My journey as a Python plugin developer for KiCad

Mitja Nemec

About me

- Working at Faculty of electrical engineering
- Programming experience:
  - low level C mostly
  - Built a desktop app based upon pyQt5 and python3
- Previously used OrCad, had to switch due to obsolescence
  - Kicad due to licensing (students)

About the talk

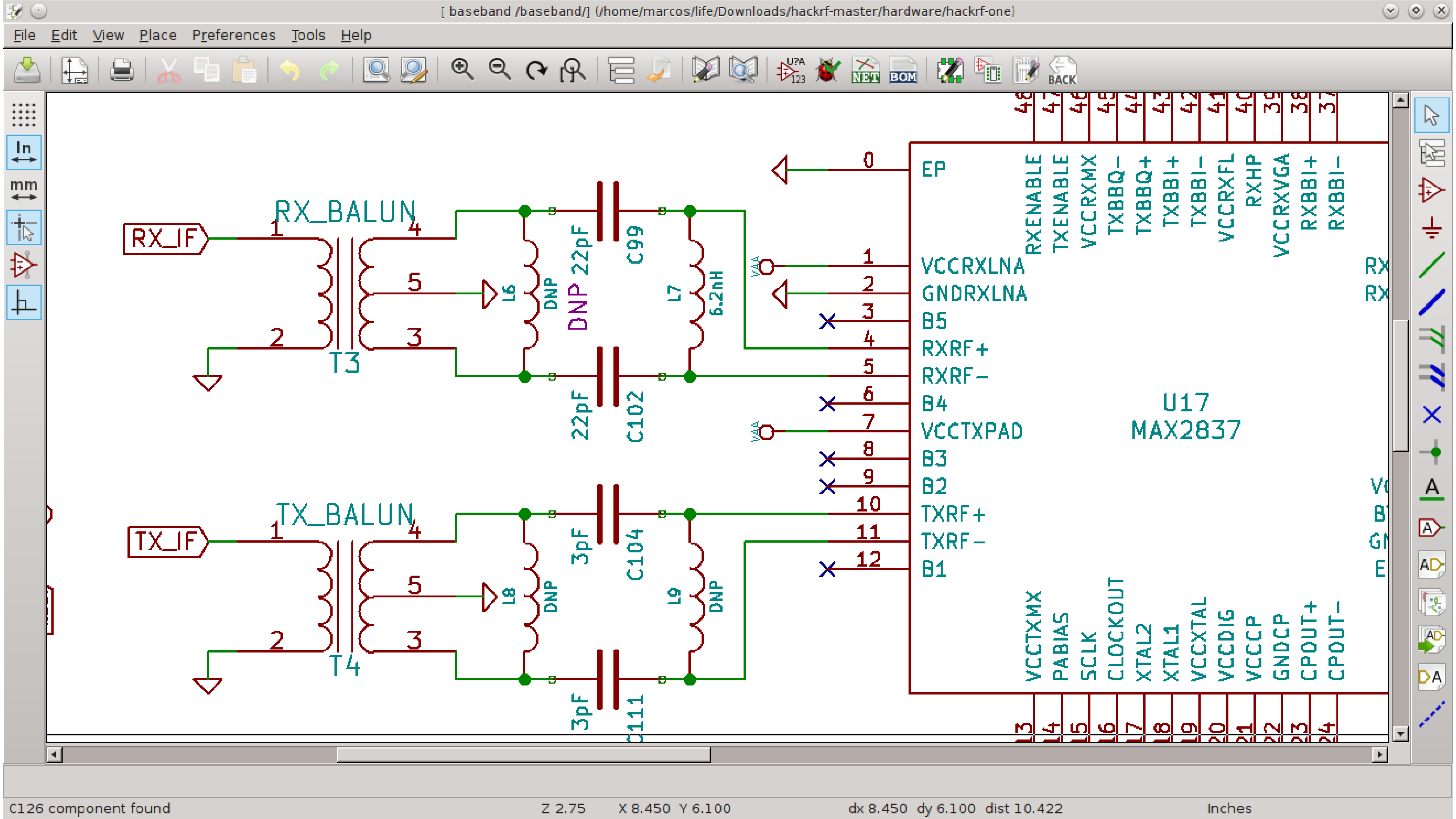
Contents:

