You may contact me at EldadHellerman@gmail.com.

This document is meant to accompany the “Arduino project design” practical workshop.

Contents

Getting started with KiCad

In KiCad, the general workflow is as follow:

I. Creating a schematic – “Schematic Editor”.
All the component symbols should be places and wired, then all values assigned. Also, it is at this point that the footprint of each component is assigned – which is what makes the schematic geared towards a physical design.

II. Creating additional symbols if necessary – “Symbol Editor”.
We will often come across symbols which we do not have in the built-in libraries that come with KiCad. When this happens, we can make our own symbols.

III. Creating additional footprints if necessary – “Footprint Editor”.
Just like with symbols, we often have to make out own footprints.

IV. Creating the PCB according to the schematic – “PCB Editor”.
This is where the PCB itself is designed. All the different footprints should be laid out and then routed using traces.

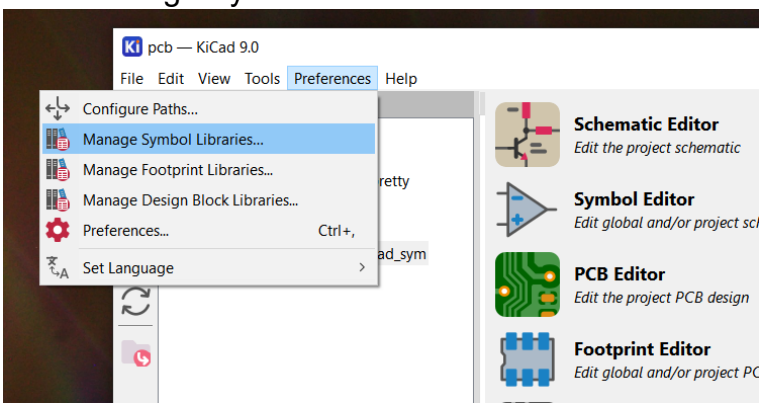
V. Exporting the PCB.
The default format for this is the “Gerber Format”.

VI. Inspecting the result – “Gerber Viewer”.

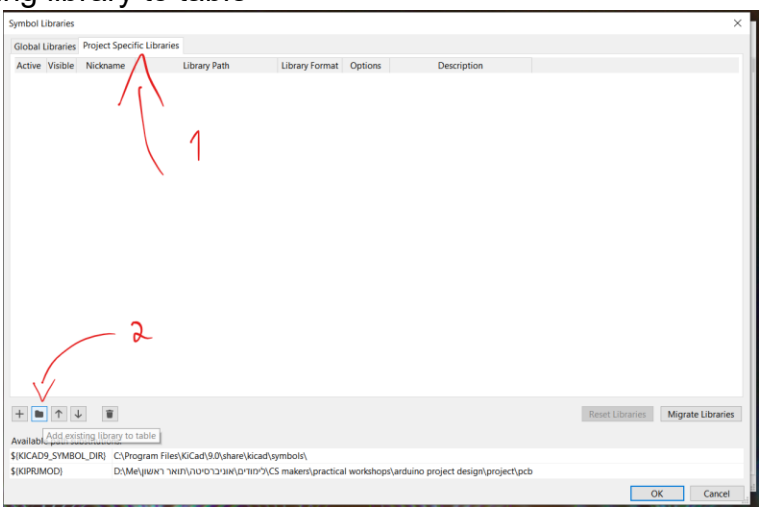
Let’s get started!

Creating a new project

1. Open KiCad (version 9 will be used throughout).
2. Start a new project: “File -> New Project”
3. We want to add the included files for this exercise into our project. Move the included pcb files to the newly created KiCad project folder.
4. “Preferences -> Manage Symbol Libraries…”



5. “Add existing library to table”

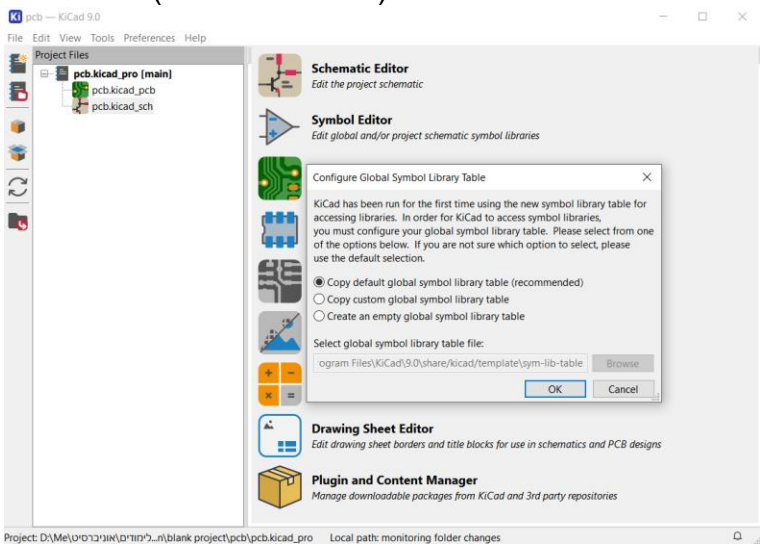


6. Selected the given symbol library file.
7. Similar to steps 2-4, go to “Preferences -> Manage Footprint Libraries…”, then “Add existing”, then select the given footprint library directory.

