# Raspberry Pi 3x3x3 LED Cube Workshop Demo

March 20, 2015

# Table of Contents

# 1 Introduction

A design for an Raspberry Pi 3x3x3 LED Cube Shield has been implemented in KiCad. The completed board can be found here:

https://github.com/DonovanSquires/KiCadDemo

The project tries to illustrate several of the concepts common in a KiCad project such as:

- Project folder organization – separating library files, documentation, prints, and fabrication outputs in folders along with the project files
- Multi-sheet hierarchical organization – one root sheet with other sheets represented as hierarchical sheet symbols
- Custom schematic symbols organized in library files
- Adding parameters such as Manufacturer and Supplier part numbers to a schematic symbol to aid in bill of materials (BOM) generation
- Wires – connections between symbols that form a net
- Net labels – implied connections between two or more wires to form a net; net labels only make implied connections on a single sheet
- Ports – implied connections between a wire on a sheet and its corresponding sheet symbol
- Associating schematic symbols with PCB footprints
- Custom PCB footprints organized in library files and folders
- Two-layer PCB design
- Fulfilling predetermined mechanical requirements – lining up connectors and mounting holes of the shield board with the Raspberry Pi
- Fabrication files – generated RS274X Gerber and Excellon Drill files

To become familiar with some of the more difficult concepts in KiCad, this exercise was developed in which the four mounting holes are removed from the completed project. In this exercise you will have to draw a schematic symbol and store it in a library, add the symbols to the schematic, draw the footprint of the mounting hole and store it in a library, synchronize the schematic files with the PCB files, place the mounting hole footprints on the board, and generate fabrication files.

This document and the project files for the exercise may be found here:

https://github.com/DonovanSquires/KiCadDemoExercise

This project and exercise was developed on KiCad (2015-01-16 BZR 5376) on the Windows platform.

# 2 Exercise

## 2.1 Download and open project files

### 2.1.1 Purpose

You will use a set of project files where most of the design has been completed except for the steps that you will perform in this exercise. The project files need to be downloaded and opened in KiCad.

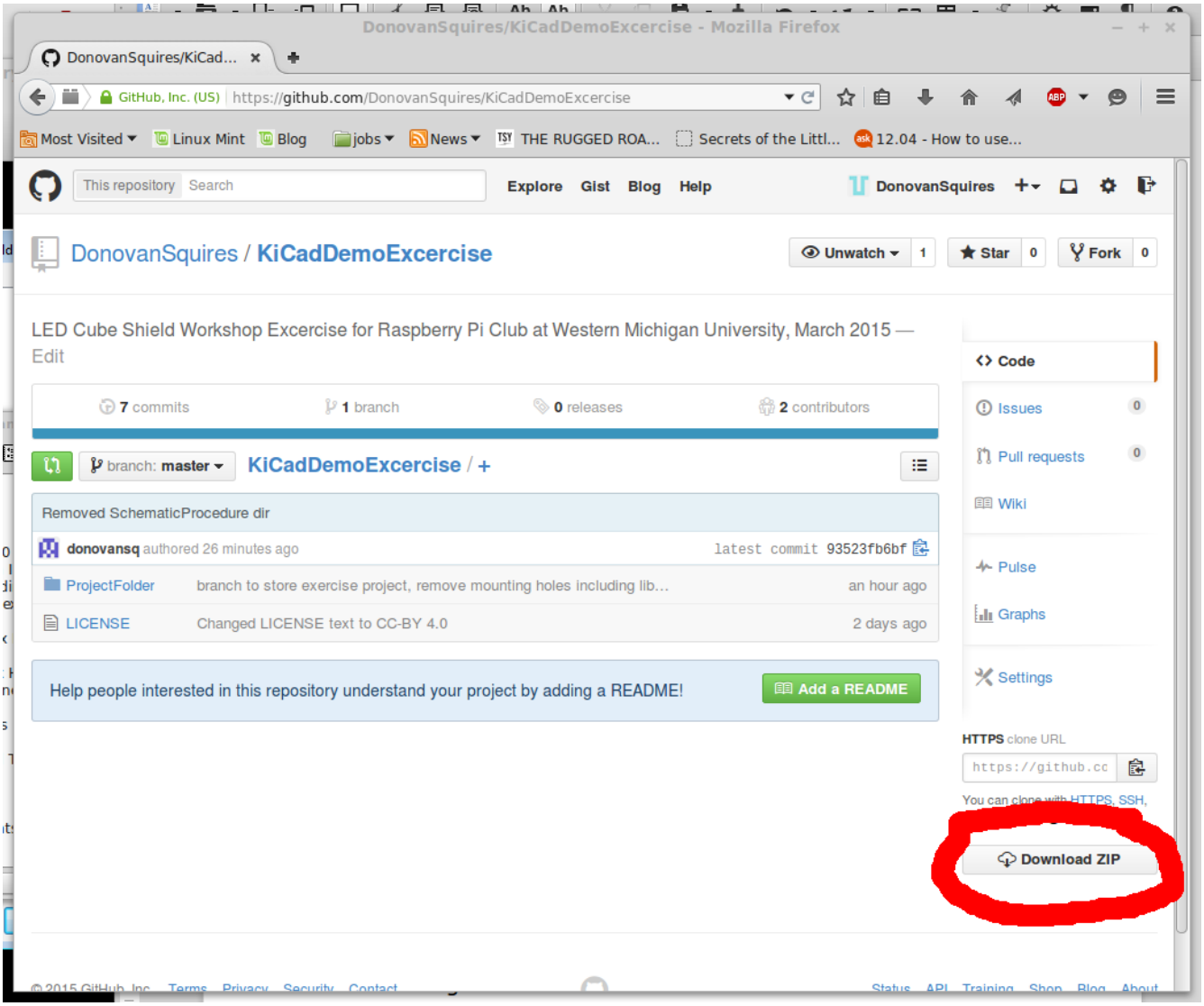
### 2.1.2 Background

The project files are stored in a GitHub repository. The repository may be downloaded as a zip file from the website, so no knowledge of git is necessary.

### 2.1.3 Procedure

#### 2.1.3.1 Navigate in a web browser to https://github.com/DonovanSquires/KiCadDemoExcercise

#### 2.1.3.2 Click on the link to download a zip file of the repository



#### 2.1.3.3 Unzip the downloaded file

#### 2.1.3.4 Run KiCad and open the project file ProjectFolder\KiCadDemo.pro

## 2.2 Create new schematic library

### 2.2.1 Purpose

Create a symbol to represent a mechanical mounting hole. This symbol will be associated with a footprint that will KiCad to automatically add to the PCB.

### 2.2.2 Background

Reusable schematic elements are store in plain text “*.lib” files. Multiple symbols may be stored in a single library file, but I prefer to make a separate library file for each symbol.

Here we will create a symbol representing a single hole that can be placed on the board and is often not electrically connected to the circuit (but it may be in some cases). Placing a symbol on the schematic allows the mounting holes to be tracked.

Pins added to symbols are used to represent an electrical connection. The schematic design rule check (DRC) can enforce rules such as not connecting two outputs to one net. I often set the pins as “Passive”, though, to avoid DRC errors that are not really applicable to the schematic.

