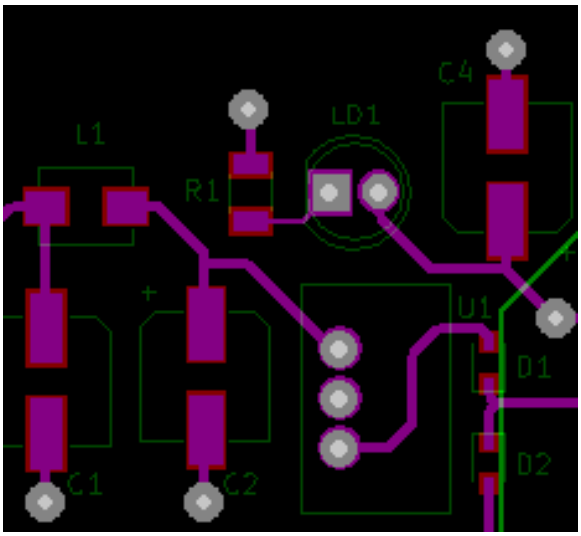
PRINTED CIRCUIT BOARD DESIGN SEMINAR



PCB Design Tutorial

Vicente Jiménez

Document History

Date	Version	Responsible	Description
22/12/2017	1.0	V. Jiménez	First Version
1/2/2018	1.1	V. Jiménez	Small errata corrections Updates for library changes in KiCad 4.07
12/9/2018	2.0	V. Jiménez	Update for KiCad 5.0
18/1/2021	2.1	V. Jiménez	Update for KiCad 5.1.8

Contents

1. Introduction 6
1.1 Nomenclature 6
1.2 Lab work 7
1.3 PCB design basics 8
2. The Initial Design Review 10
2.1 Functional operation of the circuit 11
2.2 Circuit schematic 12
2.3 Component information 13
2.4 Mechanical information 13
2.5 Other miscellaneous data 13
3. Schematic Capture 17
3.1 Schematic Basics 17
3.2 Introduction to Eeschema 19
3.3 Adding labels 34
3.4 The MCU sheet 35
3.5 The One Wire sheet 40
3.6 The USB-Debug sheet 42
3.7 The Comm sheet 43
3.8 The Outputs sheet 45
3.9 Electrical Rules Check (ERC) 48
4. Pre-Layout Review 53
4.1 Mising connector footprints 53
4.2 Button footprint 68
4.3 The inductor footprint 68

4.4 Updating the schematics 70
4.5 Naming power nets..... 71
4.6 Netlist generation 72
4.6 Bill Of Materials..... 73
5. Drawing the layout..... 76
5.1 KiCad layers..... 76
5.2 Initial configuration..... 77
5.3 Board limits 81
5.4 Fixed elements placement..... 82
5.5 Placing the rest of components 87
5.6 Define a routing strategy..... 91
5.7 Routing the power supply 93
5.8 Thermal dissipation on U2..... 97
5.9 Routing the clock 100
5.10 Routing MCU power pins..... 102
5.11 Routing the differential lines..... 102
5.12 Last critical tracks 107
5.13 Routing the rest of lines 107
5.14 Drawing the power planes..... 109
5.15 Cleaning up..... 114
6. Post processing..... 118
7. Optional work..... 121
7.1 Two layer conversion..... 121
7.2 Milled board 124
7.3 Design for hand soldering..... 129
8. PCBD Workflow in brief..... 132
Appendix A: System Description 134

Appendix B: Mechanical data 138
Appendix C: Fabrication capabilities 139
References 140

1. Introduction

This document provides a PCB design tutorial to be developed as laboratory work on the Printed Circuit Board Design (PCBD) Seminar. Although designed for a seminar, you can also use the tutorial on your own.
The tutorial will guide you to all the stages of a PCB design. The tutorial is designed to be carried out using the KiCad software Electronic Design Automation (EDA) suite version 5.0. It would probably be adequate to any future version of KiCad. It can also be followed with any other PCB design suite but the tool methods will differ and some operations would probably need to be performed in a different order.

This document should be included inside a ***PCBD_Files.zip*** file with an optional version number. You need the other contents of this file to easily follow this tutorial.

1.1 Nomenclature

Along the text you will find several special text boxes with special markings. Some indications only emphasize a part of the text:

Information box.

Gives some information that complements the main text.

Watch out warning.

Indicates some consideration to watch for that can give problems if not taken into account.

Some text boxes are related to work that can or must be performed by the student:

Homework.

Work to be performed at home before reaching this point of the document inside the lab.

Laboratory Work.

Work to be performed on the laboratory or at home using the CAD tools.

Finally, there is an especial kind of text box that indicates that the current work needs to be checked by the lab professor:

☑ **Check point for the seminar**

Place where you need to notify to the lab professor
On ETSETB Lab sessions work should not continue until the work is verified by the lab professor.
Some check points require you to upload data to ***Atenea***.
If you are using this document outside of the seminar, you don't need to bother about checkpoints.

Disclaimer on figures

This document has undergone several edit iteration using different versions of KiCad and different versions of the demo design. Some figures in the document are not updated in every new version. That means that sometimes, the images you see on the KiCad application do not exactly match what you see on the document figures. Don't expect the symbols and footprints always match what you see in this document.

1.2 Lab work

This document will guide you through a KiCad PCB design tutorial. The work with this application will start at section 3.2. All previous sections give the necessary context information for the work and contain several ***homework*** tasks you need to carry out before start working on the KiCad application.

The work will be centered on designing a medium complexity four layer PCB. This is the baseline ***Base PCB design***. After completing this work you can carry out several optional modifications to learn new methods. They are all included in ***section 7 "Optional Work"***. That includes a two layer conversion on section ***7.1*** and a design modification tailored to PCB fabrication using a CNC mill on section ***7.2***.

The optional work also contains a ***7.3*** section that explains how to design the board so that eases ***hand soldering*** of the components. PCB design shall take into account the assembly of the board and the soldering method used is an important input for the work. As hand soldering affects all the design process, you would need to start over if you want to test this option after you have ended the design. That's why, if you really want to proceed with the 7.3 section it is better to read it in advance to the lab work and carry it out in all the base design so that you will end up with the baseline and manual soldering option in one go. Note that, as you will see if you read the 7.3 section, you will need to do more work.

1.3 PCB design basics

The design of a PCB, depending on its requirements, can be an easy fast process or a very long complex work. Either way it must follow a defined workflow. Although the PCB design workflow is quite universal, it is not written in stone. Different engineers or companies follow different workflows. But you must have a workflow. The reason is easy to understand. Designing a PCB can be very complex and following a predefined workflow guarantees that you don't leave out anything important. Moreover, following a workflow also prevents that you need to redo your previous work on later stages of the design.

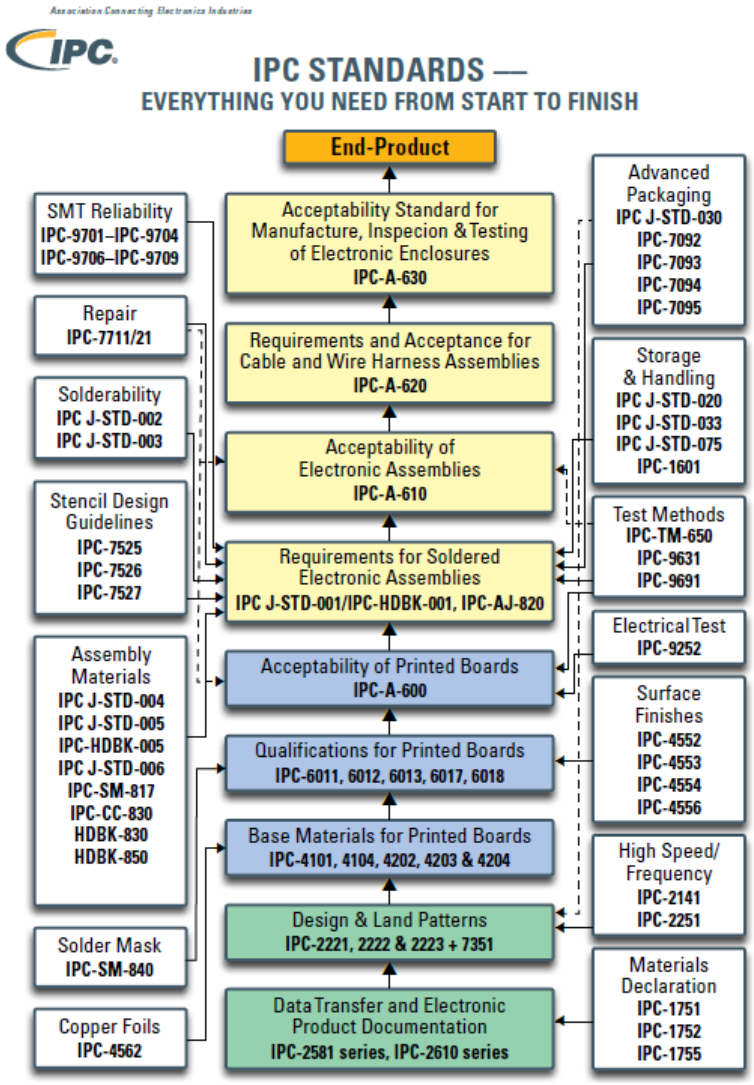
In this tutorial we will follow our own PCBD workflow. It is somewhat based on the workflow defined on the PCB Designer's Handbook released by Optimum Design Associated. You can obtain a copy following the references at the end of this document.

The PCB design workflow starts with a circuit specification and ends with all the information needed to fabricate the PCB. Due to the amount of decisions a PCB designer must take, a lousy specification will provide a lousy PCB. Moreover, a lousy specification will require future decisions that can compromise previous work, so it is important for the designer to know in advance that he has all the needed information.

As in all time consuming workflows, making mistakes can be expensive if undetected. Selecting a bad footprint, for instance, can be undetected across all the workflow until the boards are fabricated. Correcting the error when the boards are at the assembly stage can be costly due to the time needed for the rework, the missing of the fabrication deadlines and the cost associated to the of the fabricated bad units. That's why you need to be sure that the work has no errors. That can be done in two ways: checking and automation. Most errors are human errors, when a task can be automated, most error sources are eliminated. When an automatic procedure can generate errors, there is always an automated check available that can prevent those errors to become unnoticed. For tasks that cannot be automated, we need to detect the possible errors that can creep in and include specific tests to check against those errors.

The fabrication of electronic goods is a mature technology. In order to guarantee proper reliability and quality control most processes have known standards. IPC, the Association Connecting Electronics Industries, is a trade association whose aim is to standardize the assembly and production requirements of electronic equipment and assemblies from a functional point of view. This association generates standard proposals for all stages on the fabrication of electronic devices.

That means that nearly every question you can ask about PCB design has a response included in one of the IPC documented standards. They are not free, however, as the IPC needs to finance the work it does. The following figure shows a summary of IPC documents.



If you work in a company on the electronic goods market, it should have the proper documents to guide you in most decisions about the PCB design.
Inside this document we will sometimes make references to some of the IPC standards.

In general the IPC standards are not mandatory. They are just a way for you to benefit from the expertise of the industry people that write the documents. They also serve as a common reference to define acceptance criteria for products.

Note, that depending on the kind of product your PCB is associated to, there can be other standards that are mandatory that are associated, for instance, to EMC or safety regulations. If your PCB will end up inside a product sold to consumers, it is crucial to know and comply with any mandatory regulation.

2. The Initial Design Review

DESIGN PHASE 1

The first stage on the PCBD workflow is the Initial Design Review. At this stage, the designer makes its first contact with the system associated to the PCB he will design. The more information the designer has about the system, the better decisions the designer will make. That makes ideal that a circuit designer also designs the associated PCB circuit. Unfortunately that is not usually the case. There is a limit on the number of things a person can be expert on. For easy designs it is normal for an engineer to design its own PCBs, but for complex designs, the work must be usually shared between several engineers.

From this stage, the PCB designer must obtain the following information:

- Functional operation of the circuit. What it does and how does it work.
- Circuit schematic. Which components include and how are they connected to each other.
- Component information. Exact information about each component. For most components that means having the component datasheet available.
- Mechanical information. Information on components whose location is fixed. It also includes PCB size constraints.

There is other miscellaneous information that is convenient to consider during this phase of the PCB design as we will see later.

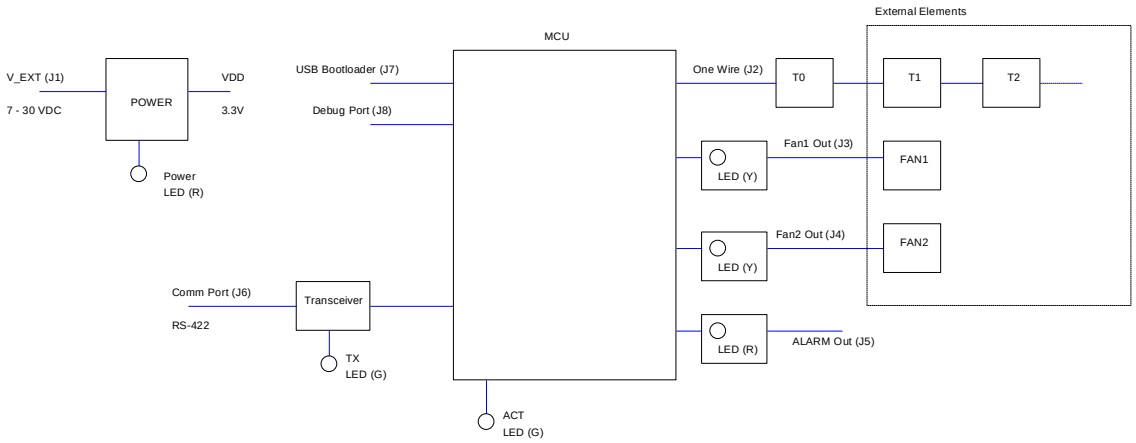
Design disclaimer

This document is a tutorial about PCB design.
The circuit we will implement is only an example and it should not be used as a reference in any way. It does not pretend to be a properly designed circuit as its sole purpose is to define a set of components to lay down on a PCB.
The final obtained PCB is also not expected to suit any specific purpose.

2.1 Functional operation of the circuit

This stage of the initial design reviews enables the PCB designer to know what the purpose of the circuit is. That way, the designer can detect whose are its critical elements.

In this tutorial we will design the PCB of a temperature control system built around a 32 bit microcontroller. The block diagram of the system can be seen on the figure below.



The system is connected to a set of one wire temperature sensors. One sensor, T0, is included on the PCB itself and senses the circuit temperature, the rest of the sensors T1, T2... are external to the PCB and are connected to the one wire bus by a dedicated connector J2.

Using information from the temperature sensors, the system can actuate on two fans to control the temperature of two external devices. Those fans are controlled by two digital lines, each one associated to one connector (J3 and J4). In order to have a visual indication of the estate of the fans, each fan is associated also to a yellow LED indicator. The system also includes a green activity LED blinks each time the system performs a measurement so that we know that it is operating as expected.

In the case that the system is unable to control the external elements temperatures, an alarm signal will be generated on connector J5 that will shutdown the external system to prevent any hazard. That alarm condition will also be signaled by a red LED.

In order to monitor the operation of the controller, it is connected to a central PC using a half duplex RS-485 link associated to connector J6. A green LED is included that blink when there is an active communication.

The system includes a USB port J7 that can be used to download a new firmware on the controller. In order to do that, the system uses the bootloader libraries included on the microcontroller.

During the development stages of the system, it is advisable to have information about the internal operation of the microcontroller. For that, we include also a debug port available on connector J8. As this port is only useful during the system development, it is only available when the system case is open.

In order to power the system, a switch mode power supply generates the internal 3.3 V supply voltage from an available 7 V to 30 V DC feed line.

Finally, the system also includes a reset button, not shown on the schematic, that enables it to restart the operation after an alarm condition or a system malfunction.

2.2 Circuit schematic

Appendix A of this document includes the complete schematics of the system to design. As is usually the case of complex system, the schematic is divided in several pages. In our case we have the following pages:

- Power Supply
- MCU that controls all the system
- USB and Debug connectors
- One Wire interface
- Communication link to an external PC
- Outputs for the fans and the alarm

Note that most I/O lines are protected with metal oxide varistors (mov) elements. This is the minimum protection required against ESD hazards. This is not, however, a fully protected industrial design. It also doesn't feature any specific electromagnetic compliance component. A real design will require more protection elements.

Although the PCB designer is not always the same engineer that draws the schematics, it is important that he fully understands them. This is vital so that he can take the correct decisions during the PCB design. Any lack of knowledge about the circuit operation should be solved during the Initial Design Review stage. In some cases, the feedback provided by an experienced PCB designer will require a modification of the circuit schematics.

Take some time to examine the schematics of the system. You should be able to understand what each component is used for. If you have any question, contact with the lab professor. You can also ask for schematic modifications if you find that you have good enough reasons for the proposal.

About the figures on this document

The figures of the schematic you will find in this document are only approximations to what you will see on the screen. The main source of information is the schematics on **Appendix A**, not the figures scattered inside this document.

2.3 Component information

All the datasheets of the components are provided in the Datasheet folder of the **PCBD_Files.zip** file. This file, with an optional version number, should be available on the same location where you found thins manual. A typical work of the PCB designer during the Initial Design Review is to find the component datasheets. In this case, we provide them to you to reduce the time requirements for the project and to mimize errors in the component footprints.

Components should have a detailed enough description so that the PCB designer knows exactly the package used and can select a proper footprint.

The PCB designer also need to know any specific component requirement like thermal dissipation requirements. For this project we suppose that natural convection is enough for all components without needing any specific heatsink, but in other circuits it is usual to have specific thermal requirements for some of the components.

Review the component information. Check that you know the footprint requirements for all the components in the system.

2.4 Mechanical information

The system case is being designed at the same time that the PCB, so all external connectors J1 to J7, all LED indicators, and the reset button must be on fixed locations. We will call those components ***fixed components***. Moreover, both the board size and the mounting holes are also specified.

The board will be assembled by automatic pick and place machines. That requires the presence of, at least, two fiducidals marks on the component side of the PCB.

Appendix B of this document shows the mechanical information requirements for the board. The PCB designer should usually verify that there is no lack of mechanical information. It his demo project, however, we will directly assume that there is no error at all.

Note that the mechanical information is only two dimensional. There is no information about the vertical (perpendicular to the PCB plane) requirements of the system enclosure. The PCB designer usually needs to check that there is no conflict between the components and the enclosure, but in this project we will relax the checks and we won't perform any test on the vertical requirements.

We won't also take into account panelization or tooling requirements for the board. There won't be also any tie down requirement for components to prevent vibration. Nor there will be any RF shielding or high voltage isolation requirements.

2.5 Other miscellaneous data

In this final part of the Initial Design Review we will try to define some other miscelaneous information:

- Component sides
- Assembly method
- PCB Stack-up
- Special layers used
- Fabrication capabilities

All this information is not always defined on this initial review. A design can be performed one way only in the easiest cases. It is quite usual to modify a previous requirement when we see, later in the design, that it is difficult to achieve. Any late modification, however, can be costly both in time and money, so you should always do your best to prevent backtracking on the design.

Component sides

The higher the density of components in a board, the more difficult it is to route it. For very high component densities, you usually need to use at least a four layer board with two routing layers. That way, tracks on the component side are only used to fan out the signals trough vias to the routing layers.

For yet higher density designs you can put components on both sides of the PCB. In some special cases, as capacitor decoupling on high speed BGA components, using both sides for component placement is a requirement regardless of component density.

You will see that our mechanical requirements give us plenty of space in the board for all the components. As we don't have any specific requirement to place components on the bottom side, we will only place components in the top side of the PCB.

Assembly method

There are several methods to assemble the components on the board. We will consider that the board will be assembled by putting solder paste on the top side landing pads. Placing the top side SMD component using pick and place and soldering them by reflow of the solder paste.

After SMD components are in place, through hole components will be manually added ad the will be soldered using wave soldering.

As no SMD component will be wave soldered, we can use the standard package footprints provided in the CAD software.

PCB Stack-Up

The PCB stack-Up is one of the important decisions to be made during the PCB design. That defines the number of copper layers, their thickness and the thickness of the isolation layers that separate the copper sheets.

In our case we will use a four layer stack-up. We will use both the top and the bottom layer for routing and the two internal layers for Vdd and GND. The stack-up will be:

Layer	Thickness
Solder Mask Top	20 μm
Copper Top	35 μm
Prepreg	130 μm
Copper Inner 1	17 μm
Core	380 μm
Copper Inner 2	17 μm
Prepreg	130 μm
Copper Bottom	35 μm
Solder Mask Bottom	20 μm

Note that the full set of layers gives us a board with a thickness of about 0,8 mm.

Top and bottom copper layers are ticker than the internal layers because they include the copper deposition that metalizes the via holes of the board.

From an EMC and signal integrity point of view, it will be better to use the internal layers for routing and the top and bottom layers for grounding. In our case we have selected the outer layers for routing because that eases any future rework we could need to perform on the board.

If the design used higher frequencies, it will be better to use the outer layers for grounding so that the internal signals are shielded from the outside.

Special layers

The basic set of layers we will need to design are the four copper layers and the two solder mask layers that provide access to the landing pads on the top side for SMD components and the bottom side for through hole (TH) components.

In our design we will include an additional silk screen layer on the top side. To reduce costs, we won't use any silkscreen on the bottom layer.

As this is a first demonstration design, we won't consider any other special fabrication layer during the PCB design.

Fabrication capabilities

Every PCB fabrication process is related to a set of fabrication capabilities. An example is the minimum copper track width and the minimum track separation. Going beyond the fabrication capabilities has an important impact on the board reliability.

If minimum track width is 0,2 mm, defininig a track of 0,1mm can yield to voids in the track. In a similar way, if minimum track separation is 0,2 mm drawing two track separated 0,1 mm from each other will probably yield shorts between the tracks.

Moreover, as the PCB manufacturer always checks that you are with the limits. He will probably refuse to fabricate the board if you exceed them.

Sometimes, one manufacturer provides several sets of capabilities at different prices. As expected, the most demanding capabilities are the most expensive.

In general it is best to give some margin to the capabilities limits of a process. If minimum track width is 0,2 mm, use only this width when you really need it and use a bigger value for most of your tracks.

Appendix C on this document list the fabrication capabilities we will use for our project. You should never go beyond those limits and you are recommended to give always some margin to them.

3. Schematic Capture

DESIGN PHASE 2

The second design phase during the PCB design is the schematic capture. In this phase we will introduce the circuit description in the CAD software.

3.1 Schematic Basics

The circuit description includes all its components and the connectivity network that joins them. Each component will be displayed with a symbol. CAD packages provide symbols for most used components. For special components, we will need to provide our own defined symbols. Either way, a symbol is only a drawn shape that includes a set of terminals that represent the places where the component can be connected to circuit nodes.

A resistor, for instance, can be represented with a rectangular shape with two terminals:

Most components feature two or more terminals, but there can also be components with one terminal, like in the case of ground terminals. There can be also mechanical only components that have no terminal at all like in the case of non plated mounting holes and fiducials.

It is important to include in the schematic all elements that you will implement on the final PCB. That, of course, includes all the electrical components like integrated circuits and resistors, but it also includes all mechanical elements like mounting holes.

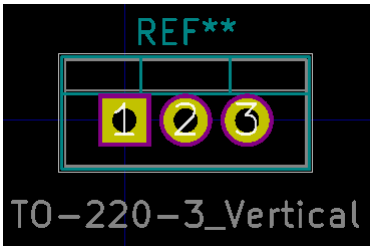
The minimum information about each component in the circuit we need to provide, besides its symbol, is:

- Component reference: Unique reference name that identifies the component, like *J1*. Component references usually include a letter that defines the kind of component followed by a sequential number for all components of a kind.
- Component footprint: Shape of the mounting pads of the component. It is usually related to the component package but it also depends on the assembly method we will use.
- Component connectivity: Node connections between components that define the circuit. That is only basic topological information: which component terminals are joined together in a node. If you have more complex connectivity requirements, like using star or ordered sequential connections in a node, you need to provide that as an specific requirement to the PCB designer.

The component footprint defines the landing pads of the component that enables us to solder it to the PCB so that it features the proper mechanical, thermal and electrical connections.

In the case of SMD components, the footprint defines the shapes of the copper and ***Soldermask*** layers for the mounting side of the component (top side in our design).
In the case of through hole components, the footprint defines the shapes of all the copper and ***Soldermask*** layers on the board.
The footprint usually also includes a placeholder for the component ***reference*** that can be drawn on a silk screen layer and, as we will see later, a placeholder for the component ***value*** that can be drawn on the ***Fab*** layer.

The following figure shows a component footprint of a through hole TO-220 package. The component features three pads (1, 2 and 3) that are replicated on all copper layers and a silk screen shape together with a **REF**** placeholder for the component name. There is also a placeholder for the component value that currently shows the footprint name "TO-220-3_Vertical".



Pad numbers on the footprint are very important because they need to match terminal numbers on the component symbol so that proper electrical connections are implemented on the PCB. We will talk about that in later stages of the design.

The component information in the CAD package will double also as the component database for the circuit. That way we will be able to generate a Bill Of Materials (BOM) from the CAD software. In order to do that, it is also advisable to include other information about each component:

- Component value: Value of the component if the reference is not enough to identify it or when the reference is not already known
- Component reference: Component reference as defined by its manufacturer
- Component manufacturer: Manufacturer that fabricates the component and defines its reference name

It is usual to include more information like second sources for the component and pick and place information for the automatic assembly machines, but for this design we will only use the previously defined component information.

Not all the component information needs to be defined at this stage. For instance, for passive elements like resistors and capacitors, we can define the component name, value and footprint and leave the manufacturer and reference information to be defined later just before the assembly of the PCB.
Beware that it doesn't mean that you can pick random resistors and capacitors as long as their values are matched. There are passive component properties like tolerance, temperature coefficient, dielectric absorption, and so on that need to be taken into account when selecting those components.

3.2 Introduction to Eeschema

As indicated before, we will use the KiCad package on the PCBD lab sessions. KiCad includes a schematic capture application, called Eeschema, that we will use to introduce our circuit information.

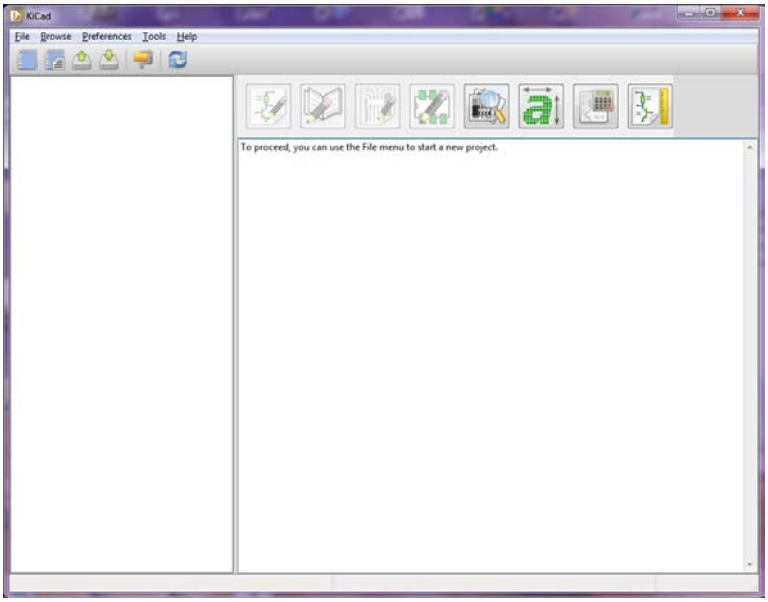
Before running KiCad, you need to create a folder structure for your design. When you log-in in Windows on the lab, a ***L:*** dive is generated that is associated with a network share associated to your user name. Obtain the **PCBD_Files.zip** compressed file (it can have an optional version number) and uncompress it o the **L:** drive on the PC. If you are working on your own, place the PCBD folder somewhere inside your PC.

The folder structure, while working on the lab, will be like this:

- **L:** Drive
  - **PCBD** *(Main folder for the PCBD seminar)*
    - **KiCadDemo** *(KiCad project folder)*
    - **Datasheets** *(Datasheets for the project)*
    - **Doc** *(Document folder with the file you are reading)*

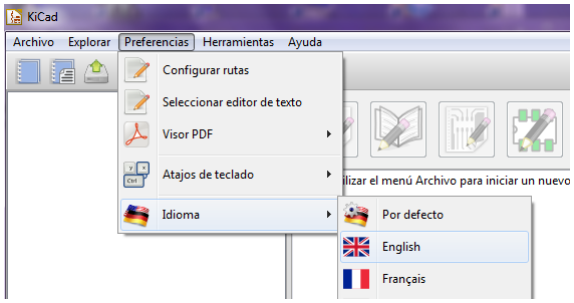
Depending on the version of the **Zip** file, there could be a more complex folder structure, but you should at least get, the folders described above.

Then we will need to lauch KiCad using the icon present on the desktop. You will obtain something like that:

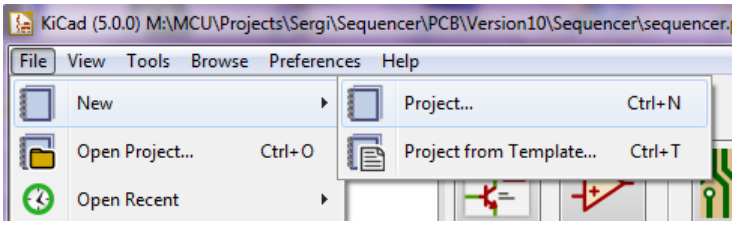


Opening screen on KiCad 4

It is possible that the default language in KiCad is not English. As this document uses the English menu names, we recommend you to set, if needed, the English language. The following figure shows the menu sequence to reach the KiCad language settings when starting in the Spanish configuration. For any other language, the menu configuration is the same although the name of the menu items will be different. Note that the image is for KiCad version 4, in other versions the menus can be different.

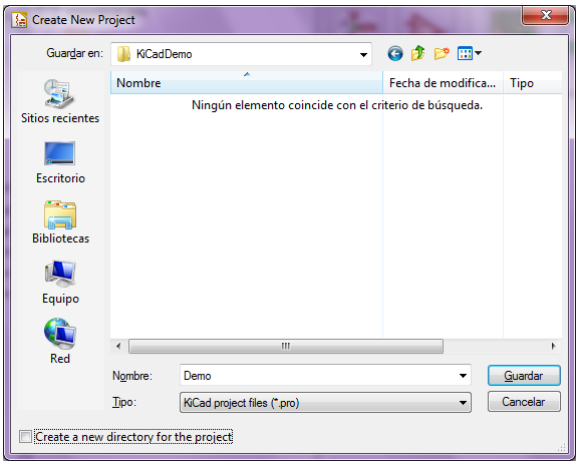


Once we have selected the English language, we can create a new project as shown in the following figure.



Select ***"Demo"*** as the project name and generate it on the previously created ***KiCadDemo*** folder that hangs below the ***PCBD*** folder in ***L:***.

Unselect *"Create a new directory for the project"* as we are using the currently empty *KiCadDemo* folder.

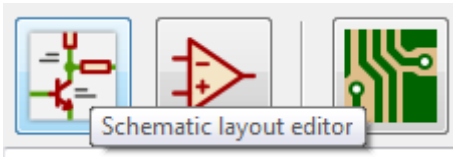


That will create three files inside the ***KiCadDemo*** folder:

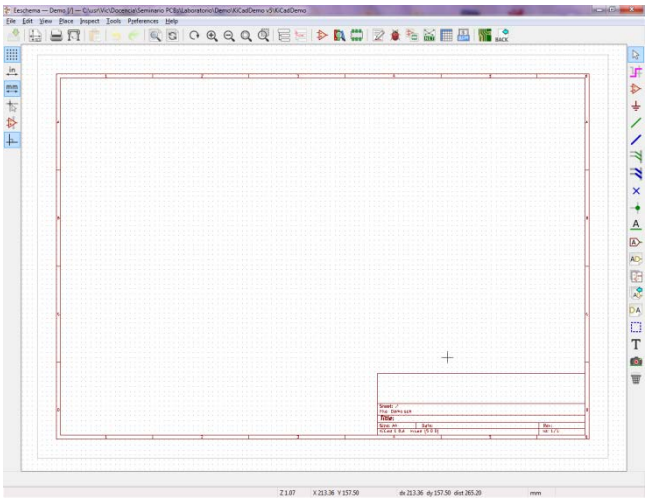
- Demo.pro : Main project file
- Demo.sch : Empty Eeschema schematic file
- Demo.kicad_pcb : Empty PCB file

All of them are text files but you should never edit them with a text editor.

As we need to start the project in the Eeschema application, just start this application using the Eeschema button:



You will open a Eeschema window like the one below:



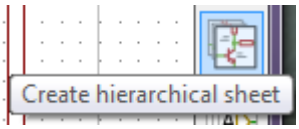
By default you have only one page for the schematic. You can define it to be any size by using the ***File→Page Settings*** menu, but trying to fit all the design in one page is not an elegant solution so we will choose a multipage approach.

Create the requested folder stucture, open **KiCad**, then **Eeschema** from it.
Check the files created inside the ***KiCadDemo*** folder.
Check from the ***File→Page Settings*** menu that page size is A4.

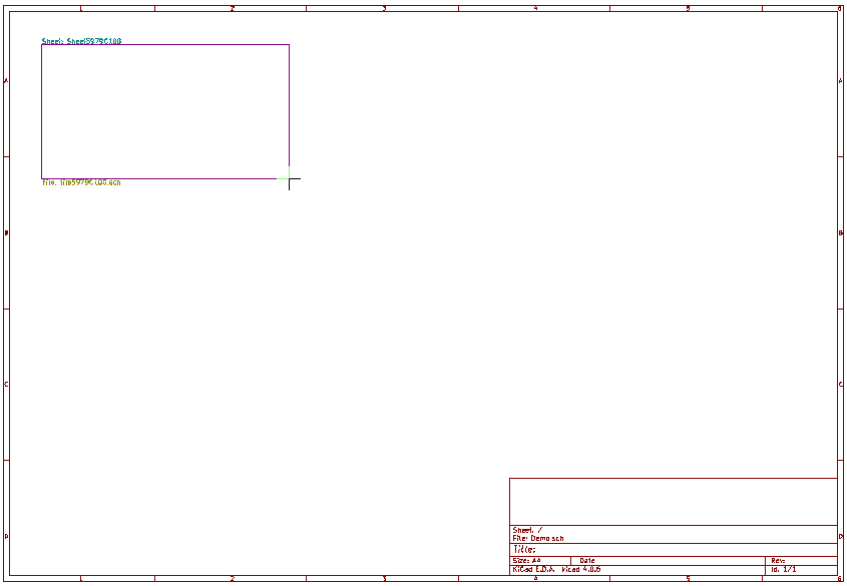
KiCad can use several workflows for multipage designs. We will use the easiest flat multipage approach. Refer to the KiCad documentation to learn about other more hierarchical workflows.

Using the flat multipage workflow we will use a set of schematic pages that have the main Eeschema page as its root. If you recall the Appendix A information, the full circuit is defined in six schematic pages, we will define the same six pages in KiCad to ease the schematic capture of our project.

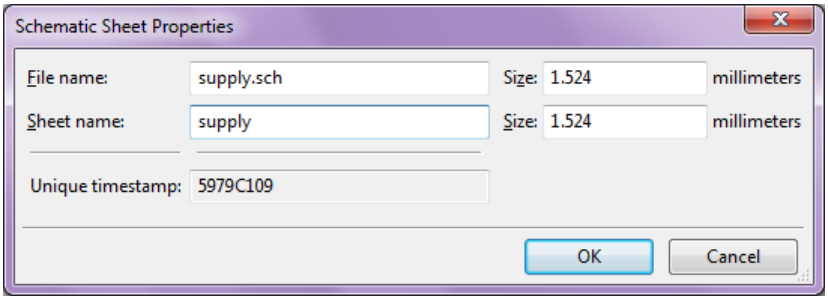
We will start creating a child page of the main one. To do that, select the ***"Create hierachical sheet"*** button on the Eeschema window right side.



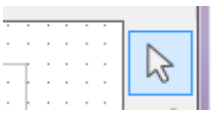
Then click and move the mouse to create a rectangle inside the main schematic page. Draw it small enough so that we can draw, at least, six boxes like that in the main page.



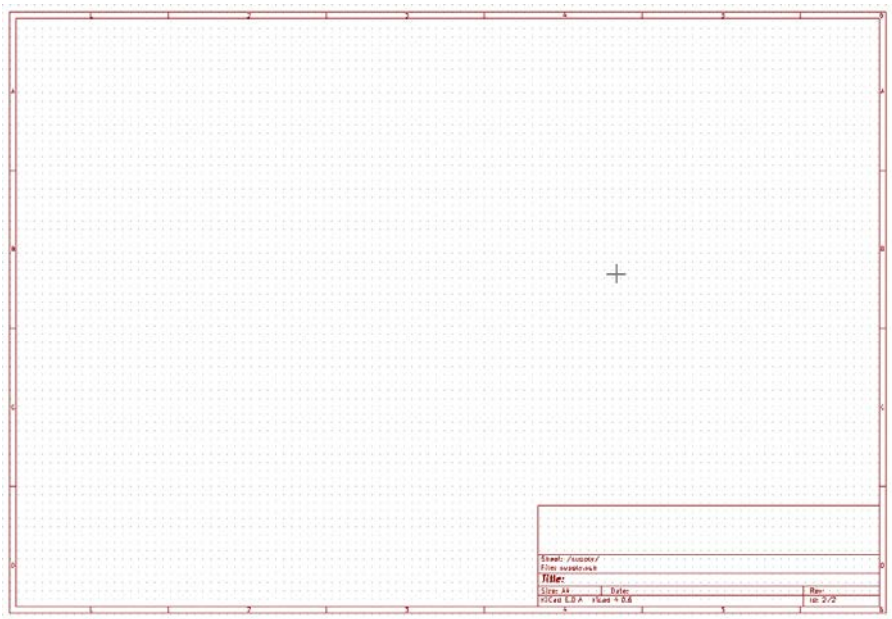
We will define the pages in the same order than in Appendix A, so the first page to define will be the supply. Set the file name to ***supply.sch*** and the sheet name to ***supply*** as shown in the following figure.



In order to edit the newly created ***supply*** sheet, use the **Selection Tool** shown below and double click on the newly created rectangle region.

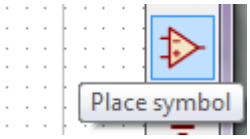


You should obtain an empty schematic sheet like the one in the figure below.

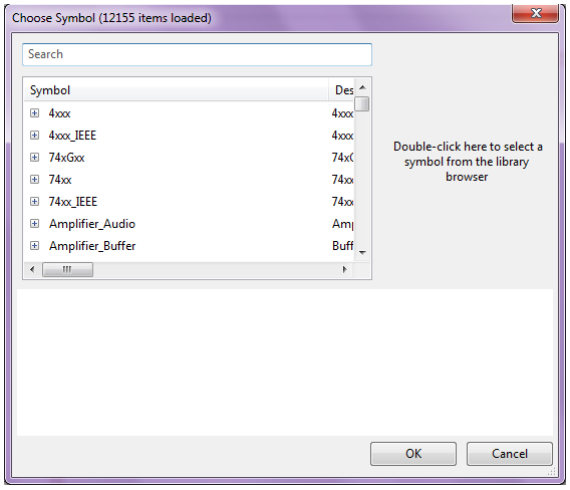


Create and enter the ***supply*** child sheet inside the main Eeschema sheet.

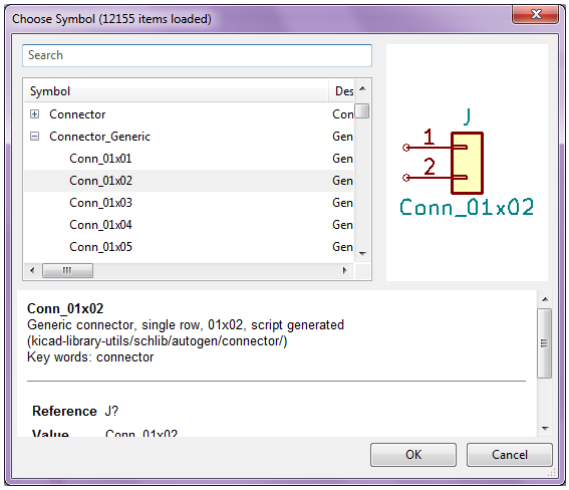
Now, it is time to populate the sheet with components. The first component we will introduce is connector **J1**. First, select the **"Place symbol"** tool.



Then click somewhere inside the sheet to open the component chooser window.



KiCad provides a set of libraries for schematic symbols. You can click on the [+] symbol next to anyone of the library names to expand its contents. The symbol we need is called ***Conn_01x02*** and is defined inside the ***Connector_Generic*** symbol library. You can access it by expanding the ***Connector_Generic*** library or by typing the ***conn_01x02*** name on the filter box in the chooser window.



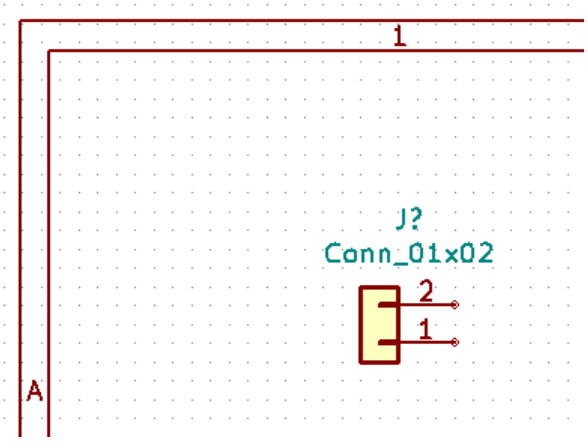
Changes in KiCad Libraries

Libraries can be changed in new KiCad versions. That usually means new component additions but, sometimes, it also means removing old components. It is possible, depending on your KiCad version, that you are not able to find some of the component symbols indicated in this document.