Creating the schematic

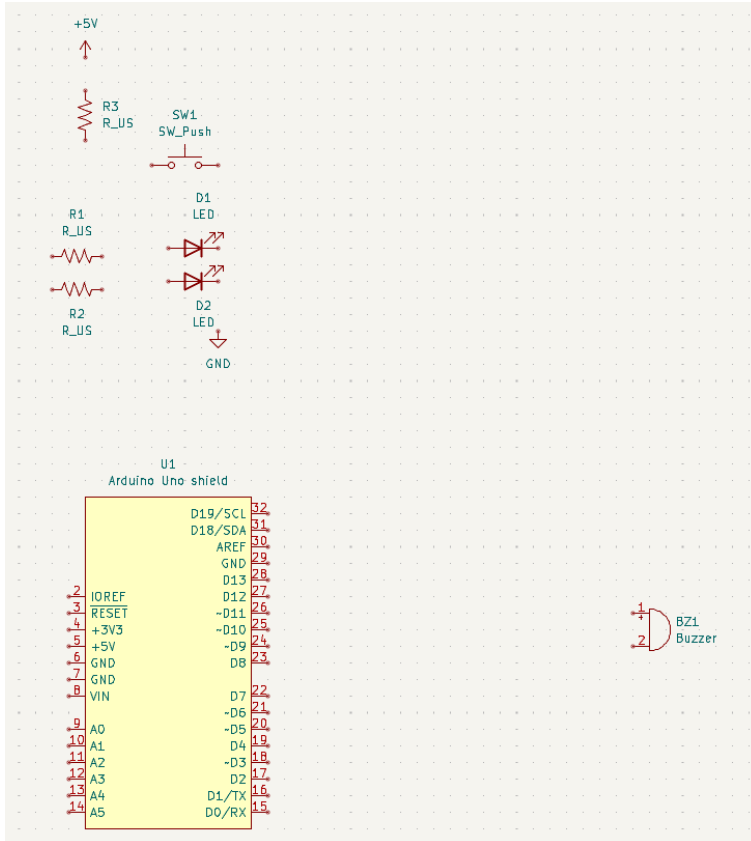
Adding symbols

1. Open the “Schematic Editor”.
   When prompted to configure library tables, go with the recommended option.
   Same goes for the PCB (or other editors) later.



2. Add all the required symbols:
   a. Three resistors – you can choose between ‘R’ or ‘R_US’.
   b. Two LEDs – ‘LED’.
   c. One button – ‘SW_push’.
   d. Power - one ‘GND’ and one ‘+5V’.
   e. One ‘Buzzer’.
   f. One Arduino shield – ‘Arduino_Uno_shield’.
      You can also find it under the symbol library we imported earlier.

   Items a-d are for one quarter region and will later be duplicated for the other quarters.
   Your schematic should now look something like this:



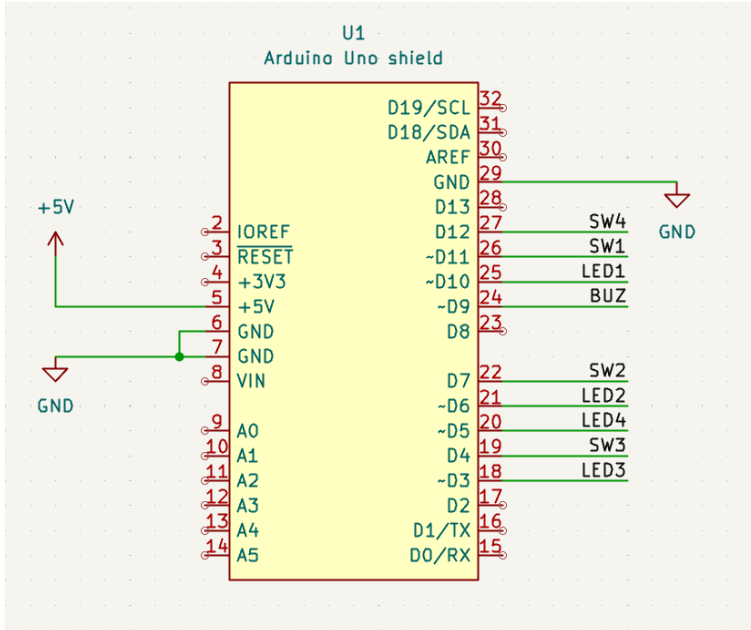
Wiring everything

First, we will connect the Arduino shield. All wires with the same label are interconnected. So, for simplicity we will just connect the used Arduino pins to labels (which we will later use), and power pins to supply.
Here is a table of all the required connections:

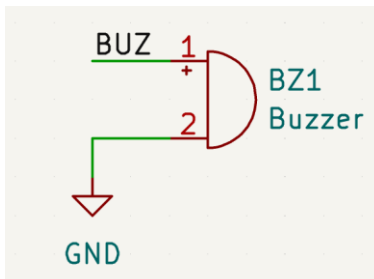
Pin	Label name
D12	SW4
~D11	SW1
~D10	LED1
~D9	BUZ
D7	SW2
~D6	LED2
~D5	LED4
D4	SW3
~D3	LED3

1. Create all the labels shown in the table and connect each one of them to the corresponding Arduino pin.
2. Also connect the following (Arduino_Uno_shield pins -> symbol):
    a. ‘+5v’ pin -> ‘+5V’
    b. ‘GND’ pins -> ‘GND’