- KiCad and EDA tools in general
- KiCad development environment
- My journey

What is KiCad

- Electronic design automation (EDA) suite:
  - Schema, simulations, pcb, gerber, 3D view
- Cross platform
  - Windows, Linux, macOs
- Collection of separate programs that are more and more tightly integrated with each release
- Part libraries (symbols, footprints, 3D models) on GitHub
- It integrates with FreeCad (which is written in python)

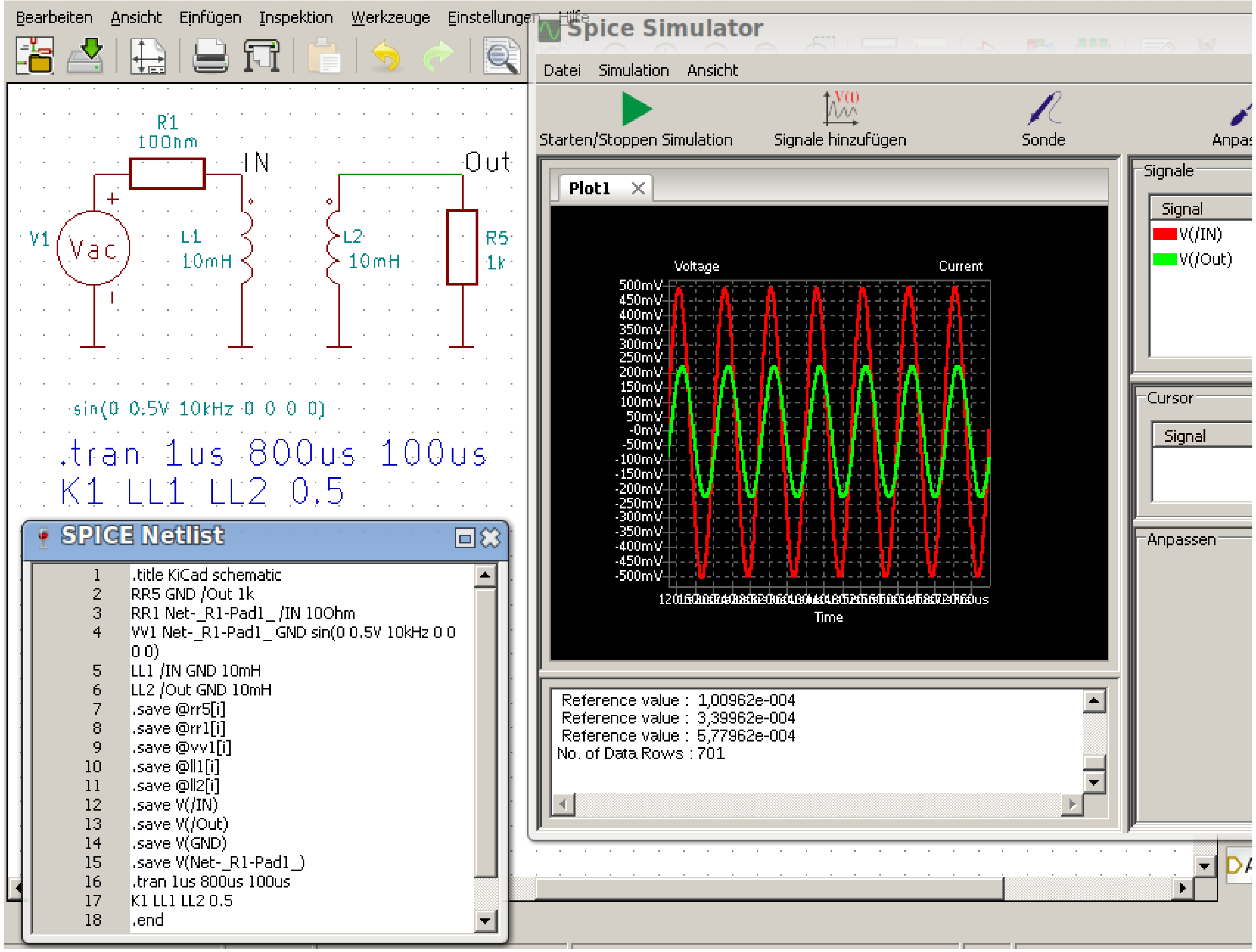
Schematics - eeschema



[baseband /baseband/] (/home/marcos/life/Downloads/hackrf-master/hardware/hackrf-one)

