



# Summer Fellowship Report

On

**eSim**

Submitted by

**Anjali Jaiswal**

B.Tech (Computer Science and Engineering)

NIT Durgapur

**Mahfooz Ahmad**

B.Tech (Computer Science and Engineering)

NIT Meghalaya

**Neel Shah**

B.Tech (Computer Engineering)

K.J. Somaiya College Of Engineering, Mumbai

Under the guidance of

**Prof.Kannan M. Moudgalya**

Chemical Engineering Department

IIT Bombay

July 22, 2019



# Acknowledgment

The fellowship opportunity we had with FOSSEE Team was a great chance for us to learn and experience professional software development. Therefore, we consider ourselves lucky to have been provided with such a wonderful opportunity. We are also grateful for having a chance to meet so many skilled and talented professionals who led us through this internship. Bearing in mind, we'd like to use this opportunity to express our deepest gratitude and special thanks to *Mr. Sunil Shetye, Mrs. Gloria Nandihal, Mr. Saurabh Bansode, Mrs. Usha Vishwanathan and Mrs. Vineeta Gharvi* who in spite of being extraordinarily busy with her/his duties, took time out to hear, guide and keep us on the correct path and allowing us to carry out our assigned tasks at their esteemed organization during the training. We express our deepest thanks to **Prof. Kannan M. Moudgalya** for taking part in useful decisions & giving necessary advises and guidance and for arranging all facilities to make our life easier. We choose this moment to acknowledge his contribution gratefully. It is our radiant sentiment to place on record my best regards, deepest sense of gratitude to *Mallikarjuna Reddy and Bharghav Katakam* for their help in testing every piece of code which was extremely valuable for our work. We perceive this opportunity as a big milestone in our career development. We will strive to use gained skills and knowledge in the best possible way, and we will continue to work on their improvement, in order to attain desired career objectives. We also hope to continue cooperation with all of you in the future.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	PyQt4 . . . . .	3
1.1.1	QtCore . . . . .	4
1.1.2	QtGUI . . . . .	4
1.2	matplotlib . . . . .	4
<b>2</b>	<b>PEP8</b>	<b>5</b>
2.1	Rules for PEP8 . . . . .	5
<b>3</b>	<b>Documentation</b>	<b>8</b>
<b>4</b>	<b>Sphinx &amp; Read the docs</b>	<b>9</b>
4.1	Sphinx . . . . .	9
4.2	Read The Docs . . . . .	12
<b>5</b>	<b>Bugs</b>	<b>15</b>
5.1	Default Component Loading . . . . .	15
5.2	TreeWidget UI . . . . .	16
5.3	Model Editor Bug . . . . .	18
5.4	Python Plotting . . . . .	19
5.5	Device Model Tab . . . . .	20
<b>6</b>	<b>Additional Features</b>	<b>21</b>
6.1	Upload Subcircuit . . . . .	21
6.2	Rename Project . . . . .	22
6.3	Logging . . . . .	22
6.4	Change lib format . . . . .	24
<b>7</b>	<b>Installer</b>	<b>25</b>
7.1	Pyinstaller . . . . .	25
7.2	NSIS . . . . .	27
7.3	Uninstaller . . . . .	29

# Chapter 1

## Introduction

FOSSEE (Free and Open Source Software in Education) project promotes the use of free and open-source tools to improve the quality of education in our country. They aim to reduce dependency on proprietary software in educational institutions. They encourage the use of FOSS tools through various activities to ensure that commercial software is replaced by equivalent FOSS tools. They also develop new FOSS tools and upgrade existing tools to meet requirements in academia and research. Incorporated to FOSSEE program, this fellowship's main aim is to introduce students to the FOSS in various engineering fields and to become a part of this big community. We were selected for this fellowship on the basis of screen tasks submitted by us. As part of fellowship program, we were introduced to open source technologies (Python and its packages like PyQt4, matplotlib) and FOSS software for electronic design simulation, like KiCad, NgSpice. At the beginning of the fellowship we formulated several learning goals, which we wanted to achieve:

- To understand the functioning and working conditions of a government organization.
- To see what it is like to work in a professional environment.
- To see if this kind of work is a possibility for our future career.
- To use our knowledge and skills and to further increase them.
- To learn about organizing an open source project.
- To learn what does it take to launch FOSS in the market, life of people, hardships to convince them about its benefits.
- To enhance our communication skills.
- To strengthen professional and social network.

### 1.1 PyQt4

PyQt is a GUI widgets toolkit. It is a Python interface for Qt, one of the most powerful, and popular cross-platform GUI library. PyQt is a blend of Python programming language and the Qt library.