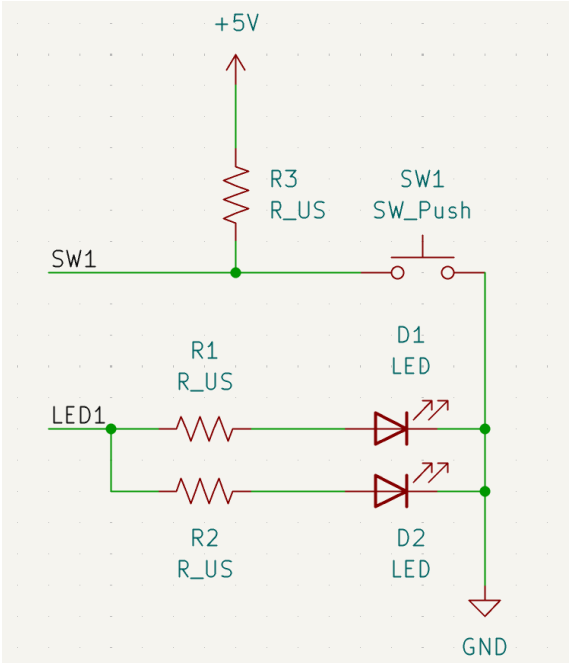
3. Check yourself - your schematic should now look something like this:



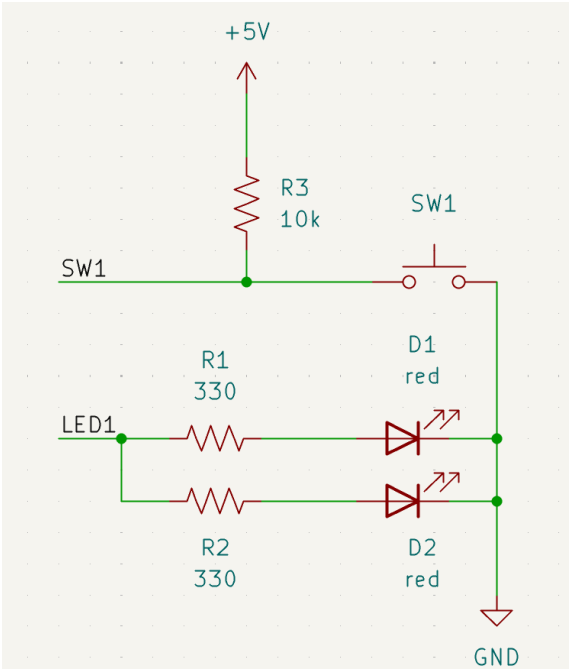
4. Now let's connect the buzzer. It should be connected as follows:



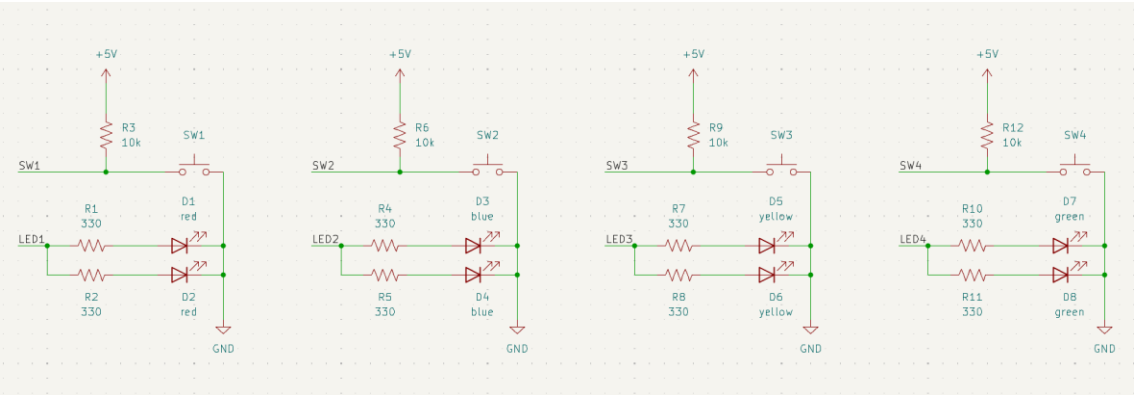
5. Now for a quarter circuit, here's what it should look like:



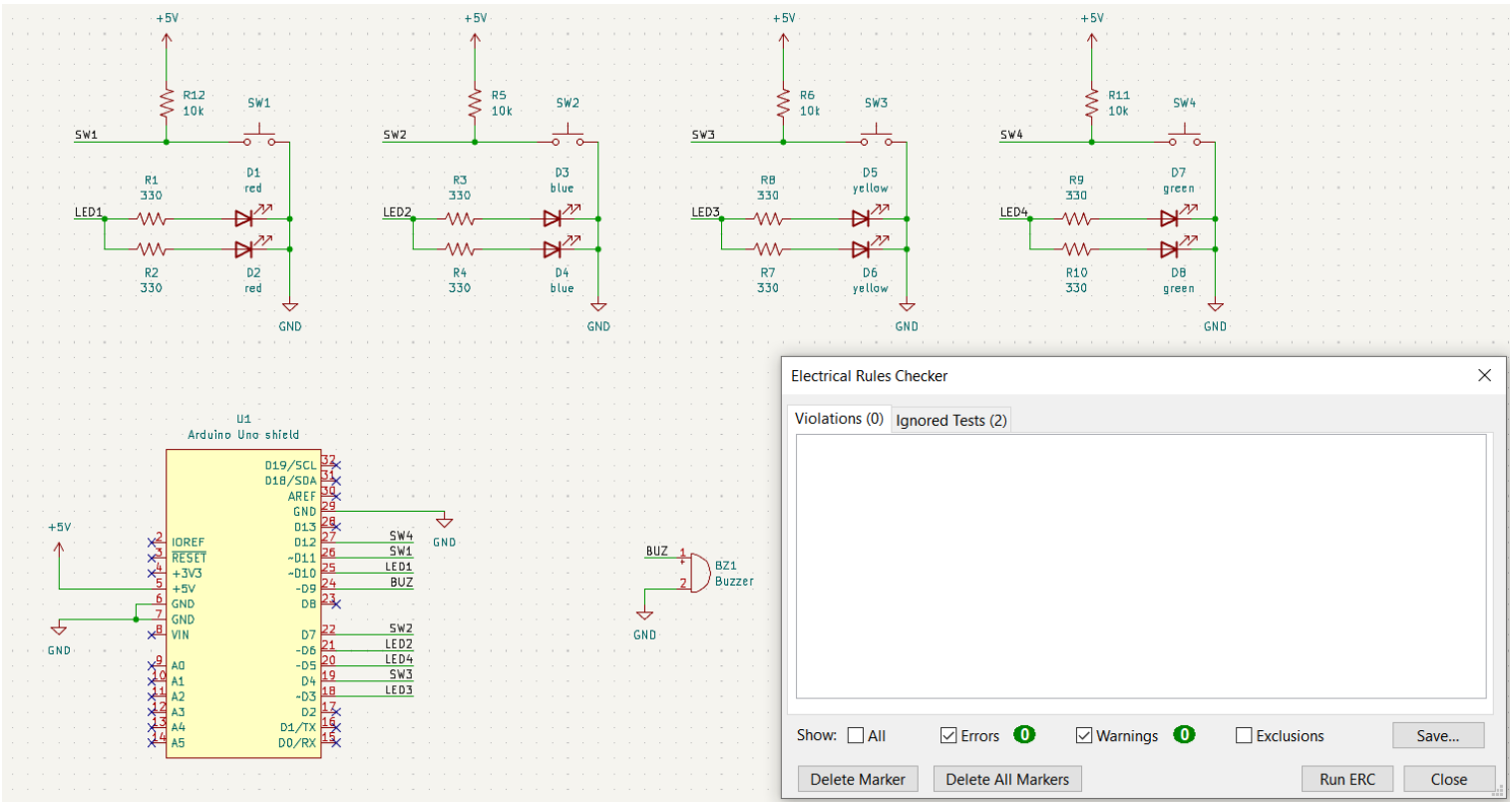
6. Let's fill in the different values. The LEDs resistors should be 330 ohms, the pullup one should be 10k, and both the LEDs have the color red. It should look like:



7. Duplicate this part another three times, for a total of four copies, one for each quarter.
8. Change the different net labels and LED colors to match the different quarters. The result should look like:



9. Run an ERC check. You should see multiple errors. To fix them, we should mark all unconnected pins as unconnected, do so with a “no connect flag”.
10. Run an ERC check again. There should not be any errors.
11. The result should look like this:



Assigning footprints

We will now assign all the footprints. Here is a table of what's needed:

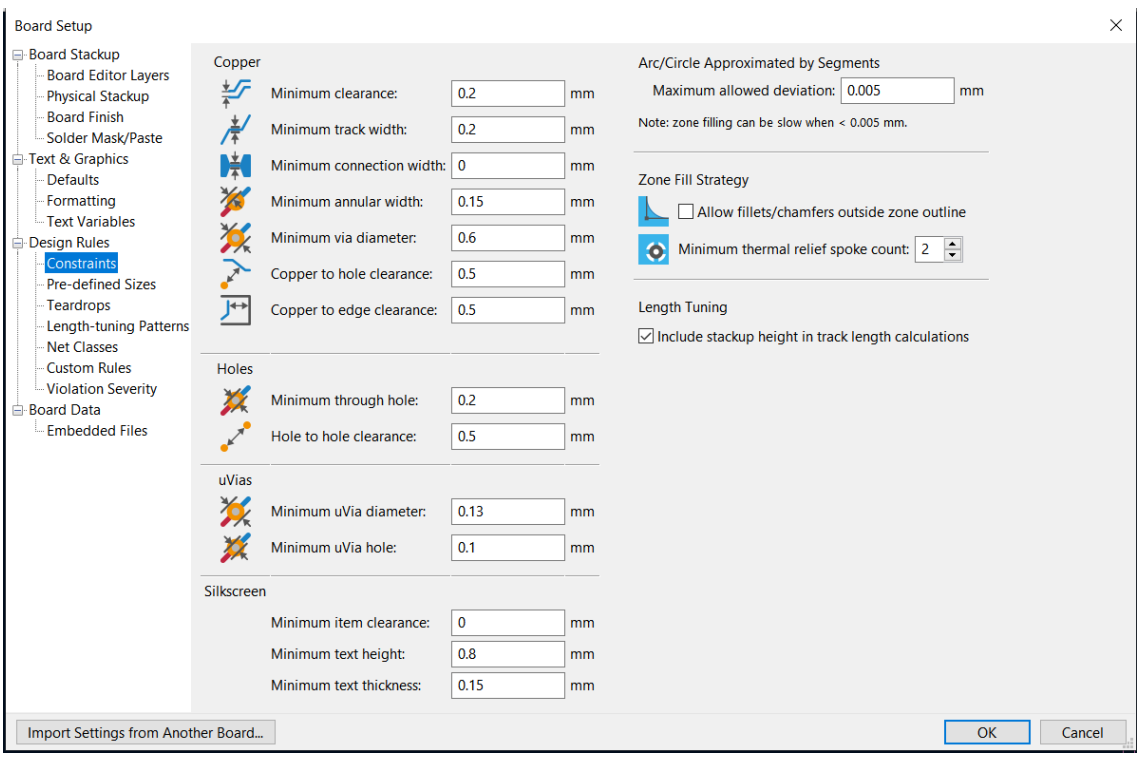
Part	Footprint	
	Library	name
buzzer	Buzzer_Beeper	Buzzer_12x9.5RM7.6
LEDs	LED_THT	LED_D3.0mm
resistors	Resistor_THT	R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
buttons	Button_Switch_THT	SW_PUSH_6mm_H8mm
Arduino_Uno_sheild	A footprint library	Arduino Uno shield