C126 component found   Z 2.75   X 8.450 Y 6.100   dx 8.450 dy 6.100 dist 10.422   Inches

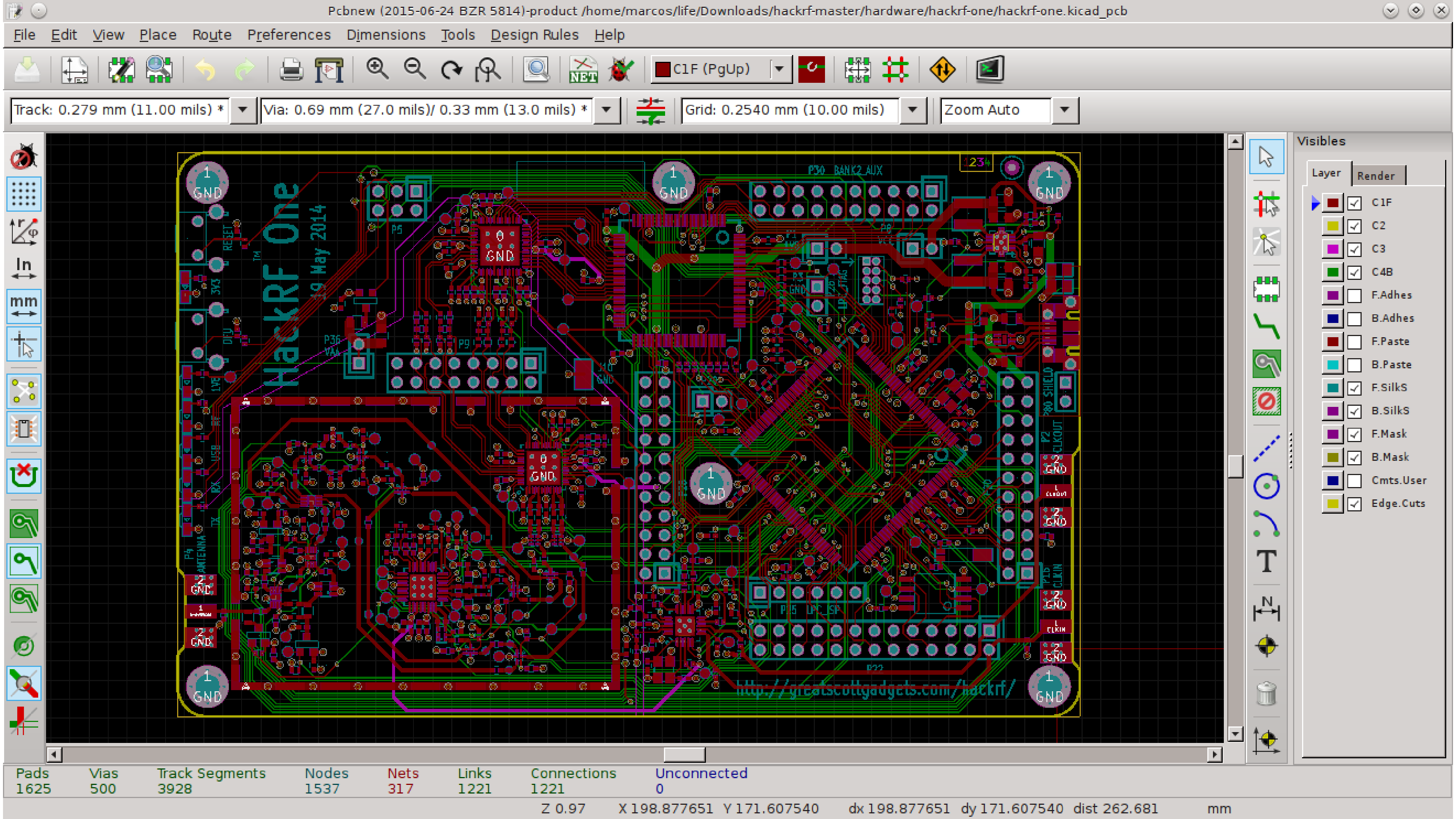
Schematics - eeschema



```
.title KiCad schematic
RR5 GND /Out 1k
RR1 Net-_R1-Pad1_ /IN 10Ohm
VV1 Net-_R1-Pad1_ GND sin(0 0.5V 10kHz 0 0
0 0)
LL1 /IN GND 10mH
LL2 /Out GND 10mH
.save @rr5[i]
.save @rr1[i]
.save @vv1[i]
.save @ll1[i]
.save @ll2[i]
.save V(/IN)
.save V(/Out)
.save V(GND)
.save V(Net-_R1-Pad1_)
.tran 1us 800us 100us
K1 LL1 LL2 0.5
.end
```

```
Reference value :  1,00962e-004
Reference value :  3,39962e-004
Reference value :  5,77962e-004
No. of Data Rows : 701
```