### 2.2.3 Procedure

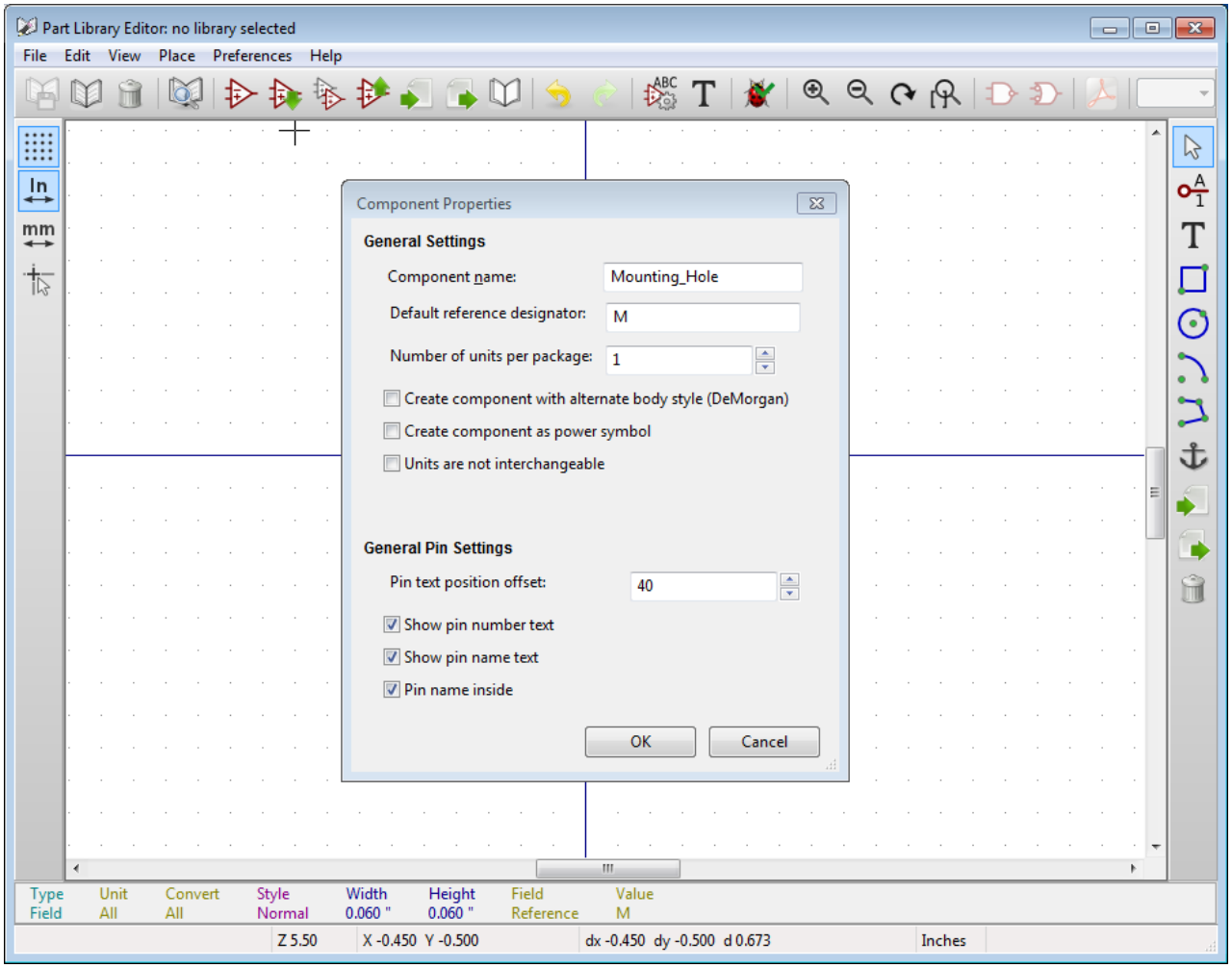
#### 2.2.3.1 Open Schematic Library Editor

#### 2.2.3.2 Click “Create New Component” icon

#### 2.2.3.3 Fill out component information

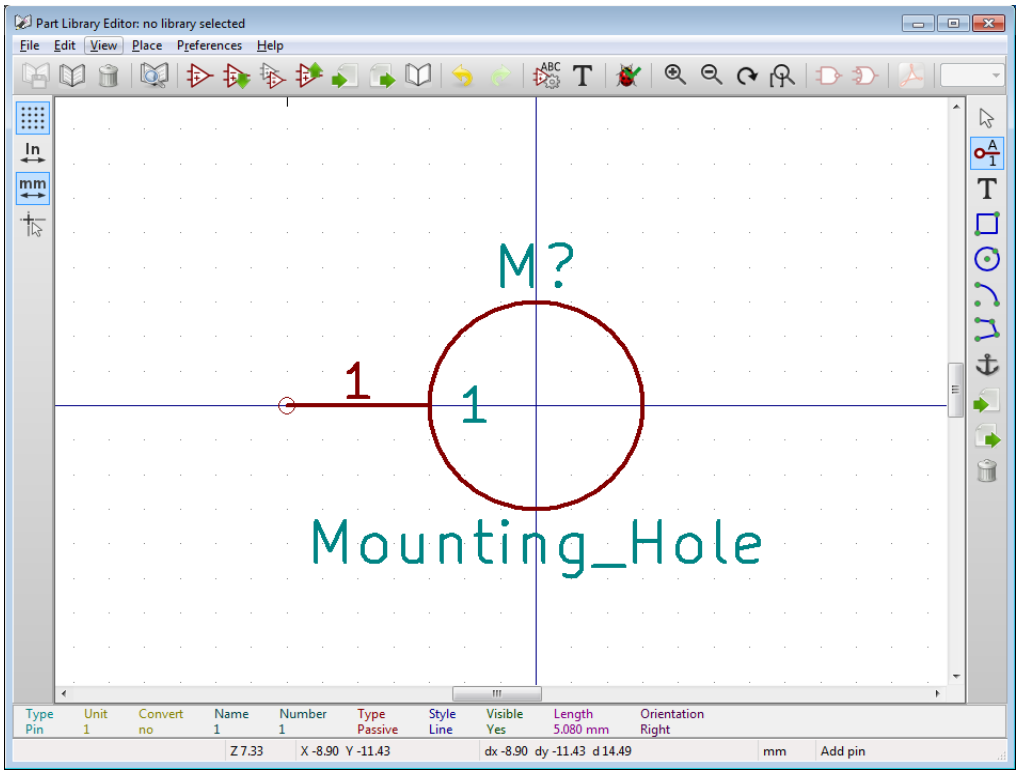
Component name: Mounting_Hole

Default reference designator: M



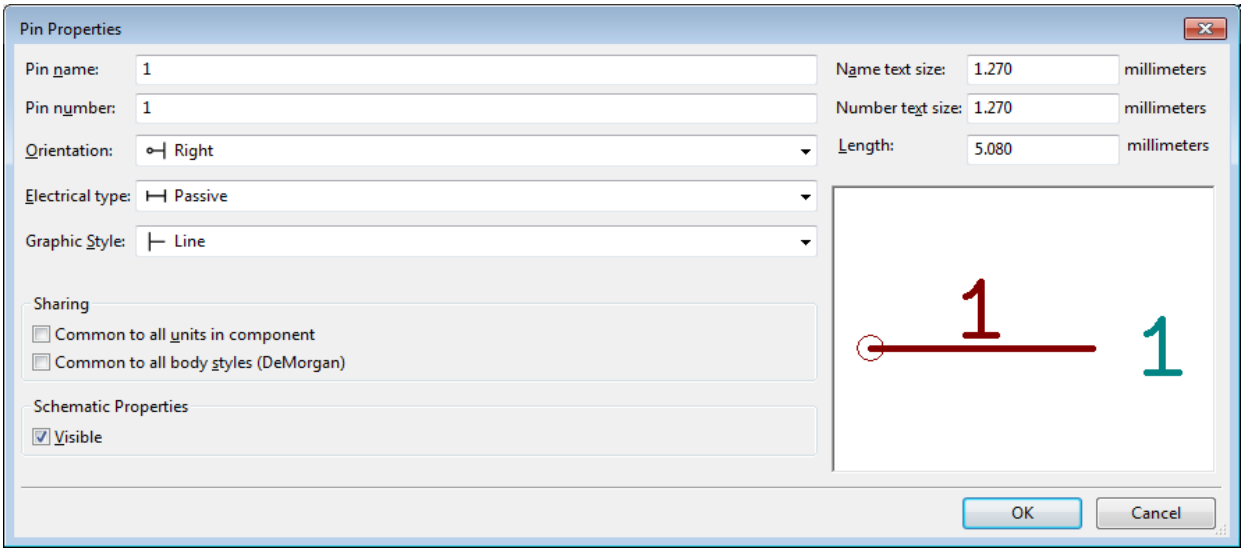

### 2.2.3.4 Draw symbol graphics

Draw a circle with a radius of 0.15”




### 2.2.3.5 Add pin

This will allow the pad of the mounting hole to be electrically connected to the rest of the circuit if desired. Make sure that “Electrical Type” is “Passive”



