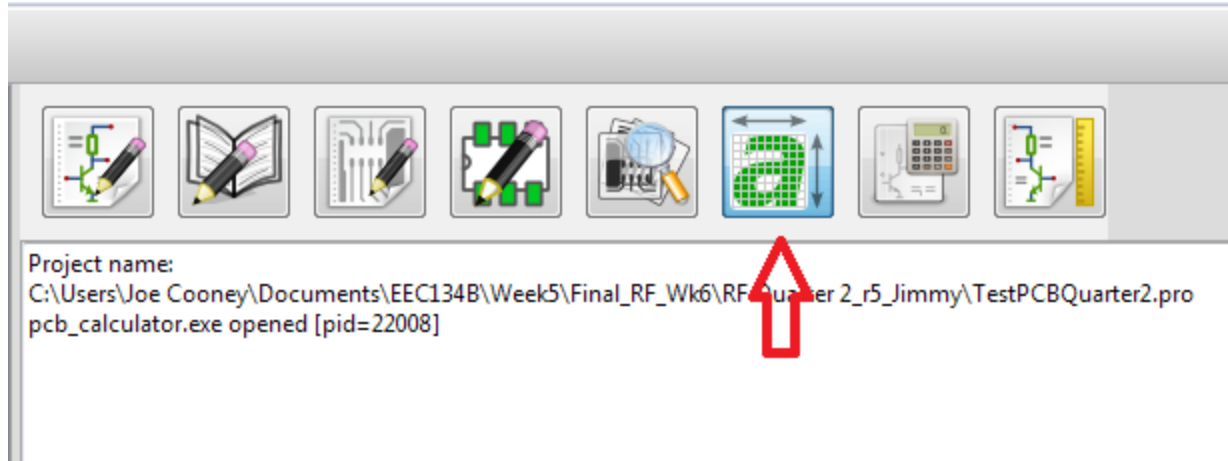


Figure #26: Bitmap to Component Converter

KiCad has a built in tool “Bitmap to Component Converter” which allows a user to upload a picture either .jpeg or .png, and then is able to convert it into a footprint that can be placed on the board. For this demo I downloaded a jpeg of the UC Davis Engineering Logo (**Figure #27**).



Figure #27: UC Davis Engineering Logo

I would recommend however that you use a .png since they are of better quality for creating a “negative space” image. Once you’ve settled on a logo, hit the “Load Bitmap” and select your downloaded Image.