1. Open the "Assign Footprints" tool – "Tools -> Assign Footprints…".
2. For each component, select the appropriate footprint from the options.
3. Click "Apply, Save Schematic & Continue".
4. Run an ERC check again to check that everything is working.

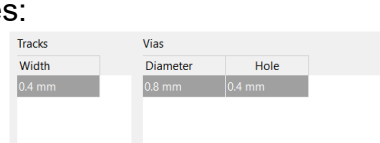
Creating the PCB

Setup

1. Open the PCB editor. It is recommended to work with two screens and keep the schematic editor open as well – they work in beautiful harmony with one another.
2. First let's define the layers we need, so the view is less crowded:
    a. Go to "File -> Board Setup…".
    b. "Board setup -> Board Editor Layers"
    c. Among the optional layers, keep only "F.Silkscreen", "B.Silkscreen", "F.Mask", "B.Mask" and "User.Drawings".
3. Set up the DRC rules:
    a. Still under "File -> Board Setup…".
    b. "Design rules -> Constraints".
    c. Fill in the following values (taken from a manufacturer website):



4. Set up track size:
    a. Still under "File -> Board Setup…".
    b. "Design rules -> Pre-defined Sizes".
    c. Add the following sizes:



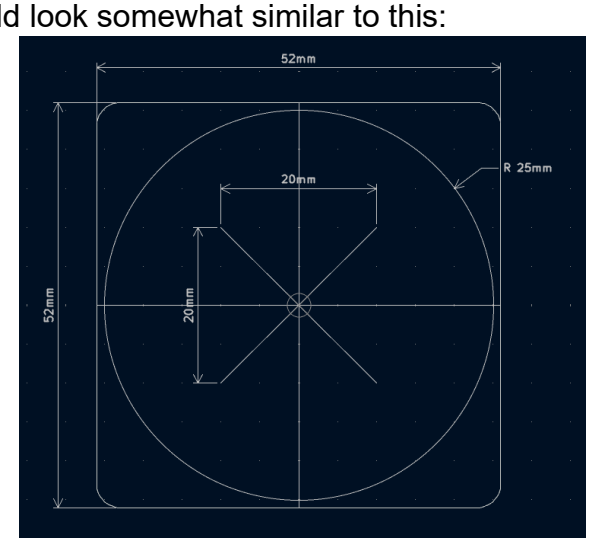
5. Also, enter those dimensions to the default net class:
   a. Still under “File -> Board Setup…”.
   b. “Design rules -> Net Classes”.
   c. Write the same sizes and clearances.
6. It would be easier if all locations would be relative to a point we chose, to do so:
   a. First set the grid origin to a comfortable point – “Edit -> Grid origin…”. (100mm, 100mm) would be ok.
   b. Then change the display origin to that point - “Preferences -> Preferences -> PCB Editor -> Origins & Axes -> Display Origin -> Grid origin”.
   c. You can also set y direction to be up – “Y axis -> Increases up”.

Defining board shape

1. Go to the “Edge.Cuts” layer and draw the board shape.
   In our case, the board is a 52x52mm rectangle.
2. Center the board around the origin.
3. Add some fillet to the corners, 3mm would look nice.
4. Have a look at the result – “View -> 3D Viewer”.

Let’s also add a user drawing of the location of the buttons and other useful data, so that it’s easier to inspect things visually later on:

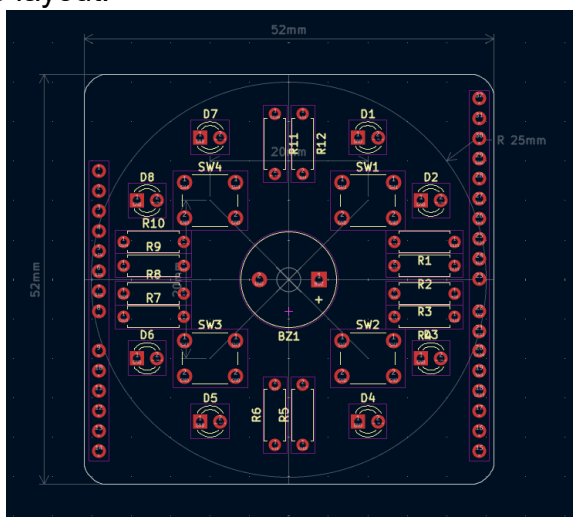
5. Go to the “User.Drawings” layer and draw both axis around the origin, as well as a 50mm diameter circle, and a 20x20mm square, whose corners will be used to indicate the center of the buttons.
   You can also do similar things with dimensions only (as was done in the following picture) – “Place -> Draw Dimensions -> …”.
   The result should look somewhat similar to this:



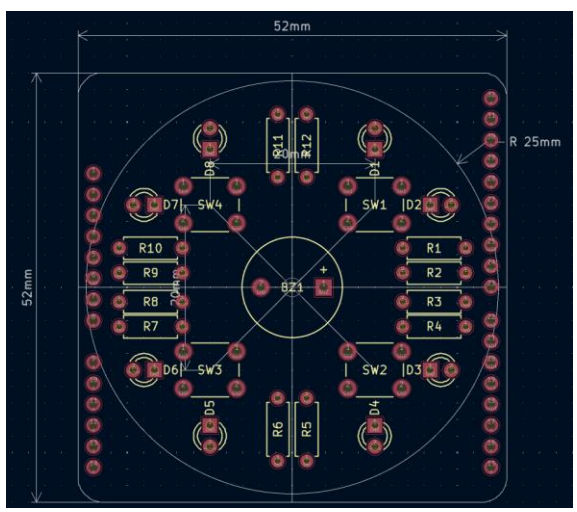
Layout

First, we need to add all the footprint to the PCB. Unlike in the schematic where we added the symbols manually, this is not needed here since each symbol already has an assigned footprint, so all we need to do is send those to the PCB.

1. “Tools -> Update PCB From Schematic…”, then “Update PCB”.
2. Place the components somewhere on the sheet.
3. Now, for the layout. Take your time and find a place for each component. All components should fit, and they should not touch each other. Here we also have many physical constraints on where we want the buttons and the LEDs to be. This can be a challenging task.
   Here’s a possible layout:



4. While we’re at it, we can also move the silkscreen to better positions. For example:



5. You can also run DRC now to check there are no collisions.

Tracing

Now to connect everything together. The ratsnest shows us what should be connected to what, but this is mainly to help in the layout process.
We will ignore GND connections, since later we will do a ground pour to connect all of those.

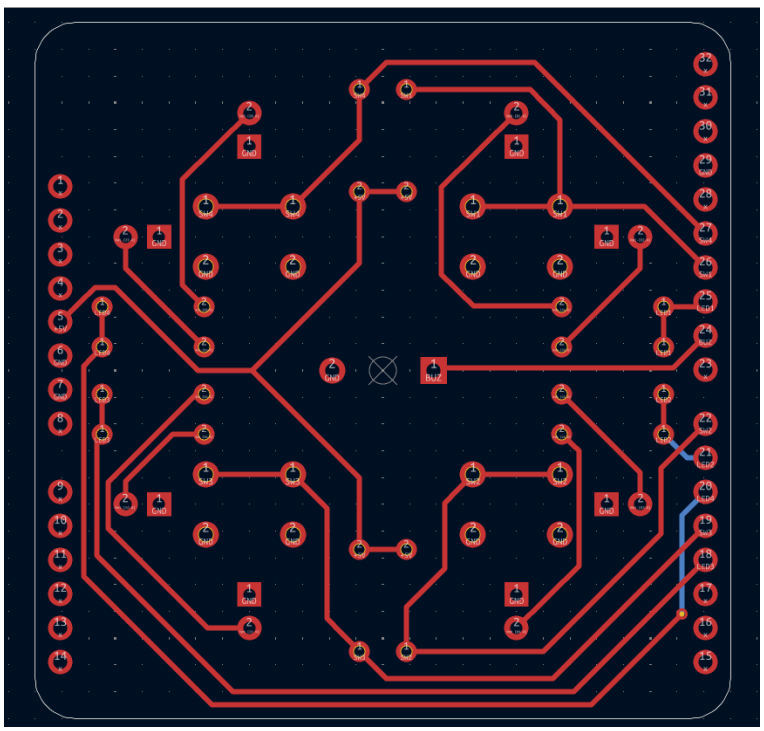
1. Go over each pin and try to trace them all.

This can be a challenging task.

There are a few things that can help us:

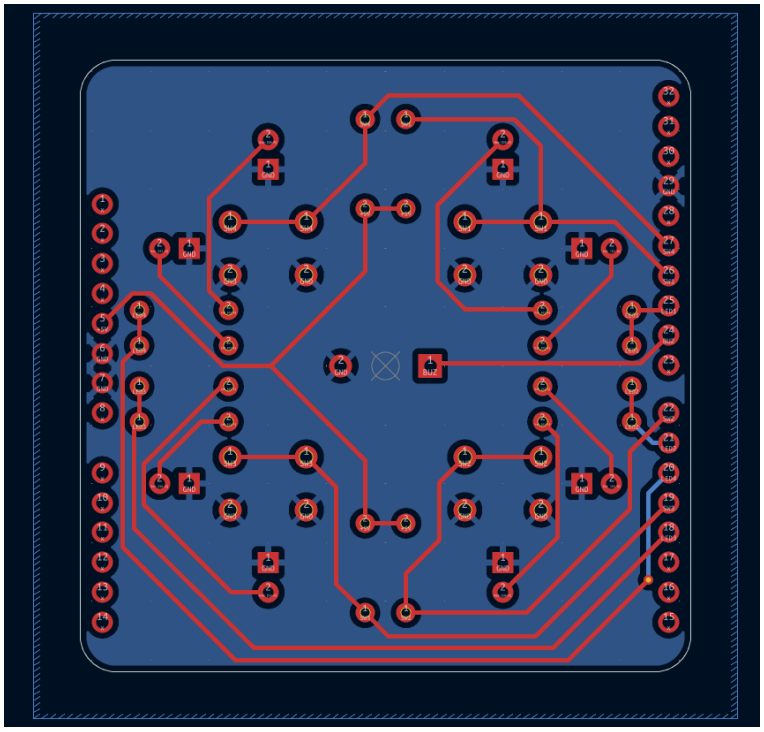
- We can trace on the other side of the board as well!
- We can add vias, which allows a single trace to cross between sides.