Layout - pcbnew



Pcbnew (2015-06-24 BZR 5814)-product /home/marcos/life/Downloads/hackrf-master/hardware/hackrf-one/hackrf-one.kicad_pcb

C1F (PgUp)

Track: 0.279 mm (11.00 mils) *

Via: 0.69 mm (27.0 mils)/ 0.33 mm (13.0 mils) *

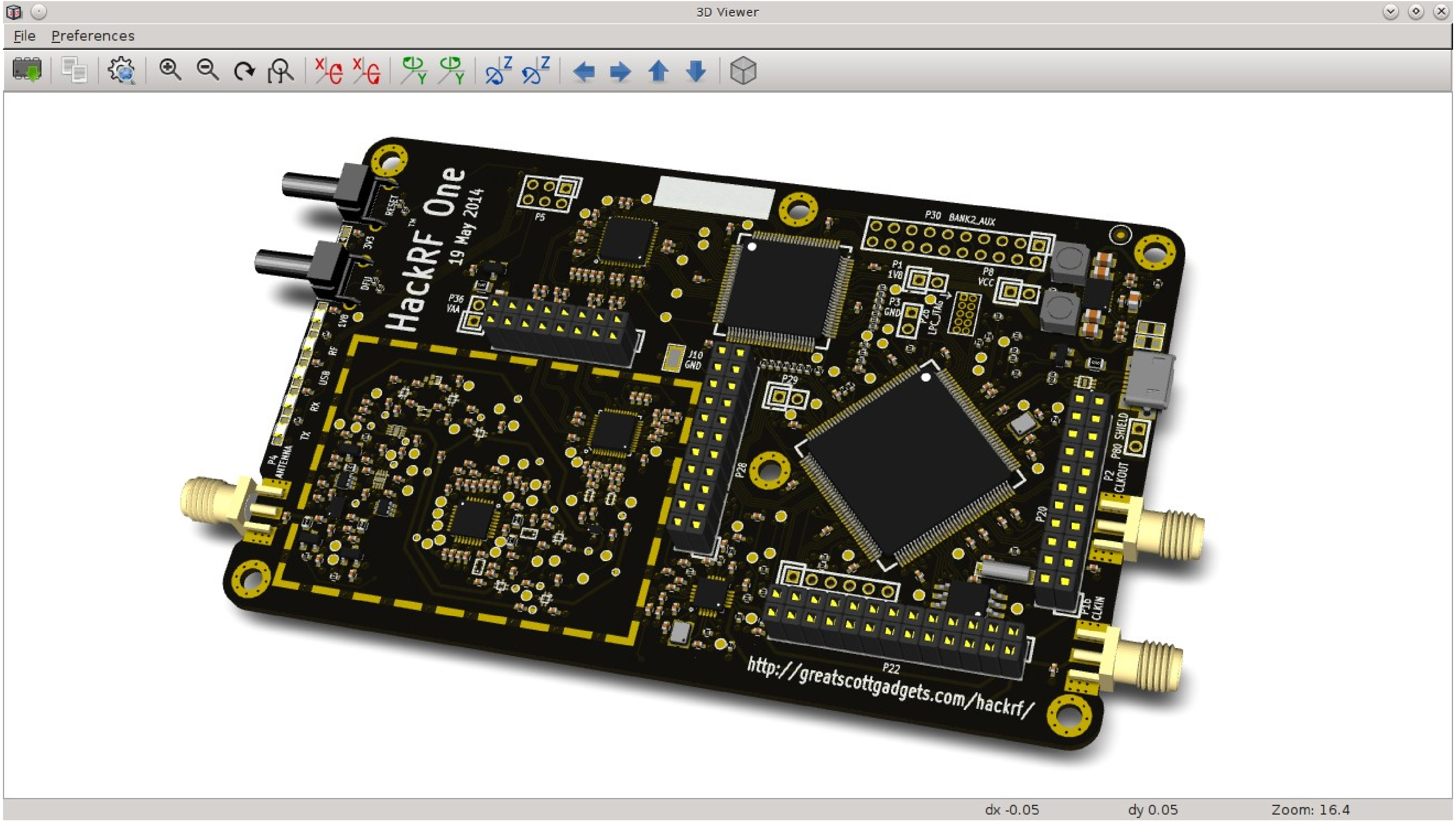
Grid: 0.2540 mm (10.00 mils)