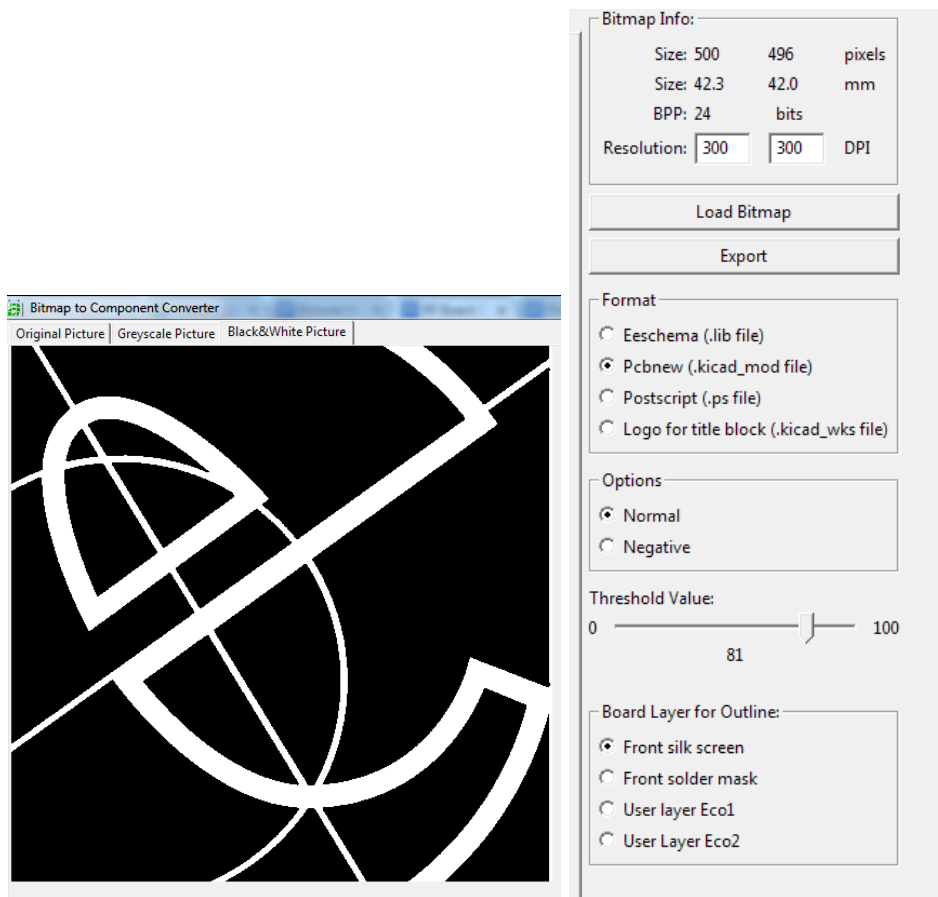


Figure #28: Exporting Picture as a .kicad\_mod file

In the editor I then select the black and white picture option and adjust the threshold until i am happy with the saturation on the image. Make sure the Format is in Pcbnew (.kicad\_mod format). You can then export, and save the logo to one of the libraries that you have already been saving footprints to.

Back in the PCB Footprint Editor, under File select "Import Footprint from File". After selecting the saved .kicad\_mod file you just created, you then should be able to see your image in the footprint editor (**Figure #29**). Note that if you would like at this time you can edit this image using the "Add graphic line or polygon" on the right side of the footprint editor. In this example I actually added some exterior lines to the logo to form the box around it.

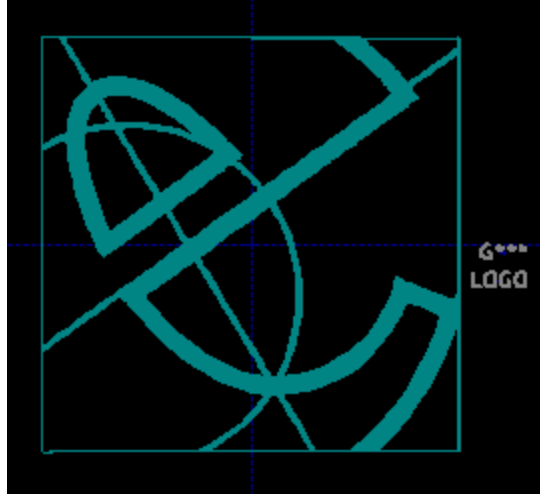


Figure #29

Once you are happy with the logo save the footprint in the current library or change libraries to one you have stored most of your created footprints in.

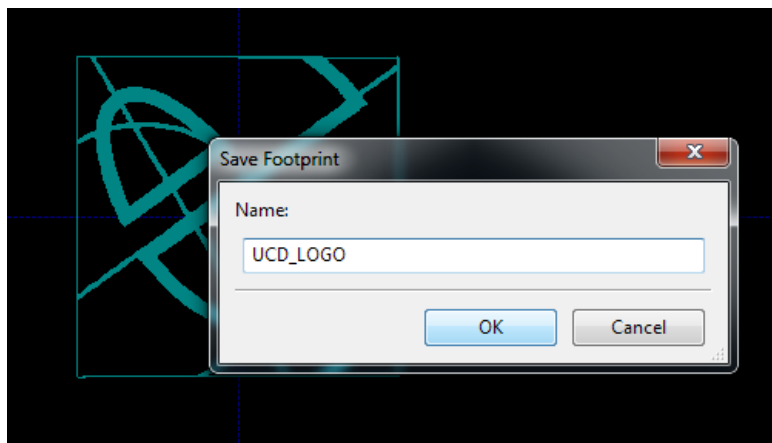


Figure #30

Now in the PCB Editor you can click on the “Place” tab and then the first option “Footprints”. Navigate to the library you have stored your logo in and select your logo. You should be able to see a preview of the logo in the box on the right.