Here is how I routed everything:



But of course, there are many different options.

For connecting the ground, create a filled zone which includes the entire board, set its net type to ground, then fill all zones. Here's what it should look like:



You should now be able to run a DRC and get no errors.

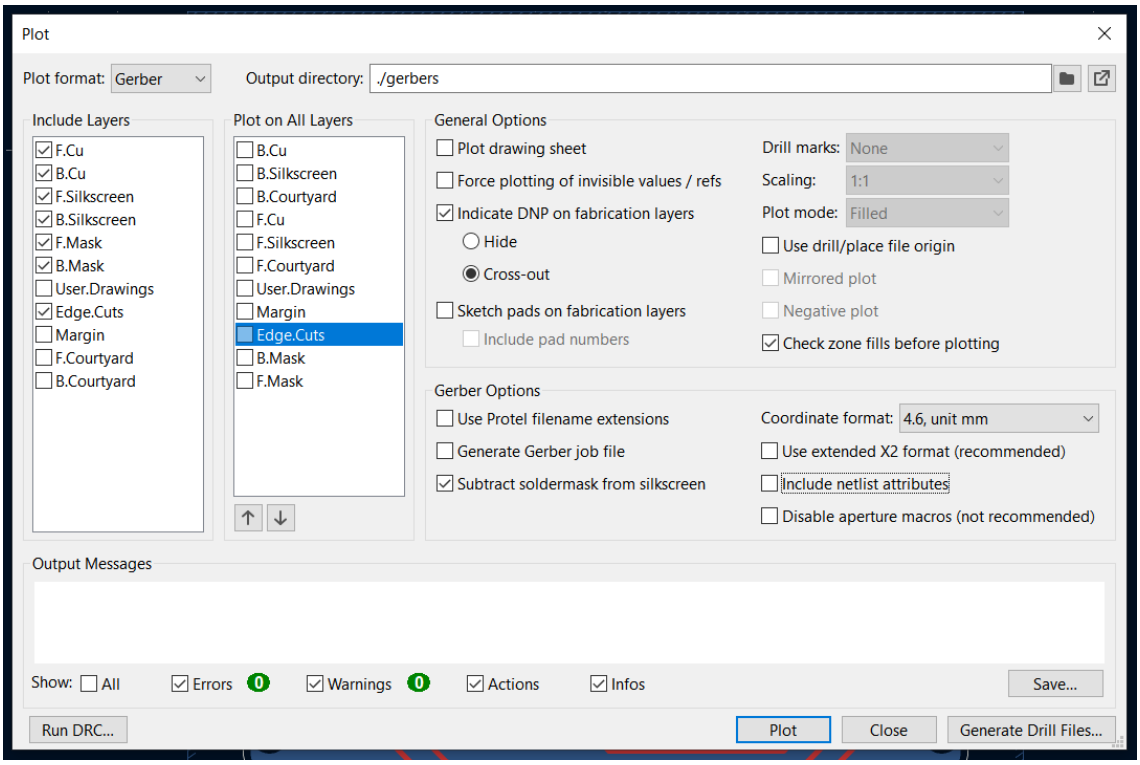
Also, open the 3D viewer and take a look at the final result!

Manufacturing the PCB

Exporting PCB as Gerber

To send the PCB to a manufacturer, we need to export it in the standard Gerber format.

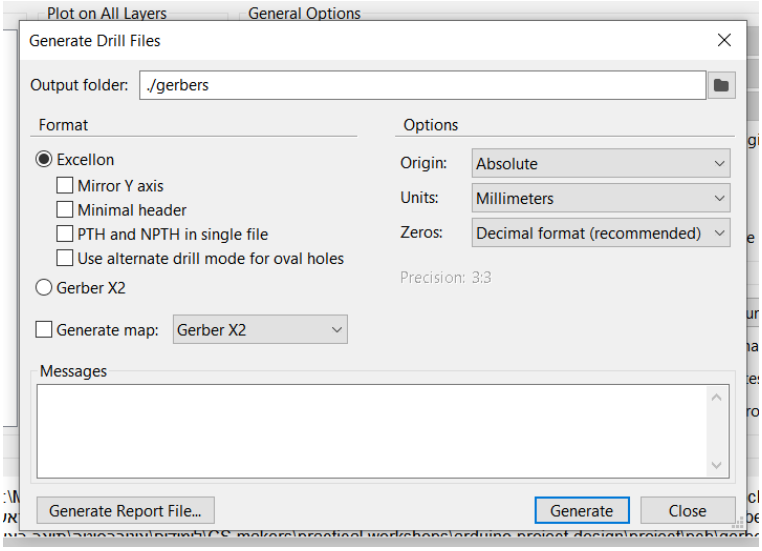
1. “File -> Fabrication Outputs -> Gerbers (.gbr)…”
2. Fill in the following settings (according to ‘PCBWay’ guidelines):



3. Click “Plot” to save the PCB files.

Now to export the drill files:

4. In the same window, click “Generate Drill Files…”.
5. Fill in the following settings (according to ‘PCBWay’ guidelines):



6. Click “Generate” to save the drill files.
7. You may also compress this folder to a zip file, for added convenience.

Inspecting Gerber files

1. Open the “Gerber Viewer”.
2. “File -> Open Zip Archive File…”.
3. Take your time, go over all layers and visually inspect everything to see that it all looks good.