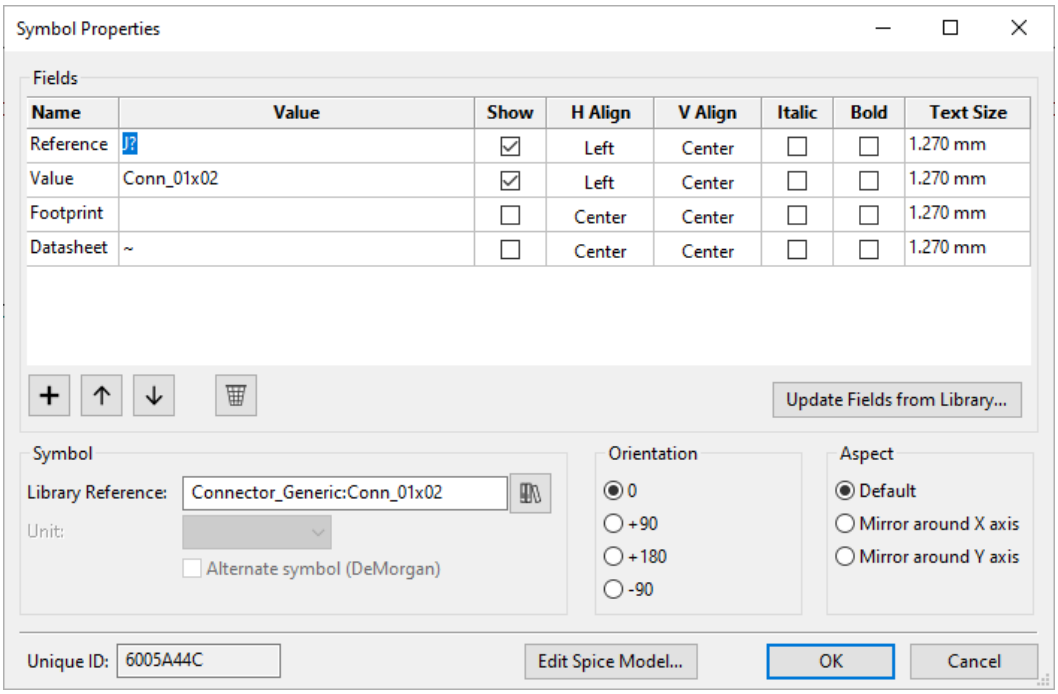
Remember that the only important thing about a symbol is the numbering of its pins. So, if you cannot find a particular two pin connector, you can always use another two pin connector as long as you use the very same use of each pin number.

When you select a symbol inside the component chooser, you will see the symbol shape and a short description. Click ***OK*** after you have selected the ***Conn_01x02*** symbol.

Place the component inside the sheet as shown in the following figure. Note that you need to rotate the component to do that (use the ***"r"*** key while moving the component for that).



After placing the symbol we need to edit its properties. To do that, place the cursor over the symbol and hit the ***"e"*** key (edit) on the keyboard. A window like the one below will open:



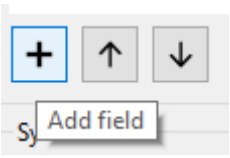
There are currently four fields of information about the component: Reference, Value, Footprint and Datasheet. The ***"?"*** character in the component reference name enables KiCad to autogenerate the component sequence number. As in our project we have fully defined the names of all components, we don't need to use the KiCad sequence generator, so change the Reference field contents to ***J1***.

Change also the name from ***Conn_01x02*** to ***Power***.

KiCad has no footprint available for this component so we will leave for now, as they are, the Value, Footprint and Datasheet fields, but we need to add two additional fields to give a unique universal reference for component.

Connector J1 is fabricated by a manufacturer called "IMO" and defined with a "20.101M/2" reference name.

The first field to add is the component reference name that we will call ***"Name"***. In order to do that click the ***Add field*** button and add a new field of name ***"Name"*** and contents ***"20.101M/2"***.



We will use the same method to add a second field ***"Manufacturer"*** with contents ***"IMO"***. The full set of component fields should, at the end, be like in the figure below:

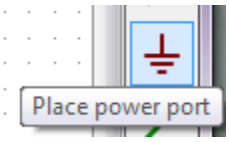
Fields

Name	Value
Reference	J?
Value	Conn_01x02
Footprint	
Datasheet	~
Name	20.101M/2
Manufacturer	IMO

Afterwards we end the component parameters edit by clicking the ***OK*** button.

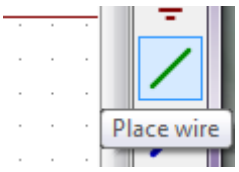
Introduce in the schematic the **J1** component and edit its properties as explained.

Pin 1 of J1 has to be grounded. We can add a ground symbol to the schematic to perform this connection. Select the ***"Place power port"*** tool.

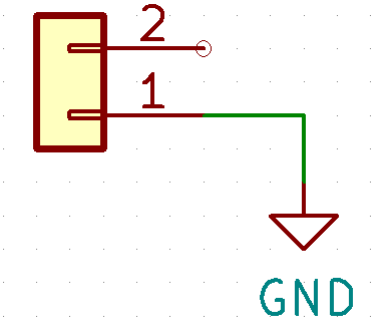


Then click inside the schematic window. You can write down ***GND*** in the filter box to select the desired symbol. Afterwards click in the position, below **J1**, when you want to place this symbol.

Now that we have the connector ***J1*** and the ***GND*** symbol we need to connect them. To do that, select the "Place wire" tool.



Then connect pin 1 of ***J1*** to the ***GND*** symbol. You should a connection like the one shown below.



Now it is time to add more components, in particular: ***F1***, ***L1***, ***C1*** and ***C2***. Use information from the following tables to locate the symbol of each component and place them in the schematic. Remember that you can easily locate a component by its symbol name using the filter box in the component selector window.

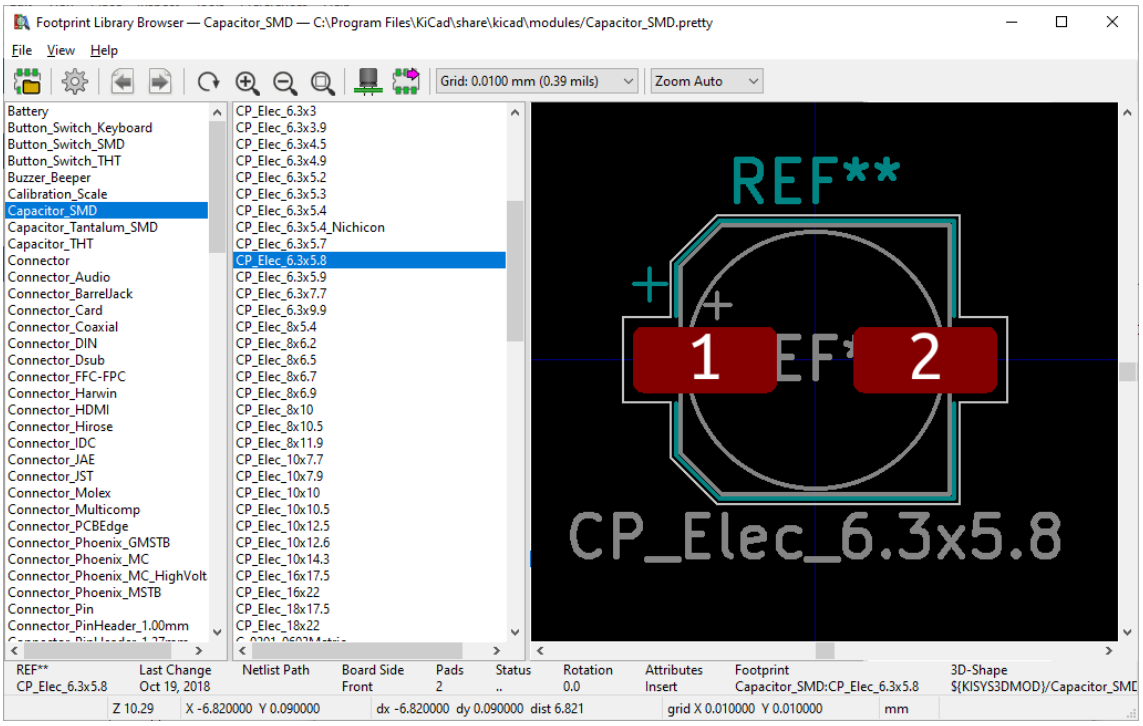
Then use the ***"e"*** key to fill-in the required fields. Remember that fields ***"Name"*** and ***"Manufacturer"*** are custom fields that need to be created. You can leave alone the standard ***"Datasheet"*** field but, if you want, you can fill in this field the name of the associated datasheet pdf file in ***PCBD_Datasheets.zip***.

Reference	**F1**
Symbol Library	device
Symbol Name	Polyfuse
Value	250 mA
Footprint library	Resistor_THT
Footprint	R_Axial_DIN0204_L3.6mm_D1.6mm_P5.08mm_Horizontal
Name	RXEF025
Manufacturer	TE

References	**C1, C2, C3, C4**
Symbol Library	device
Symbol Name	CP
Value	10 uF
Footprint library	Capacitor_SMD
Footprint	CP_Elec_6.3x5.8
Name	EDK106M063A9GAA
Manufacturer	Kemet

Reference	**L1**
Symbol Library	device
Symbol Name	L
Value	5.6 uH
Footprint library	
Footprint	
Name	tck141
Manufacturer	Traco

In order to introduce the footprint information for components ***PF1***, ***C1*** and ***C2***, you need to select the footprint field on the component edit window and then click the ***"Browse Footprints"*** button. It will open a window like the one below that includes three sub-windows. From left to right: Footprint library, Footprint name and Footprint image.



Missing footprint libraries

Depending on how KiCad is installed or on how your project is moved from one PC to other, sometimes KiCad will complain about not being able to find some of the footprint libraries.

This problem can be solved by opening the PCB editor ***PCBnew***. Opening the ***Footprint Libraries Manager*** on the ***Preferences*** menu and then deleting the missing library and appending it again using the ***Append with wizard*** button from the local libraries installed on the PC.

As ***L1*** needs a custom footprint not available in the standard KiCad libraries, we will leave the footprint information for this component alone by now.

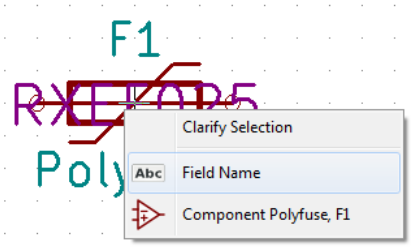
In the case of component **PF1** we want to show its name field in the schematic. In order to do that, edit the component properties, select the ***Name*** field and change its visibility options to visible by activating the ***Show*** element.

Name	Value	Show
Reference	F1	☑
Value	250 mA	☑
Footprint	Resistor_THT:R_Axial_DIN0204_L3.6mm_[	☐
Datasheet	~	☐
Name	RXEF025	☑
Manufacturer	TE	☐

That will probably show the ***Name*** text like in the figure below.

If you want, you can use the selection tool to move the name labels to put them on different locations.

Move the cursor over the component and use the ***move*** command by clicking the ***"m"*** key of the keyboard. If there is some ambiguity, a window will pop-up to clarify the selection.

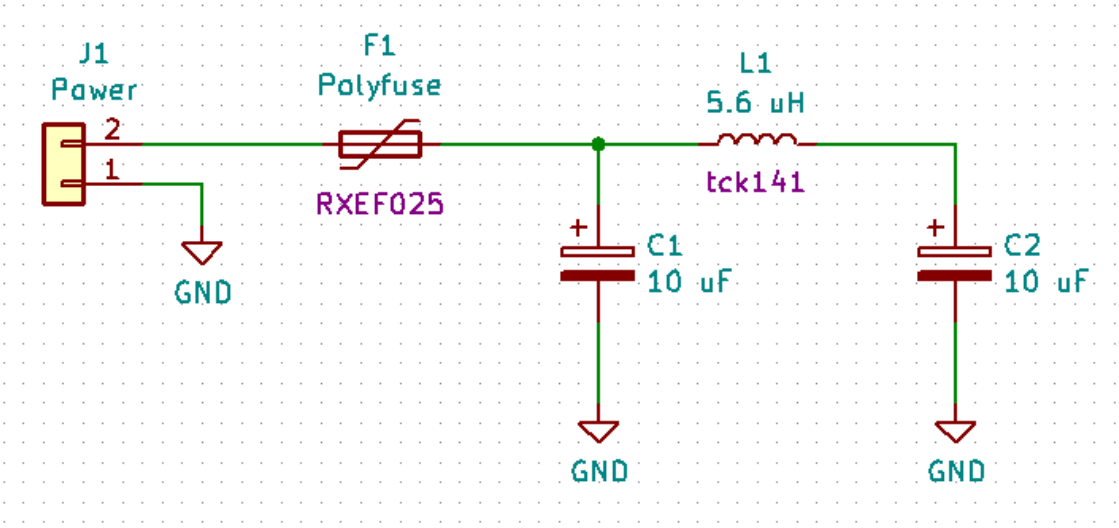


We can, for instance move the Name field below the component as shown in the figure below.

There are no hard requirements on the amount of information you need to show on a schematic. You need to balance the information so that the schematic is useful but is also not crowded with data. In this document we provide some recommendations for the project but they are not hard rules. The main purpose on this demo is that you know how to do things not that you do everything exactly as shown.
We will do the same to show the ***Name*** field also for component ***L1***.

As you see, capacitors ***C1*** and ***C2*** are almost the same. They only are different in their reference names. You can ease your work by first creating ***C1*** and entering all its field details. Then you can use the select tool, put the cursor over ***C1*** and copy it using the ***"c"*** key. That will create a new component with the same fields as C1 but with reference name ***C?***. Just put it in the correct place and edit its reference to ***C2***.

Add to the circuit the components F1, L1, C1 and C2. Add also the needed GND symbols and complete the circuit so you get something like the figure below.

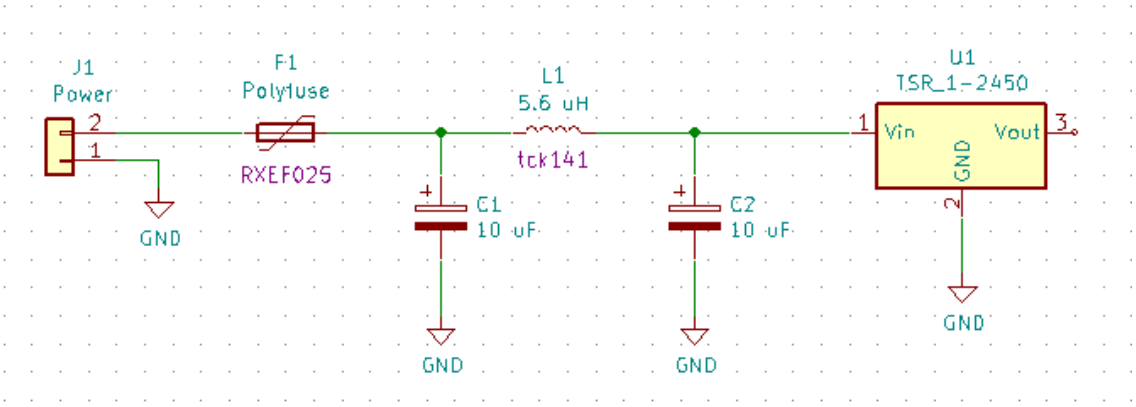


Next we will add the switch mode regulator **U1** that generates the 5V supply. We will use the following data for it.

Reference	**U1**
Symbol Library	Regulator_Switching
Symbol Name	TSR_1-2450
Value	TSR_1-2450
Footprint library	Converter_DCDC
Footprint	Converter_DCDC_TRACO_TSR-1_THT
Name	TSR_1-2450
Manufacturer	Traco

Note also that we are using the manufacturer name of the component both for the ***Vale*** and ***Name*** field. We will do that for all unique components that are not one of a kind like resistors and capacitors.

After adding ***U1*** to the circuit you should get something like:

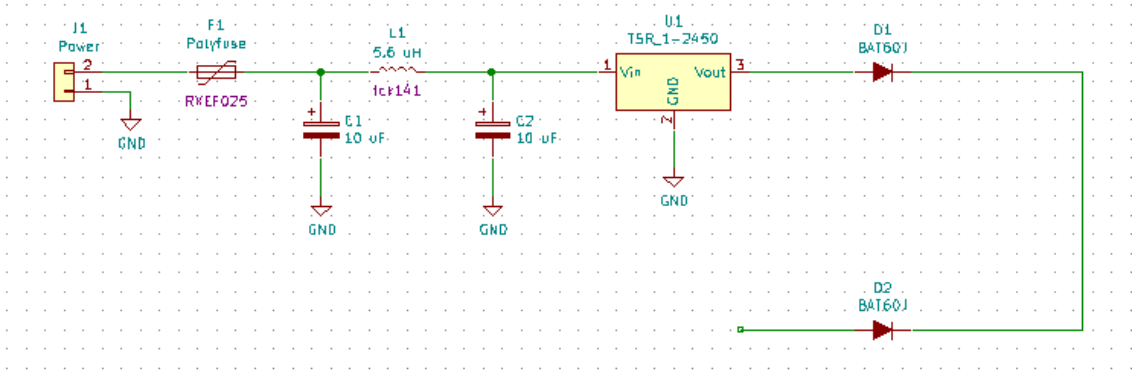


We can now add components D1 and D2:

References	**D1, D2**
Symbol Library	device
Symbol Name	D or D_ALT
Value	BAT60J
Footprint library	Diodes_SMD
Footprint	D_SOD-323
Name	BAT60J
Manufacturer	ST

Observe that you can choose the symbol name between D or D_ALT. The difference is only cosmetic so choose the one you like more.

Remember that we can add ***D1***, fill all its fields and copy it to generate ***D2***. That will get something like in the figure below. Note in the figure that ***D2*** has been copied from ***D1*** but its reference has not yet be modified from ***D?***.



Add D1 and D2 to the circuit.

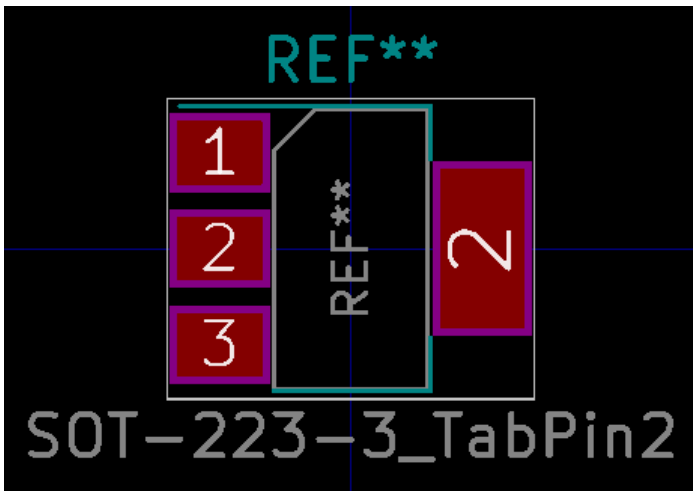
Now we need to add the second regulator U2 that generates the 3V3 supply from the 5V one. This is a UA78M33 linear regulator. This component has 3 nodes but features 4 terminals because its heatsink is soldered to the board in a dedicated landing pad.

We will use data from the following table for this component.

References	U2
Symbol Library	Regulator_Linear
Symbol Name	uA7805
Value	UA78M33
Footprint library	Package_TO_SOT_SMD
Footprint	SOT-223-3_TabPin2
Name	UA78M33
Manufacturer	Texas Instruments

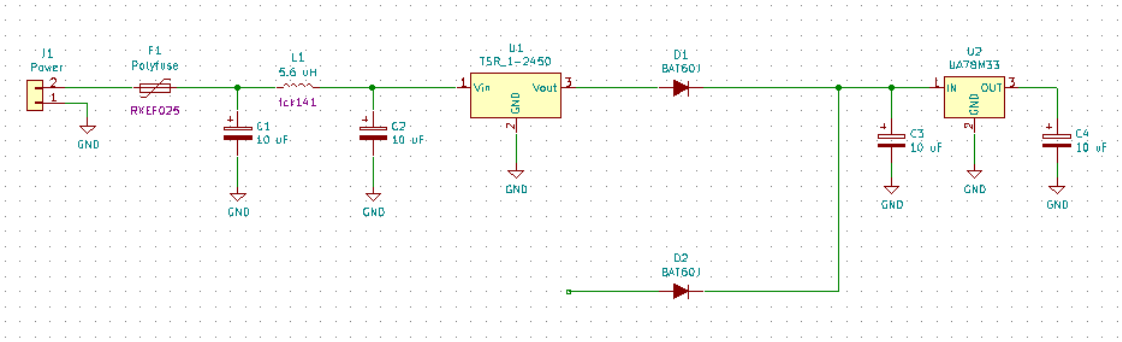
In this case we don't have a specific symbol for the component so we will borrow it from the **uA7805** symbol that also has three pins with the same numbering.

Note that the footprint used, ***SOT-223-3_TabPin2***, has two pins with the same "2" number. This is because both pins are internally connected on the package.



Add also the capacitors ***C3*** and ***C4***.

After adding this component you will have something like:



Draw the schematic so that it matches the above figure.

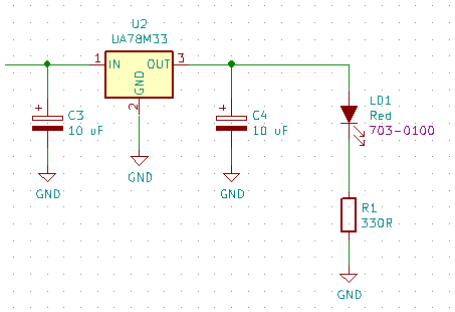
Continuing the circuit we can add the LED LD1 and the resistance R1. The data for those components is:

Reference	**LD1**
Symbol Library	device
Symbol Name	LED_ALT
Value	Red
Footprint library	LED_THT
Footprint	LED_D5.0mm
Name	703-0100
Manufacturer	multicomp

Reference	**R1**
Symbol Library	device
Symbol Name	R
Value	330R
Footprint library	Resistor_SMD
Footprint	R_1206_3216Metric
Name	
Manufacturer	

As you cannot add blank fields, don't add Name or Manufacturer fields to the resistor component.

That will yield the following circuit. Note that we make visible the LD1 component name.

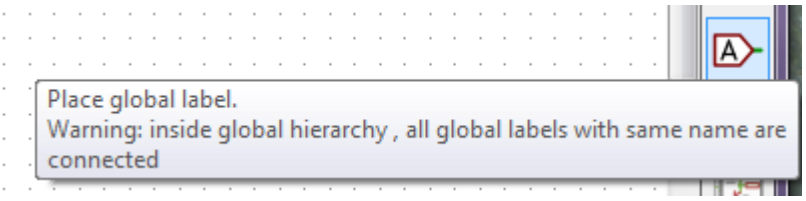


As you navigate the Footprint chooser when editing resistor R1, note that there are specific footprints defined for hand soldering. As explained before, the footprint used not only depends on the component package but also on the solder method. There are optimal footprints to choose for hand soldering, reflow and wave.

Add components **LD1** and **R1** to the schematic.

3.3 Adding labels

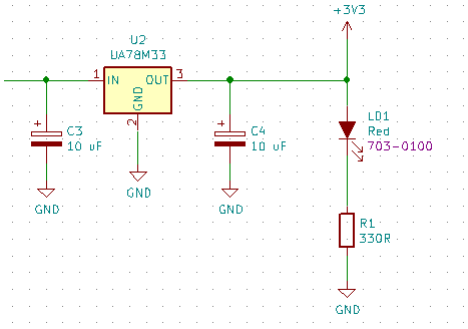
Power port signals like GND are global to the whole circuit. Any other node that should be available, at the same time, in more than one schematic sheet, must be declared with a global label when using the flat workflow. We will use a power port to identify the ***+3V3*** node and a global label ***"V_USB"*** to identify the ***D2*** anode that is connected to the USB connector in another sheet. Select the ***"Place global label"*** tool.



Now click inside the sheet. A ***"Global label properties"*** window will open. Fill the text box with ***"V_USB"*** and set the Shape bidirectional. After hitting ***OK***, move it so that it connects to the ***D2*** anode.

Now place a ***+3V3*** component from the same power library that houses the ***GND*** symbols and connect it to the ***U2*** output.

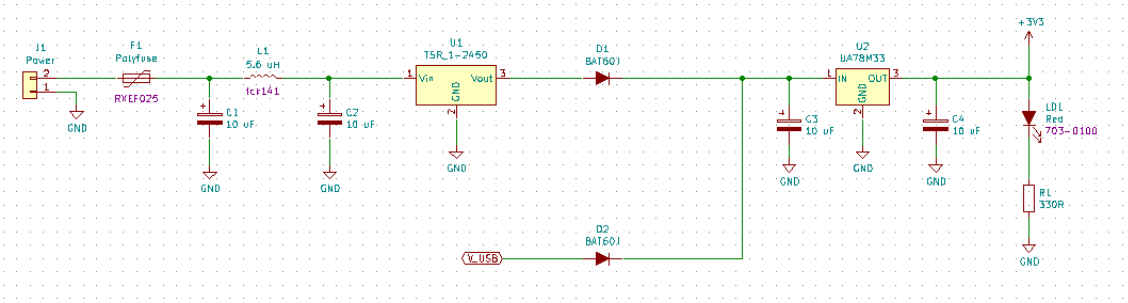
The new added components should show as in the figure below.



Escriba aquí la ecuación.

End the editing of the power supply sheet.

That ends our first schematic sheet.

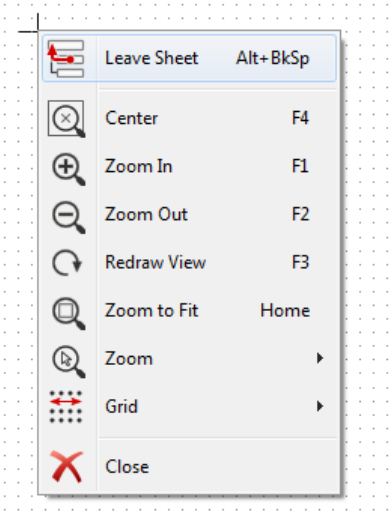


Check #1 : First Sheet

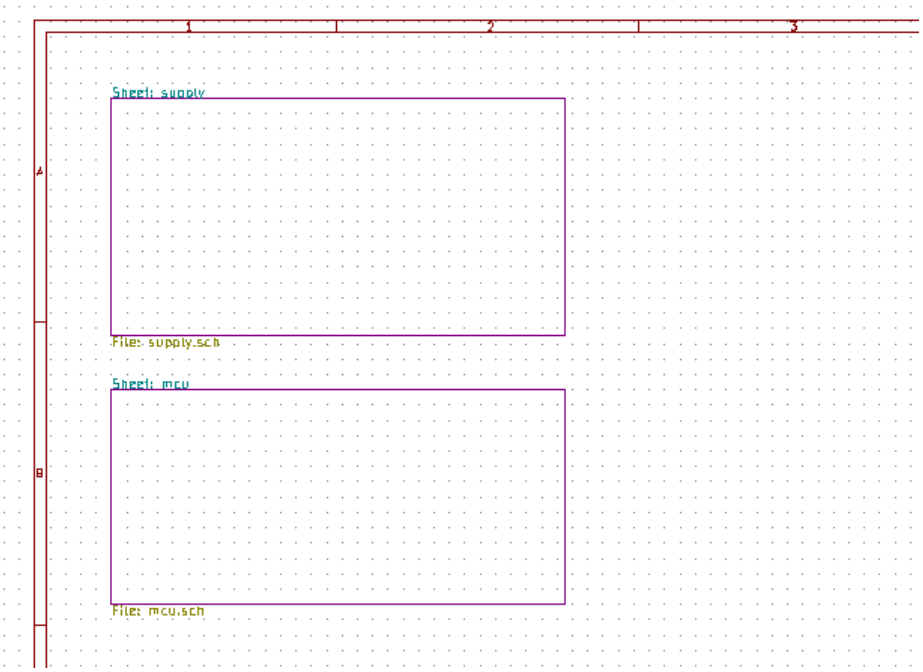
Show this first sheet of the design to the Lab professor

3.4 The MCU sheet

As we have ended editing the power supply sheet, we will continue by editing the MCU sheet. Use ***"Leave Sheet"*** option in the menu that opens after pushing the mouse ritght button to return to the main schematic sheet.



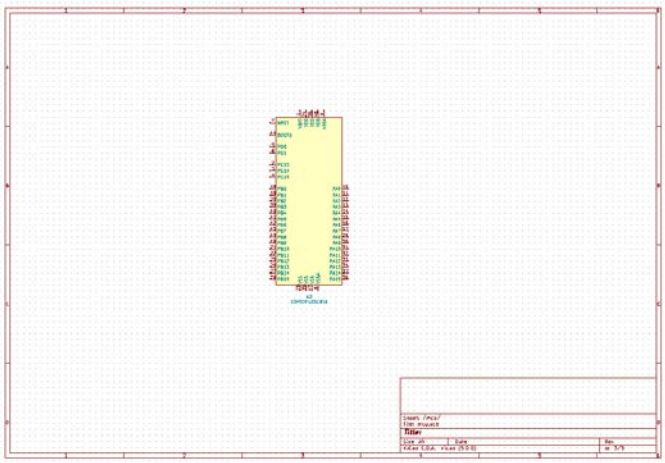
In the main sheet and create a new sheet below the power supply one.
Use ***"mcu"*** both for the name and filename of the new sheet and enter it.



The microcontroller unit (MCU) we are using in this project is the STM32F103CBT6 manufactured by ST. Data for this component is shown on the following table.

Reference	**U3**
Symbol Library	MCU_ST_STM32F1
Symbol Name	STM32F103CBTx
Value	STM32F103CBT6
Footprint library	Package_QFP
Footprint	LQFP-48_7x7mm_P0.5mm
Name	STM32F103CBT6
Manufacturer	ST

Put the MCU at the center of the sheet.



Create the MCU sheet and add the U3 component.

Now we can add the rest of components in this sheet.

References	**All capacitors**
Symbol Library	device
Symbol Name	C
Value	20 pF or 100 nF depending on the capacitor
Footprint library	Capacitor_SMD
Footprint	C_0805_2012Metric
Name	
Manufacturer	

References	**R3, R4 and R11**
Symbol Library	device
Symbol Name	R
Value	R3 4k7, R4 330R, R11 100k
Footprint library	Resistor_SMD
Footprint	R_1206 for R4 and R_0805 for R3 and R11
Name	
Manufacturer	

As there are a lot of capacitors and resistors it is recommended to use the copy method so that you don't need to edit all parameters of each one.

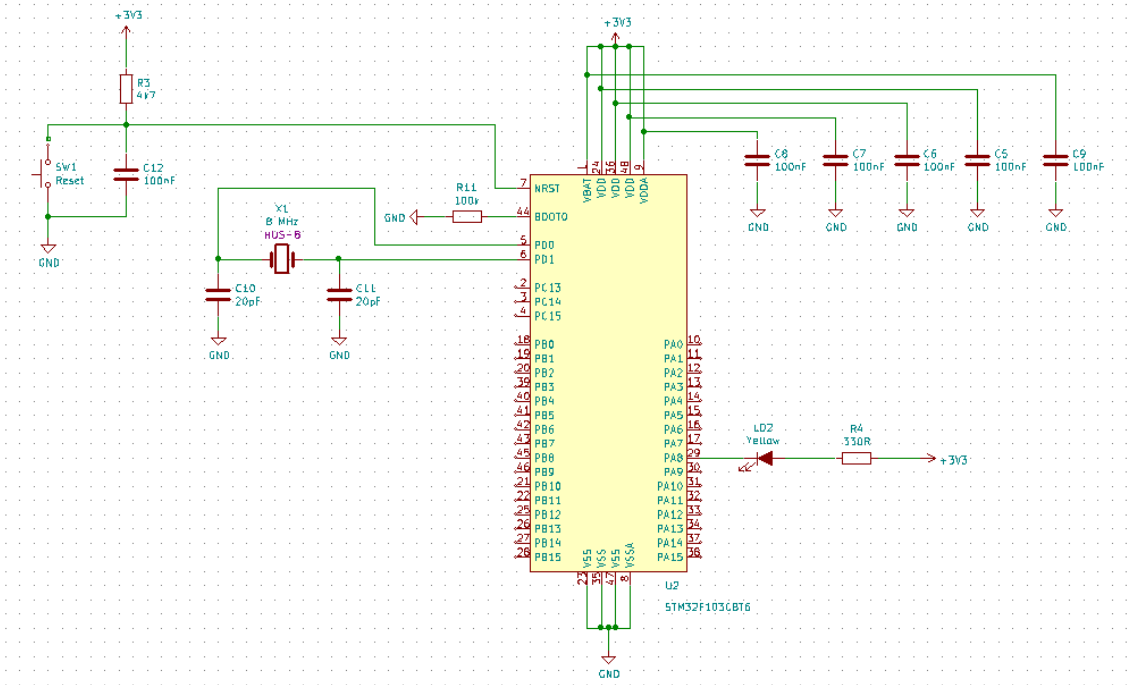
Reference	**X1**
Symbol Library	device
Symbol Name	Crystal
Value	8 MHz
Footprint library	Crystal
Footprint	Crystal_HC18-U_Vertical
Name	HUS-8
Manufacturer	Mercury

Reference	**LD2**
Symbol Library	device
Symbol Name	LED_ALT
Value	Yellow
Footprint library	LED_THT
Footprint	LED_D5.0mm
Name	703-0098
Manufacturer	multicomp

Reference	**SW1**
Symbol Library	Switch
Symbol Name	SW_Push
Value	Reset
Footprint library	
Footprint	
Name	B3F-400
Manufacturer	Omron

The ***SW1*** footprint information is empty because we will need to create a custom footprint in the future.

After adding the components the schematic will be something like:

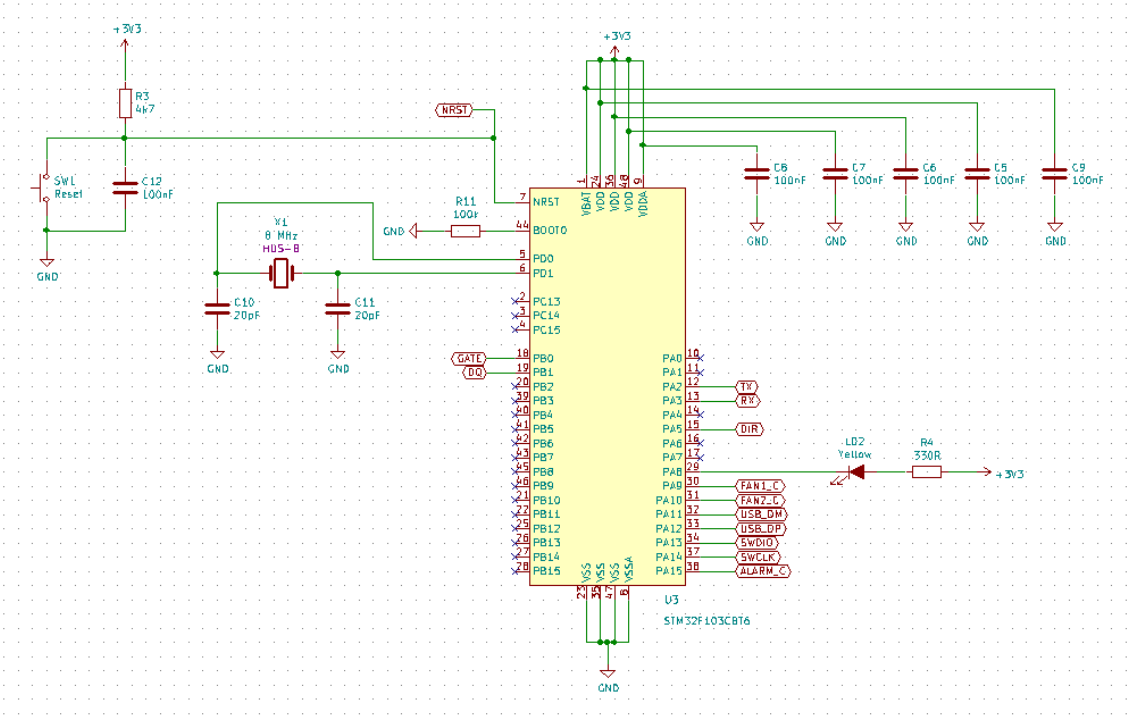


Note that we don't need to draw the decoupling capacitors as in the figure, as all are connected to the **+3V3** node. The schematic is drawn this way to remind the PCB designer that each decoupling capacitor shall be located next to its associated power pin.

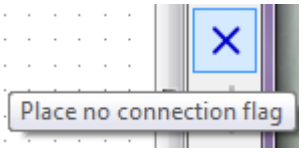
Add the rest of components of the MCU sheet.

Now we will add the global labels to the following nodes: NRST, TX, RX, DIR, FAN1_C, FAN2_C, ALARM_C, GATE, DQ, USB_DP, USB_DM, SWCLK and SWDIO.

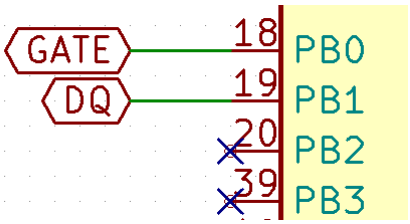
Use bidirectional global labels on all those nodes in order to get to a schematic equivalent to the one in the figure below.



Observe that there are a lot of unused pins on the MCU. Any unused pin needs to be marked as not-connected. Use the ***"Place no connection flag"*** tool for that.



The following figure shows a zoom on the left side of the MCU so that you can see the ***not-connected*** markers on the unused pins.



Add the global labels and the not-connected markers on the MCU sheet.

That ends the MCU sheet.

3.5 The One Wire sheet

Use the ***"Leave Sheet"*** right button menu option to return to the main sheet. Create a sheet named ***"OneWire"*** below the MCU sheet and enter it. Now you know how to add components so we will only provide the component tables.

Reference	**R5**
Symbol Library	device
Symbol Name	R
Value	4k7
Footprint library	Resistors_SMD
Footprint	R_0805_2012Metric
Name	
Manufacturer	

Reference	**M1**
Symbol Library	device
Symbol Name	Q_PMOS_GSD
Value	BSS84
Footprint library	Package_TO_SOT_SMD
Footprint	SOT-23
Name	BSS84
Manufacturer	ON

Transistor M1 uses a SOT-23 package. Not all MOS transistors that use the SOT-23 package have the same pinout. Check the Q_PMOS_GSD footprint against the BSS84 datasheet so that you are sure that you are using the proper footprint.

Reference	**MOV1**
Symbol Library	device
Symbol Name	Varistor
Value	MCFT000215
Footprint library	Varistors
Footprint	RV_Disc_D7mm_W3.4mm_P5mm
Name	MCFT000215
Manufacturer	multicomp

Reference	**J2**
Symbol Library	Connector_Generic
Symbol Name	Conn_01x02
Value	22-05-7025
Footprint library	
Footprint	
Name	22-05-7025
Manufacturer	Molex

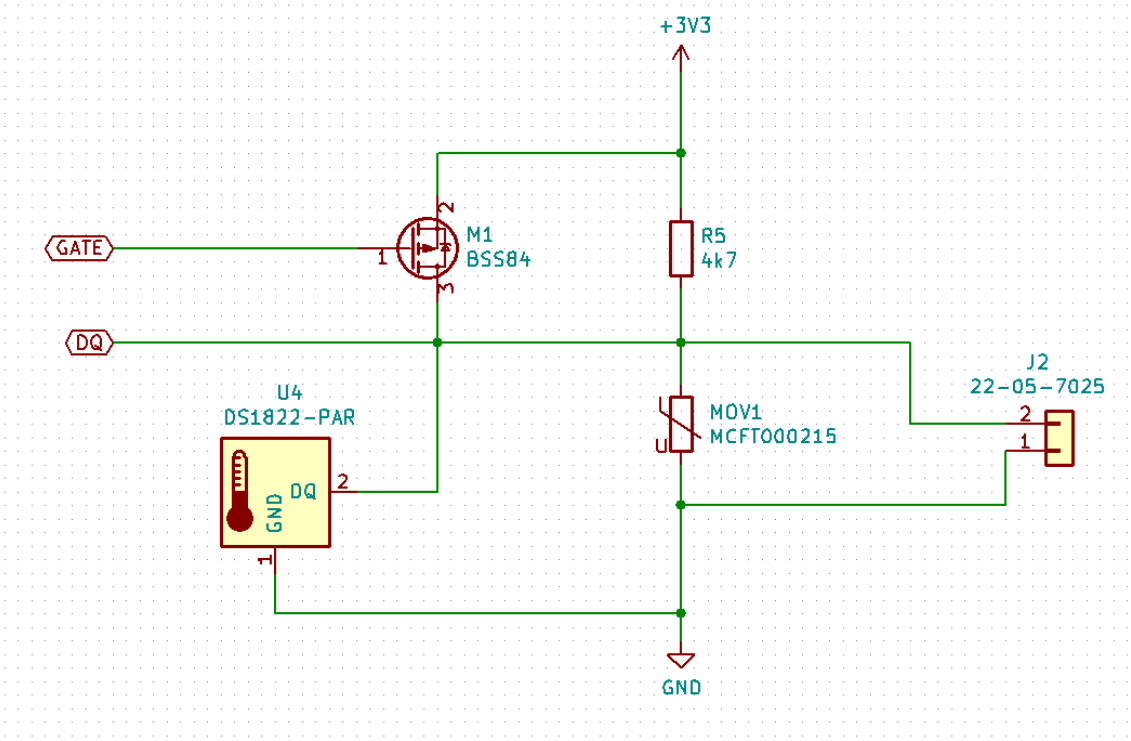
For the **J2** connector remember that the pin connected to GND shall be the number 1. Rotate or reflect as you need to get the proper symbol orientation.

The reason why we are giving the manufacturer name ***22-05-7025*** as the component value is because this component is also used in other connectors. By giving the same name, they will be easier to be grouped together in the Bill Of Materials (BOM).

Reference	**U4**
Symbol Library	Sensor_Temperature
Symbol Name	DS1822-PAR
Value	DS1822-PAR
Footprint library	TO_SOT_Packages_THT
Footprint	TO-92_Inline_Wide
Name	DS1822-PAR
Manufacturer	Maxim

The temperature sensor ***U4*** is included in the KiCad libraries. Remember to use the "PAR" version that gets "parasite" power from the DQ node at pin 2.

Now you can add all the sheet components, connect them and add the global bidirectional labels GATE and DQ. You should get a circuit like the one below:



Create and populate the One Wire sheet as described.

3.6 The USB-Debug sheet

Create a new ***"usb-debug"*** sheet at the right of the power supply one and enter it.