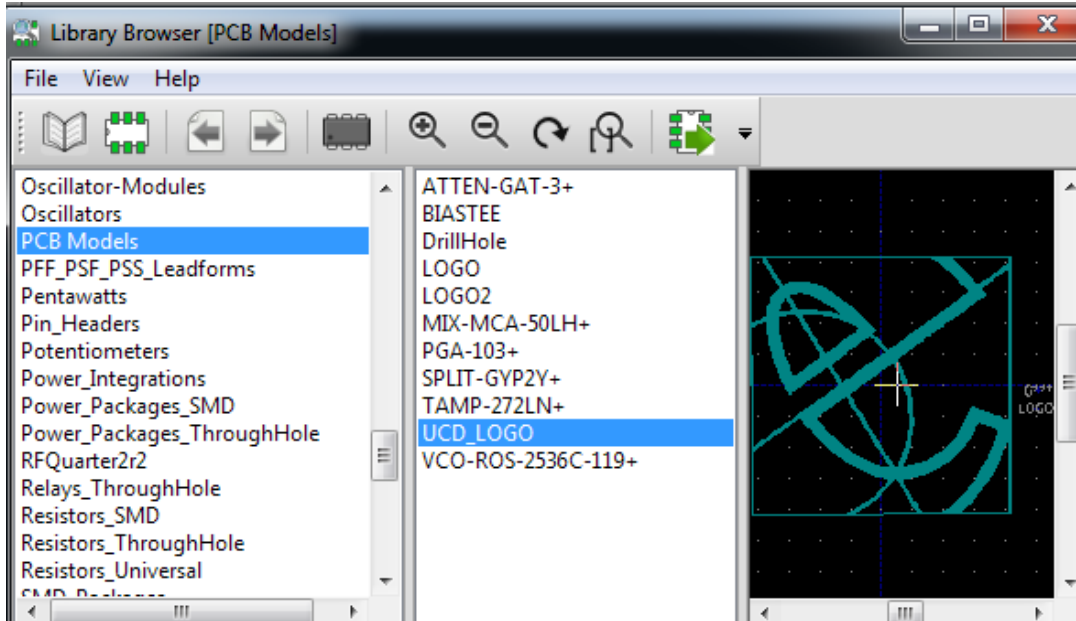


Figure #31: Import Footprint to Editor

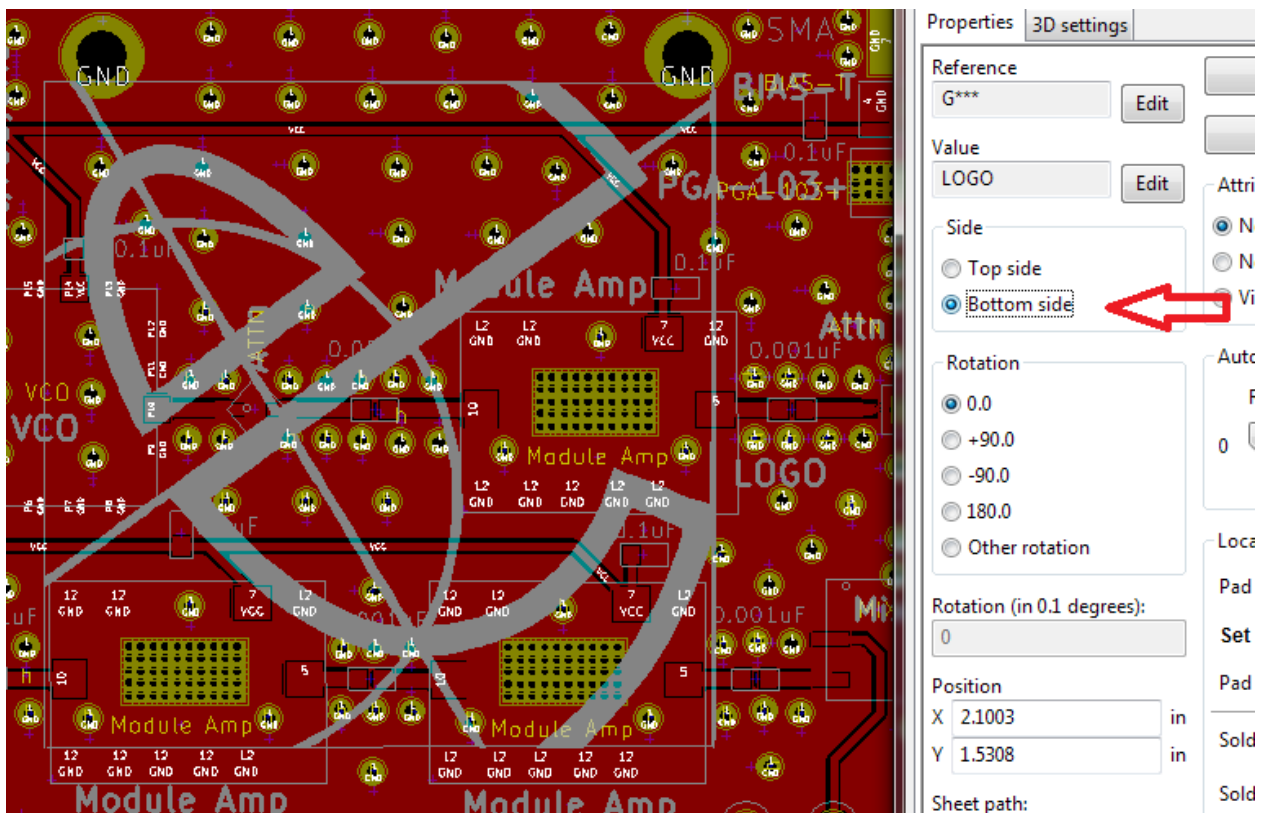


Figure #32: Edit the Footprint

You can now move your logo around on the PCB like any other component. If by misfortune your logo's dimensions do not match your board size (either too large or too small), you have to begin

the process over, this time selecting an image with a more appropriate dimension. Also, since you likely don't want your picture or logo interfering with other portions of your silk screen, I recommend putting the logo on the bottom side of the board. This can be done by hovering the mouse over the logo and hitting "e" on the keyboard (e is the hotkey for "edit"). You can now flip the logo to the bottom side of the board, and rotate it if necessary (**Figure #32**).