2.2.3.6 Click “Save component to new library” icon

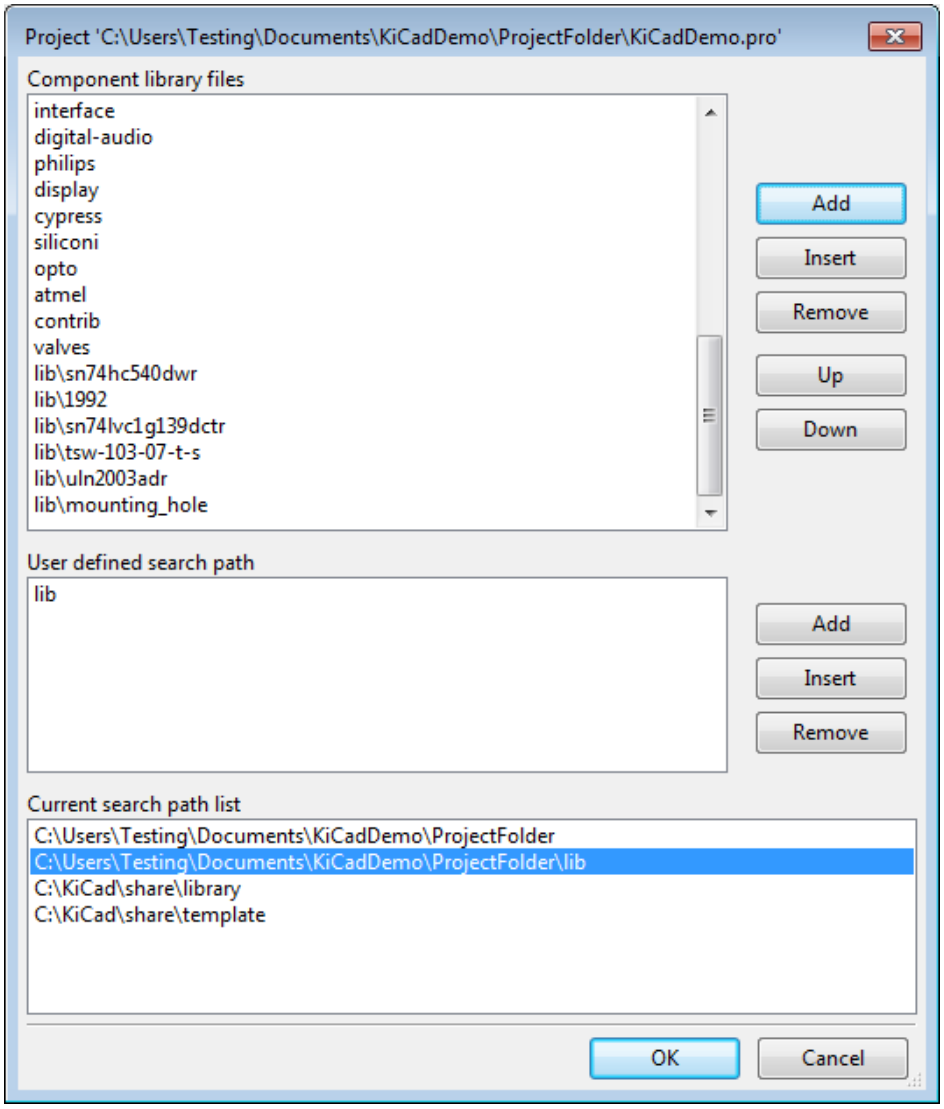
2.2.3.7 Save “mounting_hole.lib” in project “lib” directory

2.2.3.8 Add library to project

Click “Preferences”->”Set Active Libraries”

Click “Add”

Add “lib/mounting_hole.lib”



#### 2.2.3.9 Close Schematic Library Editor

## 2.3 Add components to schematic

### 2.3.1 Purpose

Add four mounting holes symbols to the schematic that will later be associated with our desired mounting hole footprint. The schematic netlist will be updated to allow the other programs in the toolchain to read the updates to the schematic.

### 2.3.2 Background

Components can be added from libraries. Connections can be drawn between component pins, but the mounting hole we are using will not be electrically connected on this board.