Zoom Auto

- ☑ C1F
- ☑ C2
- ☑ C3
- ☑ C4B
- □ F.Adhes
- □ B.Adhes
- □ F.Paste
- □ B.Paste
- ☑ F.SilkS
- ☑ B.SilkS
- ☑ F.Mask
- ☑ B.Mask
- □ Cmts.User
- ☑ Edge.Cuts

```
Pads    Vias    Track Segments    Nodes    Nets    Links    Connections    Unconnected
1625    500     3928              1537     317     1221     1221           0
                    Z 0.97    X 198.877651  Y 171.607540    dx 198.877651  dy 171.607540  dist 262.681      mm
```

Layout - pcbnew



EDA tools

- Historically by the EE for the EE
- CS concepts are slowly creeping in
- KiCad is at the front of this
  - Projects and libraries can be handled by Git
  - Libraries hosted on Github, utilizing automated testing on PR
  - Has python interface

KiCad development cycle

- I started with 4.0.x (late 2016), now on 5.1.2, V6 is being developed
- Using semver
  - Major version file format change
  - Minor version GUI changes + bugfixes
  - Patch version only bugfixes.
- Development cycle:
  - After major release, work start toward next major release
  - Nightly builds available

KiCad development cycle

- After major release:
  nightly builds are not really stable → not many users
- Towards the end of release cycle:
  nightly build become stable → bigger number of users
- The python API can change significantly even during patch releases
- Some file formats can change even during patch releases