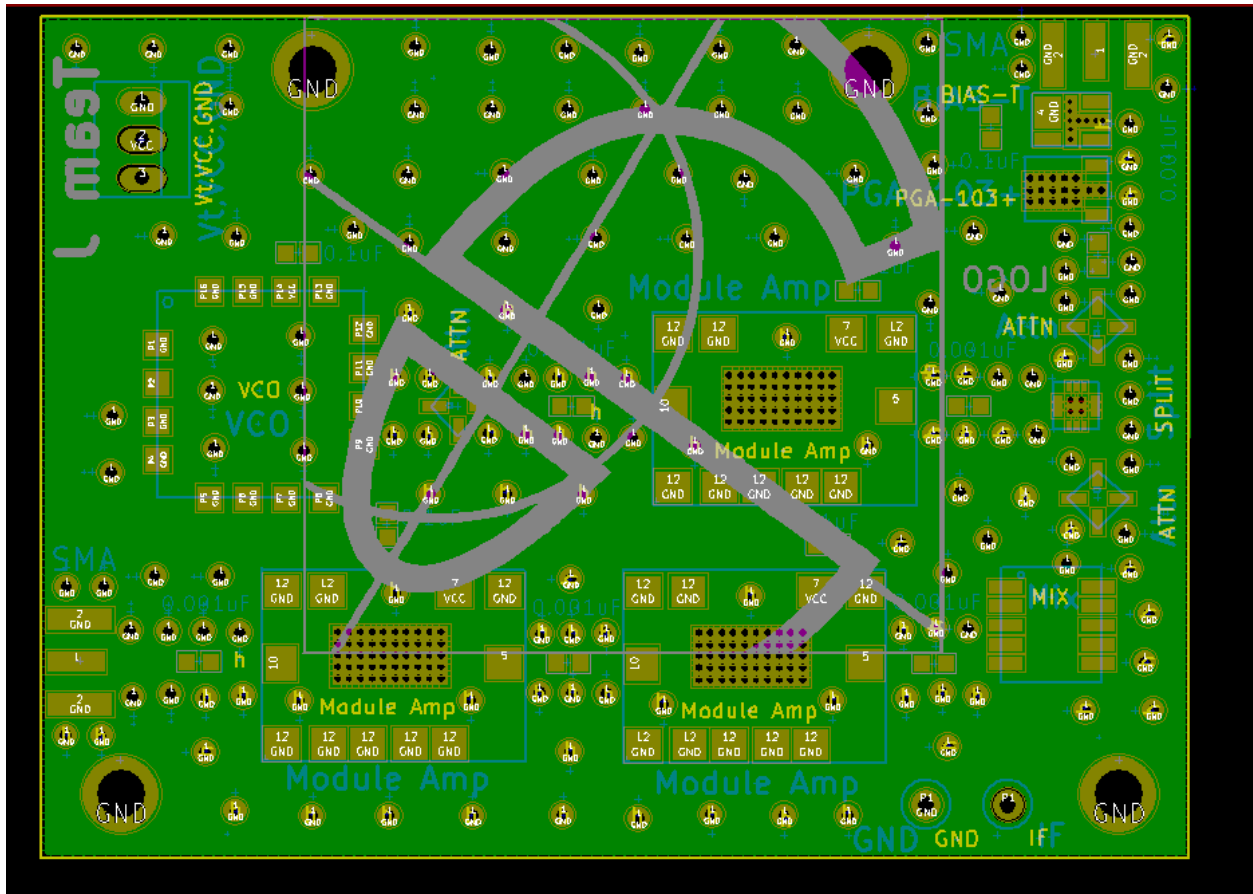


Figure #33: Final Printed Logo

Don't be alarmed that the logo is upside down, this is exactly how it should look since you are viewing the board from above (as if the board was made of glass). Remember to include whatever layer you have assigned the logo to in your Gerber File in order to have it printed to the board.

Here are two of the designs I made, which were inspired by one of my favourite cartoon shows “Rick and Morty”. They are always getting into shenanigans involving science and technology and they seemed like appropriate mascots for our RF boards.