Components for this sheet are:

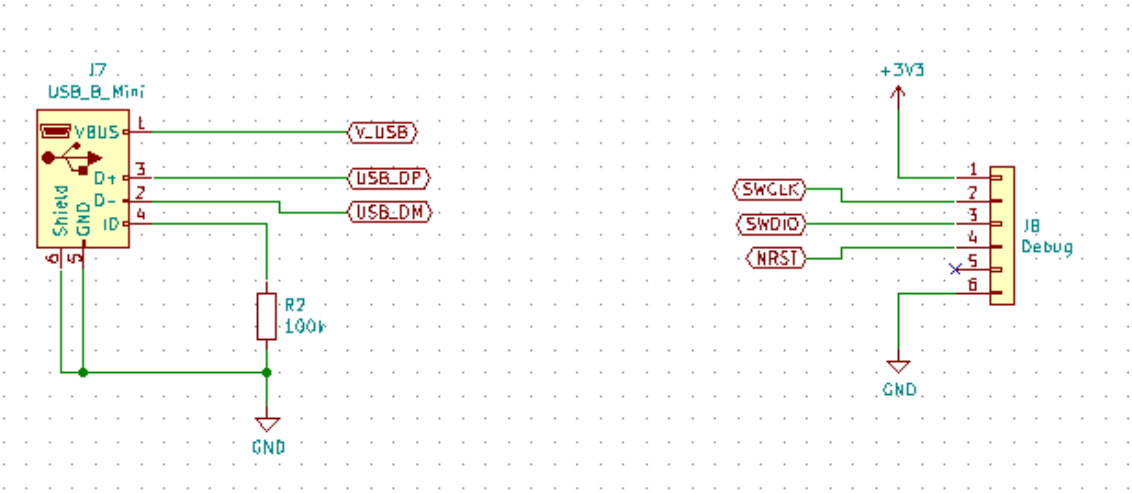
Reference	**J7**
Symbol Library	Connector
Symbol Name	USB_B_Mini
Value	USB_B_Mini
Footprint library	Connector_USB
Footprint	USB_Mini-B_Wuerth_65100516121_Horizontal
Name	67503-1020
Manufacturer	Molex

Reference	**J8**
Symbol Library	Connector_Generic
Symbol Name	Conn_01x06
Value	Debug
Footprint library	
Footprint	
Name	T821106A1S100CEU
Manufacturer	Amphenol

This component requires a custom footprint so don't fill the footprint information.

Reference	**R2**
Symbol Library	device
Symbol Name	R
Value	100k
Footprint library	Resistor_SMD
Footprint	R_0805_2012Metric
Name	
Manufacturer	

Check the pin numbers on connector J8 match this order: Vdd (+3V3) is pin 1 at the top and GND is pin 6 at the bottom. The rest of pins follow the same sequence as in the Appendix images. Global labels are shown in the following image of the completed schematic.



Create and populate the USB-Debug sheet as described.

3.7 The Comm sheet

The next step is to implement the ***Comm*** sheet that implements the circuits that communicate with the PC.
Component information is:

References	**R6 and R7**
Symbol Library	device
Symbol Name	R
Value	R6 330R, R7 120R
Footprint library	Resistor_SMD
Footprint	R_1206_3216Metric
Name	
Manufacturer	

Reference	**C14**
Symbol Library	Device
Symbol Name	C
Value	100 nF
Footprint library	Capacitor_SMD
Footprint	c_0805_2012Metric
Name	
Manufacturer	

Reference	LD3
Symbol Library	device
Symbol Name	LED_ALT
Value	Green
Footprint library	LED_THT
Footprint	LED_D5.0mm
Name	703-0097
Manufacturer	multicomp

Reference	M2
Symbol Library	device
Symbol Name	Q_PMOS_GSD
Value	BSS84
Footprint library	Package_TO_SOT_SMD
Footprint	SOT-23
Name	BSS84
Manufacturer	ON

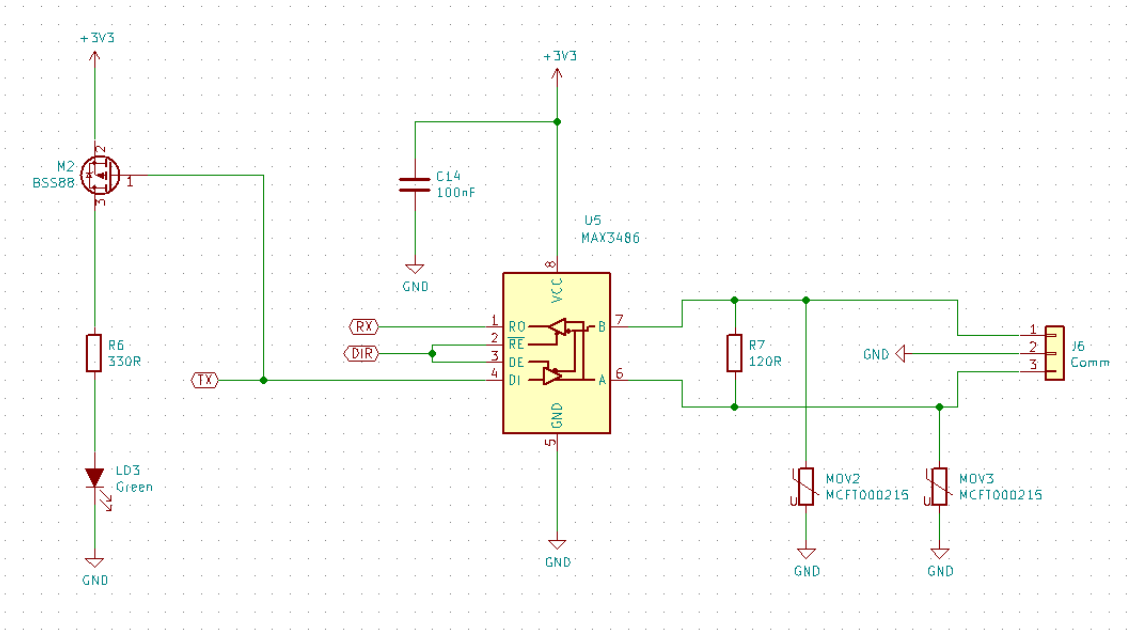
References	MOV2 and MOV3
Symbol Library	device
Symbol Name	Varistor
Value	MCFT000215
Footprint library	Varistors
Footprint	RV_Disc_D7mm_W3.4mm_P5mm
Name	MCFT000215
Manufacturer	multicomp

Reference	J6
Symbol Library	Connector_Generic
Symbol Name	Conn_01x03
Value	Comm
Footprint library	
Footprint	
Name	22-05-7025
Manufacturer	Molex

As ***J6*** will need a custom footprint, leave it black for now.

Reference	U5
Symbol Library	Interface_UART
Symbol Name	MAX3486
Value	MAX3486
Footprint library	Package_SO
Footprint	SOIC-8_3.9x4.9mm_P1.27mm
Name	MAX3486
Manufacturer	Maxim

After you end editing the comm sheet, you should get something like the drawing below:



Create and populate the Comm sheet as described.

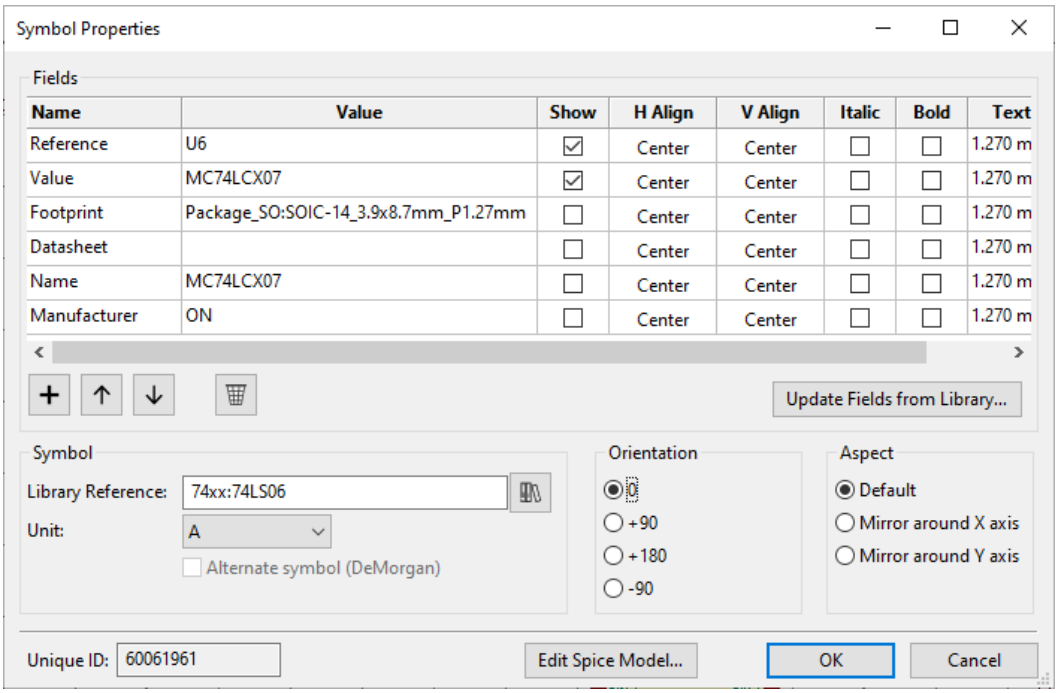
3.8 The Outputs sheet

This is the last sheet on our design. Create and enter the last sheet ***"Outputs"*** as you have done with the other sheets.
The component ***U6*** MC74LCX07 is not included in the KiCad libraries, but a similar component 74LS06 with the same pinout is included in the 74xx library.

Reference	**U6**
Symbol Library	74xx
Symbol Name	74LS06
Value	MC74LCX07
Footprint library	Package_SO
Footprint	SOIC-14_3.9x8.7mm_P1.27mm
Name	MC74LCX07
Manufacturer	ON

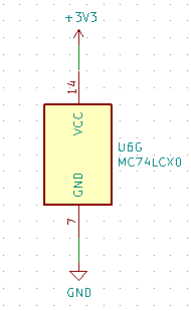
This is multipart component. It includes six inverters and you need to include a symbol for each gate.

Insert the first of the six instances and edit its fields as usual. The edit window for this component should be like in the figure below. Notice at the lower left position that there is a box ***"Unit"*** that has a value ***"A"***.



Now, copy this component five times so that you have a total of 6 instances of the component. Now edit them so that they all have the same ***U6*** reference but different unit letters from ***"A"*** to ***"F"***.

The **U6** component includes six inverters, so you need units from **A** to **F**. You should notice that there is also a **G** unit. This includes the power nodes of the device. Include this **G** unit and place it on the circuit to connect it to ***+3V3*** and ***GND***.



Add also the C13 component and connect it also between ***+3V3*** and ***GND***.

Reference	**C13**
Symbol Library	device
Symbol Name	C
Value	100 nF
Footprint library	Capacitor_SMD
Footprint	c_0805_2012Metric
Name	
Manufacturer	

The rest of the components are

References	**R8, R9 and R10**
Symbol Library	device
Symbol Name	R
Value	330R
Footprint library	Resistor_SMD
Footprint	R_1206_3216Metric
Name	
Manufacturer	

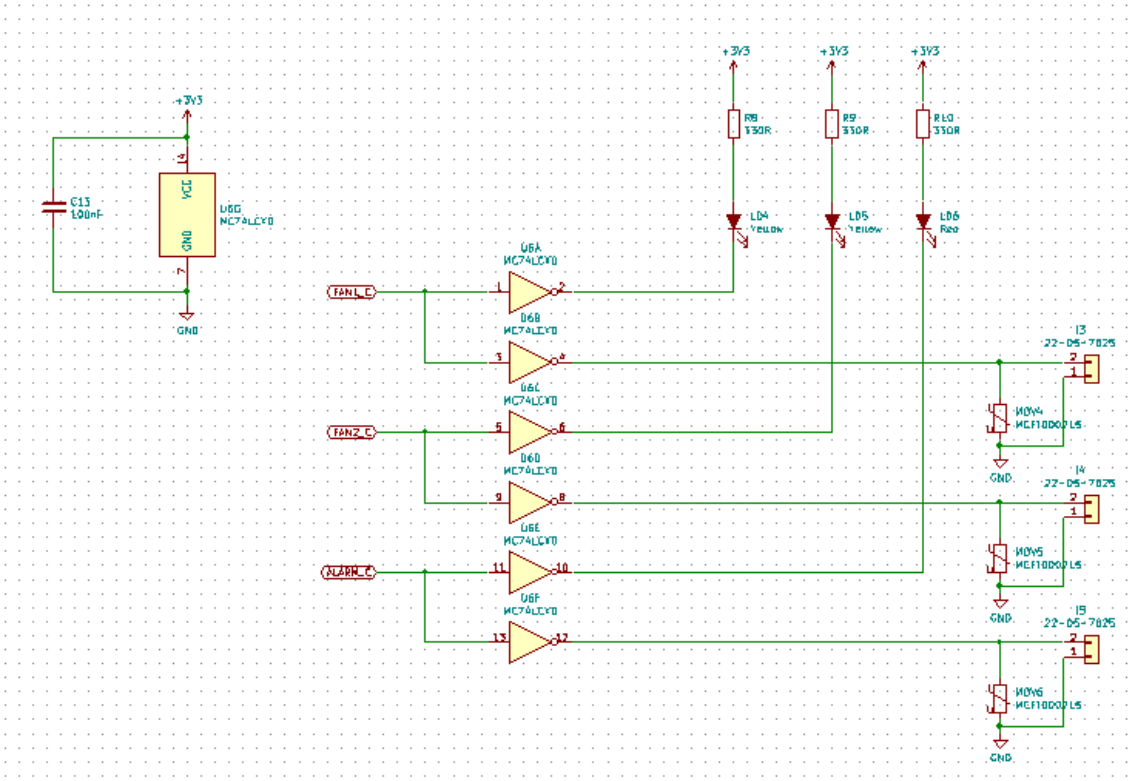
References	**MOV4, MOV5 and MOV6**
Symbol Library	device
Symbol Name	Varistor
Value	MCFT000215
Footprint library	Varistors
Footprint	RV_Disc_D7mm_W3.4mm_P5mm
Name	MCFT000215
Manufacturer	multicomp

Reference	**LD4, LD5, LD6**
Symbol Library	device
Symbol Name	LED_ALT
Value	LD4 and LD5: Yellow LD6: Red
Footprint library	LED_THT
Footprint	LED_D5.0mm
Name	LD4 and LD5: 703-0098 LD6: 703-0100
Manufacturer	multicomp

Reference	**J3, J4 and J5**
Symbol Library	Connector_Generic
Symbol Name	Conn_01x02
Value	22-05-7025
Footprint library	
Footprint	
Name	22-05-7025
Manufacturer	Molex

End the editing of the sheet. Note that pin 1 on all three connectors shall be GND.

The final circuit should be like in the figure below:



That ends the schematic capture phase of the circuit.

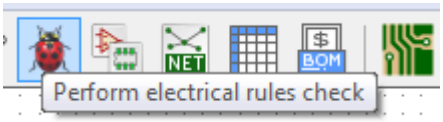
	Create and populate the Outputs sheet as described.

3.9 Electrical Rules Check (ERC)

After you draw the schematic, you can perform an Electrical Rules Check (ERC). This test verifies that you are not violating any rule of connections between components. Some of the errors it can detect are:

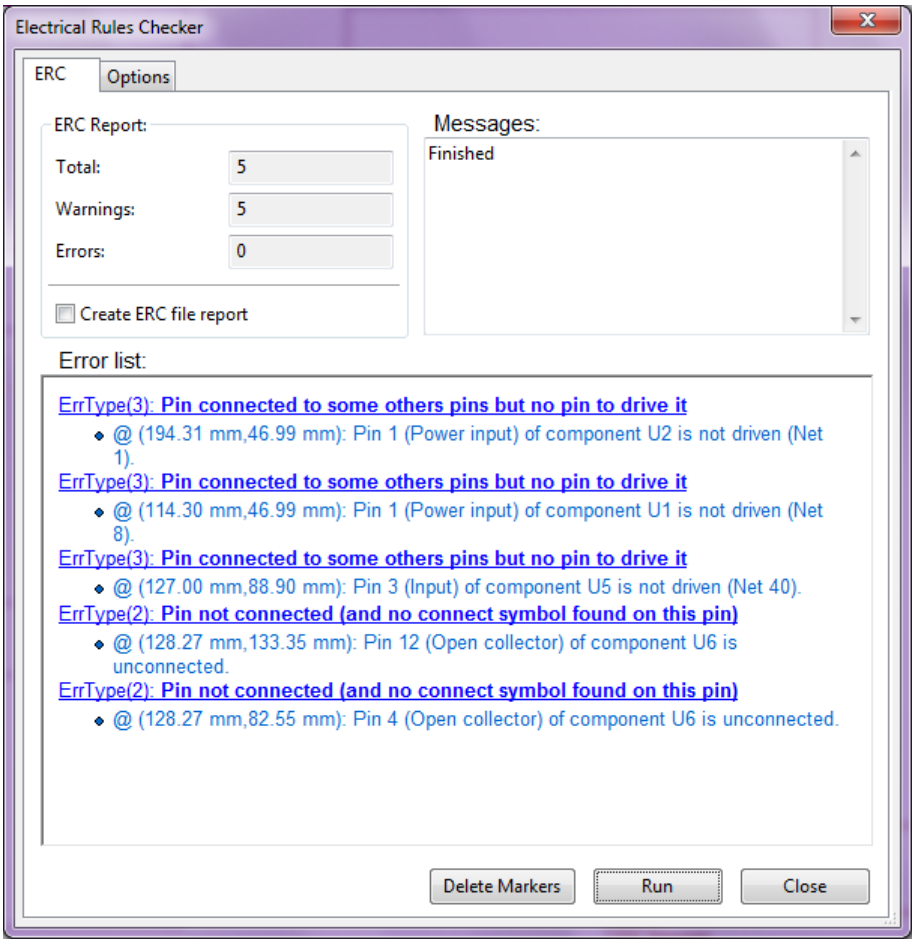
- Pin left unconnected and not marked as *Not Connect*.
- There is only a maximum of one output at each node
- There is a maximum of one power output at each node

You can open the ERC window by clicking on its icon:



Then you can click the ***Run*** button to perform the test.

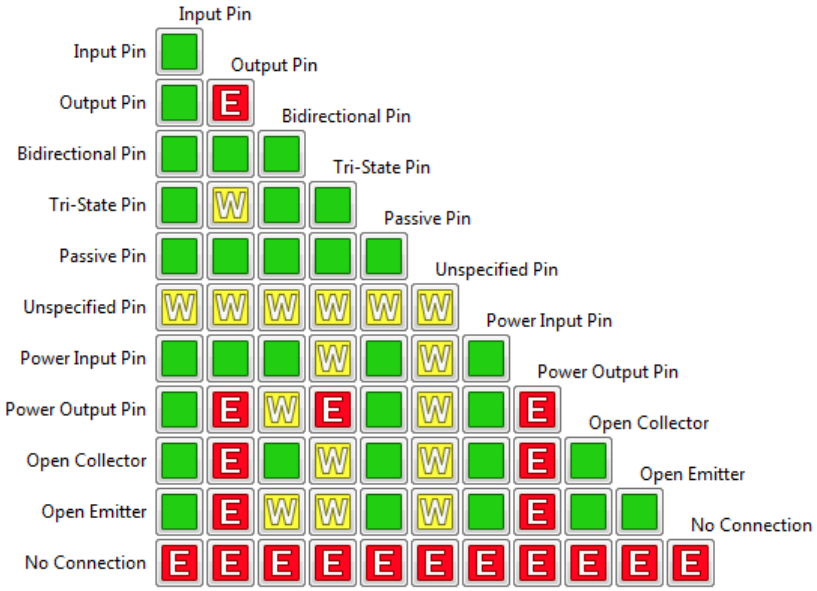
The following figure shows a result of a test run.



To locate the error you can click on the blue message on the window.

There are two kinds of messages: Errors and Warnings. An **Error** means that the circuit has to be modified so that the error is corrected. A ***Warning*** means that you are responsible to guarantee that the circuit operation is not affected.

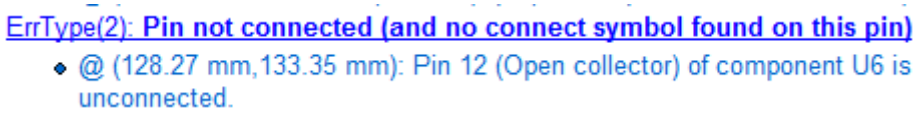
If you click on the ***Options*** tab on the window you can see how the Errors and Warnings relate to the different connection problems, as shown in the following figure:



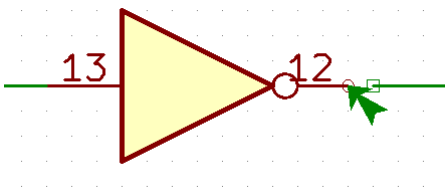
Connecting together two Output pins always gives an Error, but connecting, for instance, an Output pin with a Tri-State pin gives a Warning. The green options mean no Error nor Warning.

Both the Errors and the Warning, however, depend on using the correct definitions for all pins in all components. Sometimes, when you define a new component symbol, some pin is given a bad definition by mistake and, also, in some cases, some devices in symbol libraries have definition errors also. That means that sometimes an ERC Error is not related to a real error and sometimes, a real error is not detected due to bad pin definitions on the components.

In the above error window example, the following error:



Is related to a bad connection on the output of a gate:

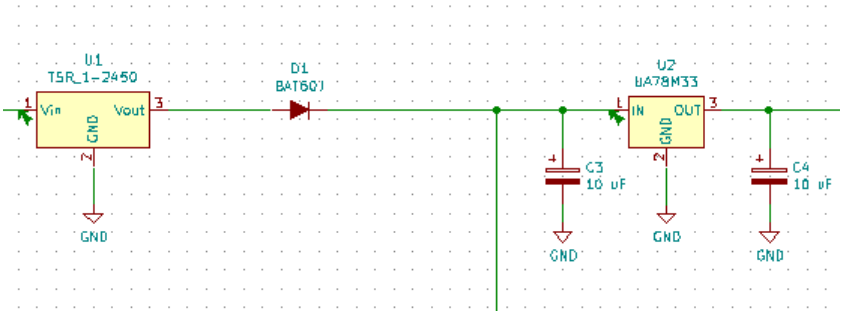


A typical case is when we declare all terminals in a power net as output and no node is declared as power input. That gives the Error message like:

ErrType(3): **Pin connected to some others pins but no pin to drive it**
- @ (194.31 mm,46.99 mm): Pin 1 (Power input) of component U2 is not driven (Net 1).

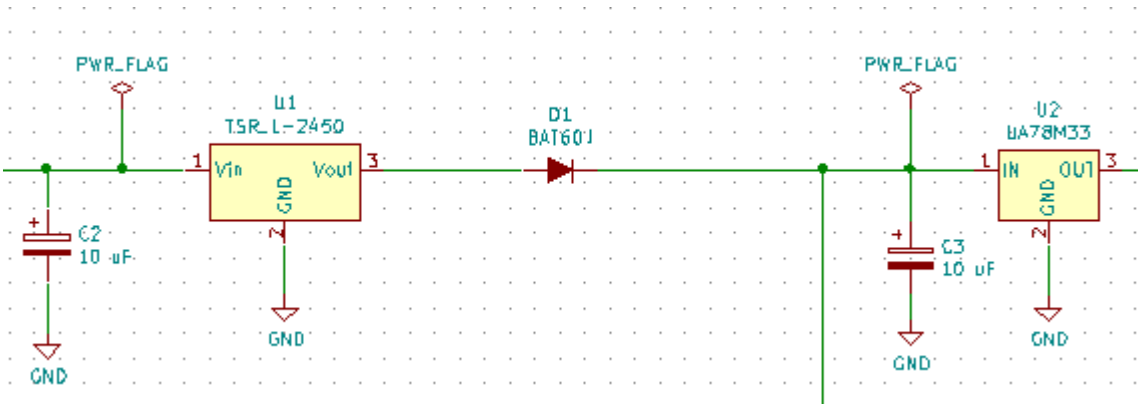
ErrType(3): **Pin connected to some others pins but no pin to drive it**
- @ (114.30 mm,46.99 mm): Pin 1 (Power input) of component U1 is not driven (Net 8).

In this case the error is in the nets that give the input to the two voltage regulator units.



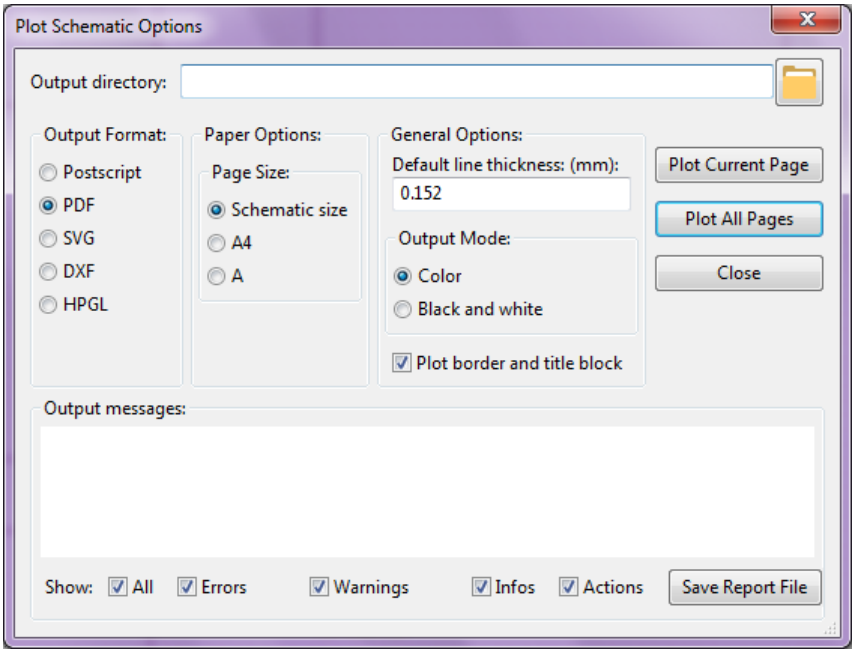
The input of the regulators is declared as power input but the nets connected are not marked as capable of providing power.

You could ignore this error but, in general, is bad practice to ignore errors because other engineer when examining your project would need to check also that the error can be ignored. The best solution is to mark the net with as power input using a power flag **PWR_FLAG** component.



Check and correct, if needed, all ERC Errors and Warnings on the project.

At this we have ended this design phase. This is a good time to generate the schematic documentation for the project. Select the ***Plot*** item in the file menu.



Select PDF as the output format, check that the Output folder is the project folder and click on ***"Plot All Pages"***. You should obtain a ***"Demo.pdf"*** file with all the schematics for the project.

☑ **Check #2 : Schematic check**

Generate the schematics PDF file. If generated during a ETSETB lab session, show it to the lab professor.
Upload the file in the proper **Atenea** task of the seminar.

4. Pre-Layout Review

DESIGN PHASE 3

The next big stage on the design workflow is to draw the PCB. But before that, we need to prepare the PCB work by performing some additional tasks.
At this point the schematic is complete and is recommended to discuss it with the circuit designer if he (or she) is different from the PCB designer. That discussion gives the final validation to the schematic.

4.1 Mising connector footprints

As you remember, some components on the schematic don't have the required footprint on the KiCad application. Those components are:

- All connectors except the USB one (J7)
- Inductor L1
- Reset button SW1

We need to create the footprints for those components before starting the drawing of the PCB.

A footprint has three main elements:

Text elements: Text elements identify the footprint during the PCB design. The first text element is the ***Component Value Placeholder***. It is shown on yellow color on the ***Fab*** layer. During the footprint editing phase it shows, however, the ***footprint name*** not the ***Component Value***. As we will see later, during the post processing stage, you can make this element visible on the ***Fab*** layer of the final PCB.
The second text element is the ***Component Reference Placeholder***. This element is shown as **REF**** on the ***Silkscreen*** layer during the footprint design. Once you use the footprint in an actual design, it changes to the reference name of the component. So, several components with the same footprint will have different references, like *U1*, *U2*,... This element can be made visible in the final ***Silkscreen*** layer of the PCB.

Pads: The Pads are electrical connections for the component. In a ***through hole*** component, Pads are composed by copper regions (usually circular) on all copper layers and a drill hole centered on the Pad. In a ***surface mount device*** (SMD), Pads are composed by just a copper region (usually rectangular) on only one copper layer: the one on the component side. You don't need to define if the component will be on the ***top*** side or the ***bottom*** side when creating the footprint. You do that later in the placement stage.

Drawings: There can be three possible drawing layers on a footprint. You can include both text and drawings on those layers. All of them are optional in the sense that you can fabricate

a board without messing with them but PCB fabrication and assembly companies sometimes request that some of them are present. Those three layers are:

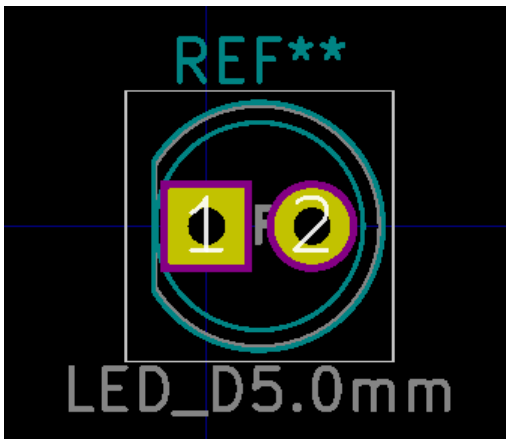
Silkscreen: The Silk Screen layer is a paint layer that can be drawn on the top and bottom of the board. It can be useful to properly identify the components during the assembly and test of the PCB.

Assembly (Fab): The assembly (or Fab) layer is a drawing that indicates how is the component and is manly used by the assembly companies to set the correct component orientation. So, it is important that the component orientation can be deduced from this drawing.

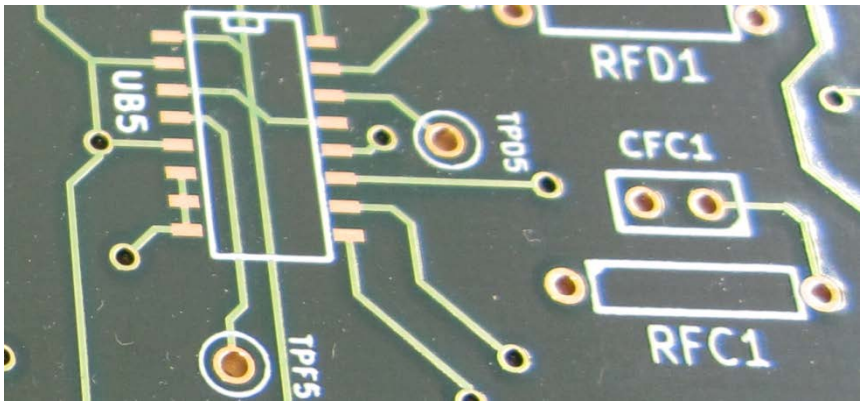
Courtyard: The courtyard layer is used to define the area that occupies a component including any necessary clearances. It usually is a box that extends from the component size to provide the proper clearance to other components. During the placement stage, this layer is used so that the courtyards of two components never overlap. You can think of it like a component bounding box.

If you want to include the ***Reference*** of the component in a layer different from the ***Silkscreen*** layer, you can use the special text ***%R*** as a placeholder. In a similar way, if you want to include the ***Value*** of the component in a layer different from the ***Fab*** layer, you can use the special text ***%V*** as a placeholder.

The following figure shows the footprint of a 5 mm LED. You can see the two text elements: the ***Value Placeholder*** "LED_D5.0mm" on the grey ***Fab*** layer and the ***Reference Placeholder*** "REF**" on the blue ***Silkscreen*** layer. You can also see the two through hole pads. The figure includes drawings in the ***Silkscreen***, ***Fab*** and ***Courtyard*** layers. Note also that the component reference is also included in the ***Fab*** layer although it is difficult to see because it is below the PADs.



As explained before, the ***Component Reference*** is usually added during the later stages of the PCB design to the ***Silk Screen*** layer. That way, the component reference and its silkscreen layer are shown on the final fabricated PCB like in the image below.



First we will define the connector footprints. The following table shows all connectors with missing footprints on our design.

Reference	Definition	Manufacturer	Name
J1	Power input (2 pin)	IMO	20.101M/2
J2, J3, J4, J5	Output (2 pin)	Molex	22-05-7025
J6	Comm I/O (3 pin)	Molex	22-05-7035
J8	Debug	Amphenol	T821106A1S100CEU

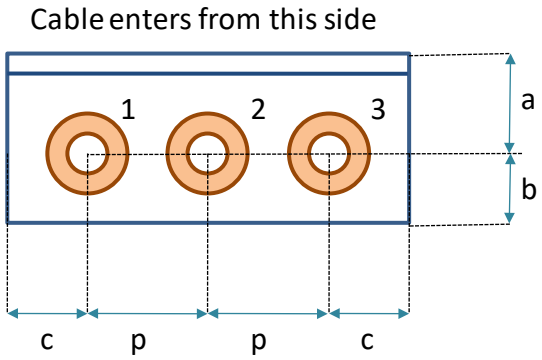
In order to define the footprints we will need to choose a footprint name for each connector kind and locate the ***pdf*** file in the ***Datasheets*** folder of ***PCBD_Files.zip*** file. As we are building custom footprints, the footprint name can be whatever we want although it is recommended not using repeated footprint names in the design.

Reference	Pins	Footprint Name	PDF
J1	2	Power	PWR Conn.pdf
J2, J3, J4, J5	2	IO2	Molex.pdf
J6	3	IO3	Molex.pdf
J8	6	Debug	T821106A1S100CEU.pdf

Footprints for through hole components are always seen from the component side. This is the usually called top view of the component. Remember that so that you don't make a wrong reflected image of the correct footprint.

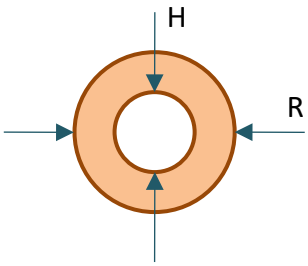
All connectors we use have trough hole pins. In the case of all connectors except J8, the pins are aligned in only one row with the cables entering from the side.

In order to define the connectors footprints (except J8) we need to obtain the values of the dimensions a, b, c and p for each connector. Note that only J6 features three holes, the other connectors J1 to J5 have only two holes.

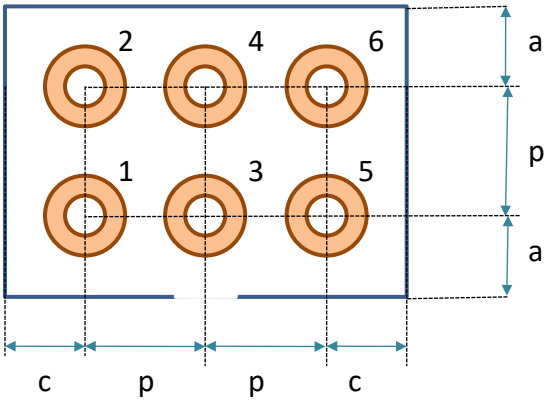


In the above figure **p** defines the **hole pitch** for the connector and the **a**, **b** and **c** parameters define the physical envelope limits. We need to know those limits in order to have the proper clearances to other components in the board.

We also need to know the hole diameter (H) and the ring diameter (R) for the pins.



In the case of connector J8, cables come perpendicular to the PCB plane and pins are arranged in two rows of three pins with an alignment mark next to pin 3.



Now **p** defines the hole pitch and **a**, **c** defines the physical envelope.
Length dimensions can be defined in a PCB in milimeters or ***mils*** (1/1000 of an inch). Most designs are defined in ***mils*** so we will convert all dimensions to ***mils*** units.

$$l[mil] = 1000 \cdot l[inch] = \frac{1000}{25{,}4} l[mm]$$

Now it is time to examine the connector datasheets in order to fill the following table.

n Pins	F. Name	a	b	c	p	H	R
2	Power						
2	IO2						
3	IO3						
6	Debug						

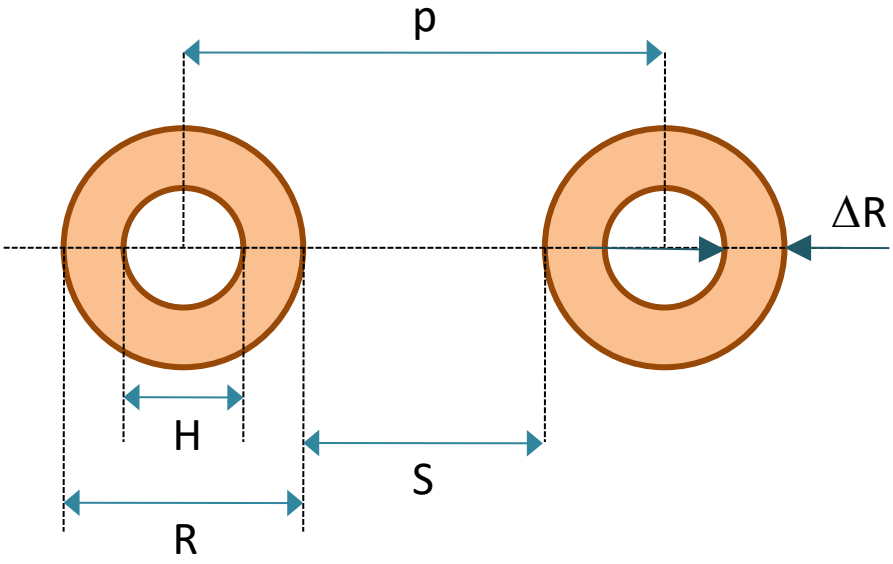
All dimensions in mils (1/1000 inch)

Dimensions a, b, c and p can be obtained from the connector drawings on the datasheets. Note that the Debug connector doesn't have a **b** parameter. Hole diameter **H** must be just over the pin diameter. In most cases the datasheet gives a hole diameter recommendation.

The power connector **J1** is the only one that doesn't give a hole recommendation. In this case we can use the **IPC-2222** recommendation for low density boards of adding 0,25 mm (10 mil) to maximum pin diameter.

In any case the hole diameter must be compatible with the fabrication capabilities for the board as seen on Appendix C.

After giving a value to the hole diameter H, we need to give a value to the ring diameter R. The **IPC-2221** document recommends adding 0.7 mm to the hole diameter for low density boards. The following figure shows all dimensions involved.



We obtain **H** from the manufacturer recommendations or we add 0,25 mm to the maximum pin diameter. Then we add a 0,7 mm to obtain the ring diameter **R**. Finally, from the pitch **p** and the previous values we calculate the copper separation **S**.

All the values need to be compatible with the fabrication capabilities (see Apendix C), so we need to verify:

$$S \geq 0{,}2\ mm$$

In general it is a good idea to be able to route one or two tracks between two pins. If both the minimum track width and spacing is 0,2 mm, then, in order to route one or two tracks between two pins we need to guarantee:

$$S \geq 0{,}6\ mm\ (one\ track) \qquad S \geq 1\ mm\ (two\ tracks)$$

The ring thickness **ΔR** affects the mechanical properties of the connector. A too thin ring will be easier to break if too much force is applied to the connector. In the case of the power connector we will double the recommendation and add 1,4 mm to the hole diameter instead of 0,7 mm.

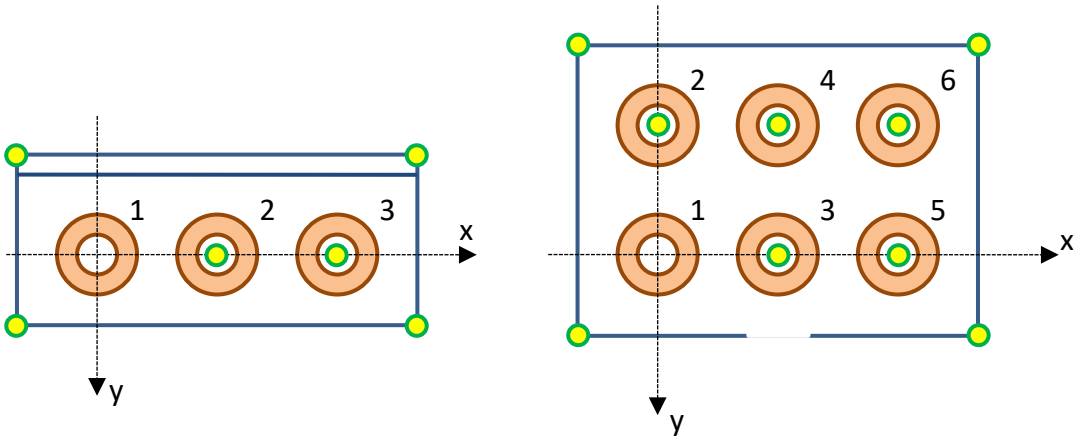
Note that the ***Debug*** connector pins are square and the datasheet gives the side size. You should use the diagonal to estimate the equivalent pin diameter.

Fill the table for all dimensions on all connectors. Use mils for units.

¿How many 0,2 mm tracks can you pass between two pins on each connector?

After all dimensions are defined, we need to draw the connector footprints. It is habitual to define the (0,0) coordinate to the center of pin 1. Also, you need to take note that the coordinates in the PCB editors are not the usual mathematical ones. We have an inverted vertical axis. That is, the X horizontal axis increases from left to right but the Y vertical axes increases from top to down.

In order to position the different elements of the footprint we need to know their coordinates relative to the (0,0) position at the pin 1 center. For the 3 and 6 pins connectors, for instance, that means that we need to know the coordinates (in mils) for the yellow points in the following figure. 6 locations for the 3 pin connector and 9 locations for the 6 pin connector.