### **1.1.1 QtCore**

This module contains non-GUI functionality for working with file and directory etc.,

### **1.1.2 QtGUI**

This module contains all the graphical controls.

## **1.2 matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

# Chapter 2

## PEP8

PEP 8 is Python's style guide. It's a set of rules for how to format your Python code to maximize its readability. Writing code to a specification helps to make large code bases, with lots of writers, more uniform and predictable, too. PEP is actually an acronym that stands for Python Enhancement Proposal.

### 2.1 Rules for PEP8

- **Indentation:** PEP8 suggests indentation we should use four spaces per indentation level. Below examples explain few cases:

```
1 # Add 4 spaces (an extra level of indentation) to distinguish
   arguments from the rest.
2 for parents, children in list(
3     self.obj_appconfig.project_explorer.items()):
4     if os.path.exists(parents):
5
6     # Aligned with opening delimiter.
7     oldFilePath = os.path.join(updatedProjectPath,
8                                 projectFile)
9
10    # Hanging indents should add a level.
11    foo = long_function_name(
12        var_one, var_two,
13        var_three, var_four)
14
15
```

Listing 2.1: Indentation example

- **Maximum Line Length:** The maximum length of a line should not exceed 72. This enhances the readability of code.

```
1 with open('/path/to/some/file/you/want/to/read') as file_1, \
2     open('/path/to/some/file/being/written', 'w') as file_2:
3     file_2.write(file_1.read())
4
```

Listing 2.2: Line Break example

- Lines should break before binary operators.

```
1 income = (gross_wages
2           + taxable_interest
3           + (dividends - qualified_dividends)
4           - ira_deduction
5           - student_loan_interest)
6
```

Listing 2.3: Binary Operators

- **Comments:**

Comments can be inline or block quotes. Inline comments should contain 2 space between code and comment. There should be space between # and comment.

```
1 # calculate a^b
2 def power(a,b):
3     c = a^^b # ^^ is used to find power of one no. to another.
4     return c
```

Listing 2.4: Comments example

- **Blank Line:**

- There should be two blank lines between upper level class definitions or functions.
- Method definitions inside a class are surrounded by a single blank line.

```
1 import os
2
3 class example:
4     self.a = "Example for blank lines rule"
5
6     def fun():
7         print(self.a)
8
9
```

Listing 2.5: Blank Line example

- **Imports:**

- Imports should be in different line.
- Imports should be grouped in the following order:
  - \* Standard library imports.
  - \* Related third party imports.
  - \* Local application/library specific imports.



```
1 from PyQt4 import QtGui, QtCore
2 from configuration.Appconfig import AppConfig
3 import sys
4 import os
5
```

Listing 2.6: Imports example

# Chapter 3

## Documentation

The first task that we were assigned to the internship was to document the whole code base of eSim. The initial thought was documentation is not easy thing and it does not even generate interest. The initial days were little tough with documentation but day by day with the help of mentors it became easy and by the end of internship we started liking it and even started doing documentation for our own projects and works. The most part of my documentation was for the front-end part of application for which I learnt PyQt4. PyQt4 is very friendly for the purpose of front-end designing. It provides lot of calsses and function that are easy to use and creates impressive designs. Class **QtGui** and **QtCore** is backbone of eSim application. Few methods of QtGui that built essential set-up

- QSplashScreen
- QWidget
- QDialog
- QDockWidget
- QVBoxLayout

In the mean while we also learned about Qt designer, which is very smart and efficient way to design applications and ease the pain to write code specially for layouts and aligns.



Figure 3.1: Splash Screen

# Chapter 4

## Sphinx & Read the docs

### 4.1 Sphinx

When we were doing documentation we came to know about very interesting thing Sphinx, we got to know that how we can even convert our documentation into html pages or other format we need like pdf, ePub etc.

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by **Georg Brandl** and licensed under the BSD license.

Sphinx uses reStructuredText as its markup language, and many of its strengths come from the power and straightforwardness of reStructuredText and its parsing and translating suite, the Docutils.

There are few conditions of reSt to parse the document

- Every class and method should have block comment outside the class or function. This line is generally author's note to user.
- Every class and methods should contain docstring inside them. This contains summary and description of whole method.
- The first line should be summary and in imperative mode.
- There should be one line gap between summary and description.