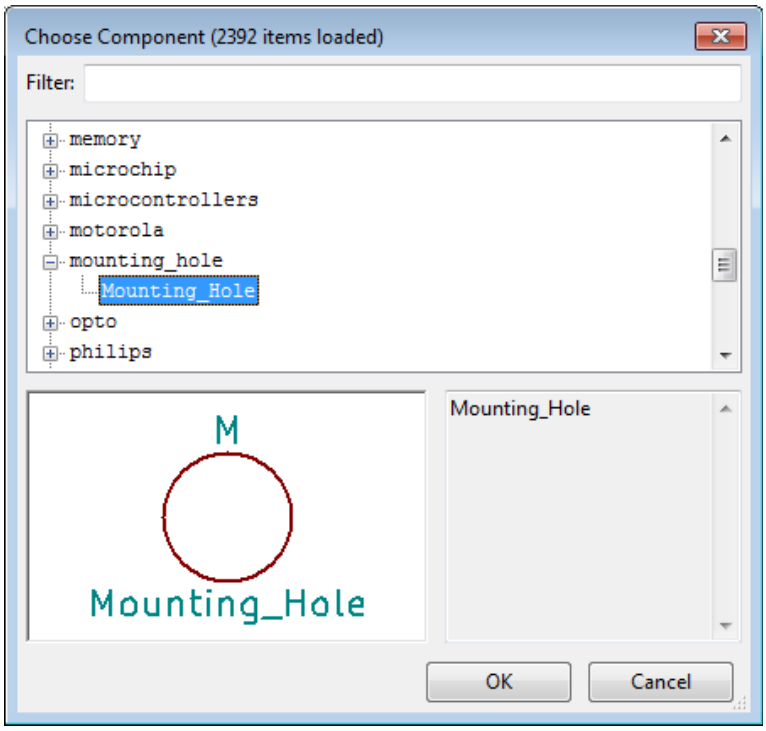
### 2.3.3 Procedure

#### 2.3.3.1 Open Eeschema

#### 2.3.3.2 On the root sheet, click “Place”->”Component”

#### 2.3.3.3 Click on the schematic

#### 2.3.3.4 Choose “Mounting_Hole”

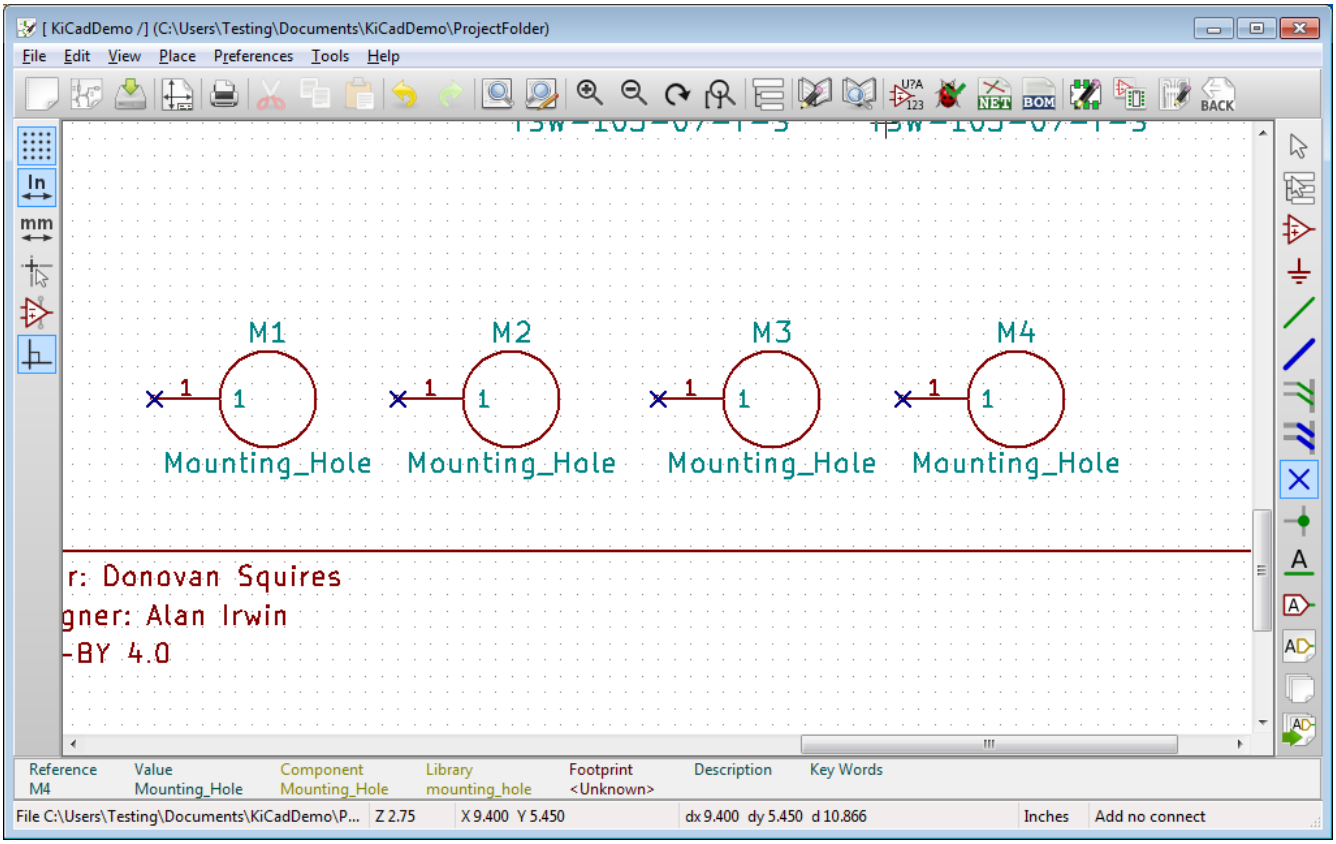



#### 2.3.3.5 Place symbol on schematic

#### 2.3.3.6 Add three more mounting hole components

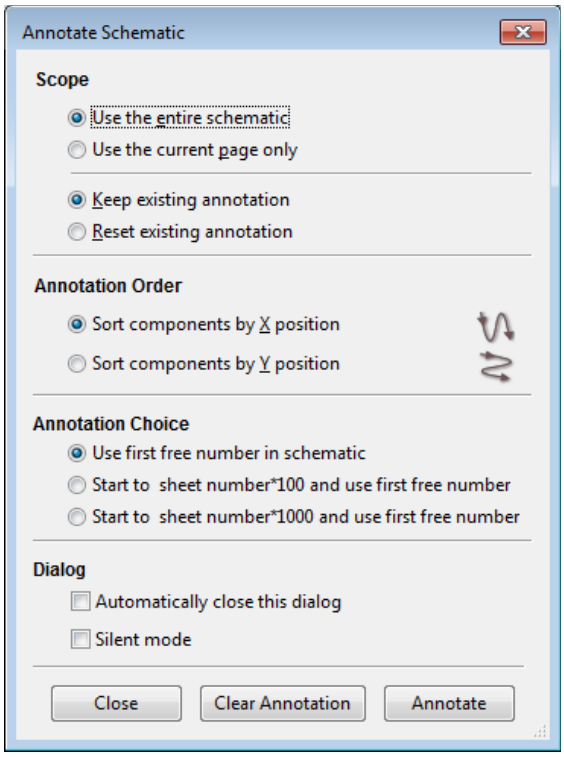
You may click on the schematic again to bring up the select component dialog, or you may copy and paste the first symbol by hovering over it and pressing “c”.

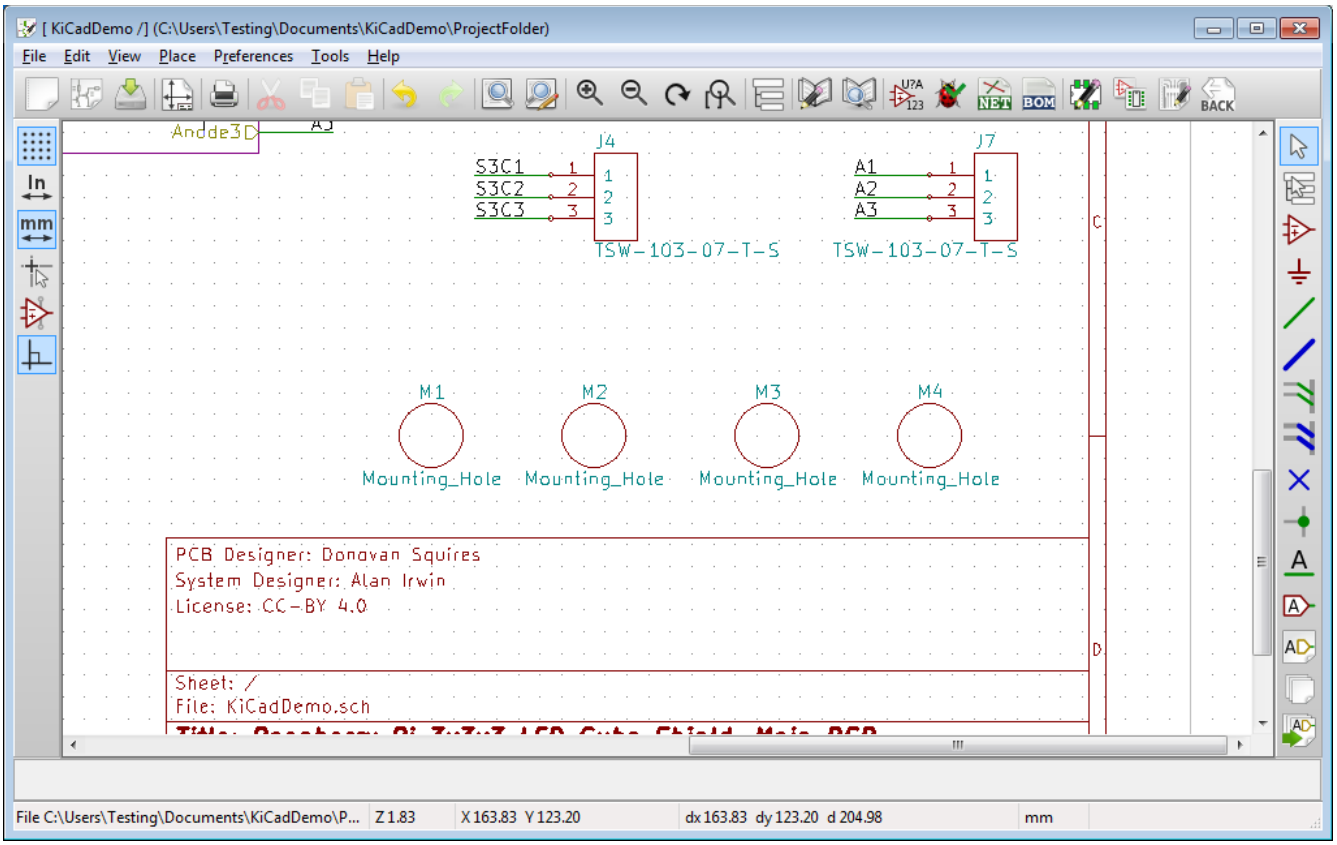
### 2.3.3.7 Click “Place”->”No Connect Flag” and add flags to pins




### 2.3.3.8 Click “Tools”->”Annotate Schematic”

This will replace the reference designators “M?” with unique numbers: “M1”. “M2”, etc. Make sure “Keep existing annotation” is selected to ensure that previously assigned designators are not re-assigned with a different number.



#### 2.3.3.9 Click “Tools”->”Generate Netlist File”

In “Pcbnew” tab, click “Generate”, and overwrite the existing netlist file.

## 2.4 Create new PCB footprint

### 2.4.1 Purpose

Create a new PCB footprint library, make it accessible to Pcbnew, and create a mounting hole footprint.

### 2.4.2 Background

A new style KiCad footprint library is a folder named “<library name>.Pretty” containing PCB footprint files “*.mod” (and 3D models and simulation models).

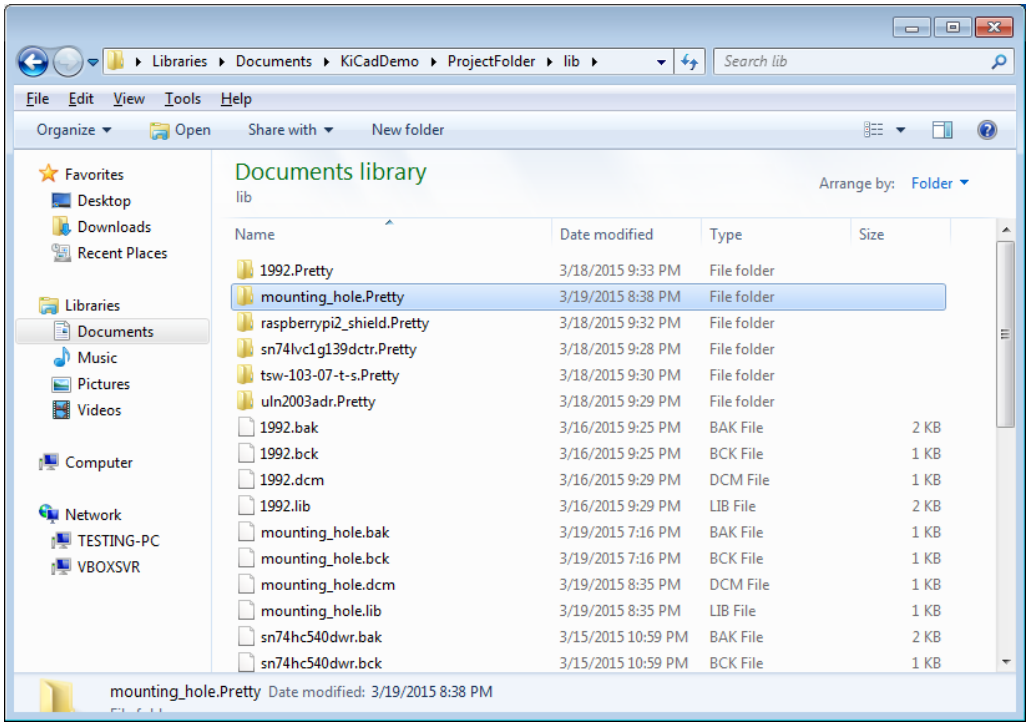
Pcbnew and the footprint editor need to be made aware of library locations before a footprint may be used or saved to the library. Adding to the list of available libraries is complicated. The following link explains some of the current challenges with library management: http://diy-scib.org/blog/working-kicads-terrible-library-management

This is the workflow I prefer, at the moment, for creating PCB libraires.