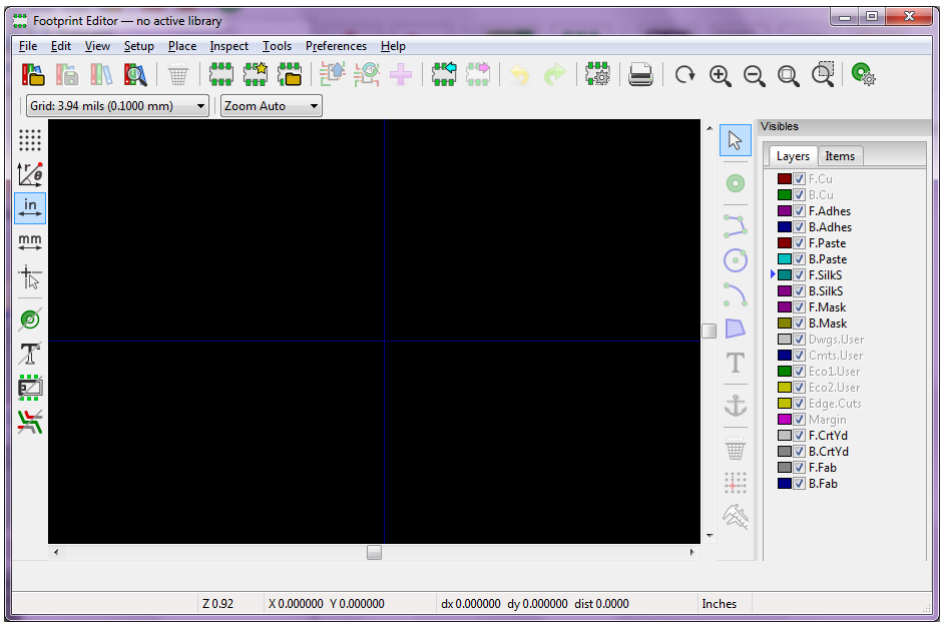


Obtain the needed coordinate points (in mils) for the four connector footprints.

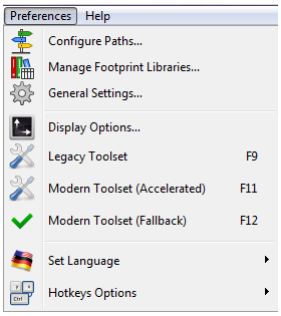
Note that several points will feature negative coordinates in one or both axes.

Once the dimensions associated to the footprints are defined, we can draw them. Close ***EEschema*** if it is already open and open the ***Footprint library editor***.

A window like the one below should open. Note that there is no active library yet.



It is possible that you get a message about enabling **graphics card acceleration**. You can enable it if you wish. You can always turn off the acceleration, if you need, selecting ***Legacy Toolset*** inside the preferences menu. This menu has three options:



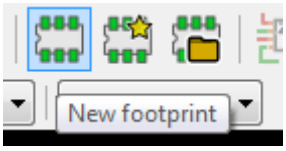
Legacy Toolset is the graphics of previous KiCad versions.

Modern Toolset (Accelerated) is accelerated graphics.

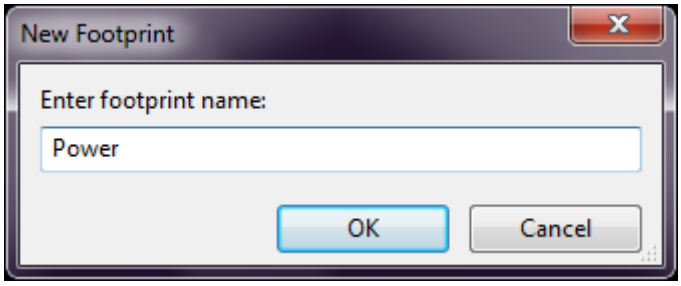
Modern Toolset (Fallback) has the same graphics as in the accelerated mode but without using hardware acceleration.

Graphics will be different depending on the mode you select, keep that in mind if you compare the images in this manual with the ones you get on screen.

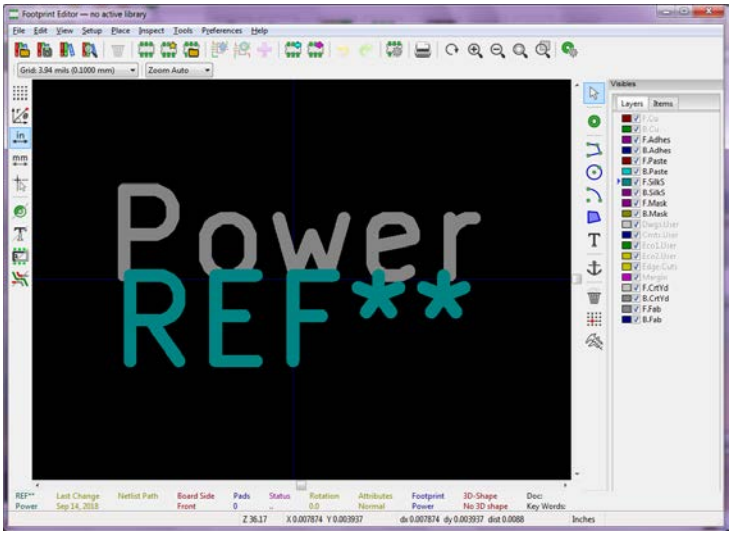
Click on the ***New footprint*** button



And give the new footprint the name ***"Power"*** we selected for the power connector J1



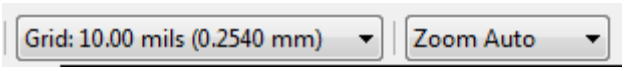
You should see a window like the one below:



When you generate a new footprint, it gets two text regions. The first one is the value placeholder and shows the name of the footprint itself , ***"Power"*** in this case. The second one is the component reference placeholder that shows the ***"REF**"*** text. In this particular case it will be **J1** in the future when we place the footprint in the PCB design.

At the bottom of the window you can see the cursor location in the selected units. If you hover the cursor inside the window you will see as they change. As we will be using ***mils*** (1/1000 of inch), make sure the units are in Inches. If you get mm units, you need to change the units using the ***In*** button at the left side of the window.

Observe that you can change the grid and the zoom in the upper region of the window. The best way to zoom, however, is by using the mouse wheel.

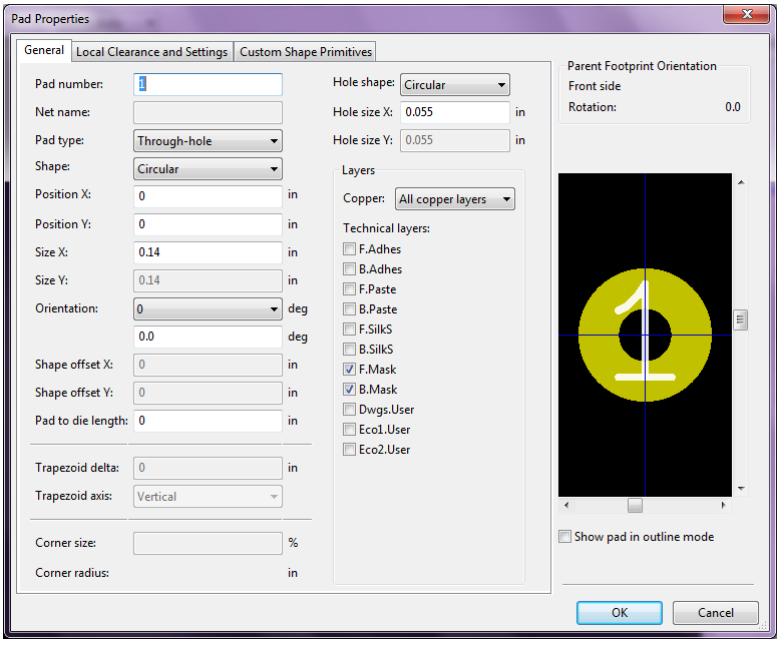


We recommend you to use in this seminar the ***mil*** (1/1000 inch) units so that you became familiar with this unit. You can, however, alternate between the ***mil*** and ***mm*** units when you are working on the footprints and the PCB. In general the best unit is the one that better aligns the component pads with the grid.

Zoom out and move the two text elements out of the way, far from the 0,0 position so that they don't annoy us while we are drawing the footprint. Then select the **Add Pad** tool to put the first pad of the footprint.

Click, then on the (0,0) position to place the Pad for pin 1. It will place a round pad, but, perhaps, it is not the right size. Always center pin "1" on (0,0) for though hole components.

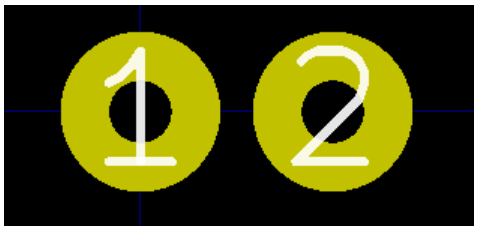
To edit the Pad, just hit the **"e"** keyboard key over it. A window like the one below will open.



As this is the first Pad, at (0,0) location, the Pad number 1 is ok. Leave Drill Shape as ***Circular*** for now and Pad type ***Circular Through-hole***. As both are circular, you only need to define the diameter as the Size X both for the Pad and the Drill hole. In this case H (hole size) is 55 mil and

R (size X) is 140 mil, but it could be slightly different on your case depending on your previous calculations.
Note the ***All copper layers*** setting. That means that this shape will be replicated on all layers on the PCB. This is usual for through hole Pads.

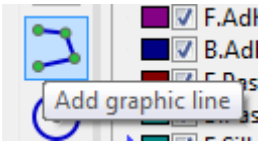
Do the same for the second Pad. Check that the Pad number is 2. Use the proper grid to place the Pad and check that it is at the proper position. As this connector belongs to the power supply, the ring is thick compared to the hole size.



Sometimes, footprint designers mark pin 1 by using a square share for its copper layers. Do that if you want in all connectors. But this is not a requirement.

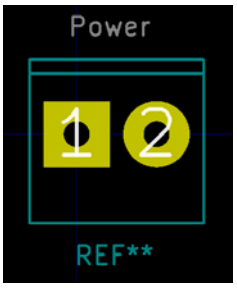
In order to define exact positions, you can place a Pad wherever you like and change its position by editing the coordinates on the edit window.

After defining the Pads, we need to add the drawing layers graphic elements. Use the ***Add graphic line*** tool for that.



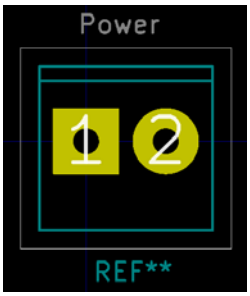
We will start with the ***Silkscreen*** layer so check that the ***F.SilkS*** layer is selected on the right of the window.

For a sequence of lines just click the different vertexes. On the last one, bring up the contextual menu with the right mouse button to select **End edge**. The most important elements of the footprint are the Pads, a small error on the silkscreen drawings has no major importance. After drawing the silkscreen box, bring close the two text elements so they are close to the drawing. You don't need to put the ***REF***** on the upper side and ***Power*** on the bottom side. Do as you please.



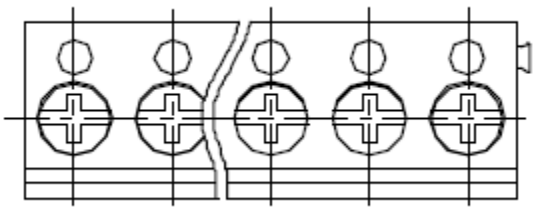
Note that the double line at the top of the component identifies cable side of the connector.

Now we will define the footprint ***Courtyard*** that defines the minimum space around the component needed to successfully mount it. Select the ***F.CrtYd*** courtyard layer and define a bounding box about 1 mm (40mil) around all sides of the component. Use a thin 2 mil thick line for this layer. Observe that the 1 mm clearance space is arbitrary. In a practical case the courtyard requirements shall be obtained from a proper reference standard like **IPC-7251** or **IPC-7351**. Those standards typically recommend that the courtyard extends from the component 0,5 mm in all directions on low density boards. Note that the manufacturer can add his own courtyard excess to this number so 1 mm seems about right for a connector.



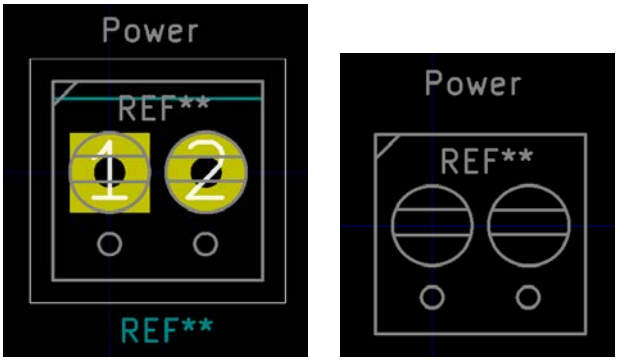
Finally we will add the assembly drawing. Select the ***F.Fab*** layer for that. The assembly layer is used so that the assembly company can give the component the proper orientation. This is especially important for components that have any symmetry.

From the component datasheet we know that seen from above, the power connector is similar to the following figure:

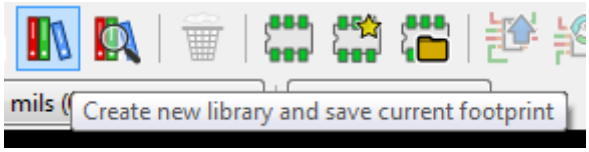


You can be creative on this layer. The following figure shows an example for the ***Fab*** layer. On the right only the ***Fab*** layer is shown. On the left, all layers are visible. Note the diagonal line on

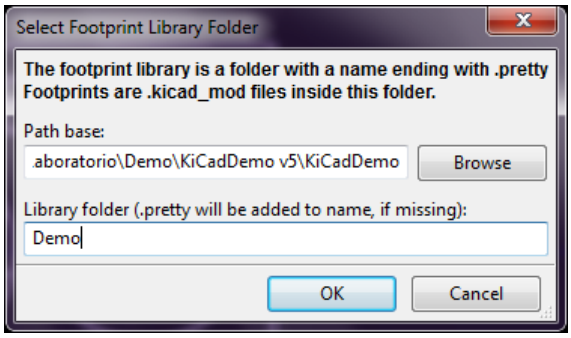
the upper left corner that identifies that pin #1 is on this side. Note also that we have replicated the component reference using a text **%R** (shown as REF**) in the ***Fab*** layer.



That ends the drawing of the footprint, we just need to save it. As this is our first footprint, we will save it on a new library using the ***Create new library and save current footprint*** button.

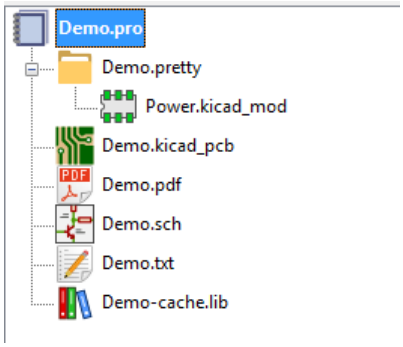


KiCad defines footprint libraries as folders with footprints, we will create a folder ***Demo.pretty*** that hangs from our project folder. Note that the ".pretty" extension is added automatically.

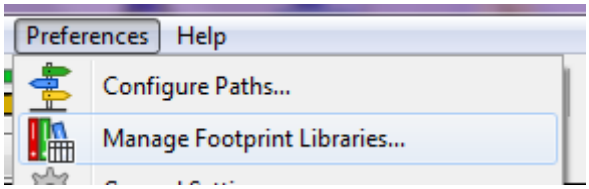


After you click ***Ok***, if you navigate to your project folder you will notice a new folder ***Demo.pretty*** and the newly created footprint ***Power.kicad_mod***.

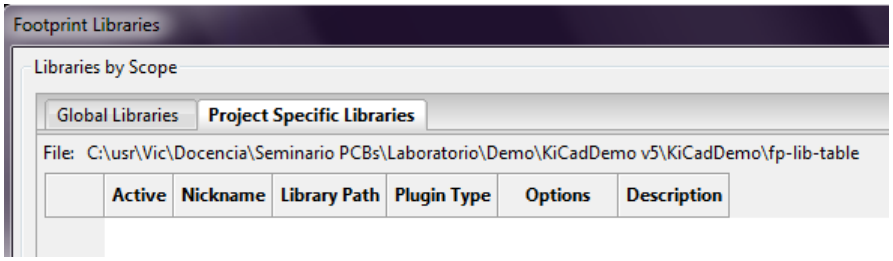
If you go to the main KiCad window you will see that the library hangs from the project folder.



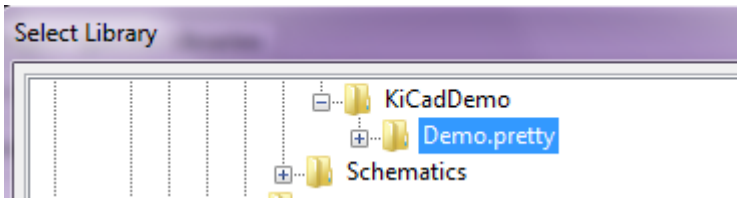
Now that we have defined a new library, we make it available to our project using the ***Manage Footprint Libraries*** on the ***Preferences*** menu.



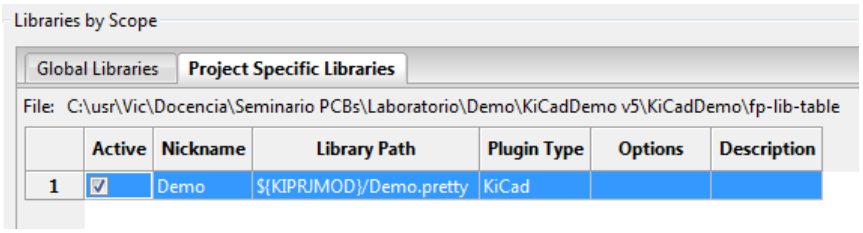
Now select the ***Project Specific Libraries*** tab



Click ***Browse Libraries*** and select the **Demo.pretty** library



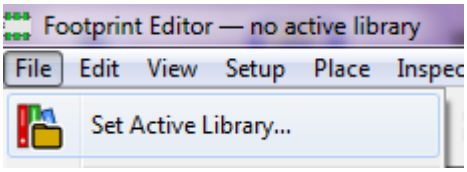
You will see that the library has been added to your project



Just click ***Ok*** to dismiss this window.

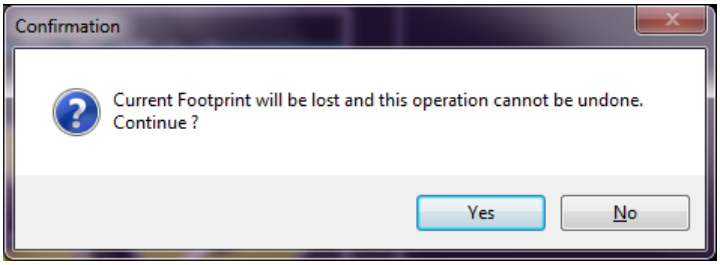
Now you can create the footprints of the other connectors **IO2**, **IO3** and **Debug**.

As we have defined the **Demo** library, we can make it the active library. Select ***Set Active Library*** from the ***File*** menu.



Navigate to the Demo Library and select it, you should see it as the currently active library.

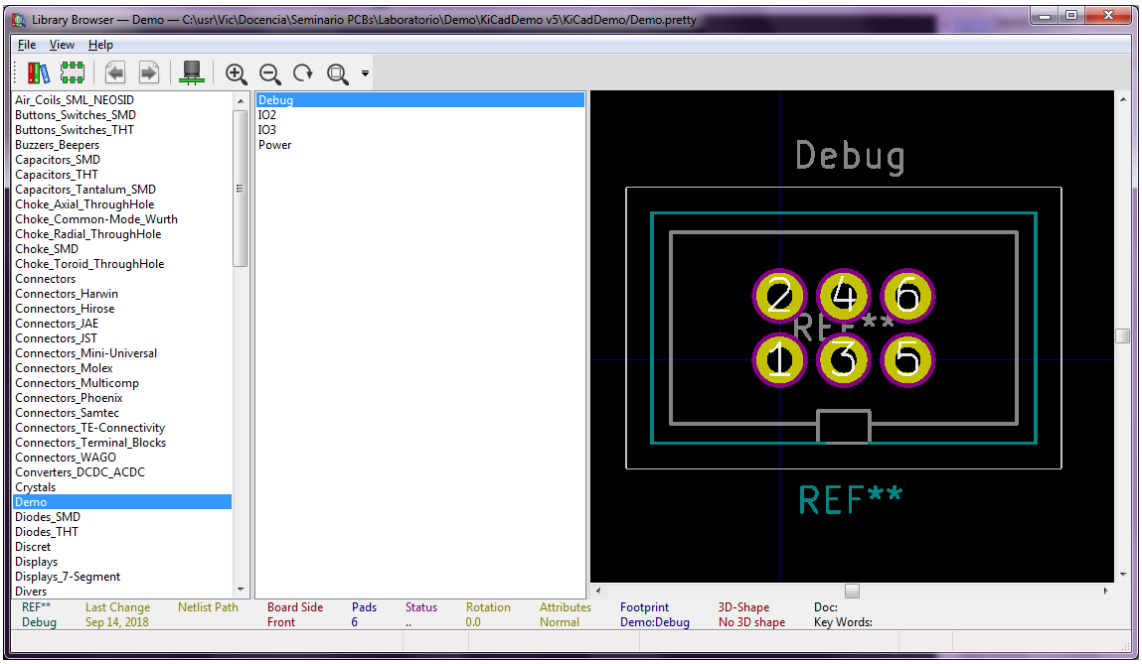
Click ***New footprint*** to start a new footprint definition. Don't worry about the warning message below if you have done the previous steps.



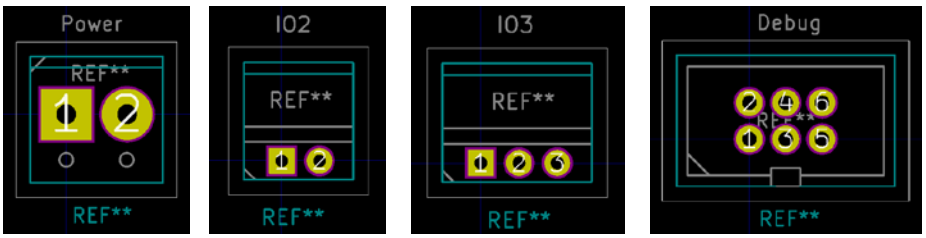
As you will enter each new footprint in the same ***Demo.pretty*** library, use the ***Save Footprint in Active Library*** on the ***File*** menu each time you end a footprint.
Don't forget to check that the pin numbers are correct on each footprint.

If two footprints are similar, sometimes you can copy a previous footprint, modify it, and save it with a new name. That could be done on **IO3** as it is similar to **IO2**. Note, however, that sometimes modifying an old design is more complex that creating a new footprint from scratch.

After you have ended the creation of the four connectors, you can close the ***Footprint editor***. If you want, you can open the ***PCB layout editor*** from the main KiCad window, open there the ***Footprint viewer*** and check the footprints you have created.



The figure below shows the four connector footprints although they are not at the same scale. Remember that you don't need to use the same shapes, especially in the ***Fab*** layer. Note that we always add a reference placeholder by adding a ***%R*** text field on the ***Fab*** layer.



Improvement of silkscreen on Power, IO2 and IO3 connectors

Those connectors are meant to reach the edge of the board. Problem is that *silk screen* layer should not reach or extend over the board edge.

You can improve those footprints by using a *silk screen* drawing that doesn't occupy the cable entering side. Note that *Fab* layer is not a problem in this case.

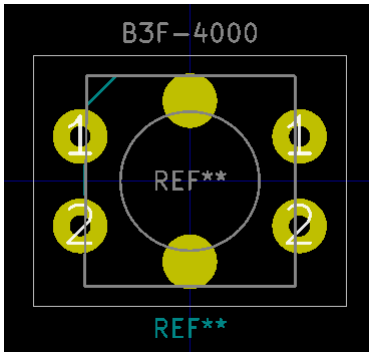
Draw all the four connector footprints.

4.2 Button footprint

There is one additional through hole missing footprint: The Reset button SW1.

Reference	Pins	Footprint Name	PDF
SW1	4	B3F-4000	B3F-4000.pdf

The reset button is somewhat special. First, the switch has two nodes but it has four electrical Pads as you can see on the datasheet. So, there will be two Pads with Pad number 1 and two Pads with Pad number 2.
Also, this button features two non electrical positioning holes. In order to generate those holes select for them the Pad type ***NPTH, Mechanical***. NPTH means *Non Plated Through Hole*.
Note that the ***Silkscreen*** layer should not intercept the Pads. The ***Silkscreen*** to pad distance shall be at least 7 mil. In some CAD tools you can draw this layer over the Pad and the tool will take care of any related problem, but it is better to do things right during the footprint design. There is no problem with the ***Fab*** layer as it is not drawn on the PCB.

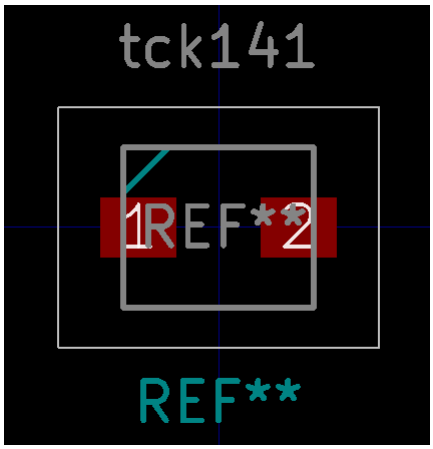


Draw the SW1 footprints and save it on the Demo library.

4.3 The inductor footprint

There is only one footprint missing. The inductor **L1** footprint.
We will use **tck141** as the footprint name.
See that the datasheet for this component ***tck141.pdf*** includes an exact definition of the Pads needed for this component. Note that the pads are not square but rectangular.
The difference between this component and the other ones is that this is a SMD component. So you will need to use **SMD** as the Pad type. Logically there is no hole in this kind of Pad.

After you end editing this component you will have a pad similar to the one below. Typically, in SMD components, the (0,0) position should be the center of the component instead of the center of pin "1". This is not critical, however, if you don't share your library.



As in the previous footprint, the ***Silkscreen*** layer should not intercept the Pads.

Draw the inductor footprint.

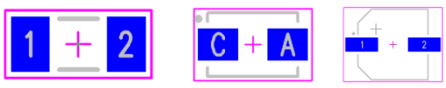
At this point you have the seven custom footprints in the ***Demo.pretty*** library folder so you have all necessary footprints to start creating the PCB design.

Check #3: Footprints

Show the seven footprints, using the footprint viewer, to the lab professor if available.
Upload the ***Demo.pretty*** folder, zipped in a file, in the proper **Atenea** task of the seminar.

Zero orientation

When creating a new footprint the orientation you use is the zero orientation without any rotation. There is a preferred zero orientation defined in standards like **IPC-7251** and **IPC-7351**. For instance, passives shall have two pads separated from left to right, not top to bottom. Pin "1" on the left.



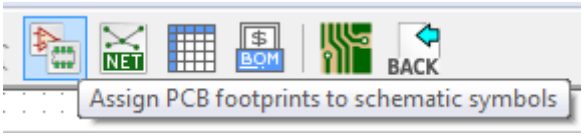
On polarized capacitors pin 1 shall be the positive. On diodes pin 1 shall be the cathode.
For ICs, pin "1" is on the upper left corner.
In general, pin "1" should be on the upper left corner or on the upper side if pin "1" is not in one corner. It seems that this is not always followed in connectors.
Note that there are other standards like **EIA-481-D** than can define other "standard" orientations.

4.4 Updating the schematics

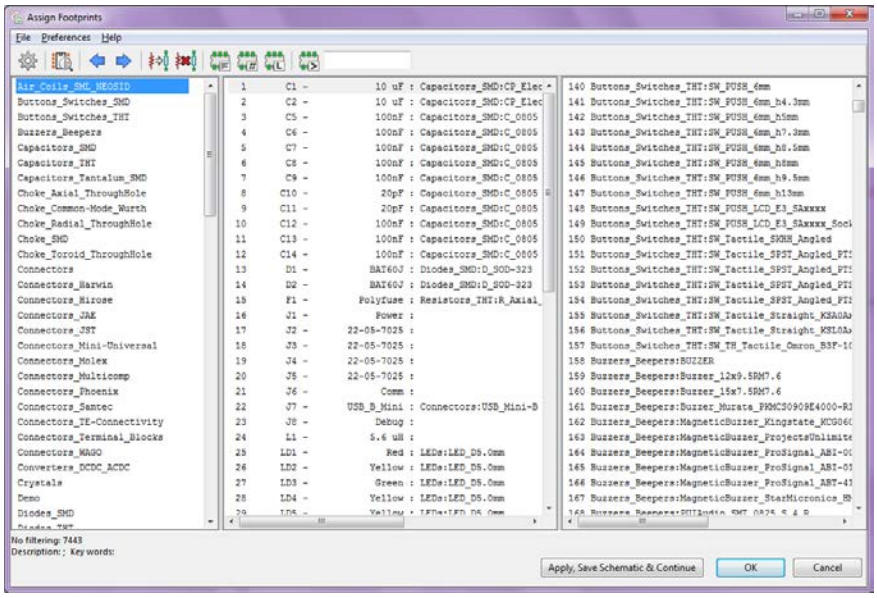
Now that we have the missing footprints, we can update the schematic so that all its components have an associated footprint. The following table shows the new footprints we have created in the custom ***Demo*** library and the associated component references.

Footprint	References
Power	J1
IO2	J2, J3, J4, J5
IO3	J6
Debug	J8
B3F-4000	SW1
tck141	L1

Open ***Eeschema*** and run the ***Assign PCB footprints*** tool by clicking on its button:



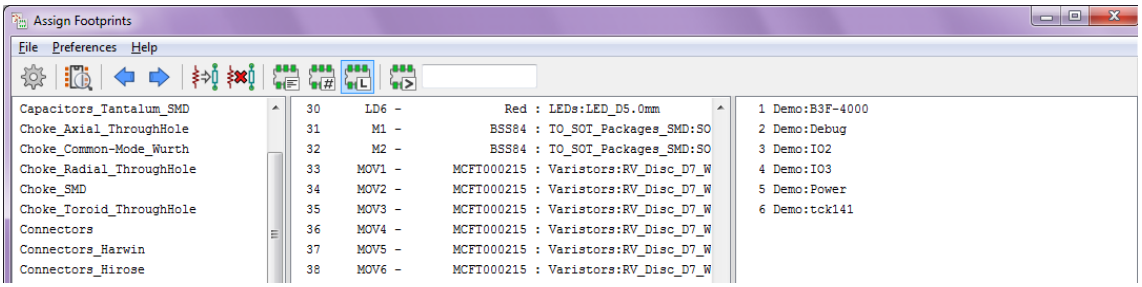
A window like the one below should open:



Observe that most components have an associated footprint as we did that during the schematic capture stage. Only nine components have missing footprints. In order to set the footprint for a component, just select the component in the center region and select the appropriate footprint in the right region.

To ease the selection of footprints you have four filter options at the top of the window:

The first one, on the left, lets you filter the footprints by keyword. We don't recommend using this filter now. The second one filters for number of pins. That means that only footprints with the same number of pins as the schematic symbol will be shown. You can select this filter although it is not critical. The last one, on the right, filters by library. That means that only the footprints in the library selected on the left panel of the window will be shown. As we only need to add footprints from our ***Demo*** library, this is a very good filter to use. The last filter selects by patter or partial footprint name.



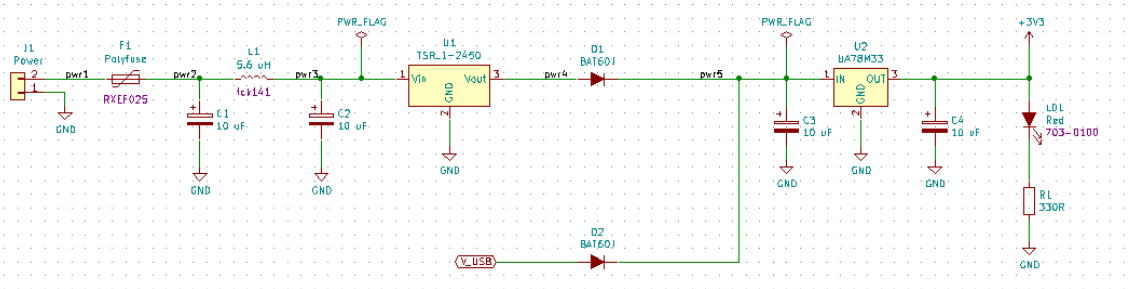
After adding footprints to the nine components, close the window saving the modifications.

Update the custom footprints in the components that use them.

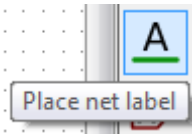
Now, if you edit one of the modified components, like **U1** on ***Eeschema***, you will see that the footprint field has been modified as expected.

4.5 Naming power nets

This step is not a requirement but eases the PCB drawing. It is recommended to group all nets depending on the preferred track width. It is normal to have the power nets thicker than the signal nets. In order to easily identify the power nets, it is recommended to have a name on all of them. Open the ***Supply*** sheet and mark the nets associated with power with local labels ***pw1*** to ***pwr5*** as shown in the figure below. You don't need to set labels on the nets already labeled like 3V3, GND and V_USB.

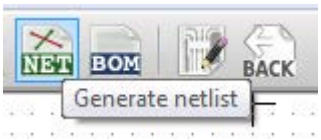


Use the ***Place net label*** tool for that

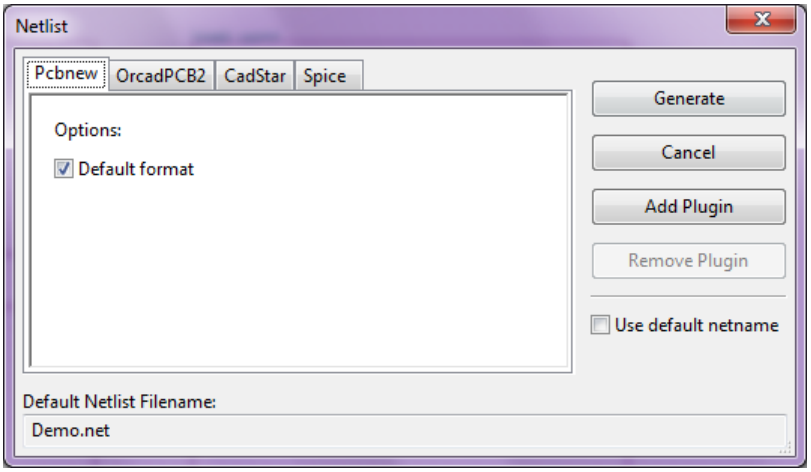


4.6 Netlist generation

The netlist is link between Eeschema and PCB New. It lists all the components in the design, he footprints associated with them and the connection topology.



Click on the ***Generate netlist*** button. Don't change anything and just click on ***"Generate"*** on the window that will open.

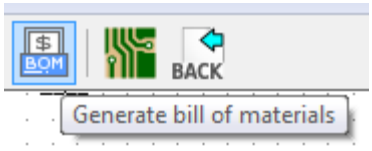


Save the netlist as ***Demo.net*** in the project folder.

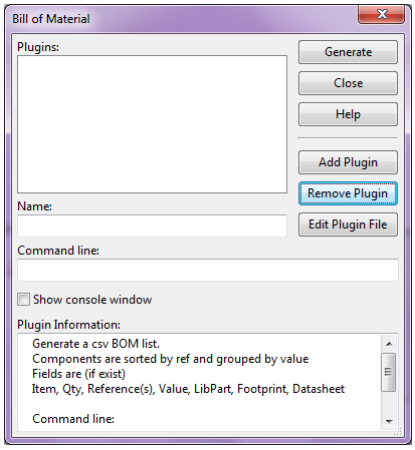
Do all previous steps so that you can generate the proper ***Demo.net*** file.

4.6 Bill Of Materials

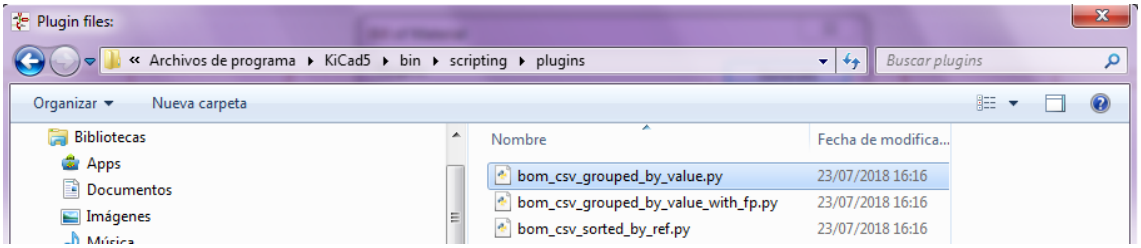
At this point we will obtain the ***Bill Of Materials*** (BOM) for our project directly from our schematic data. This document is a list of all components in our design that will be provided so all the needed components could be obtained from their manufacturers. Start by clicking on the **Generate bill of materials** button.



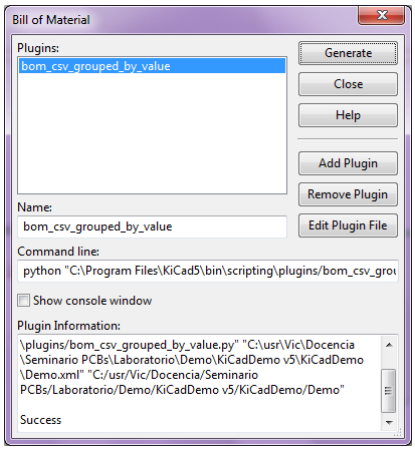
You should obtain a window like the one below and, probably, no *plugin* is in the upper box.



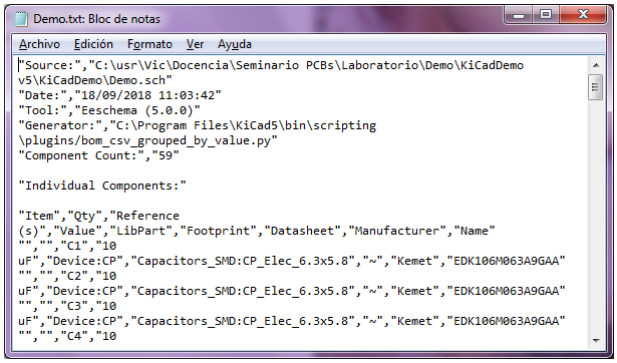
We need the *boom_csv_grouped_by_value.py* plugin, if it is not in the list, select the ***Add Plugin*** button and navigate to the scripting/plugins folder to select the needed plugin.



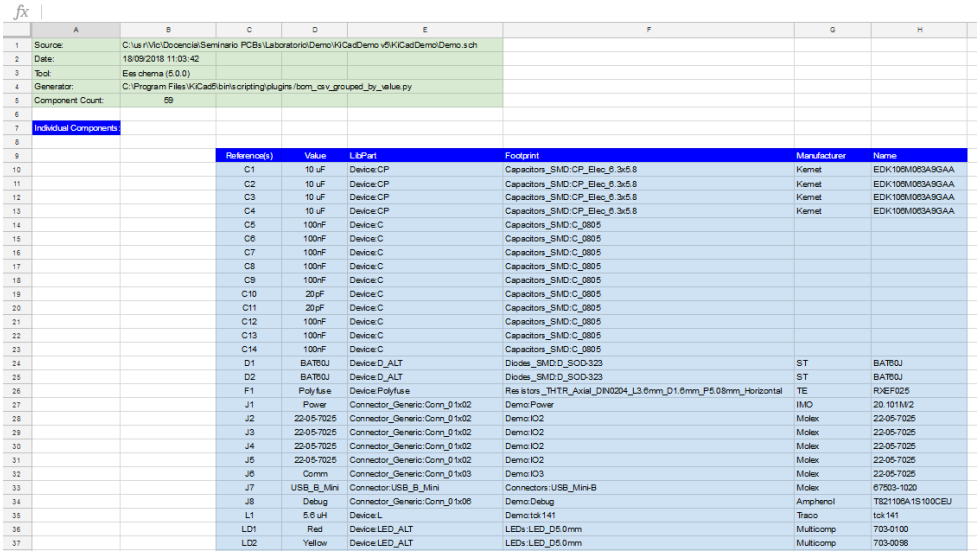
Once you add this plugin to the list, select it and click the Generate button. The plugin will run and generate a *Success* text output.



You should now have a *"Demo"* file without extension on your project folder. Rename it to *"Demo.txt"* and check that it is, in fact, a bill of materials.



The format is *comma separated value* (CSV). Import it on your preferred spreadsheet program. Below you can see the list on a Google documents spreadsheet. You don't need to use colors but sometimes it ease the document reading. Observe that the ***Datasheet*** column has been eliminated as we don't use it.



	A	B	C	D	E	F	G	H
1	Source:	C:\usr\Vic\Docencia\Seminario PCBs\Laboratorio\Demo\KiCadDemo v5\KiCadDemo\Demo.sch						
2	Date:	18/09/2018 11:03:42						
3	Tool:	Eeschema (5.0.0)						
4	Generator:	C:\Program Files\KiCad5\bin\scripting\plugins/bom_csv_grouped_by_value.py						
5	Component Count:	59						
6								
7	Individual Components:							
8								
9			Reference(s)	Value	LibPart	Footprint	Manufacturer	Name
10			C1	10 uF	Device:CP	Capacitors_SMD:CP_Elec_6.3x5.8	Kemet	EDK106M063A9GAA
11			C2	10 uF	Device:CP	Capacitors_SMD:CP_Elec_6.3x5.8	Kemet	EDK106M063A9GAA
12			C3	10 uF	Device:CP	Capacitors_SMD:CP_Elec_6.3x5.8	Kemet	EDK106M063A9GAA
13			C4	10 uF	Device:CP	Capacitors_SMD:CP_Elec_6.3x5.8	Kemet	EDK106M063A9GAA
14			C5	100nF	Device:C	Capacitors_SMD:C_0805		
15			C6	100nF	Device:C	Capacitors_SMD:C_0805		
16			C7	100nF	Device:C	Capacitors_SMD:C_0805		
17			C8	100nF	Device:C	Capacitors_SMD:C_0805		
18			C9	100nF	Device:C	Capacitors_SMD:C_0805		
19			C10	20pF	Device:C	Capacitors_SMD:C_0805		
20			C11	20pF	Device:C	Capacitors_SMD:C_0805		
21			C12	100nF	Device:C	Capacitors_SMD:C_0805		
22			C13	100nF	Device:C	Capacitors_SMD:C_0805		
23			C14	100nF	Device:C	Capacitors_SMD:C_0805		
24			D1	BAT60J	Device:D_ALT	Diodes_SMD:D_SOD-323	ST	BAT60J
25			D2	BAT60J	Device:D_ALT	Diodes_SMD:D_SOD-323	ST	BAT60J
26			F1	Polyfuse	Device:Polyfuse	Resistors_THT:R_Axial_DIN0204_L3.6mm_D1.6mm_P5.08mm_Horizontal	TE	RXEF025
27			J1	Power	Connector_Generic:Conn_01x02	Demo:Power	IMO	20-101M/2
28			J2	22-05-7025	Connector_Generic:Conn_01x02	Demo:IO2	Molex	22-05-7025
29			J3	22-05-7025	Connector_Generic:Conn_01x02	Demo:IO2	Molex	22-05-7025
30			J4	22-05-7025	Connector_Generic:Conn_01x02	Demo:IO2	Molex	22-05-7025
31			J5	22-05-7025	Connector_Generic:Conn_01x02	Demo:IO2	Molex	22-05-7025
32			J6	Comm	Connector_Generic:Conn_01x03	Demo:IO3	Molex	22-05-7025
33			J7	USB_B_Mini	Connector:USB_B_Mini	Connectors:USB_Mini-B	Molex	67503-1020
34			J8	Debug	Connector_Generic:Conn_01x06	Demo:Debug	Amphenol	T821106A1S100CEU
35			L1	5.6 uH	Device:L	Demo:tck141	Traco	tck141
36			LD1	Red	Device:LED_ALT	LEDs:LED_D5.0mm	Multicomp	703-0100
37			LD2	Yellow	Device:LED_ALT	LEDs:LED_D5.0mm	Multicomp	703-0098

You should have two lists of components in the document:

Individual Components has one line for each component sorted by the reference name

Collated Components has one line for each group of components that share the same value

Now, generate a **Demo BOM.pdf** file from the spreadsheet in **PDF** format. The following image shows an example of the *collated components* list page.