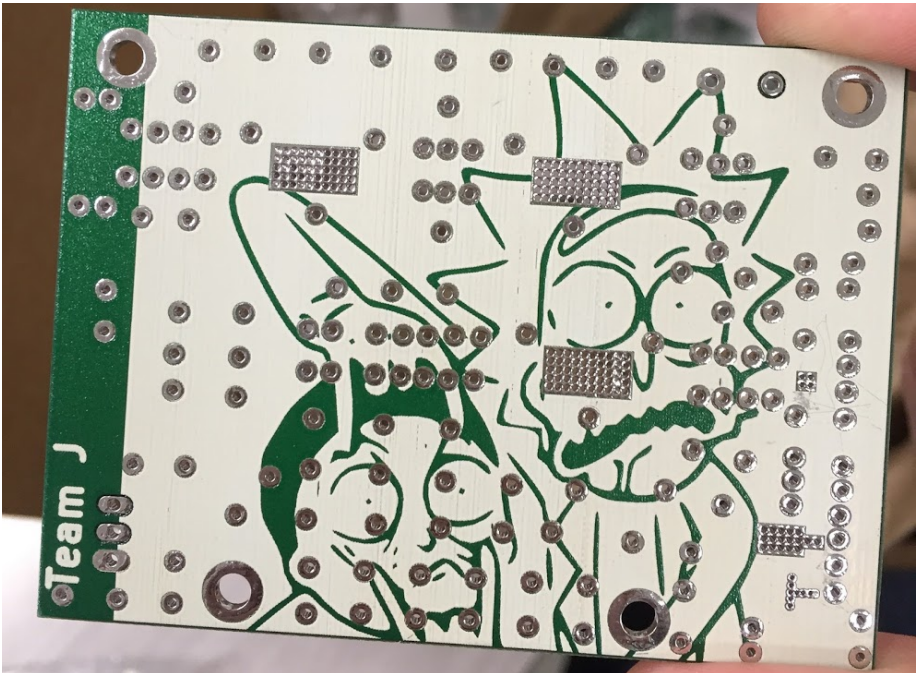


Figure #34: Printed Logo (Rick and Morty Board 1)

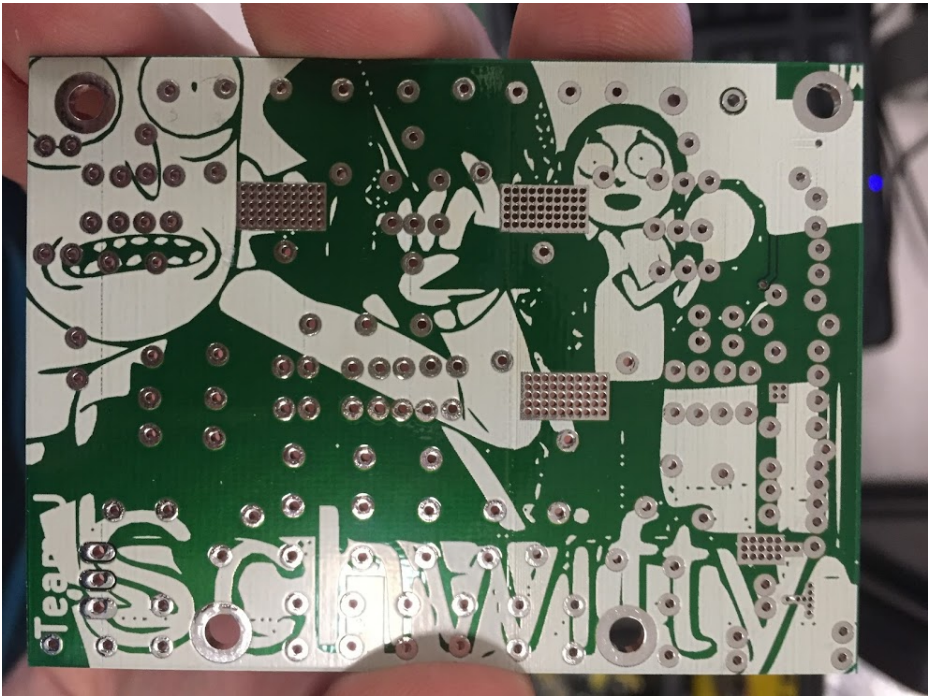


Figure #35: Printed Logo (Rick and Morty Board 2, “Schwifty”)

## **Conclusion**

Designing and building an RF PCB is a challenging task, where tiny errors can completely disable boards. This application note just barely scratches the surface on the basics of using KiCad and PCB design in general. There is a plethora of information available on the web, but as with many things, the only way to get good at KiCad and PCB design is to put in a lot of practice. Remember to always check and triple check your work, as well as have others do the same. While it is definitely not an easy skill to learn, the reward of a working board is well worth the process.