### 2.4.3 Procedure

#### 2.4.3.1 Create folder for PCB footprints

Using the OS's file manager, create folder “mounting_hole.Pretty” under “ProjectFolder/lib”.

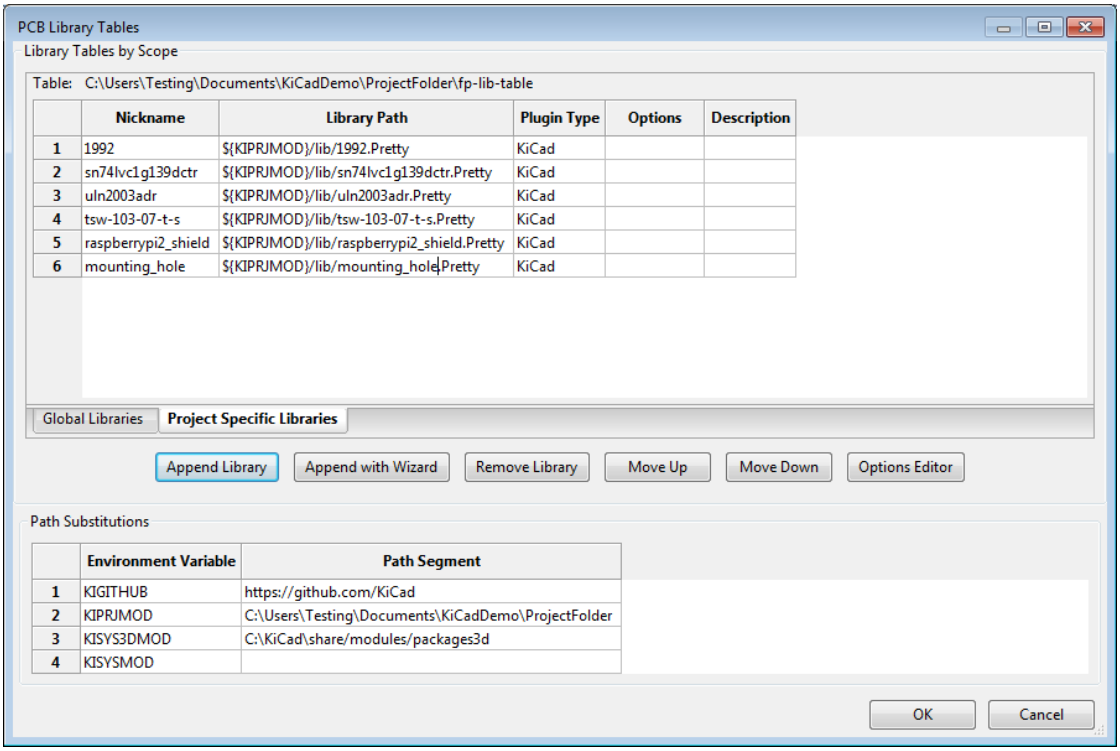



#### 2.4.3.2 Open Pcbnew

#### 2.4.3.3 Click “Preferences”->”Library Tables”

#### 2.4.3.4 Add reference to PCB library

Select tab “Project Specific Libraries”, click “Append Library”, and fill out table as shown in the screenshot.




2.4.3.5 Open Footprint Editor

2.4.3.6 Click “Select Active Library” icon

2.4.3.7 Select the “mounting_hole” library (same as the nickname field from 2.4.3.4

2.4.3.8 Click “File”->”New Footprint”

Enter footpring name: “mounting_hole_2.75mm_pth”

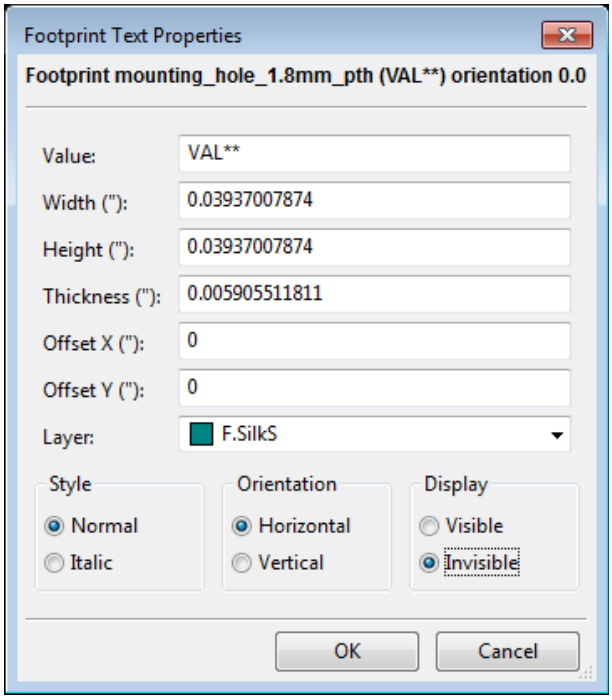
pth – Plated Thru-Hole, a copper plated (mounting) hole with a copper annular ring

npth – Non-Plated Tru-Hole, an unplated (mounting) hole

2.4.3.9 Press “e” on text to edit “Value (VAL**)” field

Set “Display” to “Invisible”

This field, if visible, will place the text in the “Value” parameter field of the schematic symbol on the silkscreen of the board. The silkscreen can get crowded on boards, and the “Reference” text is more important: it will be replaced with the reference designator of the component.




#### 2.4.3.10 Ensure that “mm” is the selected unit

Icon is along the left side of the screen. This will let us define the pad in metric units

#### 2.4.3.11 Click “Place”->”Pad”

#### 2.4.3.12 Click on grid location (0,0) to place pad at origin

#### 2.4.3.13 Hover over pad with cursor and press “e” to edit pad parameters

Set “Pad type” to “Through-hole” and “Circular” ans set “Size” to 4.75mm

Set “Drill” to “Circular” and 2.75mm (same diameter as mounting hole on Raspberry Pi B+/2)

Set “Pad number” to 1, this should correspond to the “Pin number” seen in section 2.2.3.5

#### 2.4.3.14 Click “File”->”Save Footprint in Active Library”

#### 2.4.3.15 Close footprint editor

## 2.5 Associate footprint with schematic symbol

### 2.5.1 Purpose

Tell KiCad that you want the footprint we created associated with each instance of the

symbols we added to the schematic.

### 2.5.2 Background

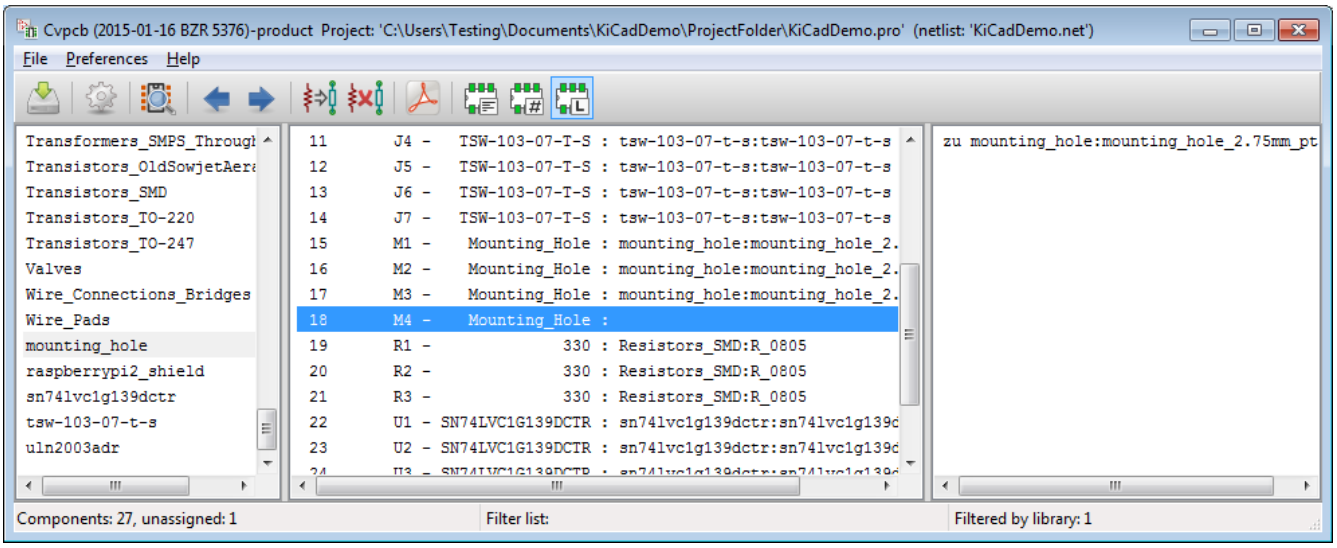
KiCad allows you to select a different footprint for each symbol in the schematic. This gives you flexibility in using different footprints for the same symbol. For instance, if you have a USB connector with 4 pins and a 0.1" pin header with 4 pins, you can use the same schematic symbol for both parts, but you can select the different footprints for the two components on the schematic. The program that assigns footprints is called Cvpcb.