Collated Components:

Item	Qty	Reference(s)	Value	LibPart	Footprint	Manufacturer	Name
1	4	C1, C2, C3, C4	10 uF	Device:CP	Capacitors_SMD:CP_Elec_6.3x5.8	Kemet	EDK106M063A9GAA
2	8	C5, C6, C7, C8, C9, C12, C13, C14	100nF	Device:C	Capacitors_SMD:C_0805		
3	2	C10, C11	20pF	Device:C	Capacitors_SMD:C_0805		
4	2	D1, D2	BAT60J	Device:D_ALT	Diodes_SMD:D_SOD-323	ST	BAT60J
5	1	F1	Polyfuse	Device:Polyfuse	Resistors_THT:R_Axial_DIN0204_L3.6mm_D1.6mm_P5.08mm_Horizontal	TE	RXEF025
6	1	J1	Power	Connector_Generic:Conn_01x02	Demo:Power	IMO	20.101M/2
7	4	J2, J3, J4, J5	22-05-7025	Connector_Generic:Conn_01x02	Demo:IO2	Molex	22-05-7025
8	1	J6	Comm	Connector_Generic:Conn_01x03	Demo:IO3	Molex	22-05-7025
9	1	J7	USB_B_Mini	Connector:USB_B_Mini	Connectors:USB_Mini-B	Molex	67503-1020
10	1	J8	Debug	Connector_Generic:Conn_01x06	Demo:Debug	Amphenol	T821106A1S100CEU
11	1	L1	5.6 uH	Device:L	Demo:tck141	Traco	tck141
12	2	LD1, LD6	Red	Device:LED_ALT	LEDs:LED_D5.0mm	Multicomp	703-0100
13	3	LD2, LD4, LD5	Yellow	Device:LED_ALT	LEDs:LED_D5.0mm	Multicomp	703-0098
14	1	LD3	Green	Device:LED_ALT	LEDs:LED_D5.0mm	multicomp	703-0097
15	2	M1, M2	BSS84	Device:Q_PMOS_GSD	TO_SOT_Packages_SMD:SOT-23	ON	BSS84
16	6	MOV1, MOV2, MOV3, MOV4, MOV5, MOV6	MCFT000215	Device:Varistor	Varistors:RV_Disc_D7_W3.4_P5	multicomp	MCFT000215
17	6	R1, R4, R6, R8, R9, R10	330R	Device:R	Resistors_SMD:R_1206		
18	2	R2, R11	100k	Device:R	Resistors_SMD:R_0805		
19	2	R3, R5	4k7	Device:R	Resistors_SMD:R_0805		
20	1	R7	120R	Device:R	Resistors_SMD:R_1206		
21	1	SW1	Reset	Switch:SW_Push	Demo:B3F-4000	Omron	B3F-400
22	1	U1	TSR_1-2450	Regulator_Switching:TSR_1-2450	Converters_DCDC_ACDC:DCDC-Conv_TRACO_TSR-1	Traco	TSR_1-2450
23	1	U2	UA78M33	Regulator_Linear:uA7805	TO_SOT_Packages_SMD:SOT-223-3_TabPin2	Texas Instruments	UA78M33
24	1	U3	STM32F103CBT6	MCU_ST_STM32F1:STM32F103CBTx	Housings_QFP:LQFP-48_7x7mm_Pitch0.5mm	ST	STM32F103CBT6
25	1	U4	DS1822-PAR	Sensor_Temperature:DS1822-PAR	TO_SOT_Packages_THT:TO-92_Inline_Wide	Maxim	DS1822-PAR
26	1	U5	MAX3486	Interface_UART:MAX3486	Housings_SOIC:SOIC-8_3.9x4.9mm_Pitch1.27mm	Maxim	MAX3486
27	1	U6	MC74LCX0	74xx:74LS06	Housings_SOIC:SOIC-14_3.9x8.7mm_Pitch1.27mm	ON	MC74LCX0
28	1	X1	8 MHz	Device:Crystal	Crystals:Crystal_HC18-U_Vertical	Mercury	HUS-8

The printing should use columns wide enough so that all text is visible

Do all previous steps so that you can generate the proper ***BOM*** spreadsheet and the **Demo BOM.pdf** file.

Check #4: BOM

Upload the **Demo BOM.pdf** file, in the proper **Atenea** task of the seminar.

5. Drawing the layout

DESIGN PHASE 4

This is the main phase on a PCB design. Placing the components and drawing the tracks.

5.1 KiCad layers

Before dealing with the layout, it is important that you understand the layers in KiCad. A PCB is basically a stack of layers. Some layers are useful for fabricating the board. Some layers are useful for the board assembly. Finally, some layers are useful only for documentation purposes.

KiCad can use a maximum of 50 layers in three categories: Copper layers, Technical layers and Auxiliary layers.

Copper layers: There can be a maximum of 32 copper layers. Two of them are the external copper layers ***F.Cu*** (Top) and ***B.Cu*** (Bottom). The rest are internal layers from ***In1.Cu*** (closer to Top) to ***In30.Cu*** (closer to Bottom).

Technical layers: There are 14 technical layers. Most of them are grouped in pairs as they are associated to one of the two board sides Top and Bottom. Only two layers are not paired:

Edge.Cuts: This layer defines the board edge and any cutout inside the board. No component or track can be placed outside the board edges.

Margin: This layer defines a keep out margin next to the edges where no components can be placed. Currently we don't use this layer.

The rest of layers are given in pairs where F means Front or Top and B means Bottom. Six of them are used to design and fabricate the PCB:

F.Mask and B.Mask: Solder mask that define the locations where the lacquer layer that protects the PCB should be opened. It usually has elements over the soldering pads of the components. It is also opened on test points.

F.SilkS and B.SilkS: Silk screen. This layer defines text or drawings that will be printed on the two PCB surfaces. As this printing adds to the cost, it is habitual to only provide silk screen on the sides that has components.

F.CrtYd and B.CrtYd: Courtyard. We know about those layers from the footprint design. This layer indicates the space each component occupies so that we can keep the proper clearances.

Six layers are not used during the PCB fabrication but on the PCB assembly of components or for documentation purposes:

F.Fab and B.Fab: Fabrication assembly. We know about those layers from the footprint design. This layer indicates the assembly companies the

component orientations on the board so they can properly adjust the place machines. Those layers are also used to generate the board documentation as they give the most close to life drawings of the components.

F.Paste and B.Paste: Solder paste. This layer indicates where soldering paste will be applied during the assembly. It usually contains elements on the soldering pads of SMD components. Those layers are typically used to fabricate a solder paste stencil.

F.Adhes and B.Adhes: Adhesive. When using wave soldering, we apply solder by running the bottom surface of the PCB against a solder wave. If this surface features SMD components they need to be glued so that they don't fall before the soldering. This layer is used to indicate where the glue should be applied. We won't use those layers.

Auxiliary layers: There are four auxiliary layers. The usage of those layers is up to the designer although sometimes the PCB manufactures gives specific requirements about them. We won't use any of those layers.

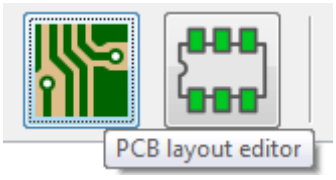
Dwgs.User: Drawings layer. This layer contains any drawing that can help to understand anything about the PCB.

Cmts.User: Comments layer. This layer contains any text comment that can help to understand anything about the PCB.

Eco1.User and Eco2.User: Those are user defined layers. They can serve any purpose the designer chose. Sometimes the manufacturer request specific information to be placed on those layers.

5.2 Initial configuration

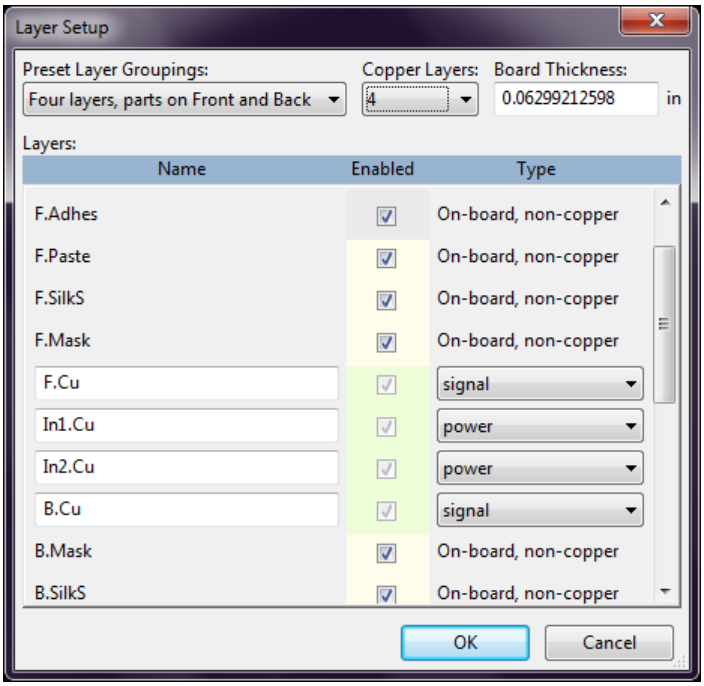
Before editing our project PCB we need to configure some options.
Open the ***PCB layout editor***.



Like in the case of the footprint editor, it is possible that you get a message about enabling **graphics card acceleration**. You can enable it if you wish. You can always turn off the acceleration, if you need, selecting ***Legacy Toolset*** inside the preferences menu.

Select ***Layers Setup*** on the ***Setup*** menu. A window will open.

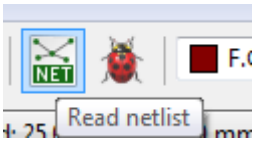
Remember that our ***Stack-up*** has four layers, select 4 on the ***Copper Layers*** section. Remember also that we used the top and bottom layers for signals and the inner ones for power. So select the appropriate type for each copper layer.



You can indicate the board thickness but this is not critical for the design of the PCB. In the above figure, the values is the typical default one of 1.6 mm.
Click ***OK*** to close this window.

You can also disable the ***F.Adhes***, ***B.Adhes***, ***Margin*** and the four ***User*** layers as we won't use them. However, this is not a requirement as we can just ignore them.

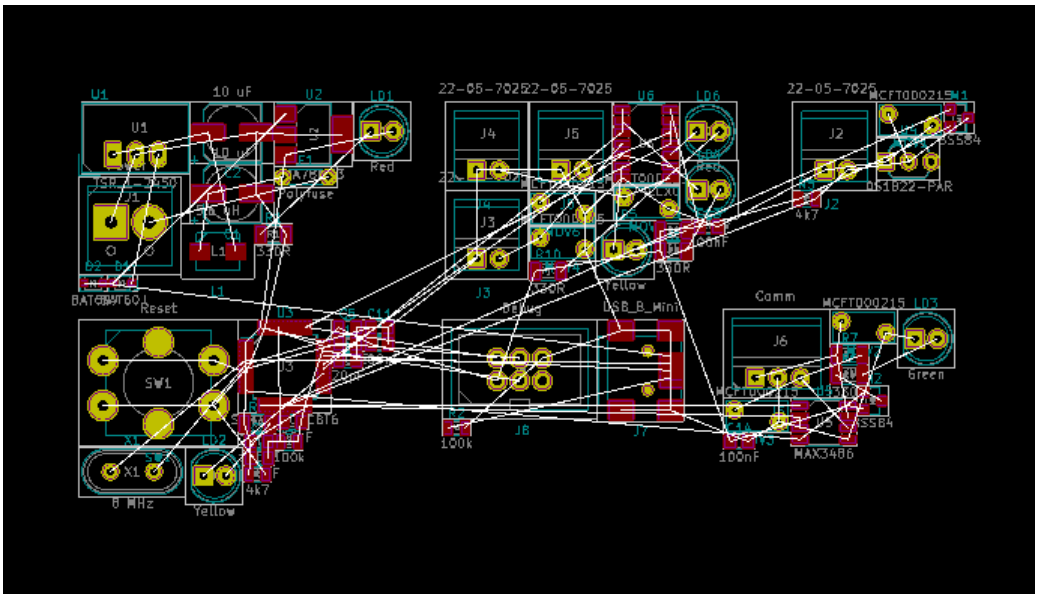
Now we can load the project ***netlist*** on ***PCBnew***. Just click on the ***Read netlist*** button and select ***Read Current Netlist*** in the window that will open.



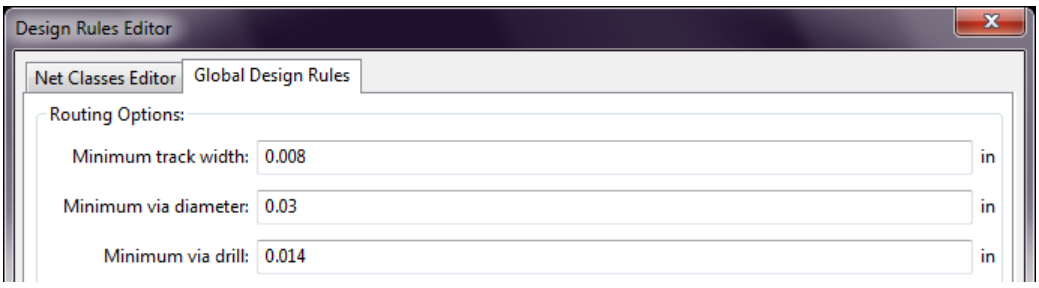
After reading the netlist there should be no errors. Ask the lab professor if you get any.

The ***netlist*** generated at the end of the schematic capture stage contains two main elements.
First element is a ***list of components***, each one with its reference and associated footprint.
Second element is a ***list of nets***, each one associated to a set of component terminals.
You can add new components on the PCB editor and manually set the net name for each component terminal. It is not the recommended way to work, but it could be useful for adding a few components to existing nets.
The proper way to add components is, however, by returning to the schematic editor and reloading a new netlist.

You should see all the project components close to each other like in the figure below. We will deal with that later.

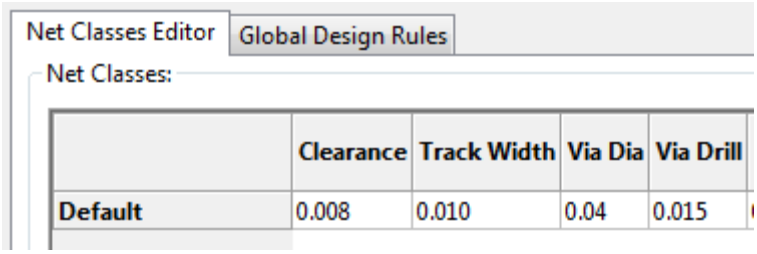


To complete the configuration, open the ***Design Rules Editor*** by selecting ***Design Rules*** on the ***Setup*** menu. Then go to the ***Global Design Rules*** tab on the window that will open. Afterwards fill the minimum track width, via diameter and via drill to comply with the fabrication capabilities shown on Appendix C using the values shown below. Don't bother with micro vias as we don't allow them.



After doing the changes, return to the ***Net Classes Editor*** tab to change the default settings. We will set the track width to 10 mil, just over the minimum. We will set the track clearance (spacing)

to 8 mil so that the editor won't complain with the MCU fine pitch package. Via diameter (40 mil) and Via drill (15 mil) is also just over the minimum. Don't change micro vias as we don't use them.

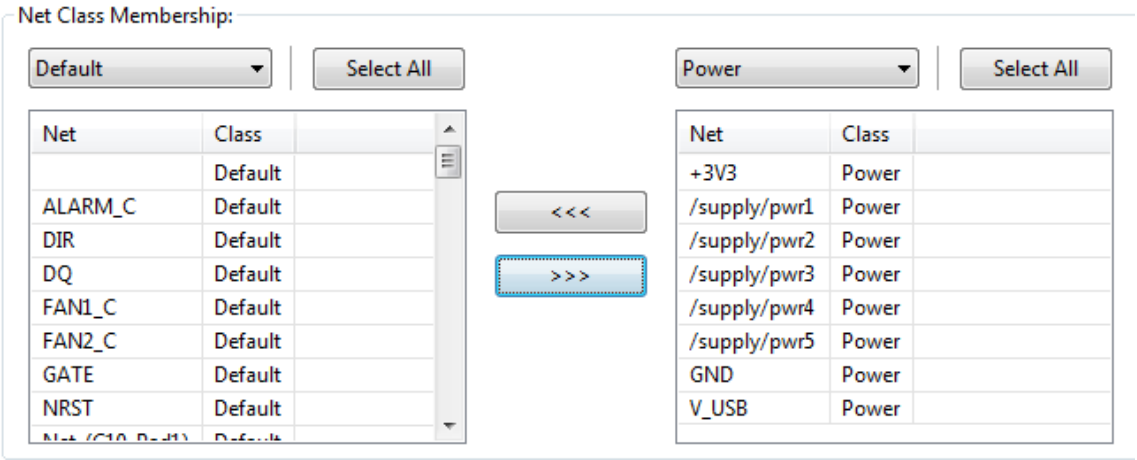


	Clearance	Track Width	Via Dia	Via Drill
Default	0.008	0.010	0.04	0.015

The above configuration is only for the normal tracks. We will use a special configuration for power tracks, so click on the ***Add*** button to generate a custom net class.
Give it ***Power*** as the name and set the values shown below. Basically we double the track width. Set also the ***Diff Pair Width*** to 0.008 for now so that the program does not complain.

	Clearance	Track Width	Via Dia	Via Drill	µVia Dia	µVia Drill	Diff Pair Width
Default	0.008	0.010	0.04	0.015	0.011811(	0.0039370(	0.008
Power	0.008	0.020	0.08	0.03	0.011811(	0.0039370(	0.008

After defining the ***Power*** custom class, we need to associate all power nets, including ***GND***, to this class. Do that with the lower region of this window. After you finish, just click ***OK***.



As this is only a demo, we are just doubling the width, diameter and hole for the power tracks. In a real design you shall calculate the tracks and vias to be able to withstand the current on them. You can use the ***PCB calculator*** available on the KiCad main window to do ease the calculations.

Open it and see that a 20 mil track could be adequate for a 1 A current but a 10 A current will require a much bigger track width.

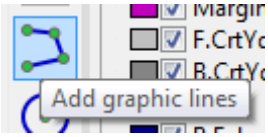
 Perform the configuration of the PCB and the loading of the netlist.

That should end the PCB configuration stage. It is time now to do some drawings.

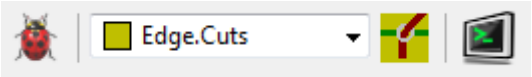
5.3 Board limits

There are two main ways to define the PSB board limits: ***After*** or ***Before*** component placement. In a test design it is possible to place and route all components and then define the PCB size around them ***afterwards***. This won't usually work on product design. There are usually some components that we will call ***fixed components***, like buttons, connectors and lights that are physically related to the product enclosure. You could design the PCB and then design the enclosure around it, but that requires more time as the enclosure design cannot start until the PCB is designed. If you want to design the enclosure at the same time than the PCB, you need to define the fixed component positions ***before*** the PCB or the enclosure is designed. You will need also to define the board dimensions ***before*** the PCB is designed.

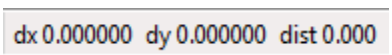
In our case, to comply with the mechanical data on Appendix B, we need to define the board size ***before*** placing the components. As the size is 100 mm x 80 mm, select dimensions on mm and a grid of 5 mm. Then select the ***Add graphic lines*** tool.



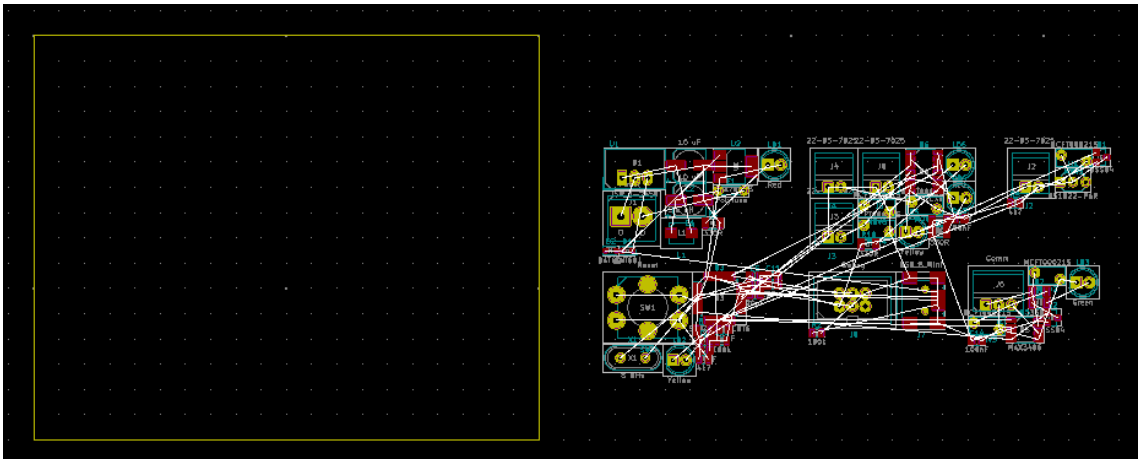
Afterwards, select the ***Edge.Cuts*** layer that is used to define the board limits.4



Then draw a box of 100 mm x 80 mm. You do that by clicking on the upper left corner and then drawing the four sides of the board in sequence. To end the drawing, use the right mouse button and select ***End drawing***.
If you hit the ***space*** key while the cursor is on the first corner of the board, all relative positions shown on the lower side of the window will refer to this point. That eases drawing the board to the required dimensions.



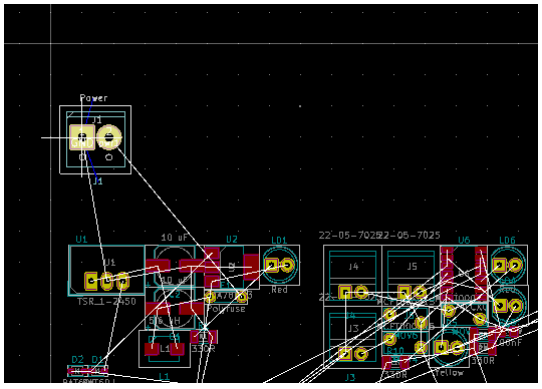
Now you should have the proper bounding box for the board. In general, it is good to move all the components out of the board space.



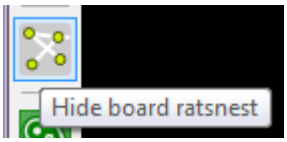
Define the board limits for the PCB.

5.4 Fixed elements placement

Now it is time to place the fixed component at their positions. Refer to ***Appendix B*** to place all 14 fixed position components in the place they should be.
As there are a lot of footprints, it is difficult to select a particular component. To ease the task, you can use the ***Get and Move Footprint*** command on the right mouse button contextual menu or just use the ***"t"*** key on the keyboard.
Make sure that the ***F.CrtYd*** courtyard layer is visible as we shall make sure that no component invades the courtyard of another component.
Let's start with the power connector J1. Hit "t" and fill in **J1** in the window that will open. The footprint for J1 will be selected and you will be able to move it inside the window.



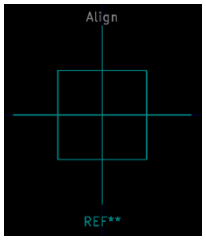
Notice two lines that go from the Pads on J1 to the other components. This is the ***ratsnest*** that enables you to know how are the Pads on different footprints connected. As we don't need that now, you can remove those white lines by unselecting the ***ratsnest***.



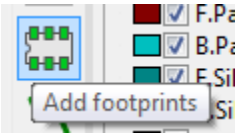
You can also ***rotate*** the component while moving by using the key **"r"**.
If you want to place the component you are moving, just hit the left mouse button. To ***move*** the footprint below the cursor, just use the ***"m"*** key. Note that you can move the footprint but you can also independently move the two text regions. Sometimes a disambiguation window will show up when the program is not able to determine what element you are trying to select.

Placing a component in a exact position is complex because the (0,0) coordinate for a component is the center of the first Pad and you don't usually know the distances of this Pad to the component edges.

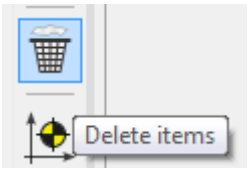
One way to ease the positioning of components is to create an alignment footprint that only has lines and no Pads at all. The center of this footprint shall be the (0,0) position so that you can place the component wherever you want to set a target location for a component. Below you can see an example of a footprint named ***Align*** that measures 20 mm x 20 mm.



To place this aiding component in the design you can use the ***Add footprints*** tool.



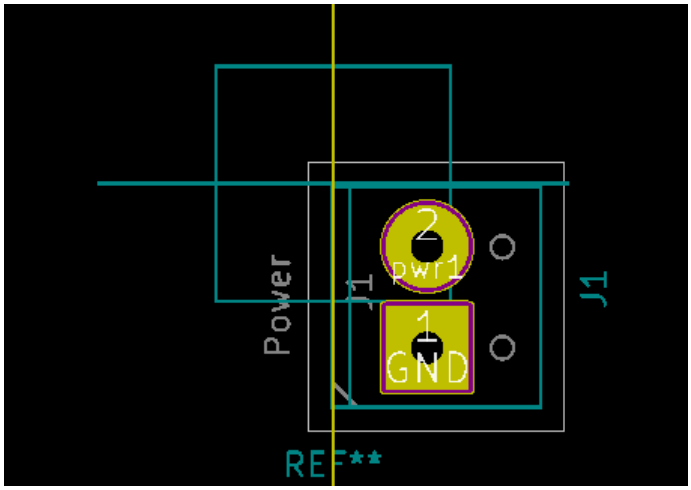
When you no longer need it, you can erase it with the ***Delete items*** tool.



Editing this component by hitting the **"e"** key when you are over it, you can change its location.

Just note that the position is ***absolute***, so you need to take into account the position of the upper left corner of the board.

If your upper left corner is at (x,y) = (50 mm,50 mm), in order to place J1 at (0 mm, 15 mm) you can place the Target at (50 mm,65 mm). Then, just move the connector to align it properly with the Target. Don't forget to use a grid fine enough to place the connector with an error below 1 mm. In the figure we only show the ***Silkscreen*** drawing. We could also use the ***Fab*** layer to align the component if it properly represents the edge to align. We cannot use, however the ***Courtyard*** box for alignment purposes as it extends outside of the component.

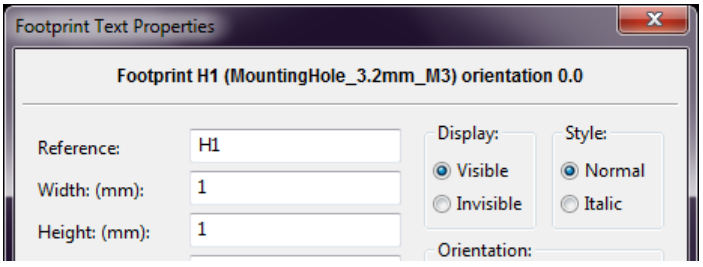


You are not required to define a Target footprint and it doesn't need to have this particular shape. Do whatever you need to place all fixed components at their locations.

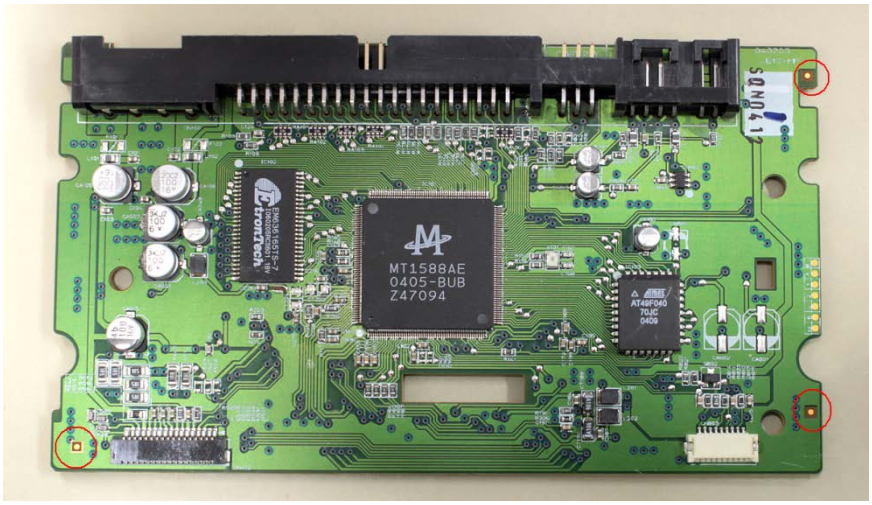
Note that not all footprints provided in KiCad feature all drawing layers. Some of them can miss a layer like ***Courtyard*** or ***Fab***. As they are not really required to fabricate the board you can just ignore the omission. If that bothers you, feel free to create new footprints from the ones provided in KiCad and include them in your ***Demo*** library.

After you have placed the 14 fixed components you need to place the four M3 mounting holes. Use the ***Add footprints*** tool to add the four mounting holes. Library is ***Mounting_Holes*** and footprint name is ***MountingHole_3.2mm_M3***. As those are centered components, you can just use the edit window to place them in the proper location.

In order to identify properly the mounting holes edit their properties and set the ***reference*** field to values from ***H1*** to ***H4*** for the four holes.



Now it is time to place the ***fiducials***. The ***fiducials*** are markings on the component side of the board that are used to calibrate the pick and place machine during the assembly stage. For manually place and solder, they are not really needed.
The following figure shows an example of ***fiducial*** marks, inside the three red circles, on a HDD controller board.



IPC documents **2221**, **7251** and **7351** recommend using three ***fiducial*** marks to correct translation offsets, rotation and non linear distortions on three corners of the board as separated as possible.

The optimum marks are 1 mm diameter circles on the component side copper layer with a minimum 2 mm diameter (3 mm preferred) clearance to the ***Soldermask***. The background shall be the same for all marks so the same copper layers shall be below all marks.

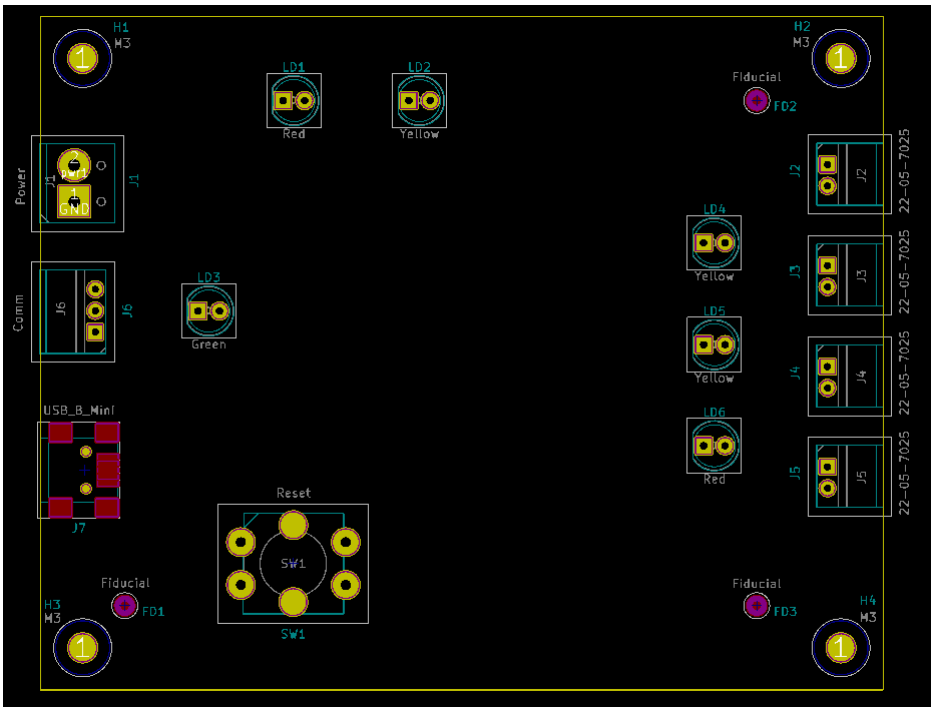
There is a specific footprint library ***Fiducials***. From that, we will select the ***Fiducial_1mm_Dia_2.54mm_Outer_CopperTop*** fiducial and place in the three positions indicated on the Appendix B document.

Note that there is no exact location for the fiducials, but they should be more or less on the indicated location. Also, FD1 and FD3 shall have the same Y position and FD2 and FD3 shall have the same X position.

Don't forget to change their reference to FD1 ... FD3.

Place the fixed components, the mounting holes and the ***fiducials***.

After the above steps you should have a board like the one in the figure below.



Note that we have changed the **value** of the M3 holes to ***M3*** and the value of the Fiducials to ***Fiducial***.

Although they are not shown in the figure, the rest of components should be close together somewhere.

5.5 Placing the rest of components

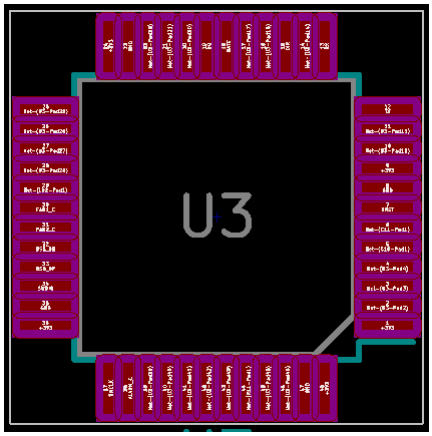
The next stage is a critical one. The 14 components we have placed before are easy to place because they are at fixed locations and we cannot place them in any other location. In the case of LEDs and the push button we can rotate them, but that's all we can change. The problem is placing the rest of the components because you have infinite options and they strongly affect the rest of the PCB design and operation:

- Placement affect the routing stage
- Placement affect the operation of the circuit

The component placement is one of the stages that make PCB design something like **"Art"**. You need experience to do that properly as the number of variables you can change is staggering high. However, there are a set of rules that can aid in the place stage:

- Follow the signals. From one connector to other, the components need to flow the path of the signal. That means that components that are connected together should be next to each other. Use the ***ratnest*** to follow the footprint connections.
- Place critical components first. Those are the components whose bad placement can mostly affect the circuit operation.
- Leave decoupling capacitors to the end but put them as close to the power pins as possible. Note that the U3 microcontroller has one capacitor ***C5..C9*** for each power pin **VDDA**, **VBAT** and **VDD_1...VDD_3**. You cannot use the ***ratnest*** to place those components because they could end all together instead of each one just next to the power pin they belong to.
- Oscillator components shall be close to the MCU. Place the footprints ***X1***, ***C10*** and ***C11*** as close to ***U3*** as possible to prevent adding stray capacitances to the crystal nodes.
- Keep high speed lines short and unobstructed. Try to have a clean path from the MCU ***U3*** to the USB and Communication connectors. The same applies to the serial debug lines.
- Keep the proper clearances between components. That means that the ***courtyard*** boxes of two components shall not overlap. Be sure to have the ***courtyard*** layer visible.

The MCU ***U3*** deserve an additional consideration. Place it in the orientation shown below.



That way, the USB lines will be on the left side so they will align properly with the USB connector.

Note that all ***U3*** MCU power pins are connected to the 3V3 rail.

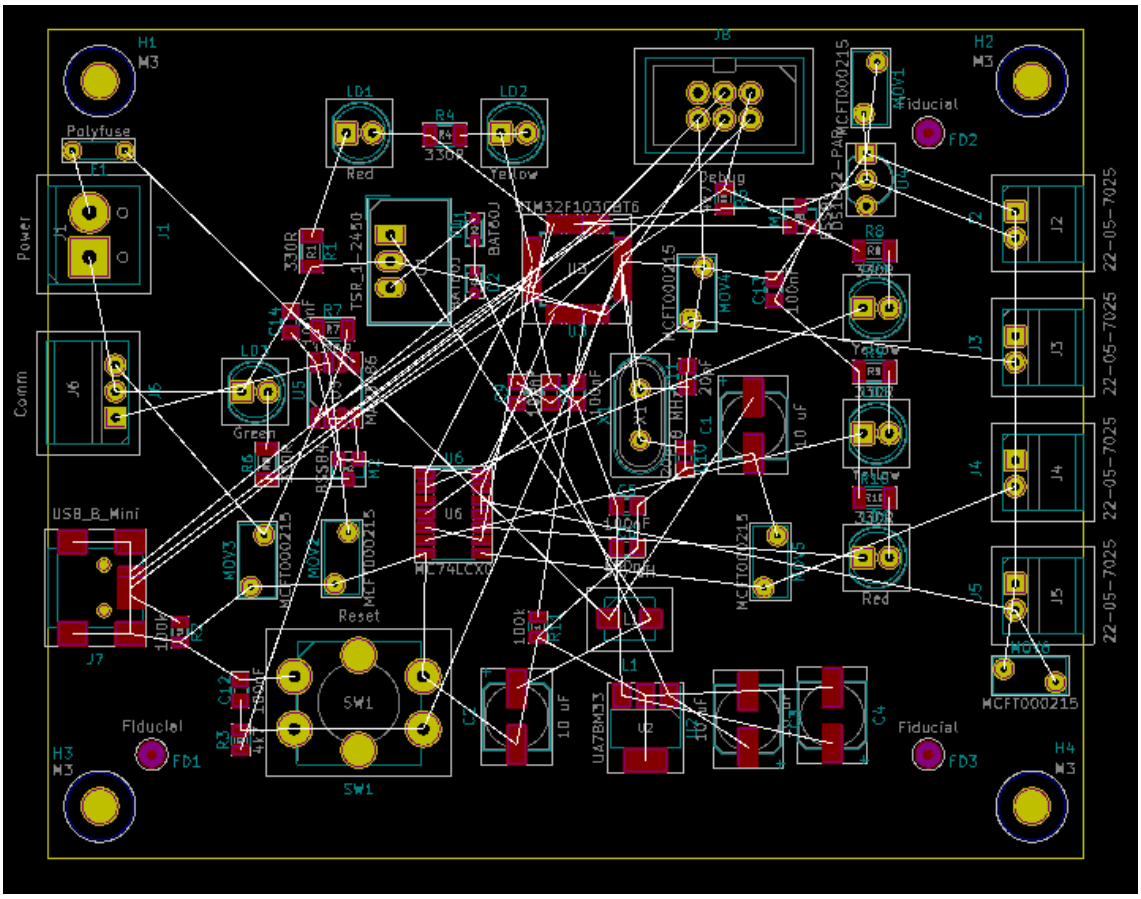
You cannot identify VDDA as different from VBAT, for instance. However, there are 5 capacitors C5..C9 for 5 power pins. Just assign one random capacitor to each pin and put it as close as possible to it but leave space to fan out the pin to the power plane.

The ***U2*** linear regulator has special requirements as we will see later. Leave some free space (about 10 mm x 20 mm) next to the big GND terminal of this regulator. If you are curious about it, read section 5.6 before starting the routing.

You don't need to place the components in the final exact position, you can always change the position before the final routing.

Place the rest of components in the design.

After you end the place stage you will get something like:



The component positions **shall** be different in your design.

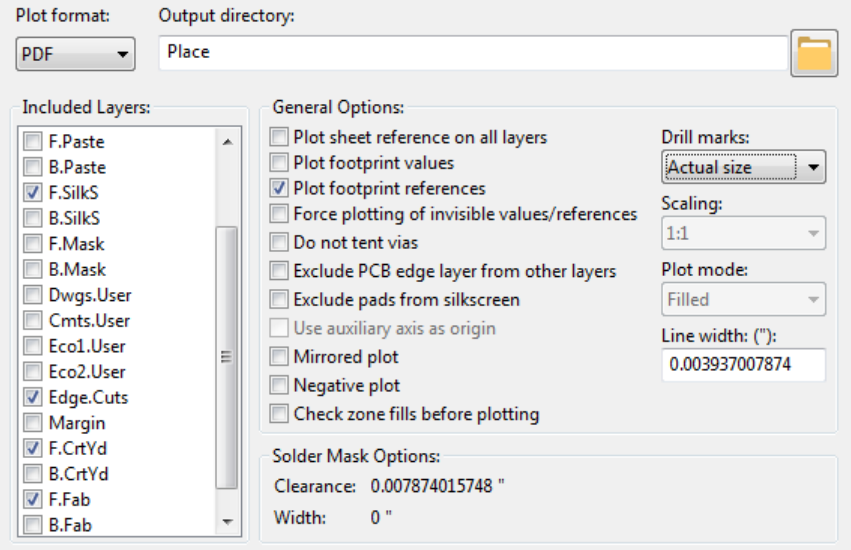
This is only one example. In fact, it is a very bad one. It should be evident that component placement is very bad. You should do much better.

The placement is not difficult in this design because there is plenty of space for all components. One way to know in advance how difficult will be to place the components is to calculate the total area associated to the components and compare it to the total board area.

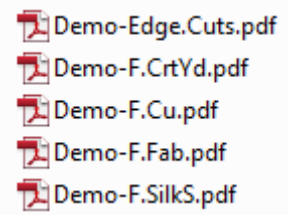
In our design, all the components together, excluding mounting holes and fiducials, occupy about 2230 mm^2. You can check this number from the sizes of all footprints. As the board is 100 mm x 80 mm, the total PCB area is 8000 mm^2. That means that the components take about 30% of the PCB area. For a four layer design this means that we have plenty of free space.