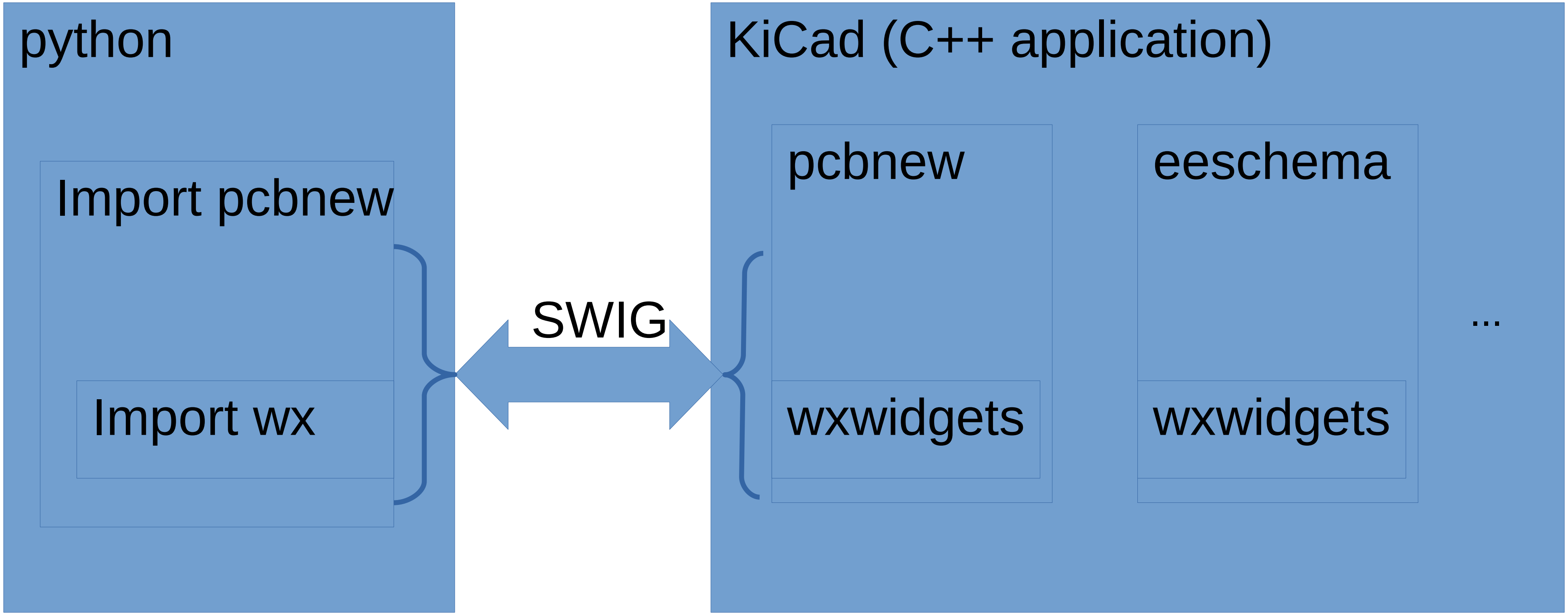
KiCad - internals

- C++ application (mainly C++11)
- GUI based on wxWidgets 3
- Uses OpenGL for graphics
- Built with GCC (clang on macOS)
- Files are human readable
- pcbnew (Layout program) has python API (via SWIG)
- eeschema python API is promised for V6

Python API

- Only available within pcbnew (PCB layout program)
- Gives access to read and modify underlying data (component locations, track parameters, ...)
- Gives access to some of the methods (can export data, ...)
- Gives access to GUI elements (menu items, can create dialogs, can add toolbar buttons, ...)
- Is not fixed and can (and will) change

Python API



python

Import pcbnew

Import wx

SWIG

KiCad (C++ application)