### 2.5.3 Procedure

#### 2.5.3.1 In Schematic Editor, click "Tools"->"Assign Component Footprint"

#### 2.5.3.2 Assign "mounting_hole:mounting_hole_2.75mm_pth" footprint to components M1 to M4

In the left pane, select the footprint library. In the middle plane, select one of the components from the schematic. In the right pane, double-click the name of the footprint you want to assign to the component.



#### 2.5.3.3 Click “File”->”Save Component File”

#### 2.5.3.4 Close Cvpcb

## 2.6 Import netlist in Pcbnew

### 2.6.1 Purpose

Import the changes from the schematic editor to synchronize the components and connections

### 2.6.2 Background

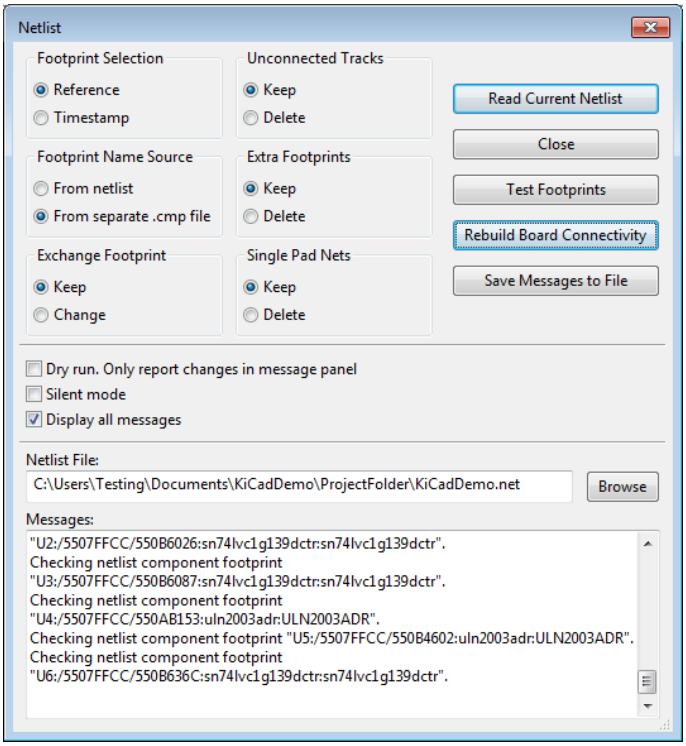
Pcbnew will ensure that all of the symbols in the schematic have their associated components on the board, and Pcbnew will keep track of which pins need to be electrically connected or not connected, according to the schematic drawing.

When you make any changes to the schematic, the changes must be imported into Pcbnew.

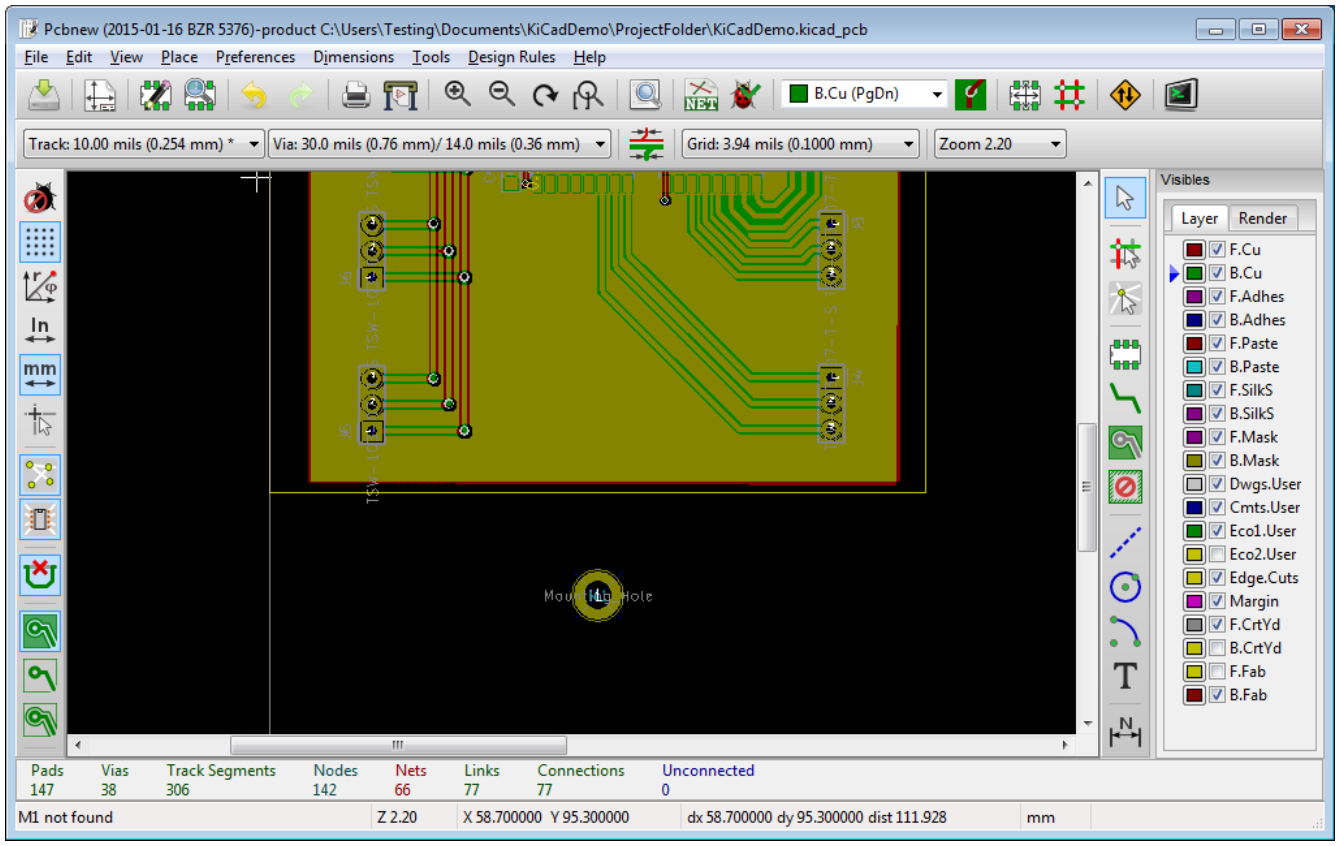
### 2.6.3 Procedure

#### 2.6.3.1 In Pcbnew, click “Tools”->”Netlist”

#### 2.6.3.2 Click “Read Current Netlist” and “Rebuild Board Connectivity”



2.6.3.3 Notice the mounting holes are placed in the PCB editor, but they are all placed at the same location, on top of each other




## 2.7 Place components

### 2.7.1 Purpose

The imported changes from the schematic are not placed intelligently. Components need to be moved so they are not on top of each other, and they need to be placed in the required location on the board.

### 2.7.2 Background

Pcbnew can move the components so they are easier to grab and move. We will have to place the mounting holes in the location that we need, which will correspond to the mounting hole locations on the Raspberry Pi B+/2

### 2.7.3 Procedure

#### 2.7.3.1 In Pcbnew, set zones to “do not show filled zones”

This will make the board easier to see.