After ending the placement we will generate a plot of the board. Select ***Plot*** from the ***File*** menu. In the window that will open select the options shown below to generate a PDF of the ***F.Cu***,

F.SilkS, ***Edge.Cuts***, ***F.CrtYd*** and ***F.Fab*** layers and click on the ***Plot*** button. Note that the ***F.Cu*** layer shall be selected although it is not shown in the figure.



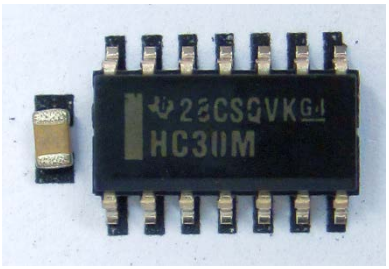
That will generate the following 5 files in the ***Place*** folder inside your project folder:



It is a good idea to examine those files to see how they look like.

Footprint check.

You should always check the components against their footprints. That means drawing the footprints at 1:1 scale and checking them against the real components both for the Pads and the bounding box. In the case of hand soldering, check that you have enough free space on the Pads for your soldering skills.

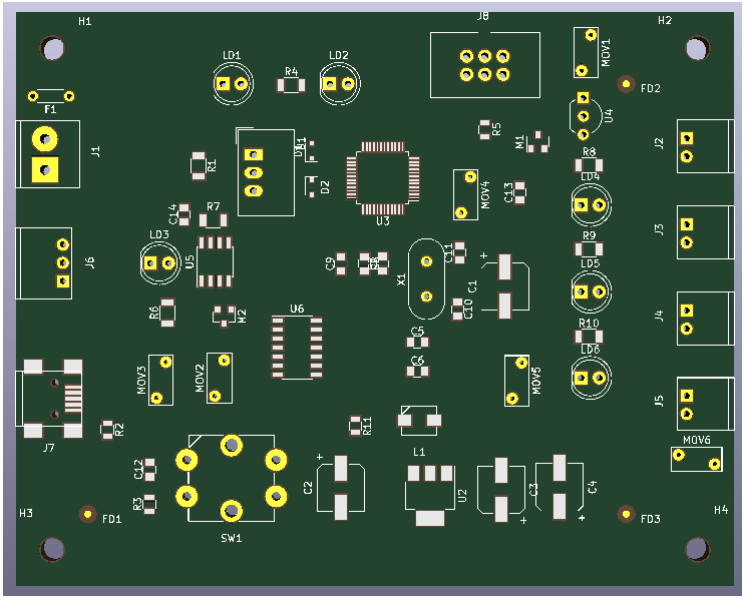


If you have not done that before, the end of the placement stage is a good place to that check because any future footprint change will be much more difficult. Use a print of the ***F.Cu*** layer PDF file for that.

As this is a *Demo* project you don't have the components at hand so you will probably skip this step, but it is very important in real projects.

If you want to check this step, the Lab professor has some of the real components available. Ask for them.

We can also obtain a 3D view of the board. Just select ***3D Viewer*** on the ***View*** menu. Note that the board below has **very bad placement**. Don't use it as a reference.



The 3D viewer is not perfect. The 3D models or components sometimes are missing or plain wrong although they improve on newer KiCad versions. That's way in the above image the components models have not been used.

Obtain a good frontal view of your board and generate a PNG file from it using the proper command of the file menu. Call this file ***"Demo Place 3D.png"*** and put it inside the ***Place*** folder.

☑ **Check #5: Placement**

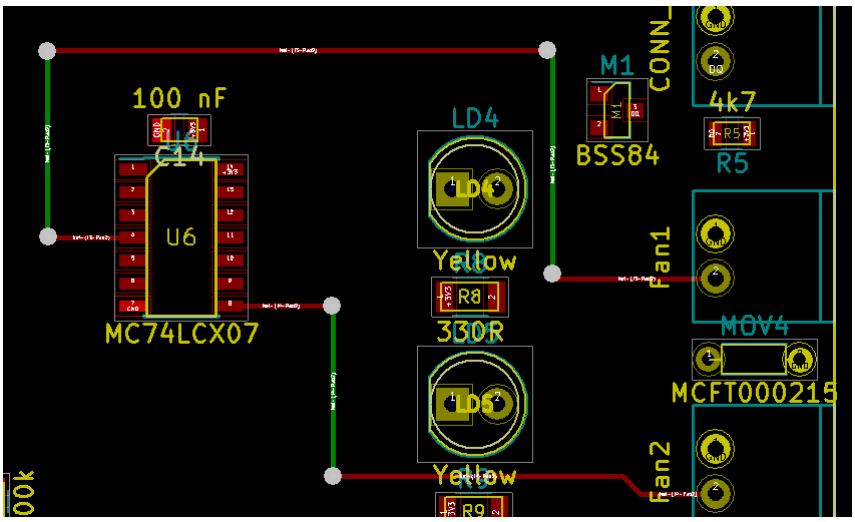
Show the placement to the lab professor if available.
Take all the files inside the ***Place*** folder, compress them inside a ***Place.zip*** file and upload it on the proper task in **Atenea**.

5.6 Define a routing strategy

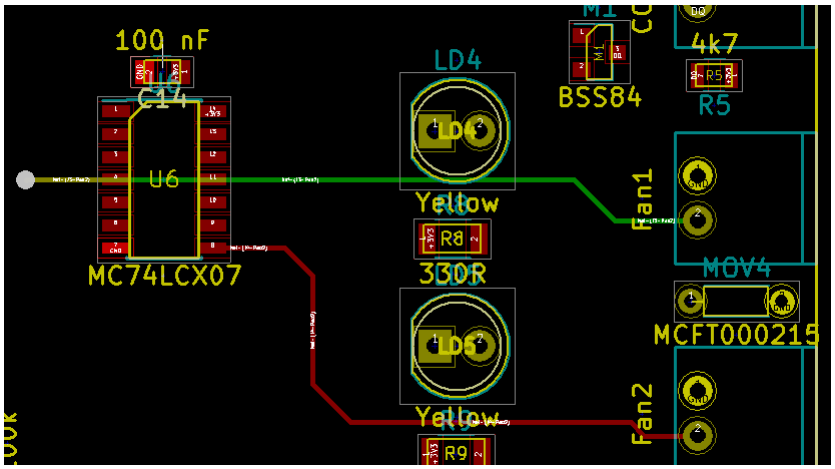
The next design stage is the route stage, but before we start the routing we need to do some planning.
As we remember we fabricate a four layer board. The two outer layers Top and Bottom will be used for routing and the two bottom layers will be used for the **+3.3V** and **GND** nodes.

For the two routing layers there are several options to choose. One is to use one layer to draw horizontal lines (Top in red, for instance) and another to draw vertical lines (Bottom in green, for

example) like in the figure below. That way the probability of having track collisions is minimized.



This solution, however, can generate more vias than needed. It could be a good solution to a very dense design but our design doesn't really require that, at least for short distance tracks. Compare the above solution to the one below. You can see that now we use only one via instead of six. Moreover, the tracks are shorter.



Although we don't require using different layers for horizontal and vertical tracks, this method can come handy in long tracks. If you don't plan in advance the routing of long tracks, you can have problems completing the routing. Using one layer for horizontal and one for vertical may help in this case. As we commented before, you don't need to use this dual layer routing for local tracks, just for long tracks that could yield routing problems if not planned in advance. As connectors are in the left and right side of the board, most tracks ending on connectors will be horizontal, so we will use the ***Top*** layer for horizontal long tracks and the ***Botton*** layer for vertical long tracks.

As our design has several SMD components that will be placed on the top side, all tracks that land on SMD Pads will exit them on the ***Top*** side. So we can define the Top layer as our main routing layer for local, or short distance, interconnects. Then define the bottom layer as an auxiliary layer to solve whatever is difficult to be done locally only with the Top layer.

If we use mainly the Top layer, it is best to use the Inner 1 layer, the one closer to the Top, for GND as this will provide a proper ground plane. That gives the following distribution of copper layers:

Copper Layer	Usage
Top	Main local routing / Horizontal long tracks
Inner 1	GND Plane
Inner 2	3V3 Vdd Plane
Bottom	Auxiliary local routing / Vertical long tracks

After defining the plane usage, we need to plan the routing. It is a good idea to define a routing grid so that all tracks align with it. As our default tracks are 10 mil thick, we can choose a 10 mill grid for general routing.

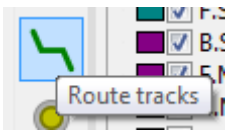
The plan is to route first the most important tracks and leave the less important to the end. The latter you draw a track, the easier is to need vias and alternate layers to draw it. In general it is good to have the most important track as free of vias as possible. A good routing order for our design is:

1. Route the power supply tracks
2. Route the MCU oscillator
3. Route the MCU power pins
4. Route the communication lines (USB and RS-485)
5. Route the debug tracks
6. Route the one wire tracks
7. Route the rest of tracks

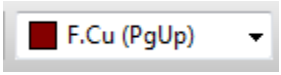
Having a routing strategy gives a cleaner design. As this demo design is not very demanding, there are several valid routing strategies. But you need to choose one and stick to it.

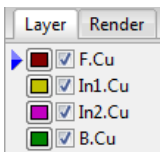
5.7 Routing the power supply

Once all components have been placed and a routing strategy is defined, it is time to route them by adding the tracks that connect them. To do that, you will use the ***Route tracks*** tool.



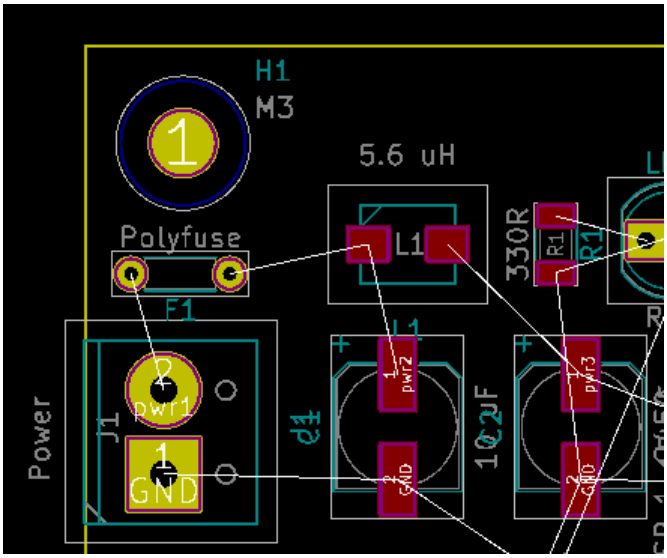
You will use also the ***layer selection box*** or the ***Layers Management Toolbar*** at the right of the window.



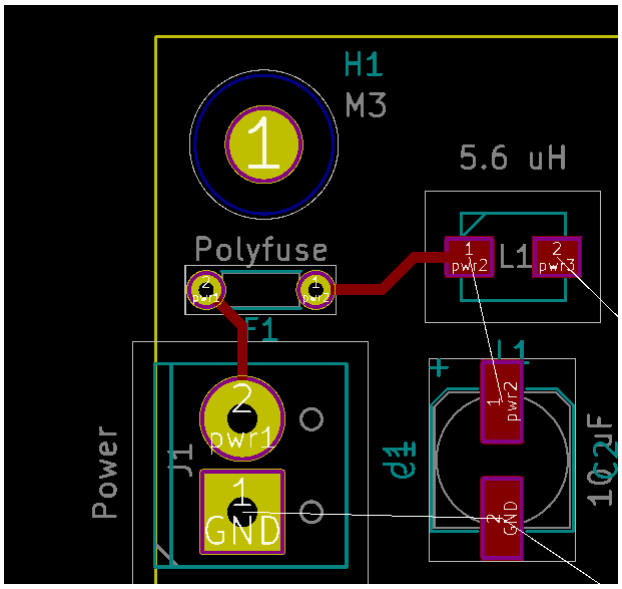


The route stage consists on drawing all tracks that connect the components together. It is based on eliminating all ***ratnest*** lines by substituting them with real copper tracks. The routing ends when there are no ***ratnest*** lines left.

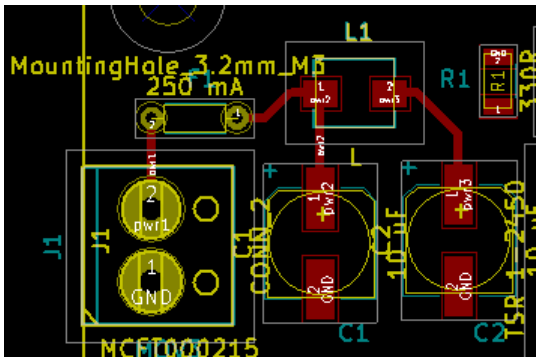
Let's say that in the layout below we want to connect the power lines between J1, F1, L1, C1 and C2.



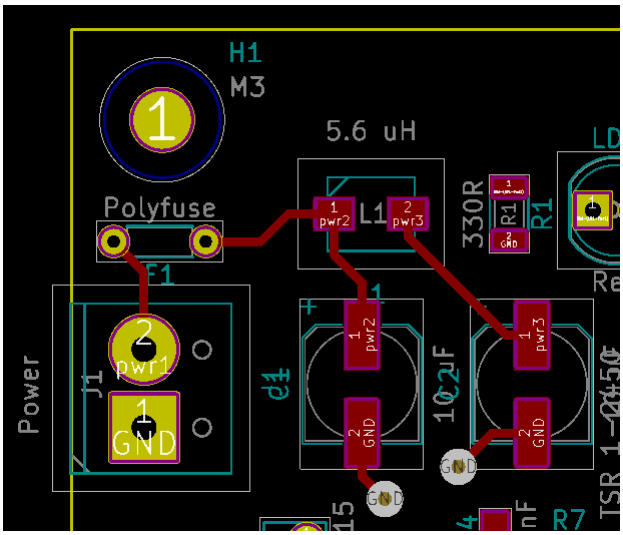
We will lay the tracks on the top copper layer, so we select ***F.Cu*** on the ***layer selection box***. Then we hover the mouse over the pin 2 of J1. We click over the pad and drag the line that is created to reach the fuse F1. Then we click on the pad. As we have ended this track we can use ***End Track*** on the right mouse button menu. That will give something like in the figure below.



We can do the same for the other components until we obtain something like in the figure below. If you have properly set the track classes in the configuration stage, the tracks should feature the proper width that is 20 mill in this case as those are power tracks.

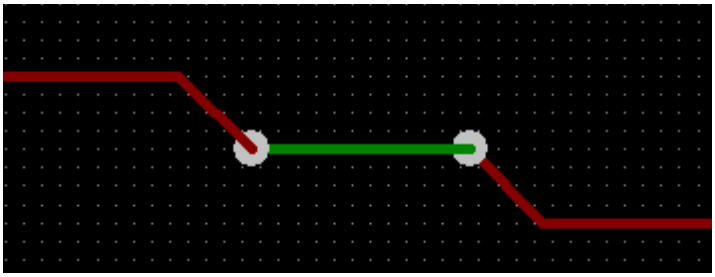


For connections to +3V3 or GND the solution is different because those nets belong to copper planes. You just need to fan-out the nodes so they end on vias or holes as in the figure below.



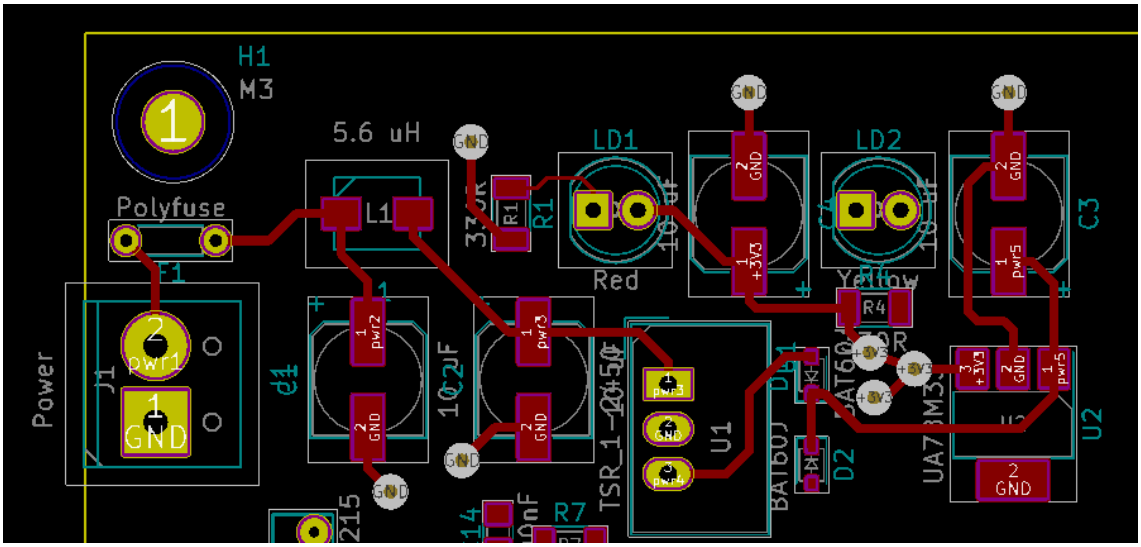
To place a via when you are drawing a track, just hit the "**v**" key on the keyboard. In the above figure we have just drawn tracks from the GND pins of C1 and C2 that end on vias. In the case of J1 we don't need any track for GND at all because the pin is already ***though hole***.

When you use "**v**" to place a via PCBnew automatically changes the layer between ***Top*** and ***Bottom***. That eases drawing tracks. In the figure below we were laying a track on ***Top*** (Red), then we used "**v**" to place a via and switch to the ***Bottom*** (Green) layer. Then, again, we placed a via to return to the ***Top*** layer.

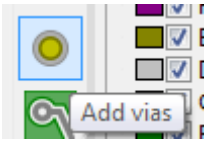


When you are drawing a track, hitting the left mouse button sets a vertex on the track. That comes handy when drawing long tracks. Note that the normal configuration limits the tracks to have a multiple of 45º angle.

In order to complete the power supply routing you need to set the power supply path from ***J1*** to the ***U2*** output. As the ***U2*** output defines the 3.3V node, it will connect to the Vdd plane. To guarantee that there will be no voltage drops on the vias, end the track that exits ***U2*** with three or four vias, all coming from a Top layer track. The figure below shows an example of routing for the power supply. Note how all ***GND*** connections end on vias or through hole terminals.



There is a specific tool for placing vias.



In general you don't need to use it. If you place a via in a location that is not connected to any track, it won't have an associated node and you will not be able to connect to it before editing the via and setting the proper net node for it.

You can change the footprint positions, if needed. But if you do that after the routing, you will need to previously erase the track that connects to the footprint.

You can probably route the power supply using only the ***Top*** layer. If you use the ***Bottom*** layer, remember that SMD components need to be accessed from the Top layer whereas though hole components can be accessed both from the **Top** and ***Bottom*** layers.

5.8 Thermal dissipation on U2

We have not yet ended the supply routing. Remember that we requested to have some space around ***U2*** in a previous section. Now we will make use of it.
Linear regulators generate heat. The power dissipated by a linear regulator can be approximated as:

$$P_{Heat} = (V_{in} - V_{out})I_{out}$$

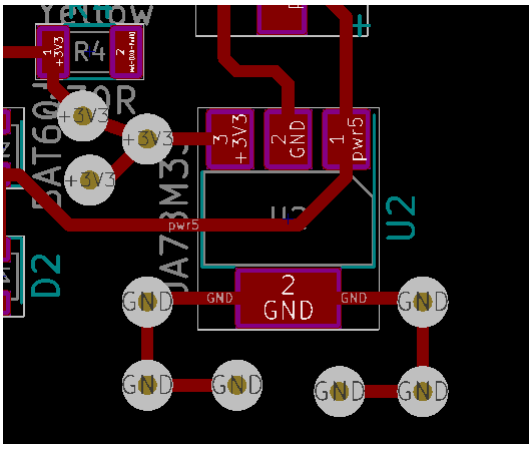
Notice that ***U2*** features two GND pins: 2 and 4. The big one, pin 4, also doubles as the thermal link to the lead frame that holds the integrated circuit.

SOT-223 packages are not very good to dissipate power by themselves. If you search in the ***U2*** regulator datasheet you will find that the thermal resistance between the silicon die and the ambient is 53ºC/W. Let's say you have a 200 mA output current, as the regulator converts from 5 V to 3.3 V, the dissipated power will be 340 mW and the temperature increase will be 18 ºC.
That means that if ambient temperature is 30 ºC, the dice will be at 48 ºC. The datasheet states that the maximum dice temperature shall be 150 ºC so we have still a lot of margin.

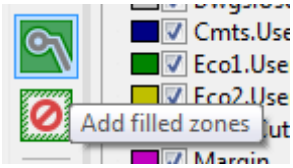
As our design don't require high currents, we will be fine although the package is not very good dissipating power. But, as we are learning about PCB design, we will design a thermal path for the regulator. The idea is to send the heat to the GND plane at one of the inner layers.

Thermal design is outside of the goals of this seminar so we will take the thermal design of ***U2*** as a requirement we shall fulfill. Our request is to send the heat generated by ***U2*** to the GND layer through the parallel connection of six vias like the ones associated to the power nets (80 mil diameter and 30 mil hole). Each via will have some thermal resistance so having several of them reduces the total thermal resistance.

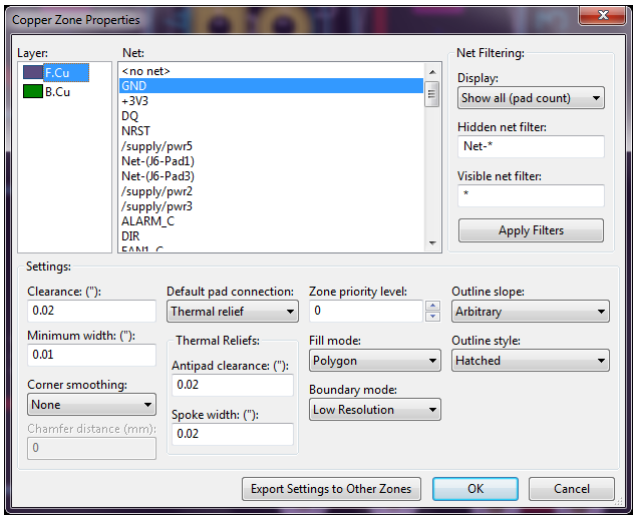
We will first add the six vias connected to the pad #4 using a copper track on the ***Top*** layer.



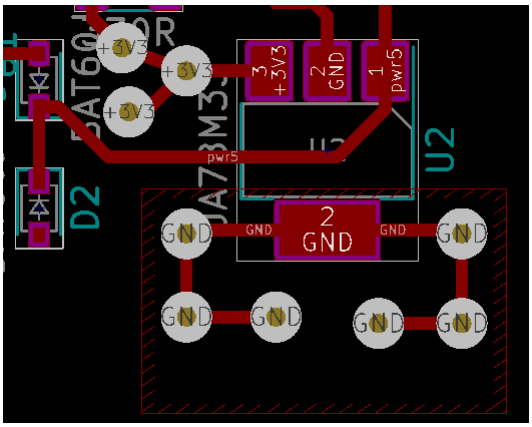
Then we will use the ***Add filled zones*** tool to add a rectangular copper area surrounding all vias.



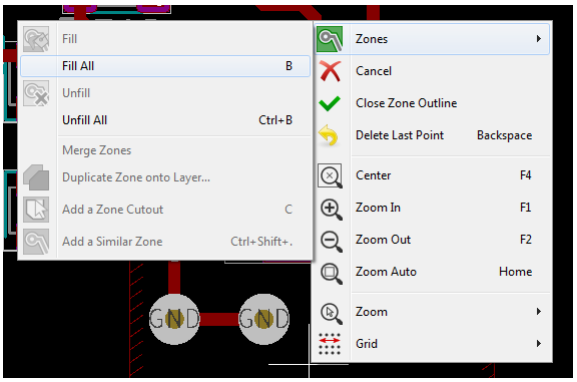
When you click to enter the first point of the copper area, a dialog will show up:



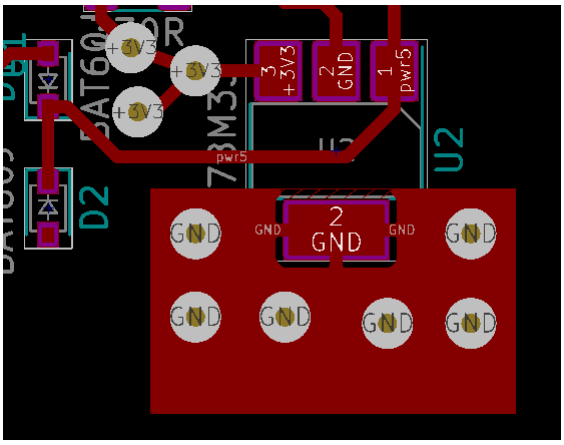
Layer should be ***F.Cu***, and the associated net should be ***GND***. Leave the rest of parameters as they are and click OK. Then enter the other corners and select ***Close Zone Outline*** in the right button menu to end the zone. You should get something like in the figure below:



Now you can fill the region using ***Fil All or*** from the right button menu.



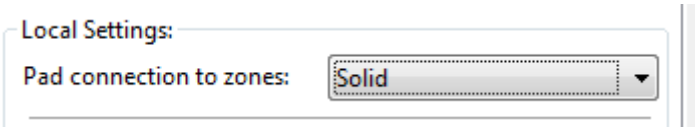
That will give something like:



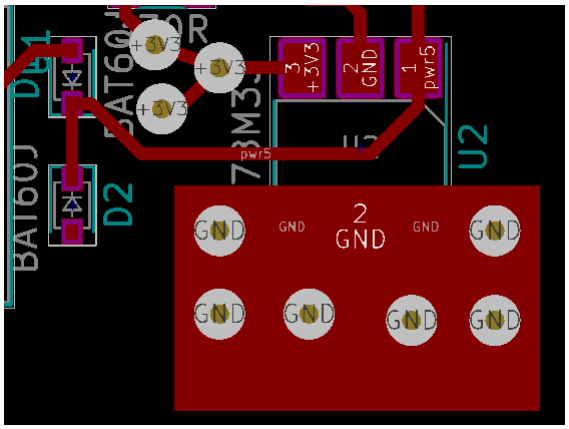
In order to show properly, the Show filled areas in zones needs to be selected on the left side icon strip.

You now have a solid copper region connected to six vias that will send the thermal power generated on the regulator to the ***GND*** plane. But there is something wrong. The ***U2*** Pad #3 connection to the copper area is not solid. It only connects with four lines to the top, bottom, left and right sides. This is a ***Thermal relief*** and is usually a good solution because it eases the soldering of the component but is not good for thermal dissipation. To do the proper connection we need to edit the properties of the ***U2*** footprint.

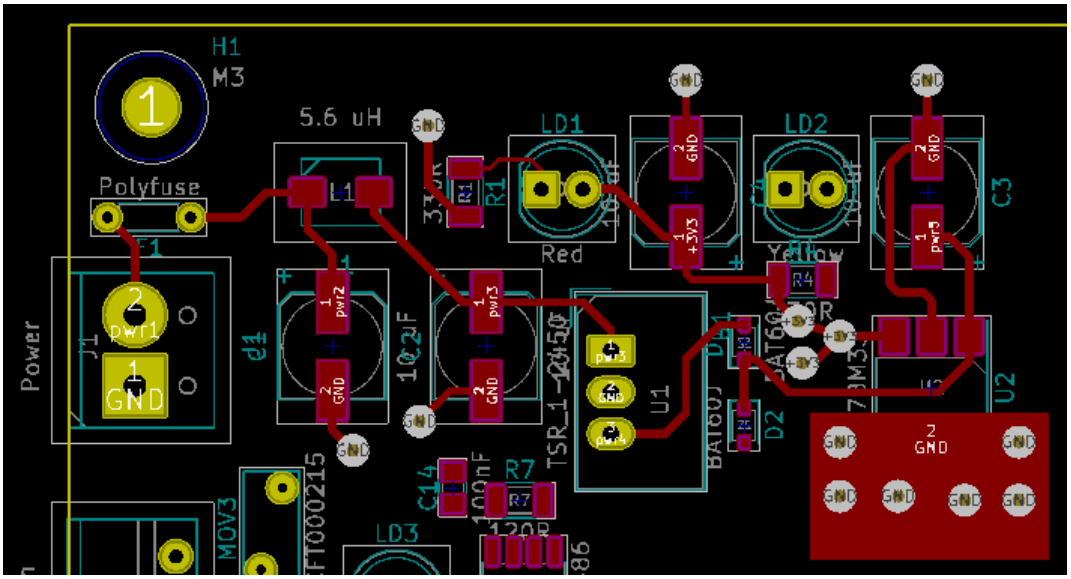
In the local settings section, change the Pad connection to zones from ***Themal relief*** to ***Solid***.



Now redo the filling of the region. You will get something like:



This gives a much better thermal coupling between the ***U2*** package and the copper region. And that ends the supply routing.



Route the power supply components between ***J1*** and ***U2*** output.

Route also the connections to the **Vdd** and **GND** planes and the ***U2*** thermal connection to the **GND** plane.

5.9 Routing the clock

After routing the power supply we will route the MCU oscillator. That includes the components ***X1***, ***C10*** and ***C11***.

Now there is a problem. The ***GND*** net belongs to the power category, so it uses thick tracks and vias. This is ok for the supply region of the board but we don't need such dimensions for the ***GND*** node on the decoupling capacitor tracks and vias. That means that we need **exceptions** to the rules.

In order to do that, click ***Design rules*** in the ***Setup*** menu and select the ***Global Desing Rules*** Tab. Here, define a custom via and a custom track equal to the default values.

Custom Via Sizes:

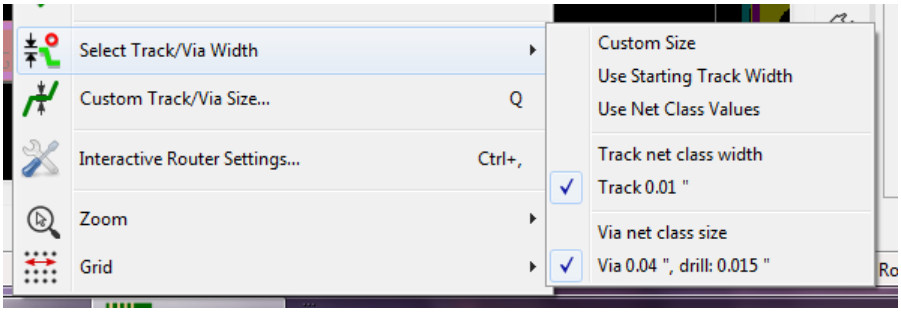
Drill value: a blank or 0 => default Netclass value

	Diameter	Drill
Via 1	0.015	0.04
Via 2		
Via 3		
Via 4		
Via 5		
Via 6		
Via 7		
Via 8		

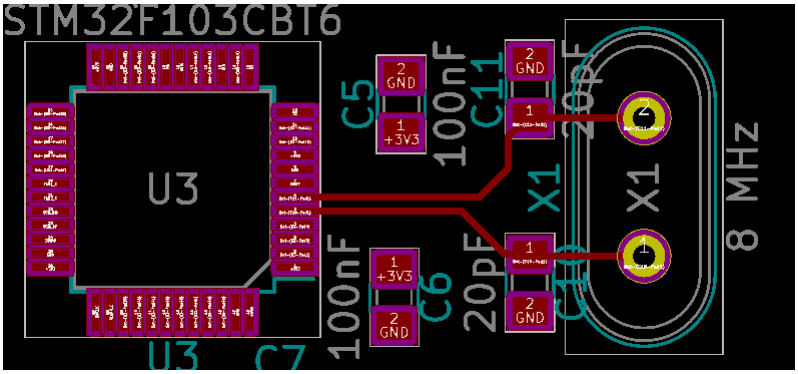
Custom Track Widths:

	Width
Track 1	0.01
Track 2	
Track 3	
Track 4	
Track 5	
Track 6	
Track 7	
Track 8	

Whenever you need to change the track size while routing, you can use the **Track 0.01"** width and **Via 0.004", drill 0.015"** override in the right button contextual menu. Just remember to set the ***net class*** option afterwards.



Now we can route the MCU oscillator. The following figure is an example solution, but it is not unique. Just keep the tracks short.



After routing the oscillator you can route the power tracks for the MCU. All Vdd pins should connect to one decoupling capacitor and the Vdd plane (using one via). The other terminal of the decoupling capacitors should connect to GND as all GND pins on the MCU.

If you look at the MCU you will see that you cannot exit from the GND and Vdd pins using the 20 mil thickness of the ***Power*** net class. So you will need to use the custom track width.

When you end drawing custom tracks or vias, remember to return to normal operation.

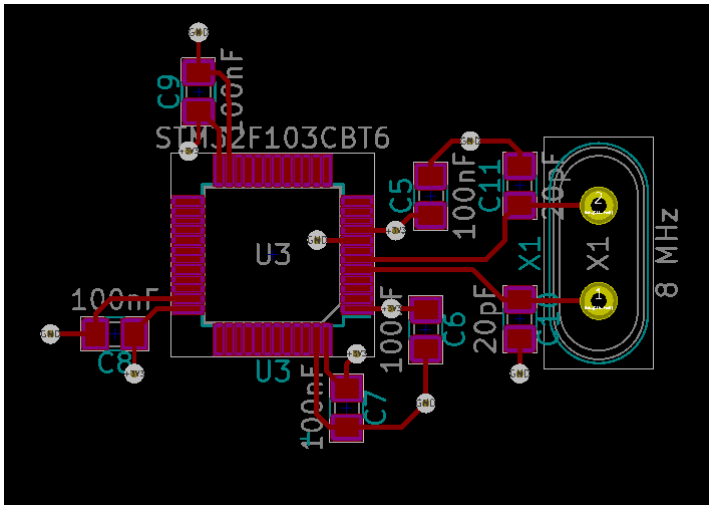
Use the right button menu to select the proper width or via on the ***Select Track/Via Width*** submenu.

The ***uses Net Class*** option will automatically select the track and via that corresponds to the net class of the track you are routing.

5.10 Routing MCU power pins

Having the **GND** and **3V3** nets on the Power class generates thick tracks and vias. This is ok for the power supply region, but is not the proper option for small components that connect to those nets. For those components use the custom 10 mil track and the ***custom via***.

After connecting the decoupling capacitors to ***U3*** we will have something like in the figure below.



Route the MCU crystal and power pins.

5.11 Routing the differential lines

Special tracks in a circuit can be differential lines, high speed lines or lines, like the USB ones, that are both differential and high speed. A low speed/frequency differential pair just needs the lines to be draw close together so that the noise is coupled to both lines in the same way. High speed lines are much more complex because they also require impedance matching.

Let's talk first about high speed and leave differential for later. The speed of a wave in a conductor depends on the dielectric constant around it and is about:

$$v = \frac{c}{\sqrt{\varepsilon_R}} = \frac{30cm/ns}{\sqrt{\varepsilon_R}}$$

There are two basic kinds of high frequency lines in PCBs, ***Microstrip*** on the external layers or ***Stripline*** on internal layers.

The ***Microstrip*** track is included in a non homogeneous FR4/air medium. But, as most field lines will go from the track to ground trough the FR4 layer, we simplify the calculations by considering only the dielectric constant of the FR4 material. That also, in the worst case, underestimates the signal speed and this is better that overestimating it.

The minimum wavelength of a signal limited by a BW bandwidth can be calculated from its frequency and signal speed:

$$\lambda_{min} = \frac{v}{BW} = \frac{c}{BW\sqrt{\varepsilon_R}}$$

For the case of FR4, that has a relative dielectric constant of about 4.5, we get a signal speed of about 14 cm/ns. We don't need to bother with high frequency designs if the linear circuit length, or track lenght, is much lower than the minimum wavelength. So, for a maximum circuit size ***"L"*** it means:

$$L \leq \frac{c}{20 \cdot BW\sqrt{\varepsilon_R}}$$

For an analog circuit it is easy to know the bandwidth. For a digital signal the bandwidth does not depend on the clock frequency but on the signals rise and fall times. For a symmetrical rise/fall signal, the equivalent bandwidth is:

$$BW = \frac{0{,}35}{t_R}$$

So, we can recalculate the previous expressions:

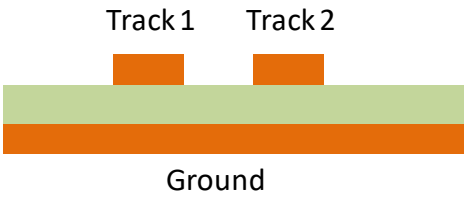
$$\lambda_{min} = \frac{t_R c}{0{,}35\sqrt{\varepsilon_R}} \qquad L \leq \frac{t_R c}{7\sqrt{\varepsilon_R}}$$

Our design features only Full Speed USB 1.1 operation. This is a speed low enough to have no problems drawing the tracks although we need to check that. In the case of High Speed (480 Mb/s) on USB 2.0 or Super Speed (5 Gb/s) on USB 3.0, the routing is much more complex to be done well.

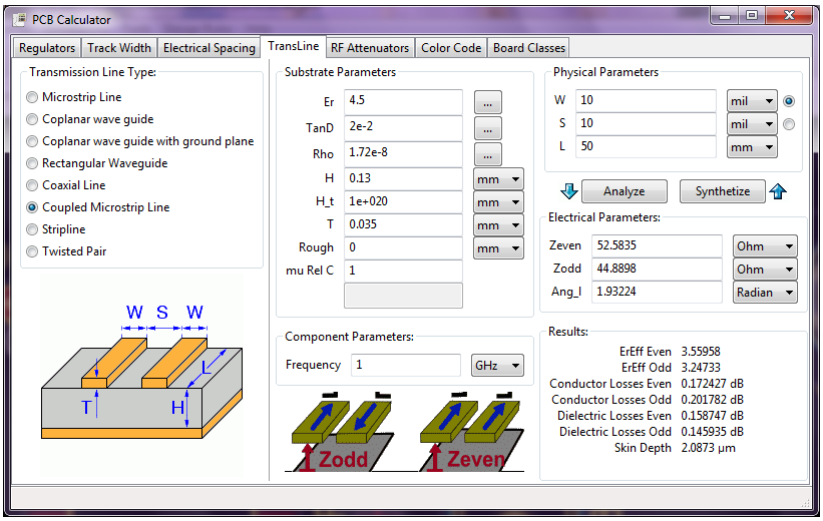
In the case of the USB full speed lines, the maximum bit rate 12 Mb/s. That gives 83 ns bit time. From the USB specs, the rise and fall times shall be between 4 and 20 ns. For the 4 ns fastest rise time, the associated length limit is 80 mm, just the short edge size of our board and less than the long edge of the board. So, our board is just on the limit of high frequency operation.

High frequency operation means that we need to use constant impedance tracks that are properly terminated on both ends. Neglecting this requirement yields reflections that degrade the quality of the signal. Although we are just on the limit of needing it, we will use high frequency tracks for the USB lines in our design. In our stack-up, that means using ***Microstrip*** because the dielectric thickness between the outer and inner layers is much smaller than between the two inner layers.

USB lines are not only high frequency but also differential. In our case we will use two ***Coupled Microstrip*** lines.



We not only need to keep constant the impedance between each line and ground but also between both lines. The critical parameter is the differential impedance of the lines than need to be 90Ω ±15% to comply with the USB specs. To calculate how the tracks need to be, you can use the ***PCB Calculator*** included in KiCad and open the ***TransLine*** tab.



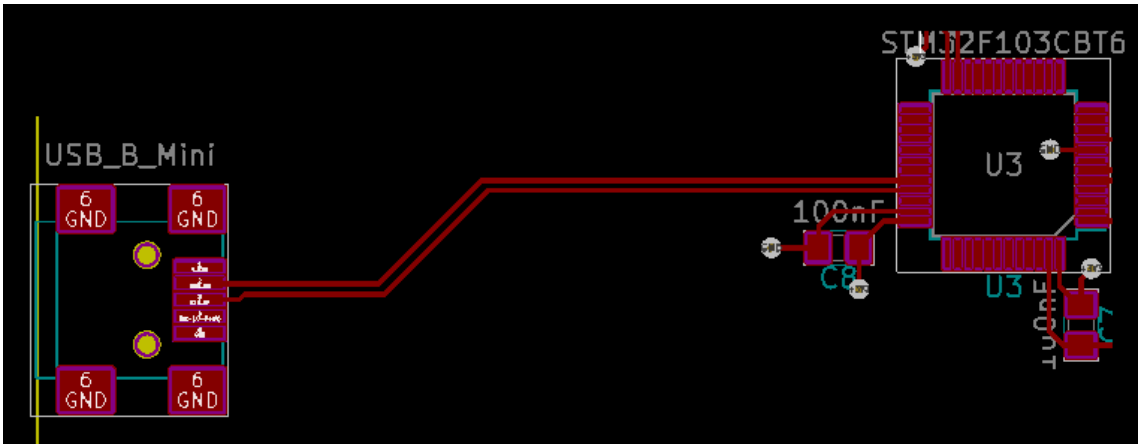
We select a ***Coupled Microstrip*** line as we will lay the tracks on ***Top*** over the ***Inner 1*** ground plane. Substrate parameters are the typical for FR4 (Er and TanD), and Copper (Rho).
H distance was defined on the stack-up to be 130 µm. T is the copper thickness that is 35 mm on the ***Top*** layer.
Setting **W** and **S** to 10 mil we get a **Zodd** of 44.89 Ω. As Zdiff is two times Zodd, we get a differential impedance of 89.78 Ω that is within specs.

In high speed differential lines, the tracks need to be drawn together so that not only the impedance is kept constant buy also so that the signals on both lines are time synchronized. The time error Δt for a difference in length ΔL can be calculated:

$$\Delta t = \frac{\Delta L}{v}$$

So, a 1 cm mismatch will give a 33 ps time skew on a typical ***Microstrip*** line. As we remember we have a 83 ns bit time in our USB connection. So it is difficult to have a length mismatch that will give any problem.