pcbnew

wxwidgets

eeschema

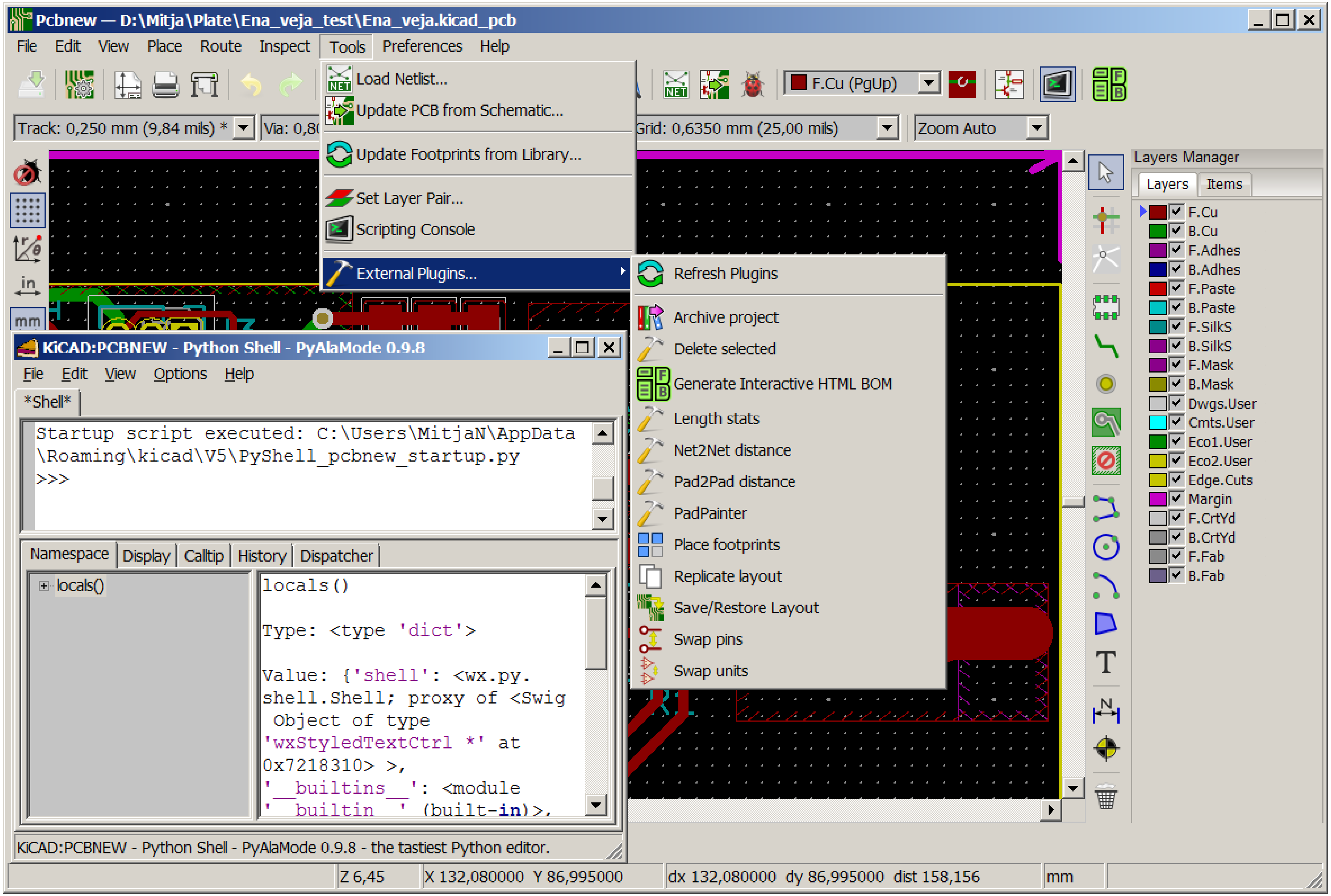
wxwidgets

...

Python API

- Generated with SWIG
- Doxygen documentation of API available:
  - History of documentation not available
- Scripting console available
- Plugin interface is available for
  - BOM generation
  - Footprint generation
  - General interactive plugins (called active plugin)
- Also available as python module which can be used standalone

Python API



```
Startup script executed: C:\Users\MitjaN\AppData
\Roaming\kicad\V5\PyShell_pcbnew_startup.py
>>>
```

```
locals()

Type: <type 'dict'>

Value: {'shell': <wx.py.
shell.Shell; proxy of <Swig
 Object of type
'wxStyledTextCtrl *' at
0x7218310> >,
'__builtins__': <module
'__builtin__' (built-in)>,
```

Python as a portable platform

- The reality is quite different

  On Windows

  - Still on python2 due to dependencies:
    - wxpython 3 supports only python2 (on windows)
    - wxpython 4 (phoenix) supports python 3
      does not build on windows using msys2&gcc
      (which are used for building KiCad)
    - KiCad does not build with MSVC (currently)

Python as a portable platform

On Windows

- Kicad’s python is built with MSYS2&gcc
- Most python packages for windows are built with MSVC
- Only pure python packages from pypi are available
- Numpy is not available due to binary incompatibility

Python as a portable platform

On linux:

- Wxpython on linux can be built against GTK2 or GTK3
- Distributions are migrating from GTK2 to GTK3, but not all libraries are available (e.g. wxpython is still on GTK2
- If main app uses wxwigets built against GTK3, you can not use wxpython built against GTK2
- Migration to python3 is taking place. As it is system python on new distributions

Python as a portable platform

macOS:

- Seems like a most stable platform from the KiCad python developer/user perspective
- Same GUI widgets that work on linux/Windows behave differently on macOS

Python as a portable platform

- All of the above are a reason for the high churn ratio among python script and plugin developers for KiCad
- As a plugin writer you have to be aware of all of this if you make your plugins public
  - Only use vanilla python modules
  - Be compatible with python2 and 3

Pythons scripts

- Intended for console input
- A lot of availability
  - Not many are supported
- Somebody had an itch, scratched it, released the code for public
- If you are lucky there is an example
- Might not work on your system (uses numpy, was tested only on 4.0.7, ...)

Action plugins

- Intended to expand pcbnew functionality through GUI elements
- Very few of them are actively maintained
- There are good examples on which you can build upon:
  - action_menu_annular_check
  - action_menu_move_to_layer
  - InteractiveHtmlBom
  - Replicate_Layout
  - Place_footprints
  - Archive_project

Action plugins

- Quite good architecture
- Makes it easy to split GUI from the business back-end
- If you are careful you can make the back-end useful on its own (headless mode, further integration, reuse ...)
- Installation is somewhat rough

My journey

- Started in November 2017
- *Replicate layout* plugin first
  - built upon previous work (Miles Mccoo, ...)
- UX one of the main requirements
- Slowly added more and more plugins
- Currently at 10 plugins and 81 GitHub stars

My journey

- Added testing quite soon:
  - As python API changes, unit test will catch this
  - Integration tests are harder to build (requires specific diff tool to compare generated output from expected one). But is worth the effort.
  - TDD not practical. You need to generate test output, check it manually and if it is alright it can serve as a reference
  - Test input and output has to be change often when adding new features

My journey

- Logging for instrumentation
  - Once your plugin is public you **will** get a bug report. It is nice to have some data behind it
  - *repr* and even *pickle* are quite useful
  - Do not put too much data into instrumentation as some users cannot share their designs (in any form)
- Cross platform issues:
  - GUI behaves differently (especially macOS)

My journey

- Embed plugin version info into it
  - The best solution would be embed commit SHA
    - Involves building compete build pipeline
  - version.info file keeping only one number, increased by pre-commit hook

My journey

- Python3 migration:
  - If you are developing something for python2

    ```
    from __future__ import absolute_import
    from __future__ import division
    from __future__ import print_function
    ```

    is a must from the start
  - Hard to do if you don’t have an environment available working on windows
  - Testers (especially persevering ones) are a godsend
  - You can turn a users into testers

My journey

- Refactoring:
  - When learning, it is likely you’ll screw up
  - Don’t delay with refactoring
  - You might need to refactor same codebase multiple times

My journey

- Documentation:
  - Documentation effort is not negligible
  - Maintaining documentation is PITA
  - README.md is quickly not enough
  - EDA tools are visual → hard to document with text

My journey

- Documentation:
  - Language issues:
    - difference in slang (don’t be stickler about it)
    - prefer simplified English
  - Production issues:
    - how to make good videos,
    - how to get rid of Slavic accent
  - Presentation issues:
    - where to store documentation
    - try not to use too much bandwidth

My journey

- Github usage
  - Issues
    - Different etiquette (who closes an issue, ...)
    - Few of the users are willing testers
  - Pull requests
    - Most of the users are EE, so PR are rare
    - Do you accept PR which is functionally correct but differs significantly in style:
      - map instead of list comprehension
      - using regular expressions for simple text search/replace

My journey

- Github usage
  - Contirbutions
    - TBD
  - Releases
    - TBD
  - Wiki
    - TBD

Conclusion

- It is really rewarding to be part of something bigger
- You learn a lot by doing something outside of your comfort zone
    - Logging is a must
    - Handle errors gracefully and let the user know or at least point in the right direction
    - Process is as important as the code