#### 2.7.3.2 Click the “Mode footprint: [...]” icon

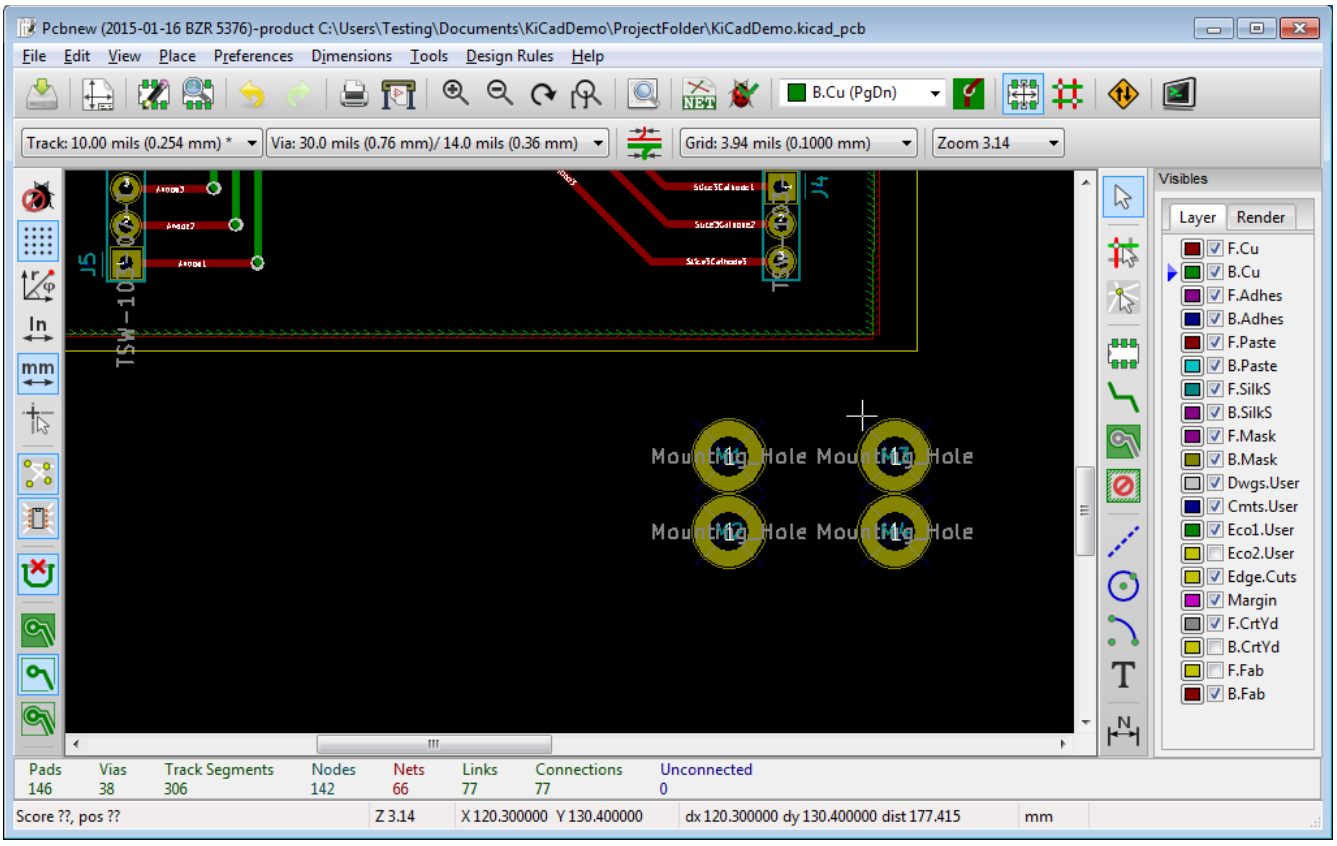
This will make the options we need available in the right-click menu.

#### 2.7.3.3 Right-click anywhere on the board

#### 2.7.3.4 Select “Global Spread and Place”->”Spread out footprints not already on board”

If this was a new board, you can select “Spread out all footprints”

Result:




#### 2.7.3.5 Enable visibility on F.Fab layer

I placed a schematic footprint that uses this layer to draw an outline of the Raspberry Pi

with its connector and mounting holes.

#### 2.7.3.6 Place mounting holes at the same locations as the Pi holes

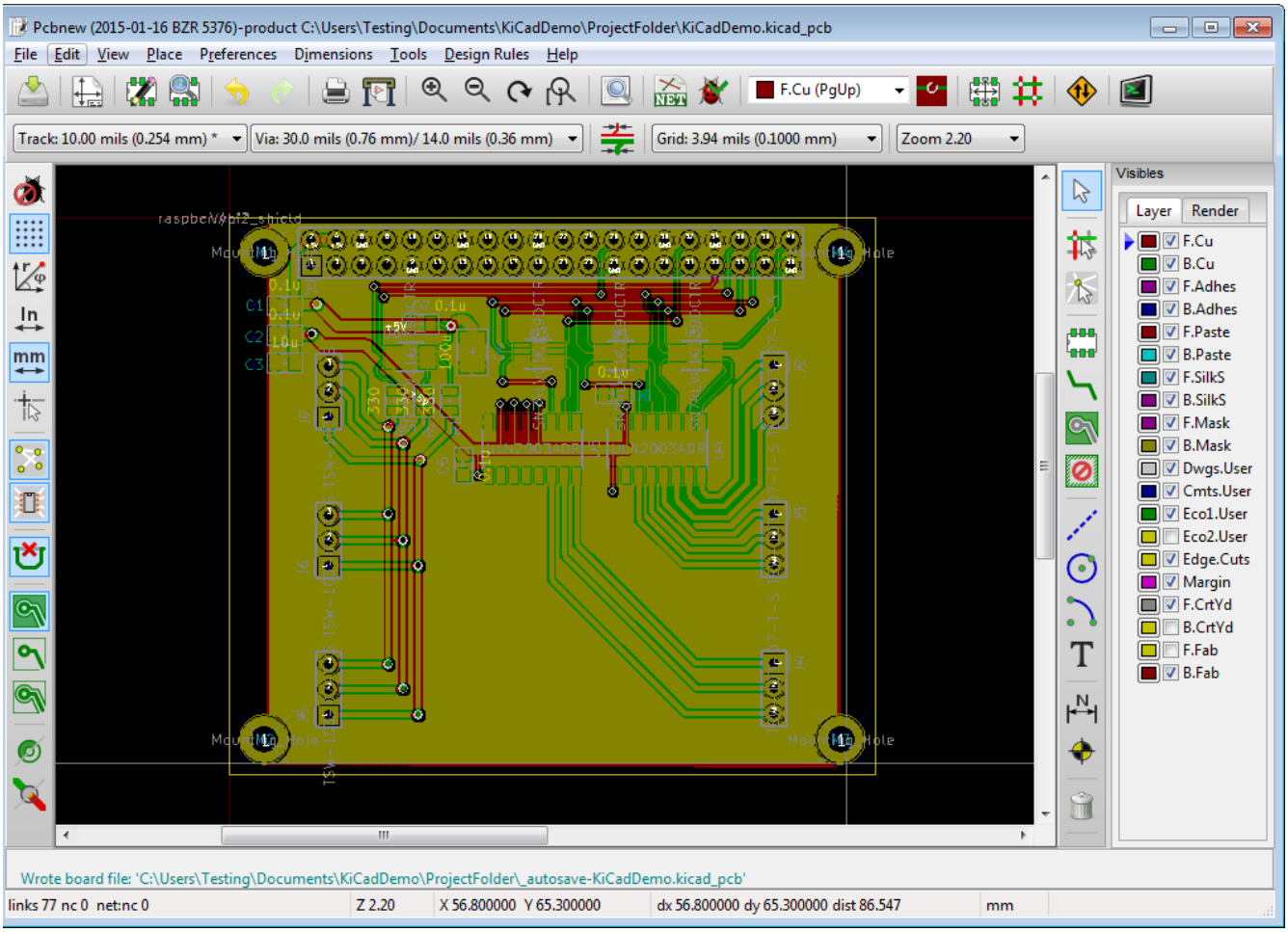
You can try to place them manually, or you can press “e” over the footprint and type the “X” and “Y” position for a more accurate result

#### 2.7.3.7 Right-click on edge of zones on front and bottom copper

For each zone, select “Zone Outline”->”Zones”->”Fill Zone”.

This makes the zone not intersect with the new component.

Zones are a.k.a. copper pours or copper planes or regions in other CAD packages.



#### 2.7.3.8 Select “Tools”->”DRC”

#### 2.7.3.9 Click “Start DRC” then “List Unconnected”

This will check to make sure that pads and nets aren't connected in a way that conflicts with the netlist.

## 2.8 Generate Gerbers

### 2.8.1 Purpose

Produce fabrication files for a board manufacturer

### 2.8.2 Background

Most manufacturers accept a standard text format for defining the board. Gerber RS-274X is the most common standard, which KiCad outputs by default when you select a Gerber output. To define the drill holes, Excellon Drill files are the standard most used.