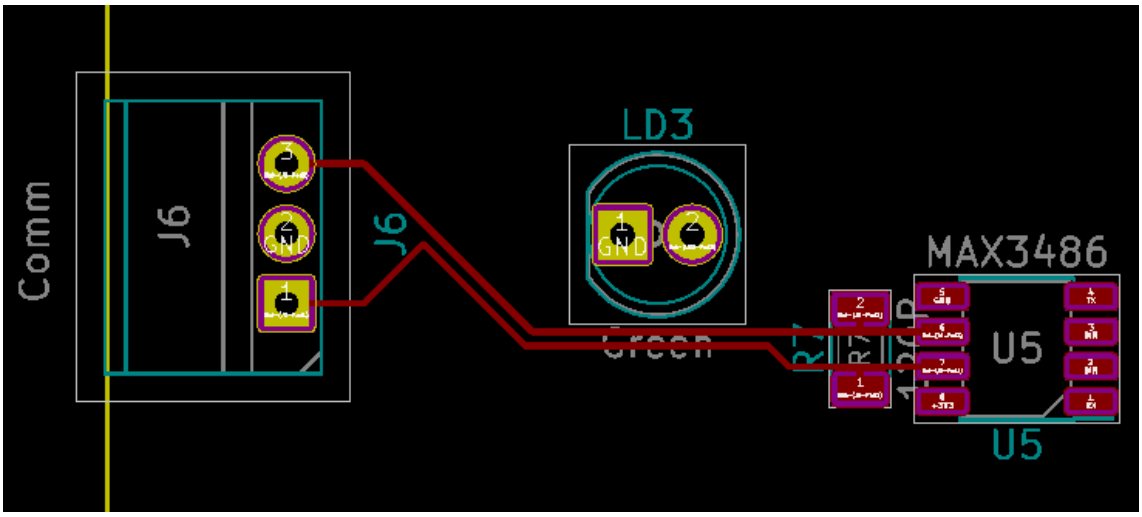
Now, after all the previous calculations, we can draw the track from the MCU to the USB connector. As explained, we will use 10 mil tracks with 10 mil spacing. The figure below shows one example of those tracks.



After drawing the USB tracks we can focus on the RS-485 lines. Those lines are meant for long distances and we don't expect to have a too high baud rate. If the communication port uses a 115200 baud rate, the symbol time will be 8,7 μs so it is difficult to have problems with time skews. If the rise time of the signal is 1/10 of the symbol time, then the rise time will be 870 ns and the equivalent bandwidth will be 402 kHz. The length limit in our circuit will be 17.6 m. As our PCB is 100 mm x 80mm we don't really need to bother with impedances for the RS-485 lines. As this is a tutorial, we will, however, work as if we need high speed operation.

The impedance requirements for the RS-485 lines that leave the driver ***U5*** are different from the USB case, but, as this is just a demo board and, also, as the data rate on this interface is not too high, we will just use the same 10 mil width/spacing as in the USB case.

Note that the connector uses pins 1 and 3 for the two differential lines. If the lines were connected to two pins close to each other, the solution will be better. Note also that the two differential tracks don't feature the same length. Depending on the bit rate this could be a problem but hardly in our case. The solution below gives an equalized line length although it is probably not really needed.



The RS-485 driver ***U5*** connects to the MCU using two single ended TX and RX lines. Route also those line minimizing the number of needed vias. In this case the line lengths don't need to be matched as the system has no synchronization between those lines.

If the RS-485 baud rate were high enough for requiring a high frequency design, so will be the baud rate on TX and RX. So those lines should also be properly terminated and drawn in constant impedance tracks.

Remember that we don't really need to consider the RS-485 lines as high speed but, if possible, try to draw the TX and RX lines with a continuous 10 mil track on the ***Top*** layer. Changing to the ***Bottom*** layer will reference the lines to the ***Inner 2*** Vdd layer, so good decoupling between the GND and Vdd layers is recommended, although not critical, in this case.

Route the two differential line pairs.

Route also the TX and RX connections that connect the ***U5*** driver and the MCU.

Special differential routing modes

KiCad now features some routing modes specifically designed to draw differential lines. To ease the tutorial, we won't use those modes but remember that they are available so they will be very helpful for differential intensive boards.

5.12 Last critical tracks

After drawing the differential lines we will route the lines that go to the debug connector. The most critical ones are SWCLK and SWDIO. NRST is not critical at all. It is possible that you cannot route those lines without any ***via***. It is no problem but try to minimize the number of vias on each track. As SWCLK and SWDIO are synchronous to each other, it is recommended to keep a similar path for both signals. The SWCLK clock signal tops at 50 MHz so the minimum period is 20 ns. Assuming a rise/fall time of 1/10 of the period, that gives 2 ns and is equivalent to a 175 MHz bandwidth and 4 cm length limit for requiring a high frequency design. Observe that those lines are much more close to operate at high frequency than the RS-485 lines.

Try to maintain the SWCLK and SWDIO lines as short as possible (if possible below the 4 cm limit) and also try to keep equal the line lengths and minimize, and make equal, the number of vias.

The last critical track is the one wire DQ track. Signals in the one wire bus have minimum edge distances of about 10 μs. So, we can assume a lower than 1 MHz required bandwidth. That means that we don't really require any high frequency design on this line. Anyway, although it is really not that critical we will draw it before the rest of tracks. If the MCU happens to be far from the ***J2*** connector, remember to use the ***Top*** layer for horizontal tracks and the ***Bottom*** layer for vertical tracks.

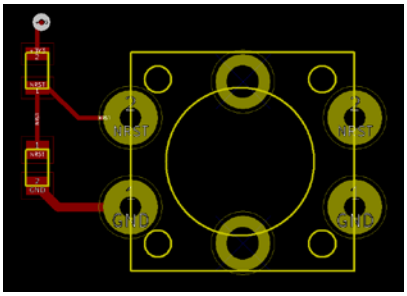
Route the SWCLK, SWDIO, RST and DQ lines.

5.13 Routing the rest of lines

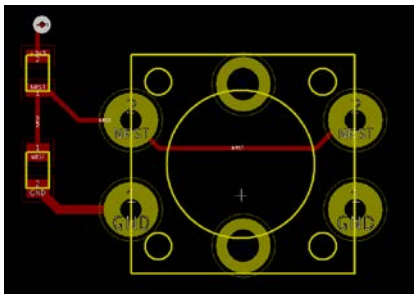
In order to route the rest of lines, you are on your own.

- Route long lines using mainly ***Top*** for horizontals and ***Bottom*** for verticals.
- Use the ***ratsnet*** lines to plan in advance the lines that cross with each other.
- Fanout all **GND** and **Vdd** connections to vias.

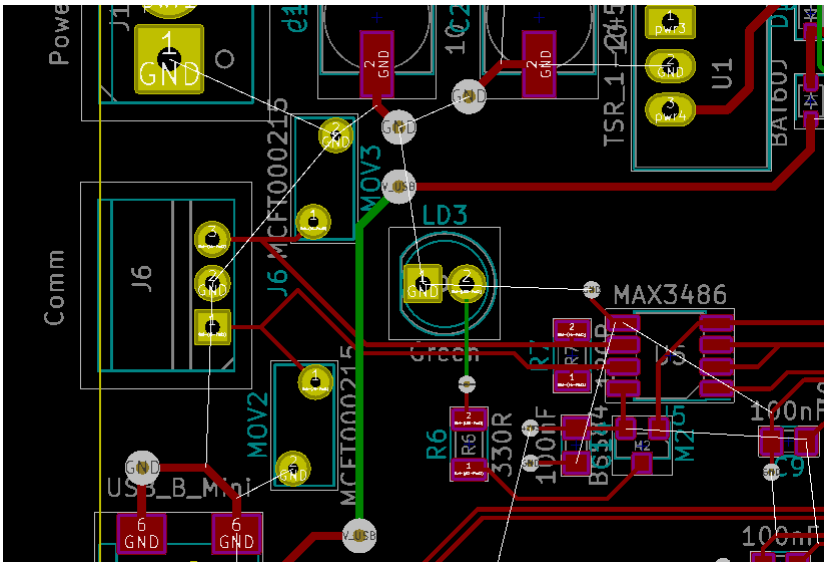
Notice that the two ***SW1*** terminals that connect to NRST are internally connected through the component. You can use the internal connection to route the NRST line like in the figure below.



But it is better to do the proper connection using the PCB tracks so that the proper connections don't depend on the assembly of the components. Perhaps sometime in the future it is decided not to populate the ***SW1*** footprint. You don't want to need a PCB rework in this case.



When you end this phase of routing the ***ratsnet*** lines should only connect Pads that either connected to Vdd (+3.3V) and GND like in the figure below.
Sometimes KiCad marks a terminal as unconnected, and do not erase its ***ratsnet***, when it is properly connected. In this case, you can just ignore the line or redraw the connection.

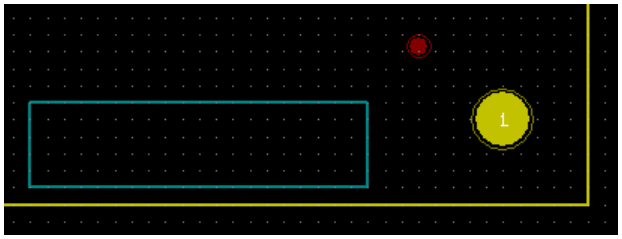


Route the remaining connections.

5.14 Drawing the power planes

After drawing the all connections routed on ***Top*** and ***Bottom*** and fanning out all ***Vdd*** and ***GND*** connections, it is time to draw the power planes.

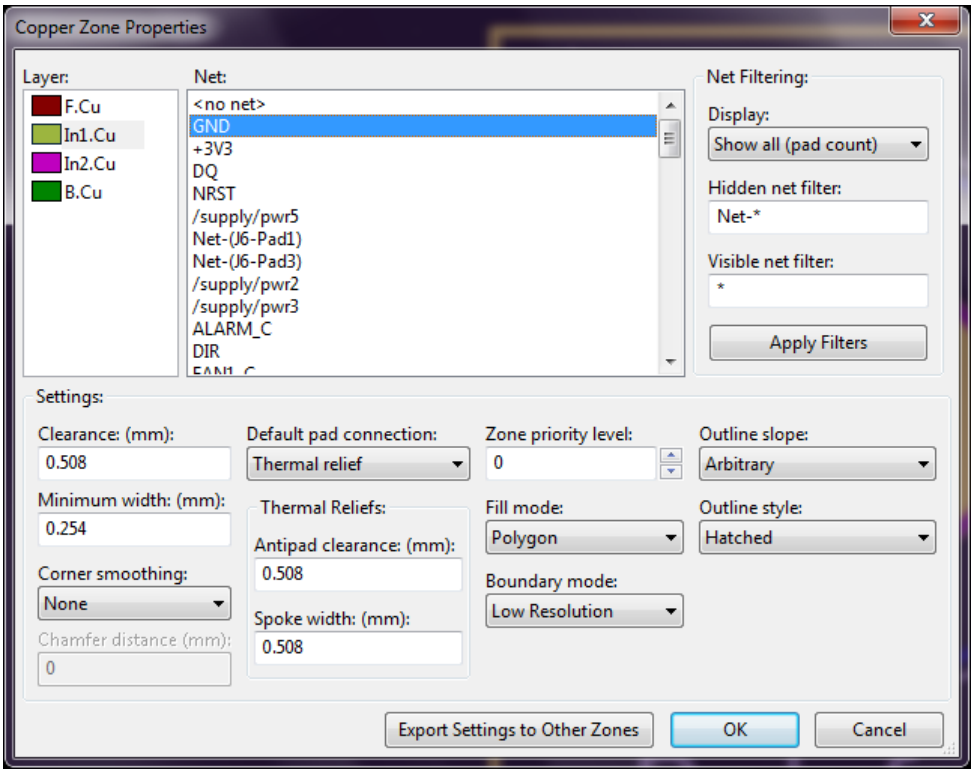
But, before that, we will draw, using the F.SilkS silkscreen layer a layer window box in the lower right region of the board like in the figure below. Line thickness shall be 0.2 mm, and box height and width shall be 5 mm and 20 mm. No copper shall be near closer than 1 mm to this box.



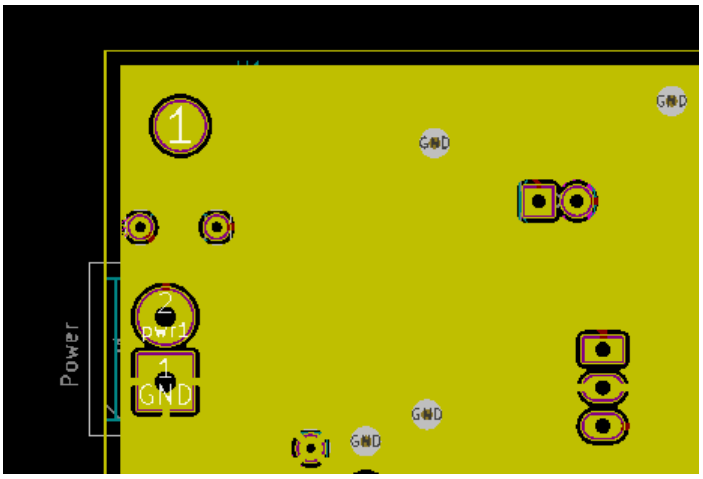
If you don't have space in this region, just place the box elsewhere along one of the sides of the board. This box will define the layer window. Then place the numbers 1 to 4 using copper layers ***F.Cu***, ***In1.Cu***, ***In2.Cu*** and ***B.Cu*** in this order like in the figure below.

The layer window is not mandatory but is useful so that we can see the distribution of layers in the future after the PCB is manufactured.

After defining the layer window we will draw the copper planes. First, we will start with the ***GND*** power plane in the ***Inner 1*** layer. Select this layer and the ***Add filled zones*** we used to draw the thermal copper region of the linear regulator, then, draw a rectangle region 1 mm inside the board limits but skip the previously drawn layer window.



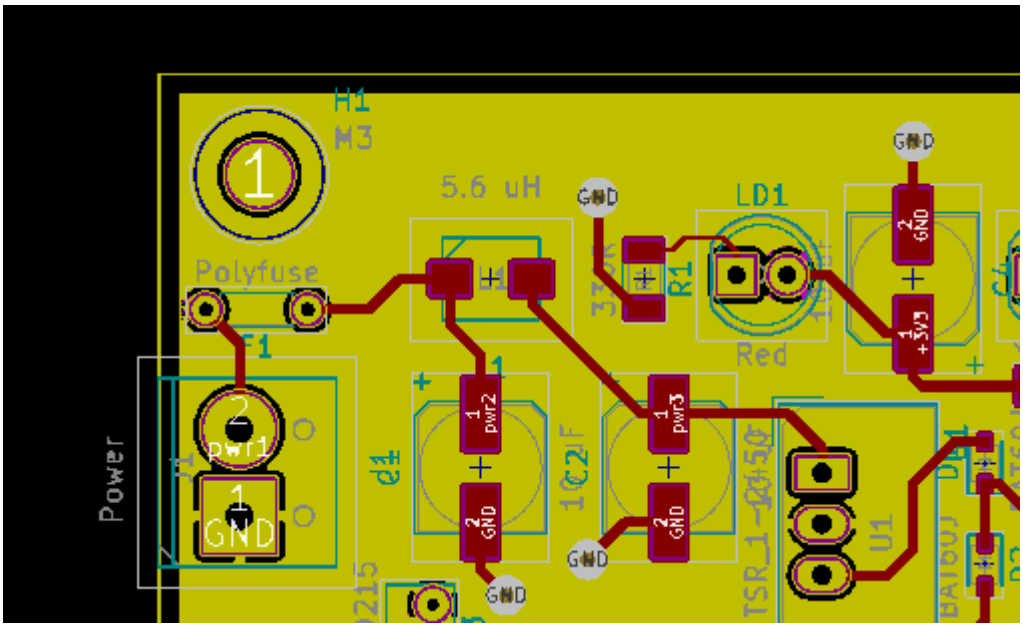
Set ***GND*** as the Net, use Layer ***In1.Cu*** and set Pad connections to ***Thermal relief***. Remember to use ***Close Zone Outline*** from the right mouse button menu to end the region. You should get a rectangle inside the board limits similar to the one shown in the figure below.



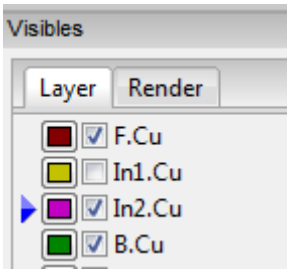
Observe that ***through hole*** Pads are connected to the ***GND*** plane by ***thermal reliefs***. Note also that this is not the case with ***vias*** that feature ***solid*** connections.

The two zone icons on the left side can be used to hide or show the filled zones

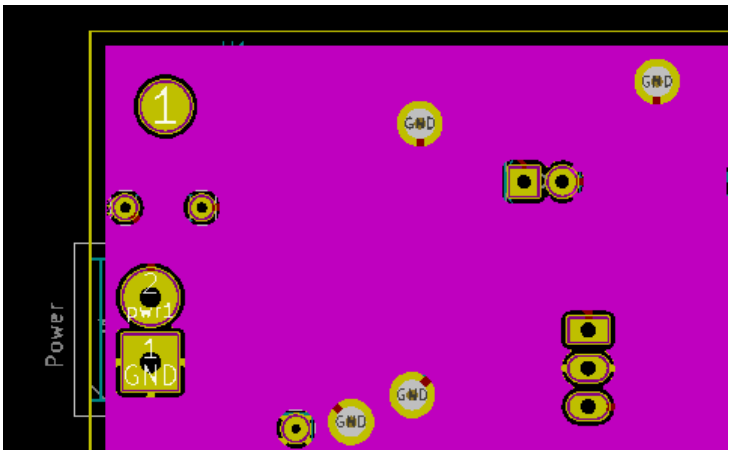
The selected layer is always visible on the top, so, if you select the ***Top*** layer, you will see it over the other layers as in the figure below.



Repeat the same with the ***Vdd*** power plane in layer ***In2.Cu*** associated to the ***+3V3*** net. When the two power planes are shown at the same time, it is difficult to see the details. You can use the ***Layers Management Toolbar*** on the right side of the window to unselect the ***Inner 1*** layer so you can better see the ***Inner 2*** layer details.

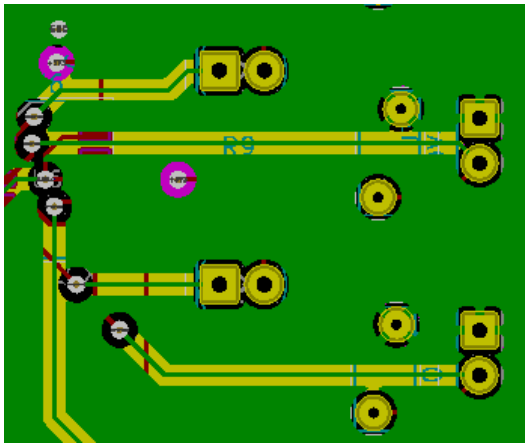


In this case you will see something like:

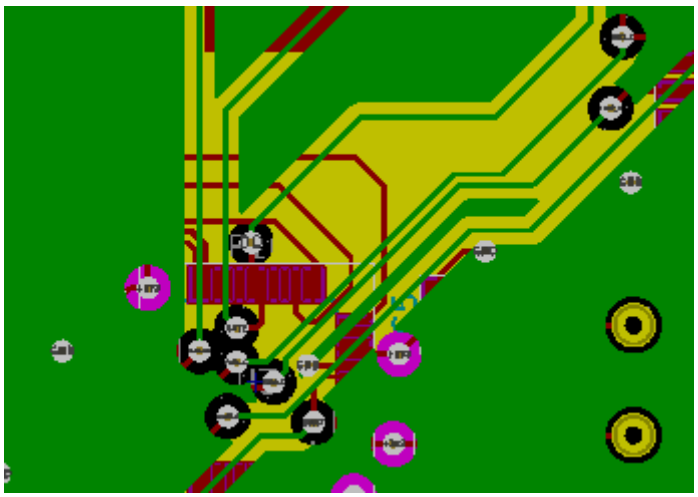


Note the solid ***vias*** that connect with the ***Vdd*** plane.
Depending on how you have made the routing it is possible that the ***Bottom*** layer has a low number of tracks. In this case you can add an additional ground plane on the bottom layer. That will convert this plane in a mixed routing/ground plane. This will decrease GND impedance and probably improve the EMI properties of the board.

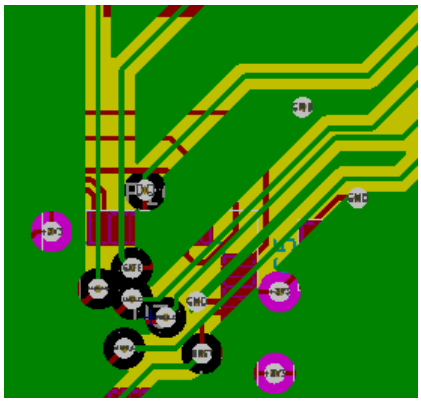
To add a plane on the ***Bottom*** layer, just add a GND plane on this layer like you did before with the ***Inner 1*** layer. The copper, as shown in the figure below, will be removed next to the signals tracks and though hole pads and vias that do not belong to the GND network.



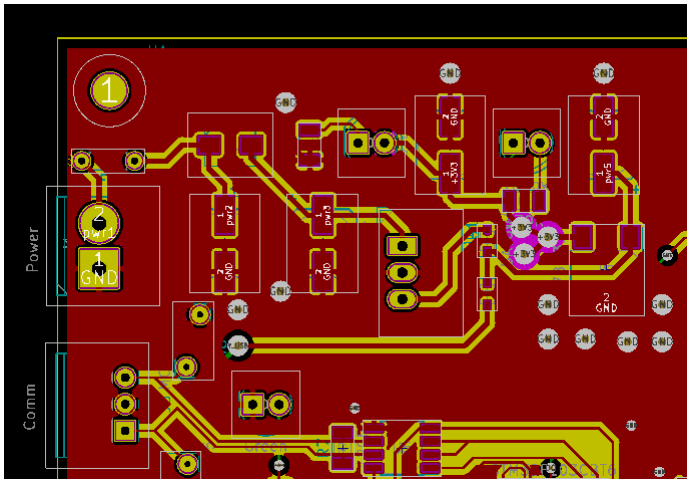
The KiCad program should take care of not generating isolated islands if you set a net for each region, but it is a good idea to check.
In the figure below, for instance, there is a region not covered by the green bottom layer because it would generate an isolated island.



If you place a via connected to the **GND** node inside this region, the region will be filled. Although you are not requested to do that.

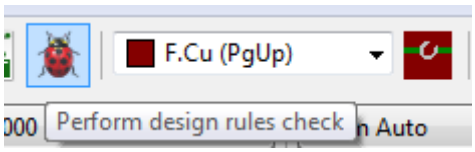


You could do the same with the ***Top*** layer. As this layer has more connections, it is easy to have less plane coverage. You should also check that there are no isolated islands although KiCad probably won't generate them. The figure below shows the ***Top*** layer for the supply regions when a ***GND*** plane is added.

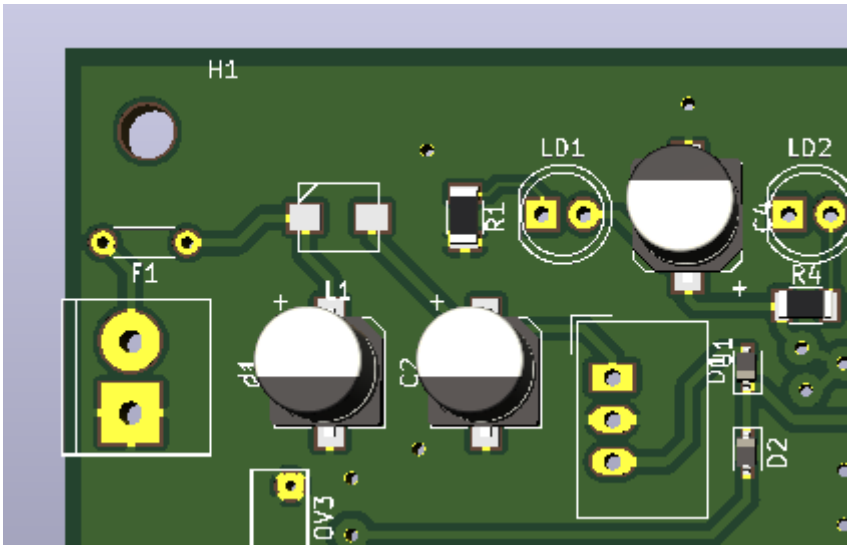


If you have correctly skipped the layer window when drawing the regions, it will show as in the figure below after filling all copper planes.

As drawing the power planes ends the routing stage, you should not see any remaining ***ratsnet*** line in on PCB. You can perform a DRC on the board to check all connections.



Check all errors to determine if they are important or not. Then, when no important errors remain, delete all markers and close the window.
You can see a 3D view of the board if you please.

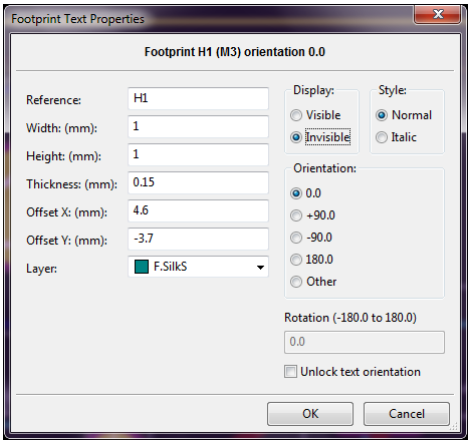


5.15 Cleaning up

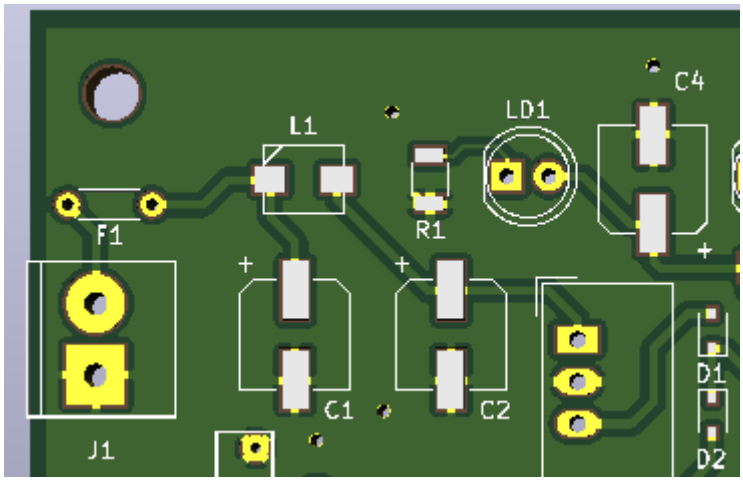
The PCB should be valid like it is, but you need to clean the text elements. When you generate the final board files to be sent to fabrication, you can transfer the component references to the ***Silkscreen*** layer so they are plotted on the PCB. You can also transfer the component values to the ***Fab*** layer.

In our board we will only use a ***Silkscreen*** layer on the ***top*** side of the board. So you need to make sure that all reference texts, like ***U1***, ***U2***, ***R1***,... are inside the board and do no collide with Pads or other markings.

In some cases, like the mounting holes or fiducials, you don't want the ***reference*** to be visible on the board. You can turn off this element visibility by changing its properties:



You can also move and rotate all component references so that they have the same reading direction. This does not improve the electrical operation of the board in any way but gives a more professional look as can be seen in the 3D detail view on the figure below:

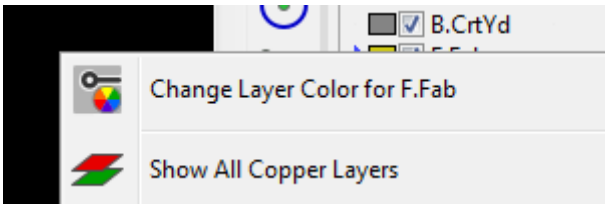


Be sure that the ***Silkscreen*** references are drawn outside of the components and their pads. If not, the text will be hidden after the assembly.
The component ***references*** that can be transferred to the ***Silkscreen*** layer are drawn, while editing, on the ***Render*** pseudo layer ***Text Front*** in blue color if ***References*** item is active.

In a similar way, clean up the component ***values*** that can be transferred to the ***Fab*** values. They are also drawn, while editing, on the ***Text Front*** layer but on grey color if the ***Values*** item is active. Make also invisible the values of all components that are not resistors, capacitors or LEDs. We don't really need them on the ***Fab*** documentation for this board.

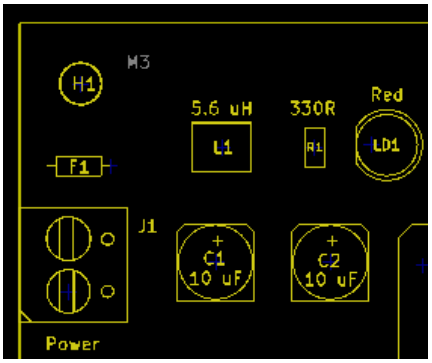
You can superpose the ***Values*** over the blue silkscreen ***References*** as they belong to different documents.

You can change a layer color if you need by using the right button contextual menu when you are over the colored rectangle of a layer at the right side of the layout window.



Finally, clean-up the text information on the ***Fab*** layer. In particular, rotate and move the references in this layer so that they read well. It is a good idea to put all references in the ***Fab*** layer inside the component bodies. Do not make invisible the mounting holes and fiducials **values** that will go to the ***Fab*** layer. Remember that this layer is used for information purposes. It is not printed on the PCB.

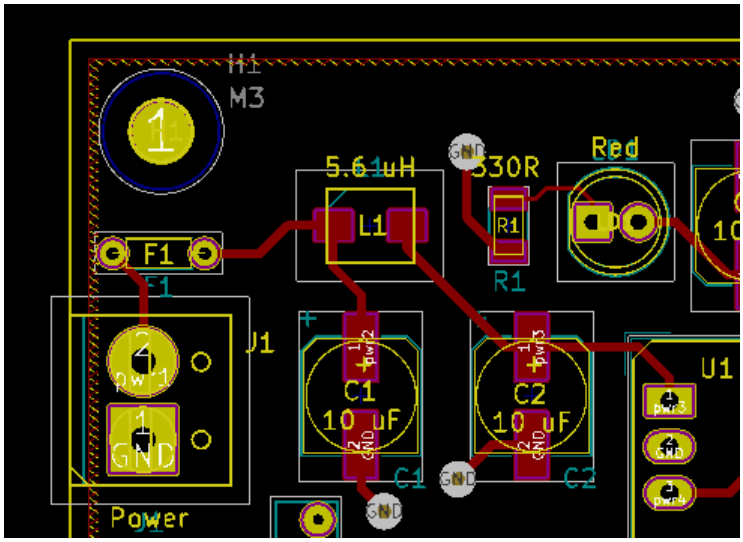
The following figure shows a figure of the ***Fab*** layer. Note how the component references in the layer have been moved over the components themselves. We have also changed the value of polyfuse to "F1" because that footprint didn't include a reference text on the ***Fab*** layer and we need to have all component references on this layer. Finally, note that we have added a circle on the ***Fab*** layer over the **H1** hole location so that it is visible on this layer.



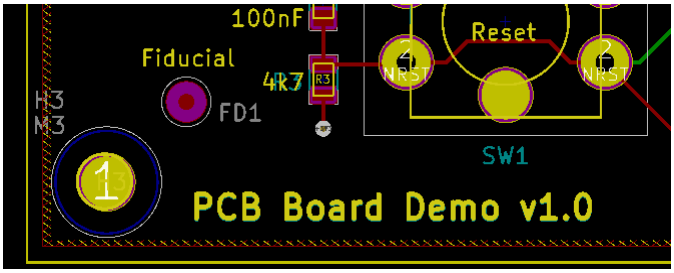
Like on the above example, if a component, like the ***polyfuse*** or the ***movs***, don't has a ***REF**** field on the ***Fab*** layer, modify its value to match the reference and make it visible on the ***Fab*** layer. It is very important that we can identify all component references on the ***Fab*** layer. Obviously you cannot make the value not visible in this case.

If any component has missing drawings on the ***Fab*** layer in its footprint, add a global drawing on this layer so that it can be identified. In the end, if you only show the ***Fab*** layer, you shall be able to locate all components in the board. This is only a quick and dirty solution. The best solution would be to modify the footprint so that it has both a drawing and a reference field on the ***Fab*** layer.

The following figure shows a region of the board cleaned up. As we have explained, there is no problem overlapping the ***Silkscreen*** and ***Fab*** layers as they are not printed at the same time.



At the end of this stage we will add some text on the ***Silk Screen*** and ***Fab*** layers to identify the board, like on the figure below:



You can also add your names if you please.

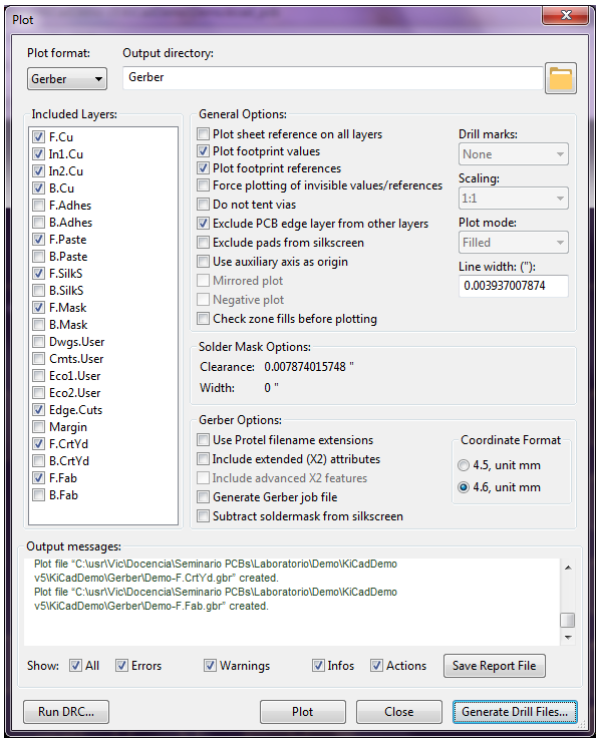
Draw and fill the power planes.

Check also that there are no important errors.
Clean-up the text elements.
Add also the board identification text

6. Post processing

DESIGN PHASE 5

The last stage on the PCB design is to generate the files needed for the board to be fabricated. Save the design and open the ***Plot*** command in the ***File*** menu. Set the options like in the figure below. Not that we will generate files on the four *copper* layers, the ***Edge.Cuts*** and the front ***Solderpaste***, ***Silkscreen***, ***Soldermask***, ***Courtyard*** and ***Fab*** layers.

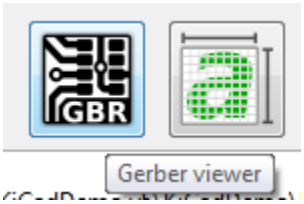


Then click on ***Plot*** and then on ***Generate Drill File*** to generate the files that contain the board hole locations. In the window that will open after clicking on ***Generate Drill Files***, leave all as is and click ***Generate Drill File***. After that, you can close both windows. You should have now a ***Gerber*** folder in your project folder with the following file contents:

Demo-B.Cu.gbr
Demo-Edge.Cuts.gbr
Demo-F.CrtYd.gbr
Demo-F.Cu.gbr
Demo-F.Fab.gbr
Demo-F.Mask.gbr
Demo-F.Paste.gbr
Demo-F.SilkS.gbr
Demo-In1.Cu.gbr
Demo-In2.Cu.gbr
Demo-NPTH.drl
Demo-PTH.drl

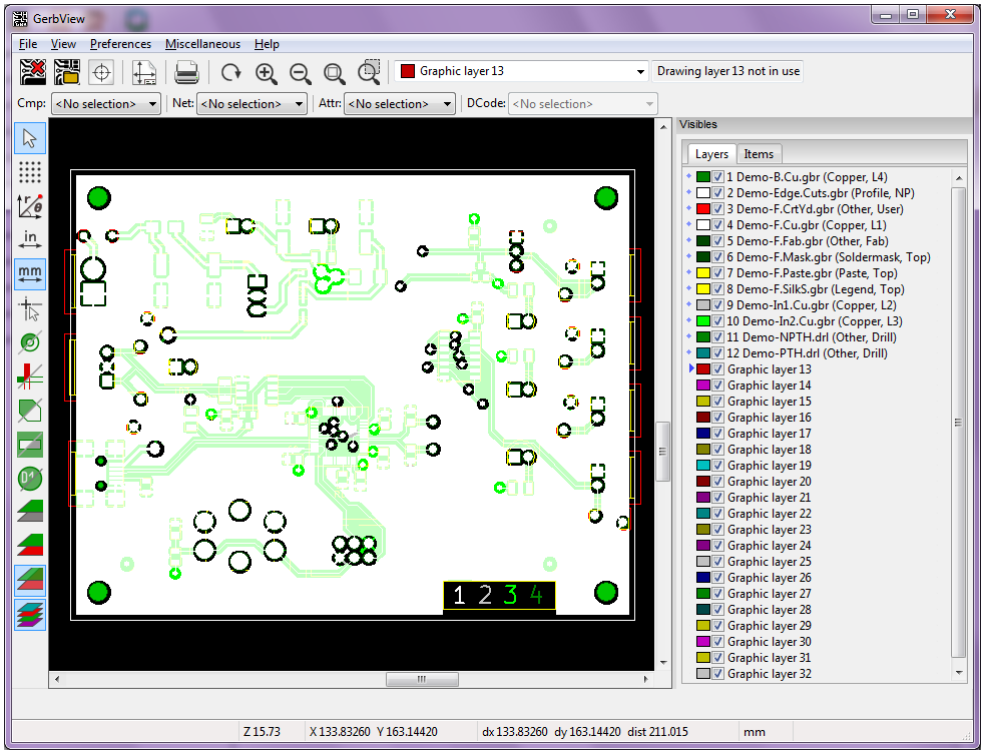
The ***gerber*** files contain most of the information a facility needs to fabricate your board. There is some information missing like the board stack-up details, soldermask colors or copper finishing options, for instance. This information can be included in the ***Cmts.User*** layer but it really depends on the manufacturer requirements.

It is always a good idea to examine the ***gerber*** files to test that they are ok. Also, you can examine the ***gerber*** files to see the design if you lack any other file from your project. KiCad provides a ***Gerber viewer*** application right from the main KiCad window.



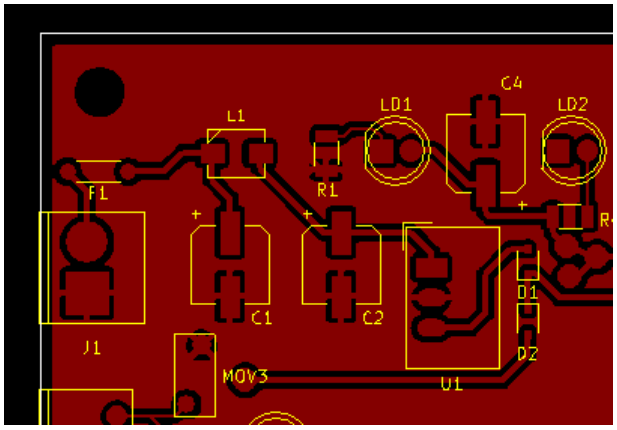
Close the PCB editor and open the Gerber viewer and go to the ***Load Gerber File*** on the ***File*** menu. Navigate to the Gerber folder inside your project folder and select, at the same time, all ***Gerber files***. Then open them.
The select ***Load EXCELLON Drill File*** from the file menu and open the two drill files on your project (one for plated and another for non plated holes). Open them.

All your board design should be visible like on the figure below. Note that colors will be different than in the ***PCBnew*** application.



You can use the right panel to hide some layers so that you can examine the other ones. For instance, the following figure shows the power supply region of the PCB with the only some of

the layers visible. Observe how al component references are shown on the ***Silkscreen*** layer. You can change the colors associated to each layer if you want or change the layers that are visible.



Check #6: Gerber files

Put all files in the ***Gerber*** folder inside a ***Gerber.zip*** file.
If you use the ***hand solder*** option call this file ***Gerber handsolder.zip***
Upload it to the proper ***Atenea*** seminar task.

That ends the base of the lab work for the PCBD seminar.
Next section adds some optional additional work that can be done if you have enough time.

7. Optional work

This section adds additional work that can be done from the base design. Do it if you have enough time after you have completed all previous tasks.
As some of the work proposals imply a strong modification of the design, it is recommended to copy the ***KiCadDemo*** folder to a new folder with another name.
Optional works don't need to be performed in order. You can start with 7.2 if you please, for instance.

7.1 Two layer conversion

We don't want to modify the original design. Close KiCad and copy the ***KiCadDemo*** folder to a new ***KiCadDemo (2 layer)*** folder. Then open the project inside this folder.

Our base design features a four layer stack-up. It is possible to convert it to a two layer design. As the board track density is low, we can fill all unused copper area to create a plane. The layer usage will then be:

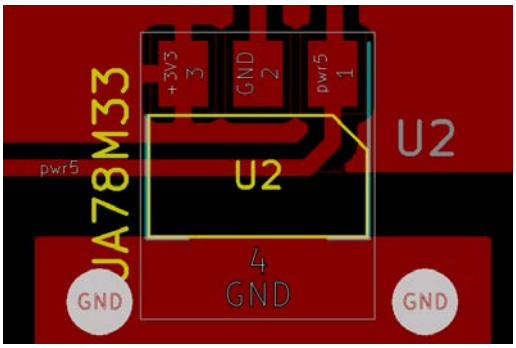
Copper Layer	**Usage**
Top	Main local routing / Horizontal long tracks 3V3 Vdd Plane
Bottom	Auxiliary local routing / Vertical long tracks GND Plane

Eliminate the copper regions in the ***Inner 1*** and ***Inner 2*** layers. Select ***Layers Setup*** in the ***Design Rules*** menu and set the board to have only 2 copper layers. That will eliminate the ***Inner 1*** and ***Inner 2*** layers.