Making the order

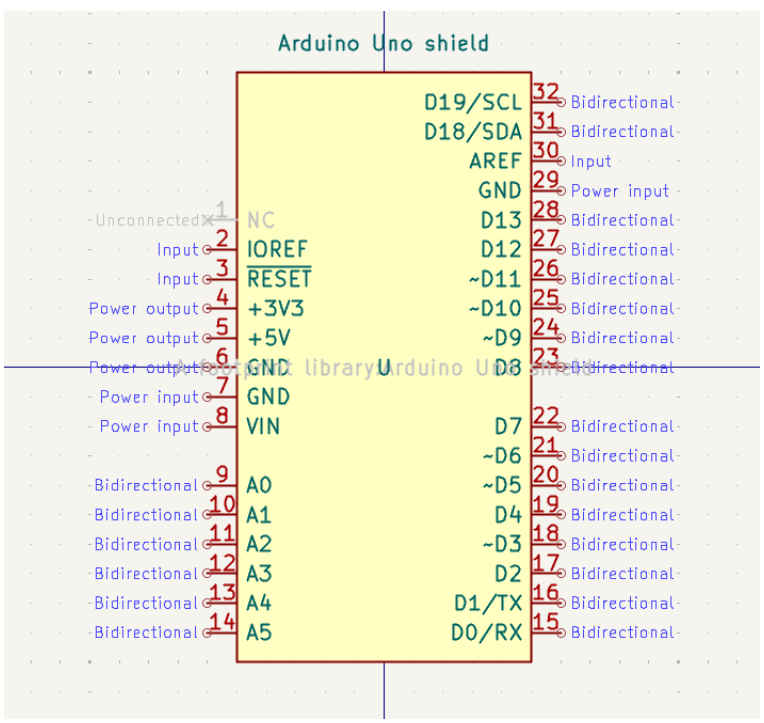
Go to some PCB manufacturer website, upload the Gerber files, and add to cart. It’s that simple. There are a lot of advanced options, but the often are explained by the manufacturer, so we won’t go into details here.

Appendix A – Creating a custom symbol

In this appendix, you will create a custom symbol for the Arduino Uno shield, instead of the one you were given. You are advised to look at it in the schematic or in the symbol editor and use it as a reference.

1. Open “Symbol Editor”.
2. “File -> New Library…”, then choose between “Global” or “Project” and name the library.
3. “File -> New Symbol…” and name the symbol.
4. Draw a rectangle to be the body of the symbol and fill it in with the body background color.
5. Add all the symbol pins. It is easier to do so with “Edit -> Pin Table…”. Again, if you open the given Arduino shield symbol, you can view its pin table to get all the required data.
6. Place all the pins to your liking.
7. Run “inspect -> Symbol Checker” to check that everything is ok.
8. You can switch to your symbol in the schematic by going to its properties, then “Change Symbol…”, then choosing your symbol and clicking “change”.
9. You now need to assign a matching footprint to your symbol. If you kept the same pin numbers, you can still use the given footprint.
10. Run an ERC, then go to PCB and run a DRC. Check that nothing broke.

Given Arduino Uno shield symbol for reference:



Appendix B – Creating a custom footprint

In this appendix, you will create a custom footprint for the Arduino Uno shield, instead of the one you were given. You are advised to look at it in the footprint editor and use it as a reference.

1. Open “Footprint Editor”.
2. “File -> New Library…”, then choose between “Global” or “Project” and name the library.
3. “File -> New Footprint…” and name the footprint by saving it.
4. Add all the pads corresponding to the symbol.
5. The pads’ location should match the physical component! Using the same pin numbers as in the symbol. Use the given footprint as reference.
6. Optional – add a courtyard around the symbol.
7. Optional – add a matching 3D model.
8. Run “inspect -> Footprint Checker” to check that everything is ok.
9. You can switch to your footprint in the schematic by assigning it to the symbol. Don’t forget to update the PCB from the schematic.

Run an ERC in the schematic and a DRC in the PCB to check that nothing broke.

Appendix C – KiCad common hotkeys

To view all hotkeys, you can press Ctrl + F1 inside any window of KiCad.

	Hotkey	Description
Common	Ctrl + S	Save changes
	Space	Reset local coordinates
	N	Change grid size
	Shift + N	Change grid size
	`	Highlight net
	Ctrl + D	Duplicate
Schematic Editor	A	Place symbols
	W	Draw wires
	/	Switch segment posture
	E	Properties
	M	Move
	G	Drag
	F	Edit footprint
	U	Edit reference designator
	V	Edit value
	X	Mirror horizontally
	Y	Mirror vertically
	R	Rotate counterclockwise
	L	Place net labels
	O	Autoplace fields
PCB editor	A	Place footprint
	X	Route single track
	/	Switch track position
	E	Properties
	M	Move
	D	Drag
	P	Pack and move
	R	Rotate counterclockwise
	V	Toggle layer / Add via
	F	Change side / flip
	B	Fill all zones
	Alt + 3	3D viewer