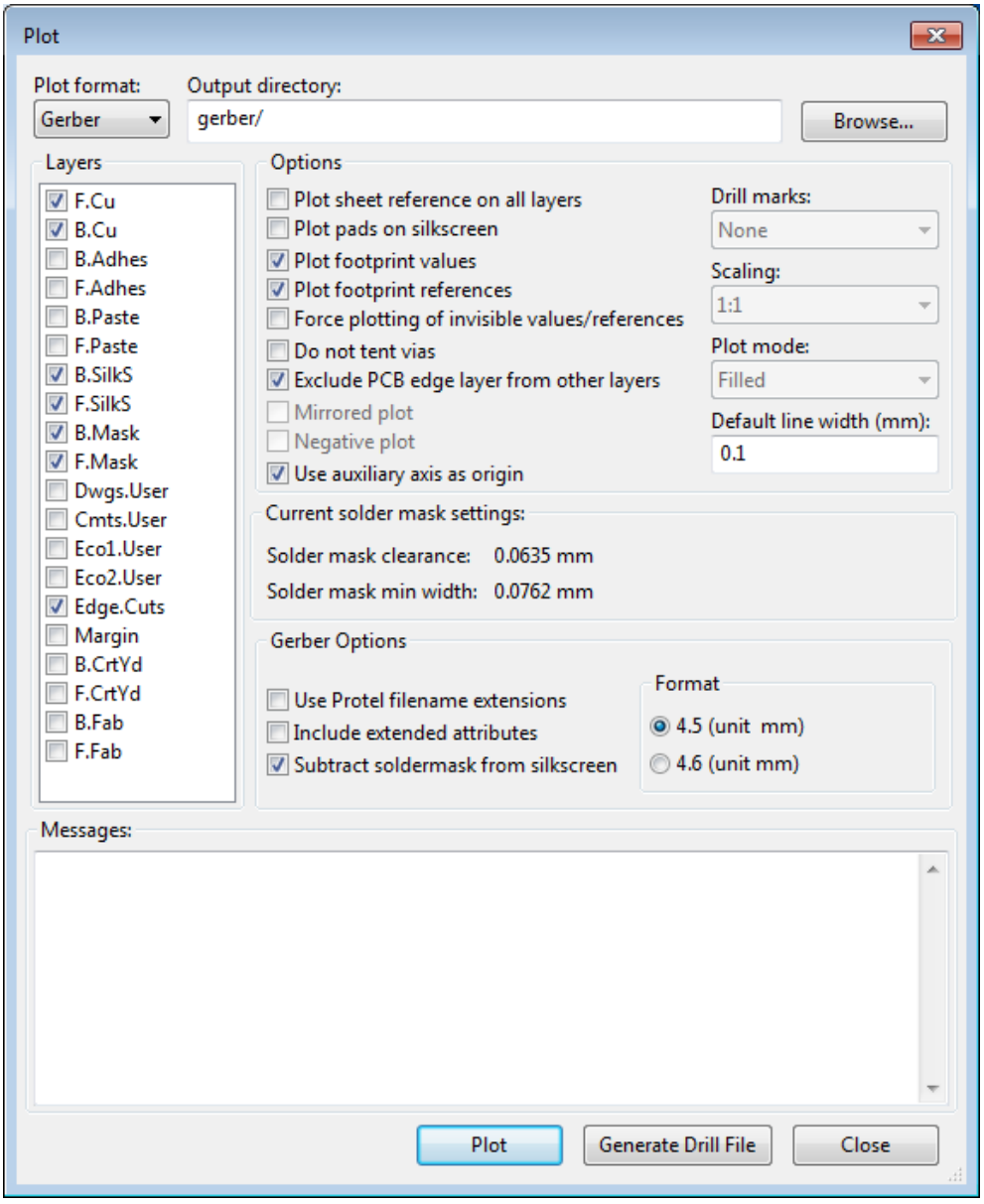
Each layer and the drill file are stored in separate files.

This version of KiCad (2015-01-16 BZR 5376) appears to only output Gerbers in metric format, whereas, most manufacturers specify imperial format (inches). The Excellon Drill file should probably be output as metric to be consistent with the Gerber files.

### 2.8.3 Procedure

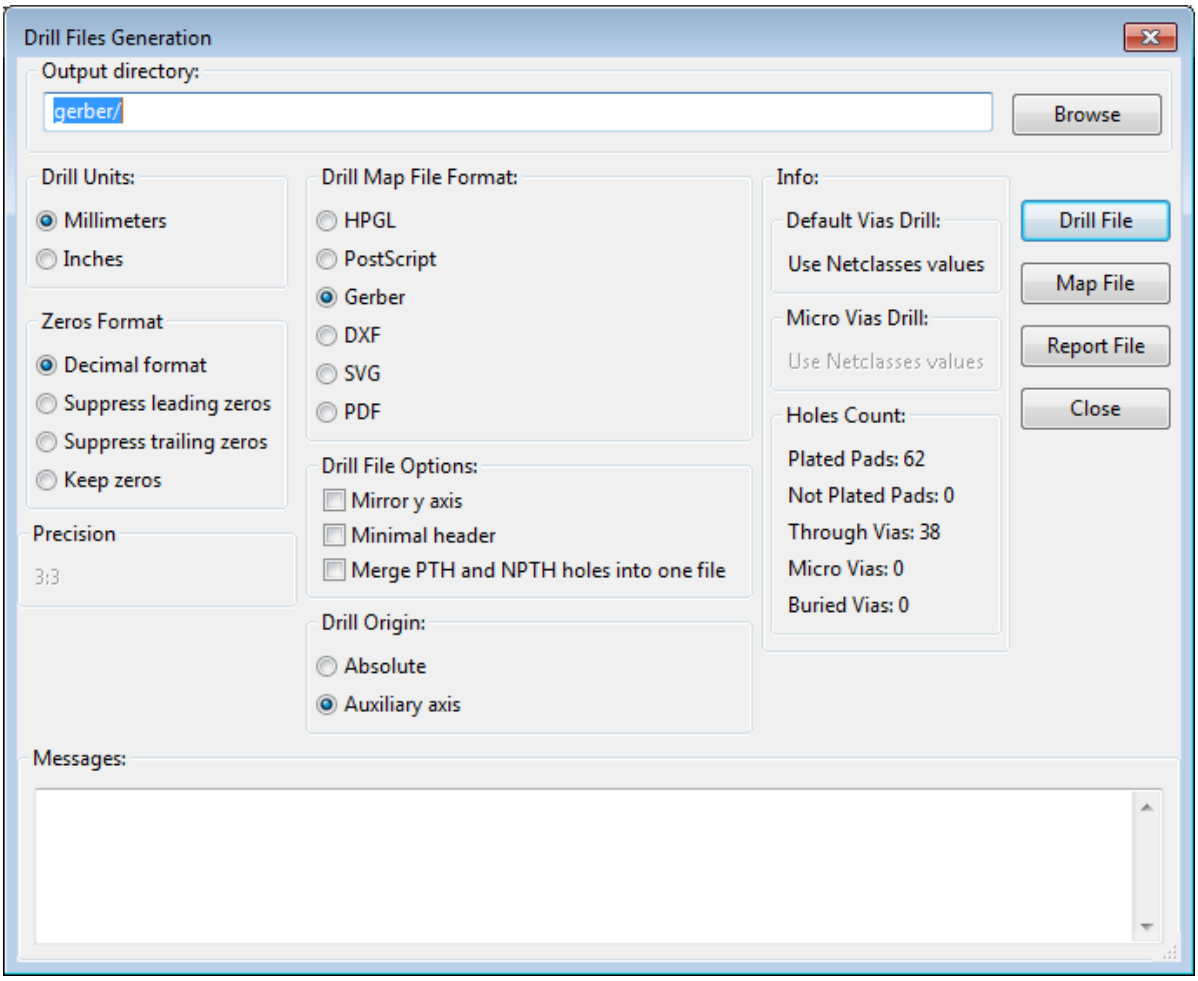
#### 2.8.3.1 In Pcbnew, click “File”->”Plot”

#### 2.8.3.2 Fill out the window as below and click “Plot”



#### 2.8.3.3 In the “Plot” window, select “Generate Drill File”

#### 2.8.3.4 Fill out the window as below and click “Drill File”




#### 2.8.3.5 Fabrication files have been created

You can close windows and exit KiCad. It's good to verify the gerber files are what you expect with a gerber viewer, such as the one included in KiCad or a 3rd party software.

# 3 Appendix

## 3.1 User trace and via bug

When you open a PCB file, the user made track widths and via sizes will not be available. Open the “Design Rules”->”Design Rules” dialog, and the user tracks and vias will be available again.