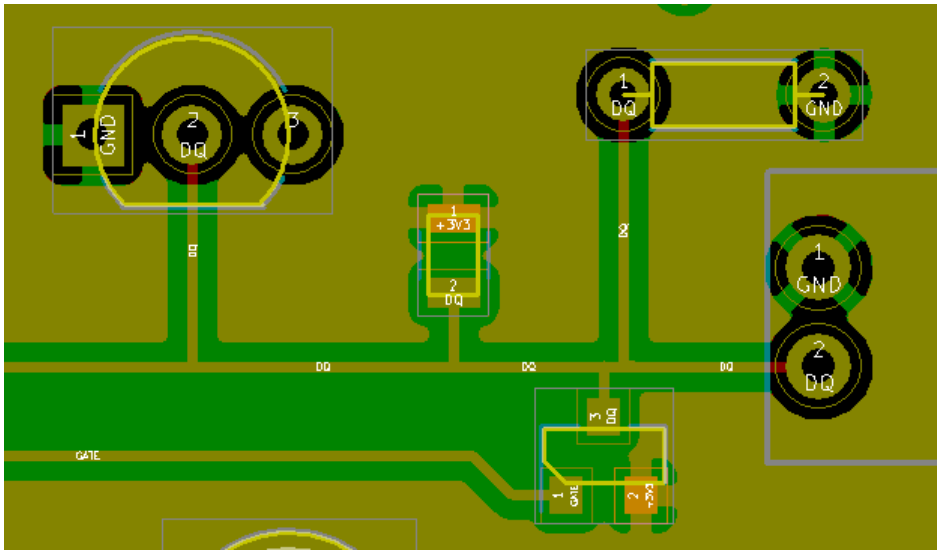
Now, modify the design so that it is fully functional.

- All unused space on ***Top*** shall define a ***3V3*** copper area.
- All unused space on ***Bottom*** shall define a ***GND*** copper area.
- There shall be a ***GND*** copper area on ***Top***, with vias, so that we draw away the power generated by U2 like in the original design. Note that there cannot be two overlapped copper areas in the same layer.
- The two differential pairs (USB and RS-485) shall be laid out over a continuous region of the ***GND*** plane. That means that those pairs shall not cross any track drawn on ***Bottom***.
- Try to lay also the RX and TX signals on a continuous region over the GND plane.
- All vias used to fanout the SMD terminals to Vdd shall be removed as a direct connection to the power plane can be implemented on ***Top***.
- Check that no ***ratsnet*** indicates a non connected Vdd or GND Pad.

The linear regulator ***U2*** was special because it required using solid Pad connections, without thermal reliefs, so that Pad #4 drains its heating power to the copper plane. The +3V3 Pad #3, however, should have a thermal relief as other SMD Pads. To do that you must take into account that the ***Pad connection to zones*** property can be selected both in the Footprint as global or at each Pad on the footprint. In the figure below we see a ***U2*** footprint without thermal relief at footprint level but with thermal relief locally defined on Pad #3.



As a final example the following figure shows a case where the ***Top*** layer doubles as Vdd plane and the ***Bottom*** layer doubles as GND plane. In the figure, the ***green region*** is the Bottom layer whereas the ***green-brown region*** is the superposition of the Top and Bottom layers.



After you end designing the two layer conversion, generate the Gerber and Drill files. They shall be the same as in the base design, but without the ***In1*** and ***In2*** layers.

☑ **Check #Op1: Two layer Fabrication files**

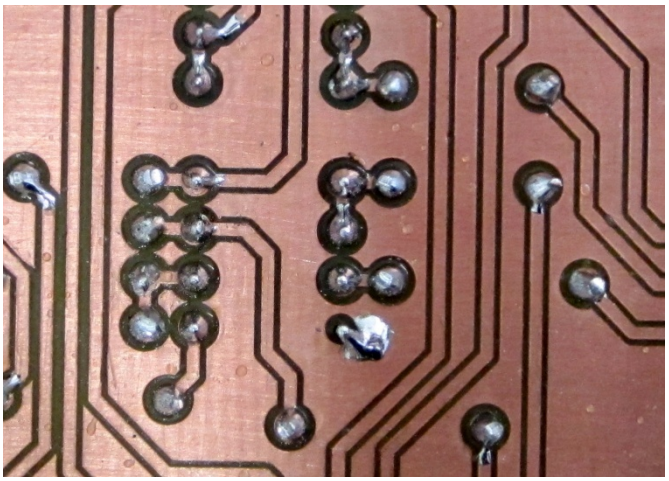
Put the ***Gerber*** and ***Drill*** files inside a ***Gerber 2 layer.zip*** file.
If you use the ***hand solder*** option call this file ***Gerber 1 layer handsolder.zip***
Upload it to the proper ***Atenea*** seminar task.

Impedance in tracks

The two layer conversion, if done properly, guarantees that the differential lines are laid out over a continuous GND plane so the impedance of the lines is kept constant.
However, as we only have two layers, they will be much more separated than the separation between ***Top*** and ***Inner 1*** layers in the four layer stack-up.
The track thickness and separation would need to be recalculated for the new two layer stack-up if you want to keep the same impedances.

7.2 Milled board

There are several ways to fabricate one PCB. In the case of PCB prototypes or one offs, one of them is milling. The milling method takes a single or doubled sided board and uses a mill driven by G-Code to isolate the different tracks. Obtaining a result similar to the one shown in the figure bellow:



Although you can rub out (eliminate) the copper in selected areas of the board, this is not recommended because the impact it has on the life of the milling bits. In the end you should have a board full of copper with thin isolation milled lines. As this technique is used mostly for prototyping, those boards are usually hand soldered.

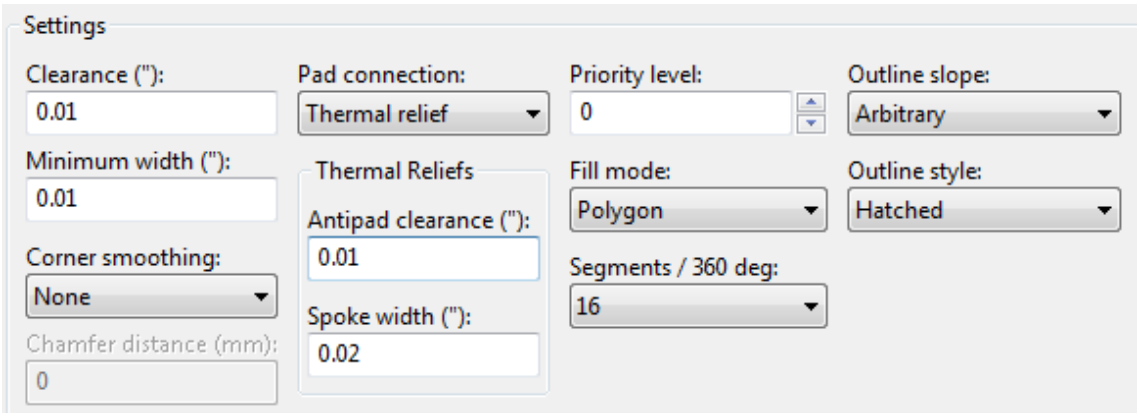
You can do milling in two sided boards if you solve the alignment requirement between both sides. The main problem in this case is that the vias on the board are not metalized, so they don't connect the ***Top*** and ***Bottom*** layers if no explicit soldering is performed on a wire on both sides. There is a rivet solution for vias but it is not popular nor cheap.

One important problem in milled boards is related to the vias problem. In order to join two tracks on ***Top*** and ***Bottom*** you can put a wire trough the via and solder on both sides. But you cannot always do the same with through hole component terminals as most through hole components cannot be soldered on the component side. Moreover, hand soldered vias cannot be placed inside the component footprints as the create bumps below the components that affect the assembly of them.

Another important problem is that milled boards easily creates isolated copper islands that can couple signals trough them. For low speed designs this is not usually a problem but can be important when fast signals are present.

Finally, the last problem is that KiCad uses solid via connections to the copper planes. This is Ok for normal PCB fabrication methods, but for milling, vias usually are hand soldered. In the case of vias connected to a power plane, not having a thermal relief can be a problem.

In KiCad we will use copper areas on ***Top*** and ***Bottom*** to simulate the copper between tracks that results from the milling process. We will suppose that the milling bit has a 10 mil thickness, so, in order to simulate it, we need to select a 10 mill clearance for the ***Top*** and ***Bottom*** copper areas like in the figure below.

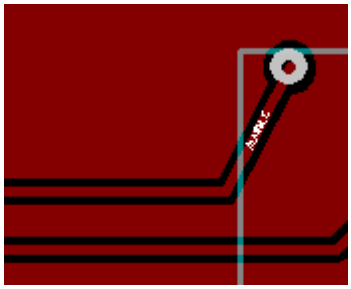


The layer usage will be the same as in the previous optional proposal:

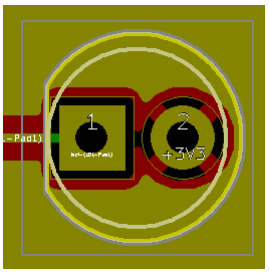
Copper Layer	**Usage**
Top	Main local routing / Horizontal long tracks 3V3 Vdd Plane
Bottom	Auxiliary local routing / Vertical long tracks GND Plane

When you fill the Top and Bottom areas you can find empty regions like the one in the figure below:

The empty region between the tracks is generated because KiCad would create an isolated island in this case. In a standard milling process this will create an isolated island. You cannot always prevent those islands but try to minimize them. In this particular case, just moving up the via and separating the tracks a little, guarantee that the ***Top*** Vdd region flows in the space between the tracks.

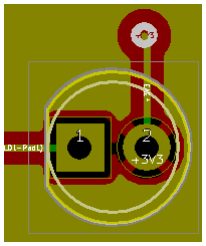


In order to ease the hand soldering of the vias, we would like to have thermal reliefs in vias connected to the copper planes. Consider the following LED that is connected, in the cathode, to a ***Bottom*** track and, in the anode, to the Vdd +3V3 ***Top*** plane.



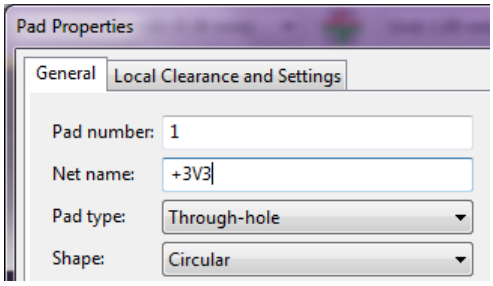
Note that the Green-Brown color is the superposition of the Red Top color and the Green Bottom color.

Remember that all ***through hole*** components, in a milled design, shall be accessed from the ***Bottom*** layer. This design cannot be soldered because we will need to solder the anode on the top layer and the component itself blocks this ***Top*** layer Pad. To solve the problem we can connect the anode to a via connected to the ***Top*** layer using a track on the ***Bottom*** layer.



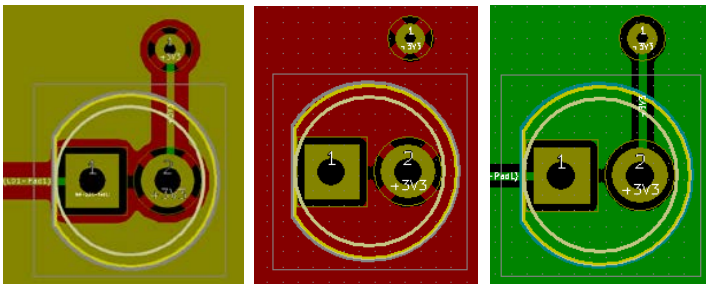
That solves the initial problem but creates a new one. The via on the top layer is a solid one and is difficult to solder because the plane draws away the soldering heat. We would like to have a via with a thermal relief but this is not directly possible in KiCad.

The solution is to define a new component with just a Pad with the same 40 mil diameter and 15 mil hole as a normal via. We will call it the ***Via40*** component. Now we can add this component in the position the via should be. In order to connect this Pad to the plane, we associate it to the ***+3V3*** network by editing its properties.



Note that we are editing the ***Pad*** properties, not the ***footprint*** properties.

Then we can route the new Pad and the anode using a track on the ***Bottom*** side. After filling the planes you should get something like in the figures below.



From left to right: Both planes, Top plane and Bottom plane.

We have now a thermal relief on the via connected to the power Vdd plane on ***Top*** so it will be easier to be soldered.

We don't want to modify the original design. Close KiCad and copy the ***KiCadDemo*** folder to a new ***KiCadDemo (milled)*** folder. Then open the project inside this folder.

Eliminate the copper regions in the ***Inner 1*** and ***Inner 2*** layers. Select ***Layers Setup*** in the ***Design Rules*** menu and set the board to have only 2 copper layers. That will eliminate the ***Inner 1*** and ***Inner 2*** layers.

If you have performed the previous optional two layer design, you can use it as the start design instead of the four layer base solution. In this case you don't need to eliminate the inner layers. If you don't have performed the two layer optional design, read its documentation because several of the comments made on this design also apply for this one.

Create the ***Via40*** footprint on the ***Footprint editor*** and save it on the ***Demo*** library. Create also a ***Via80*** footprint with 80 mil diameter and 30 mil hole for high current vias on the supply region. Remember that every time you add a ***Via40*** or ***Via80*** component you need to associate its ***Pad*** to one of the ***nets*** so that the DRC will not complain and you can perform the connection.
Remember also to make **invisible** the ***REF**** text of any ***Via40*** or ***Via80*** component you add to the design as you don't want this text to show up on the ***silkscreen*** layer.

Don't use thermal reliefs on the vias that connect the U2 linear regulator copper zone to the bottom ground plane. It would work against its functionality.

Now, modify the design so that it is fully functional. Some of the conditions are the same as in the two layer conversions and some, in blue color, are new to the milled design.

- All unused space on ***Top*** shall define a ***3V3*** copper area if possible.
- All unused space on ***Bottom*** shall define a ***GND*** copper area if possible.
- There shall be a ***GND*** copper area on ***Top***, with vias, so that we draw away the power generated by U2 like in the original design. Note that there cannot be two overlapped copper areas in the same layer.
- Minimize all non filled areas on the two routing layers.
- All trough hole component Pads shall be accessed from the ***Bottom*** layer and cannot be used to change to the ***Top*** layer.

- You can only use vias to change from the ***Top*** to the ***Bottom*** layer.
- Use special thermal relief vias based on the Via40 and Via80 components for vias that connect to the power planes.
- Don't place any via inside a component footprint bounding box.
- The two differential pairs (USB and RS-485) shall be laid out over a continuous region of the ***GND*** plane copper area on ***Bottom***. That means that those pairs shall not cross any track drawn on ***Bottom***.
- Try to lay also the RX and TX signals on a continuous region over the GND plane.
- All vias used to fanout the SMD terminals to Vdd shall be removed as a direct connection to the power plane can be implemented on ***Top***.
- Check that no ***ratsnet*** indicates a non connected Vdd or GND Pad.

After you end designing the two sided milling conversion, generate the Gerber and Drill files. They shall be the same as in the base design, but without the ***In1*** and ***In2*** layers.

☑ **Check #Op2: Milled version Fabrication files**

Put the ***Gerber*** and ***Drill*** files inside a ***Gerber milled.zip*** file.
If you use the ***hand solder*** option call this file ***Gerber milled handsolder.zip***
Upload it to the proper ***Atenea*** seminar task.

ⓘ **Copper fills on a milled board**

When you generate the ***Gerber*** files for a milled board in theory you don't need to do the copper fills on the Top and Bottom layers as the milling bit will create them for you. However, the thermal reliefs on Pads connected to the planes will only be generated when you perform the copper fills. So, don't neglect the copper fills if you depend on the thermal reliefs.
In the files you send, in any case, provide the copper fills to check more easily the lack of isolated islands in the design.

ⓘ **Milling resolution**

This is a design that uses fine pitches in some of the components.
You cannot be sure that it can be properly fabricated in one particular milling machine.
Always check the milling machine fabrication limits before start designing the board.

7.3 Design for hand soldering

This optional design adds some steps to the other designs. So, it is not really a new design but an option that can be added to the previous designs. After you end the base design you can start over to carry out this optional modification on the design. But, as explained at the start of this document, if you really want to try this option, it is better to just apply this option from the start of the lab work so you will end up with the base design with this option added.

As you should know, the footprints on a PCB depend, not only on the component packages but also on the soldering method used during the board assembly. There are four main methods for soldering:

- Automated methods for high volume
    - Wave soldering
    - Oven reflow
- Manual methods for low volume or rework
    - Hand soldering using soldering iron
    - Hot air reflow

Currently oven reflow is the main assembly method for SMD components and wave soldering is the main assembly method for through hole (TH) components.

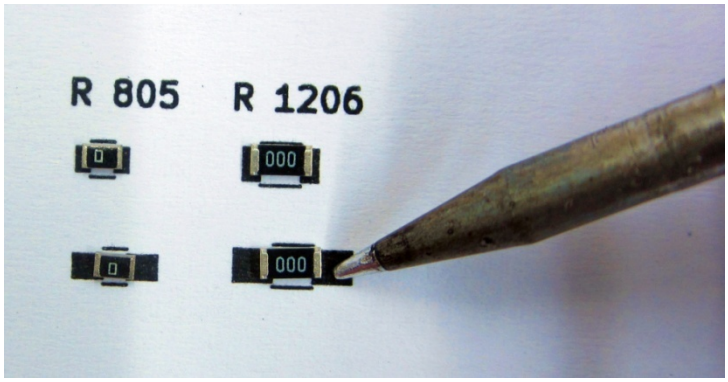
Fortunately, trough hole (TH) components don't usually require footprints tailored to the assembly method so they can be considered quite universal. That is not the case with SMD components. Each method requires a different kind of footprints so you will end up with three different footprint options for each component package:

- Reflow footprints for oven and hot air
- Wave soldering footprints
- Manual soldering footprints

As oven reflow is the main assembly method for SMD components, this is the default footprint you will get on all CAD tools. You can hand solder a board designed for reflow assembly, but it will be more difficult than performing the same work on a board specifically designed with hand solder footprints.

The main difference between SMD normal reflow footprints and SMD hand soldering footprints is due to the how you provide heat. Reflow heats all the board at once whereas hand soldering uses the tip of a soldering iron to provide the heat. Hand soldering, in general, requires longer terminal landing Pads because you need to have space to put the soldering tip and transfer heat.

The following figure shows the footprints for standard ***0805*** and ***1206*** resistor packages together with the real components and a fine soldering tip. The upper row shows normal KiCad reflow footprints whereas the lower row shows KiCad hand soldering footprints. Note that, although you could hand solder on the reflow optimized footprints, it will be much easier on the hand solder optimized ones.



In the end, you can hand solder virtually any package with exposed terminals not blocked by the package itself, but having hand solder optimized footprints eases the work. You can argue, however, about how much longer the landing pads need to be. Some say that KiCad hand solder footprints have too long pads.

If you check the KiCad footprints you should note that the pad spacing is the same in the two kinds of footprints but the pad length is about 40 mil (1 mm) wider on the hand soldering footprints.

Adding 40 mil to the pad length is perhaps too much. Adding 20 mil (0.5 mm) is probably enough. Our design uses several SMD footprints as shown on the following table.

References	Package	Footprint
J7		USB_MINI-B
C1,C2,C3,C4		CP_Elec_6.3x5.8
U2	SOT-223	SOT-223
Other caps	0805	C_0805-HandSoldering
R2,R3,R5,R11	0805	R_0805-HandSoldering
Other resistors	1206	R_1206-HandSoldering
M1, M2	SOT-23	SOT-23_Handsoldering
D1, D2	SOD-323	D_SOD-323_HandSoldering
U3	LQFP48	New, extend pads 20 mil
U5	SO-8	New, extend pads 20 mil
U6	SO-14	New, extend pads 20 mil
L1	Custom	Pads extended 20 mil

The proposal in this optional work affects in a different ways to the SMD footprints in the table:

- Components in **green** on the table will use the same KiCad provided footprints as in the base design. That includes J7, U2 and electrolytic caps C1 to C4.
- Components in **black** on the table won't use the base design footprints but other special hand solder footprints included in KiCad. That includes all resistors, diodes, MOS transistors and non polarized caps.
- Components in **red** on the table will use custom footprints instead of the KiCad provided ones. This is the main work associated to this optional work as you will need to create three new footprints on the ***Demo*** pretty library. Those footprints will be modified versions of the ones used on the base design. Basically we will add 20 mil to all pad lengths. You can use whatever footprint name you please for the new footprints but don't repeat the standard KiCad footprint names.
- The L1 component, in **blue** on the table, already required a custom footprint in the base design. In the hand solder option you just need to use 20 mil longer pads.

If you have a lot of spare time you can also design custom versions of the five footprints for components in ***black*** on the table. Design the new footprints from the original base footprints but add 20 mil to the pads instead of the 40 mil added in the KiCad hand solder footprints. In this case, provide those five new footprints also on the ***handsolder.zip*** file described below.

After you end a design with the special hand soldering footprints, create a ***handsolder.zip*** file with the three (or eight) new footprint ***kicad_mod*** files.
If you use the hand solder option in the base or any of the optional designs, add ***"handsolder"*** to the fabrication zip file name that includes the ***Gerber*** and ***Drill*** files you upload to ***Atenea*** as the final task.
If you follow the long path starting over using the hand solder option after you have ended and uploaded to ***Atenea*** the normal base design, add the ***Gerber*** and ***Drill*** files to the ***handsolder*** zip file.

☑ **Check #Op3: Hand solder zip file**

Upload the ***hansolder.zip*** file to the proper ***Atenea*** seminar task.

8. PCBD Workflow in brief

This section gives a brief description of the workflow we follow in the PCBD seminar. It is only an ordered list that shows you what our work sequence in the PCB design is.

1. Initial Design Review

- What does the circuit, how it is organized. Signal Path. Critical elements.
- Gather Schematic info for the circuit.
- Obtain information about all components
- Get mechanical data for the PCB including
  - PCB dimensions
  - Fixed position components
  - Obstacles and height limitations
- Get assembly information, PCB Stack-up and layers to use
- Check PCB fabrication capabilities

2. Schematic capture

- Detect and create symbols missing in the application libraries.
- Introduce metadata on all components including name, manufacturer and footprint
- Place components and connect them with nets
- ERC Check

3. Pre-Layout Review

- Generate missing footprints
- Schematic update with missing footprints
- Netlist generation
- Boom Generation
- Define PCB requirements
  - PCB Class
  - Component clearances
- List and definition of elements not in schematic:
  - Isolated mounting holes
  - Fiducials

4. PCB Layout

- Configure the tool
- Load the netlist
- Set board limits
- Place fixed components
- Place the rest of components
- Define a routing strategy
- Route the critical nets
- Route the rest of nets
- Draw the power planes
- Clean-up the reference text elements

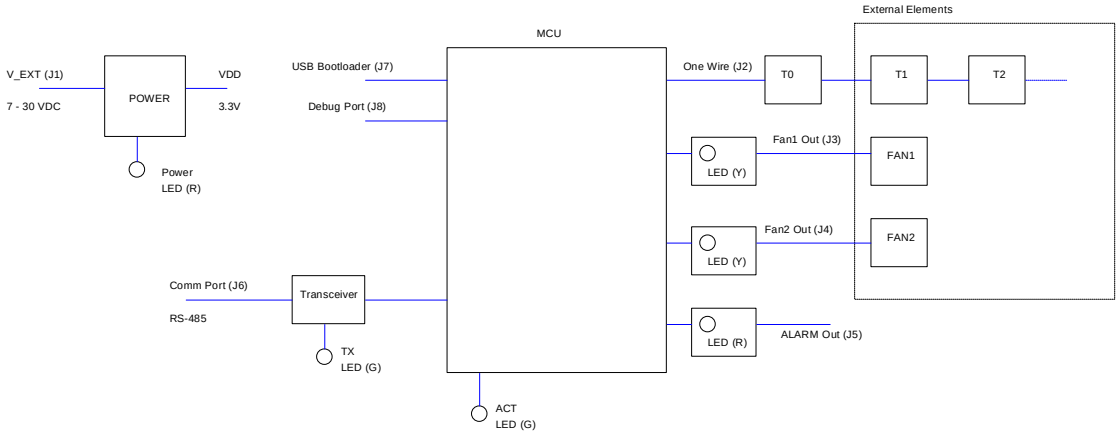
5. Post processing

Generate the ***Gerber*** and ***Drill*** files

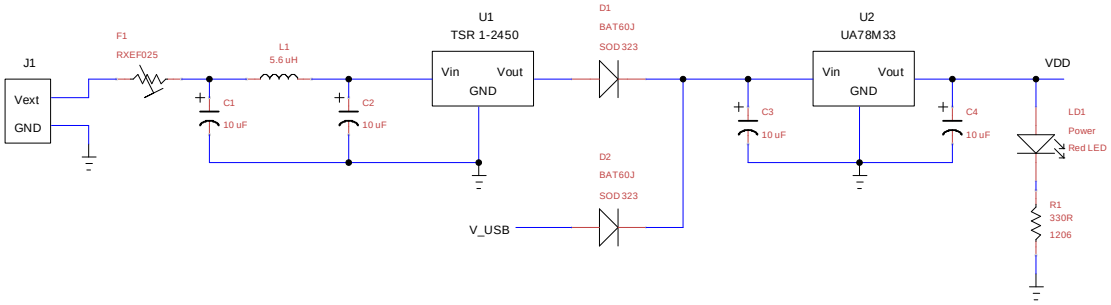
Examine the files to detect errors

Appendix A: System Description

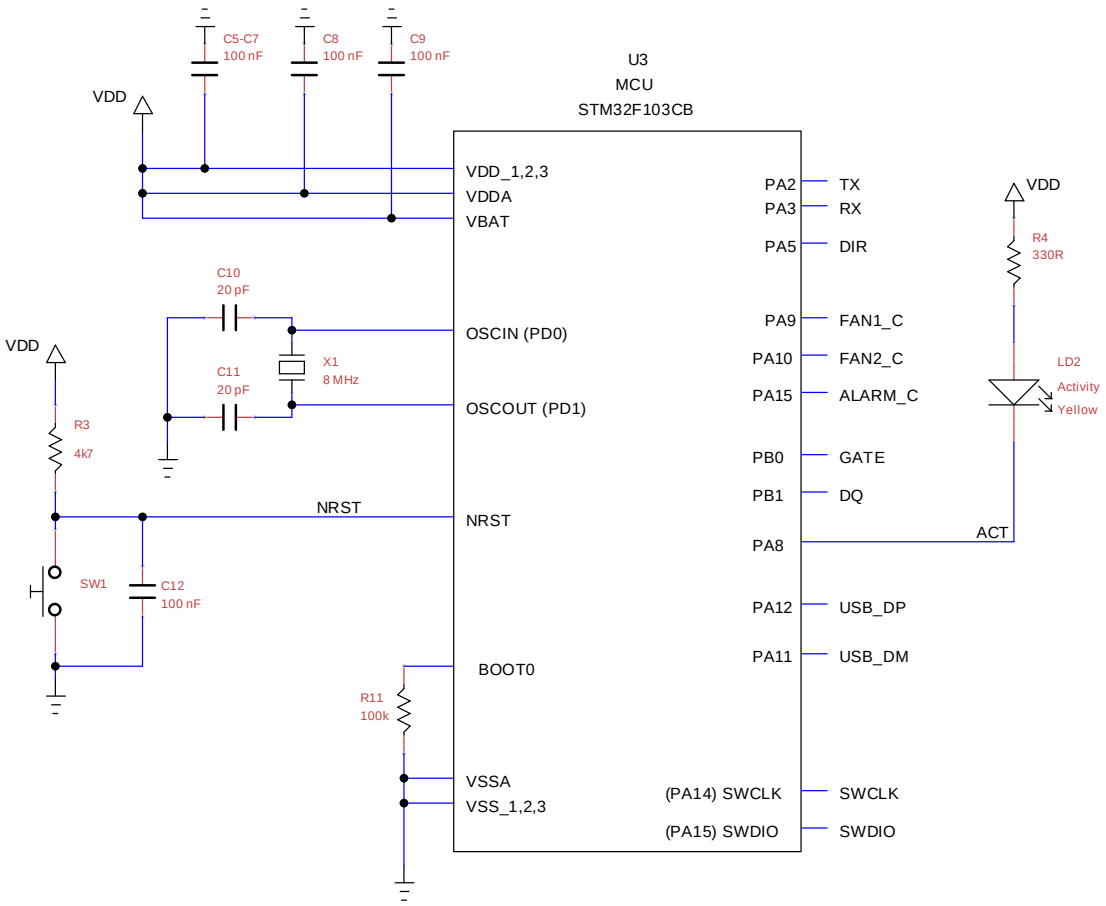
Block Diagram



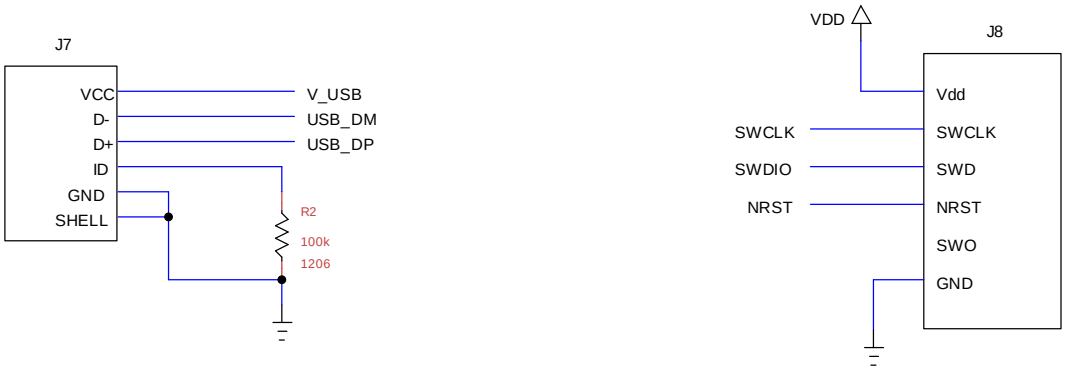
Power supply (1/6)



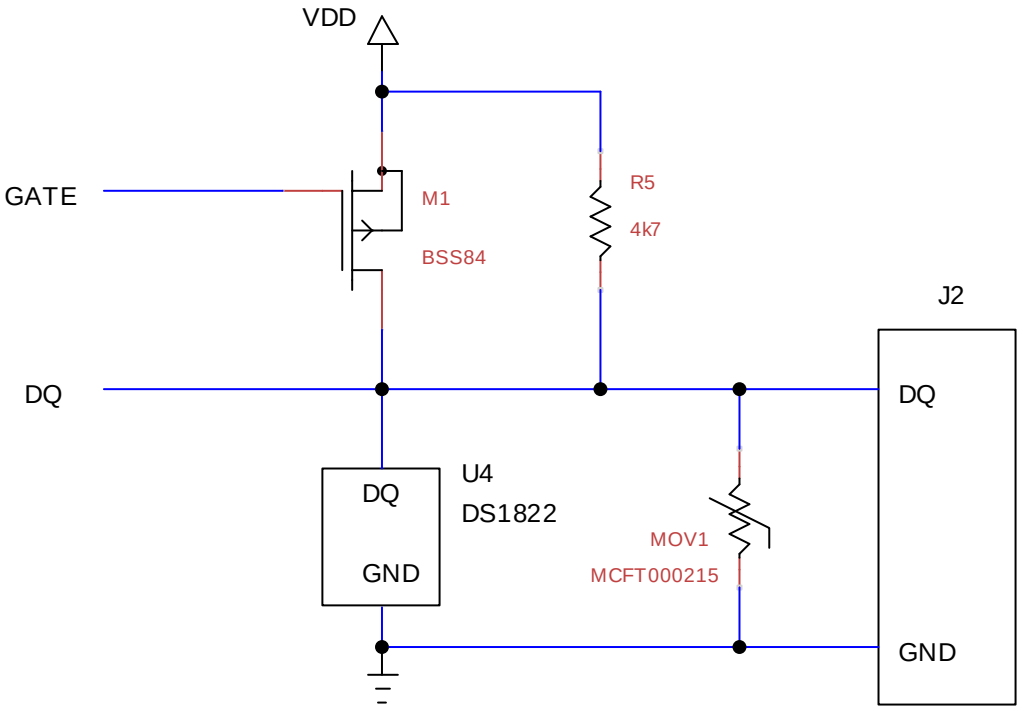
MCU (2/6)



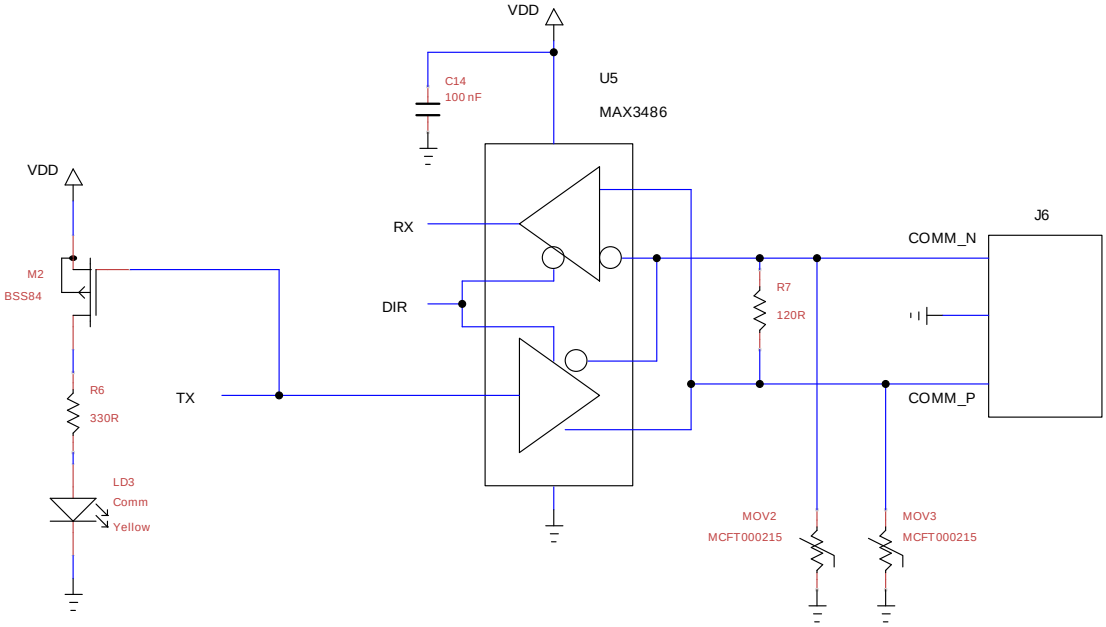
USB and Debug Connectors (3/6)



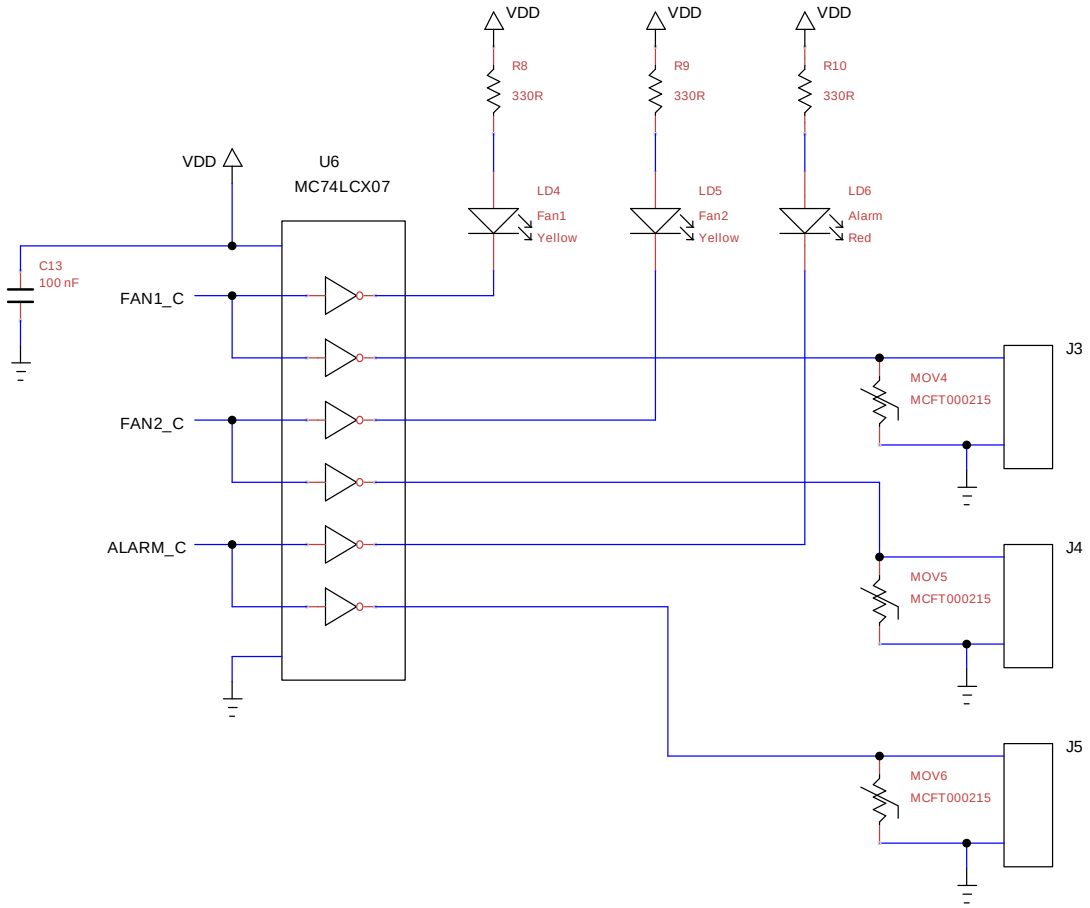
One Wire (4/6)



Communication (5/6)

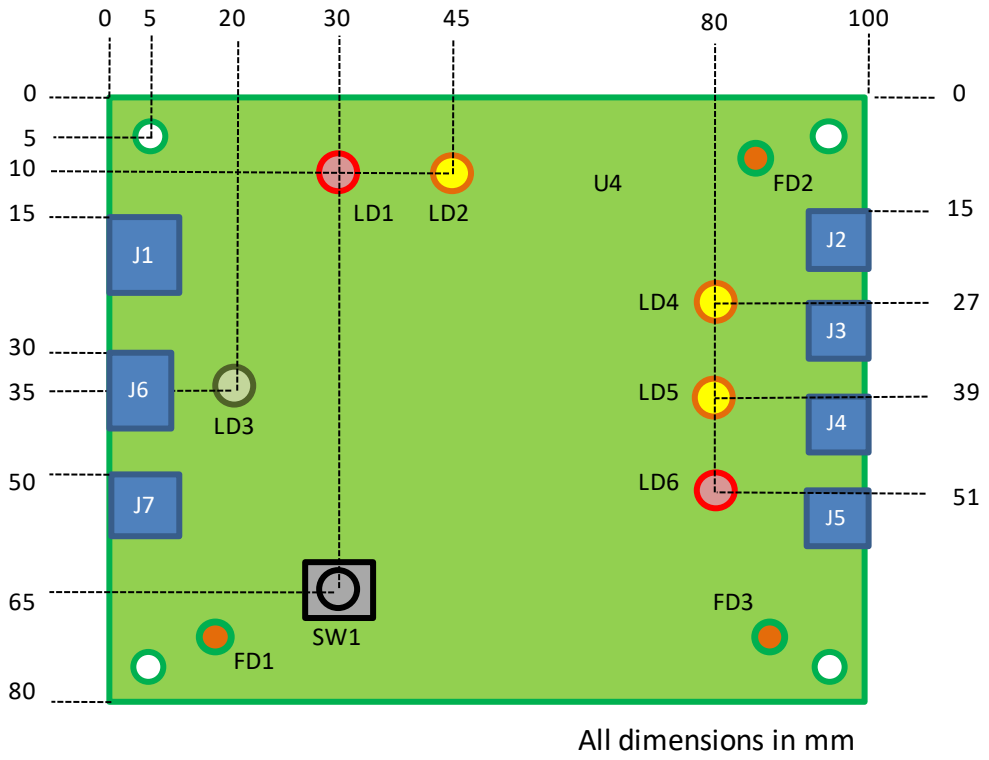


Outputs (6/6)


VDD
VDD
VDD
R8
330R
R9
330R
R10
330R
VDD
U6
MC74LCX07
LD4
Fan1
Yellow
LD5
Fan2
Yellow
LD6
Alarm
Red
C13
100 nF
FAN1_C
FAN2_C
ALARM_C
J3
MOV4
MCFT000215
J4
MOV5
MCFT000215
J5
MOV6
MCFT000215

Appendix B: Mechanical data

The following figure shows the mechanical requirements for the PCB board to be developed.



The total board size is 100 mm x 80 mm.
The board has 14 fixed position components: 7 connectors, 5 LEDs and one push button.
It also has four mounting holes for M3 screws with centers at 5 mm distance from the board edges.

In order to calibrate the cameras of the pick and place machine we need three fiducials. All them are composed of a 1 mm diameter round Pad inside a 2 mm diameter solder mask free area. They shall be at least 5mm from the board edges. Their exact location is not critical but they should be in the three corners indicated on the figure.

The position of the temperature sensor U4 is not critical, but it must be in the upper right region of the board.

Appendix C: Fabrication capabilities

The following table defines the fabrication capabilities we will use for our demo design. Always check the fabrication capabilities of the fab before starting a new design.

Rule	Metric	Imperial
Min. track width	0,2 mm	8 mil
Min. track spacing	0,2 mm	8 mil
Min. ring thickness	0,2 mm	8 mil
Min. hole diameter	0,35 mm	14 mil
Min. soldermask thickness	0,1 mm	4 mil
Min. soldermask distance to pad	0,075 mm	3 mil
Min. silkscreen thickness	0,125 mm	5 mil
Min. distance from copper to board edge	0,2 mm	8 mil

Although we will use the same fabrication capabilities table for all optional designs, beware that usually milling machine fabrication has typically worse capabilities than normal industrial PCB fabrication methods.

References

KiCad Homepage

Homepage of the KiCad Cross Platform and Open Source Electronics Design Automation Suite. You can obtain a copy of the software from this site.
http://kicad-pcb.org/

Optimum Design Handbook

PCB Design Handbook that describes in detail all the phases of a PCB design.
It describes the design workflow carried out at the Optimum Associates Company and is a very good workflow reference.
http://blog.optimumdesign.com/optimum-designer-handbook

IPC Standards Documents

IPC, the Association Connecting Electronics Industries, is a trade association whose aim is to standardize the assembly and production requirements of electronic equipment and assemblies. Some standards related to this document are:

IPC-2221 Generic Standard on Printed Board Design
IPC-7251 Generic Requirements for Through-Hole Design and Land Pattern Standard
IPC-7351 Generic Requirements for Surface Mount Design and Land Pattern Standard
IPC-A-600 Acceptability of Printed Boards

The documents associated to the standards, however, are not free.
http://www.ipc.org