The further steps on how to use sphinx are follows:

- To download sphinx in Linux

```
1 apt-get install python3-sphinx
```

- Now enter in the directory where your python code is then run command:

```
1 sphinx-quickstart
```

Now you can see following question on command prompt:

```
1 > Root path for the documentation [.]<ENTER>
2 > Separate source and build directories (y/N) [n]: y
3 > Name prefix for templates and static dir [-]: <ENTER>
4 > Project name: an_example_pypi_project
5 > Author name(s): Andrew Carter
```

```

6 > Project version: 0.0.1
7 > Project release [0.0.1]: <ENTER>
8 > Source file suffix [.rst]: <ENTER>
9 > Name of your master document (without suffix) [index]: <ENTER>
10 > autodoc: automatically insert docstrings from modules (y/N) [n]:
    y
11 > doctest: automatically test code snippets in doctest blocks (y/N
    ) [n]: n
12 > intersphinx: link between Sphinx documentation of different
    projects (y/N) [n]: y
13 > todo: write todo entries that can be shown or hidden on
    build (y/N) [n]: n
14 > coverage: checks for documentation coverage (y/N) [n]: n
15 > pngmath: include math, rendered as PNG images (y/N) [n]: n
16 > jsmath: include math, rendered in the browser by JSMath (y/N) [n
    ]: n
17 > ifconfig: conditional inclusion of content based on config
    values (y/N) [n]: y
18 > Create Makefile? (Y/n) [y]: n
19 > Create Windows command file? (Y/n) [y]: n
20

```

- After this there are autogenerated folders as build, source and Makefile. The source contains **conf.py** and **index.rst** these are the important files which will help to parse data for html pages.

- We can change index.rst as our requirement and then run command

```

1 make html
2 or
3 sphinx-build -b html srcDir dstDir
4

```

The first one will create a directory name `_build` inside and file **index.txt** including many other files. Same will be with the other command but inside the dir of your choice.

- Next step is to create a code.rst file in which we will write code to auto generate documentation.

```

1 Heading of Page
2 =====
3
4 .. automodule:: name of python file
5    :members:
6

```

- After this we will include this code.rst file in index.rst file of source directory. Run the following commands:

```

1 make clean
2 make html
3

```

Now you can see the changes by opening index.rst in web-browser.

This is general method of setting up sphinx and generating documentation for python files. We can replicate this method for more complex structure of files and folder by creating `.rst` files for each python files and including them in `index.rst` file.

Below are the few screenshots of the documentation:

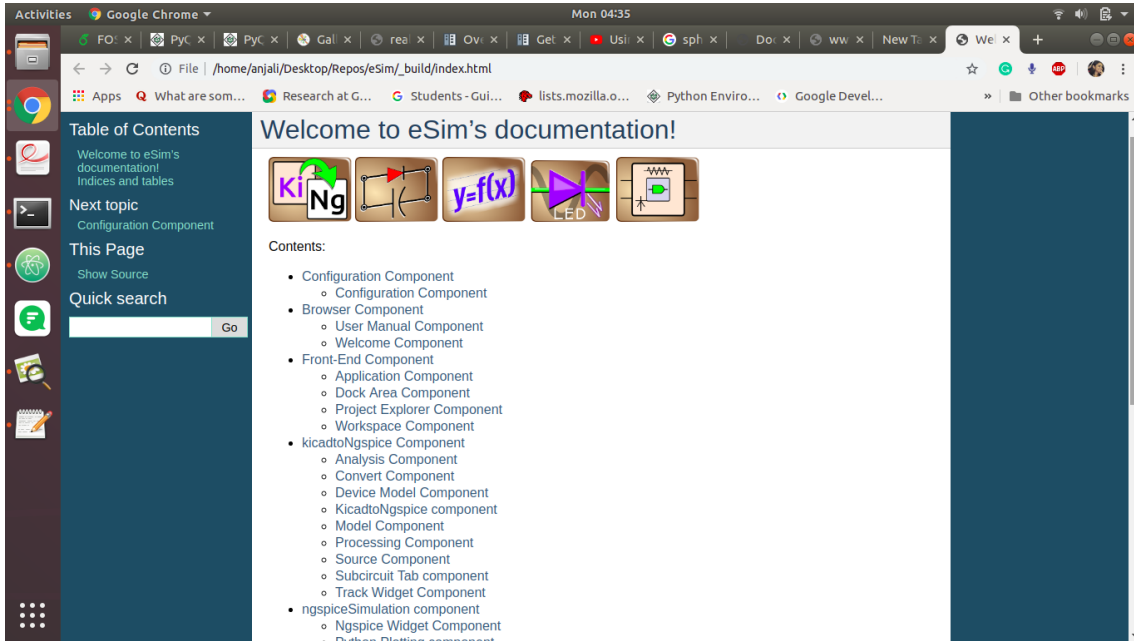


Figure 4.1: Index page

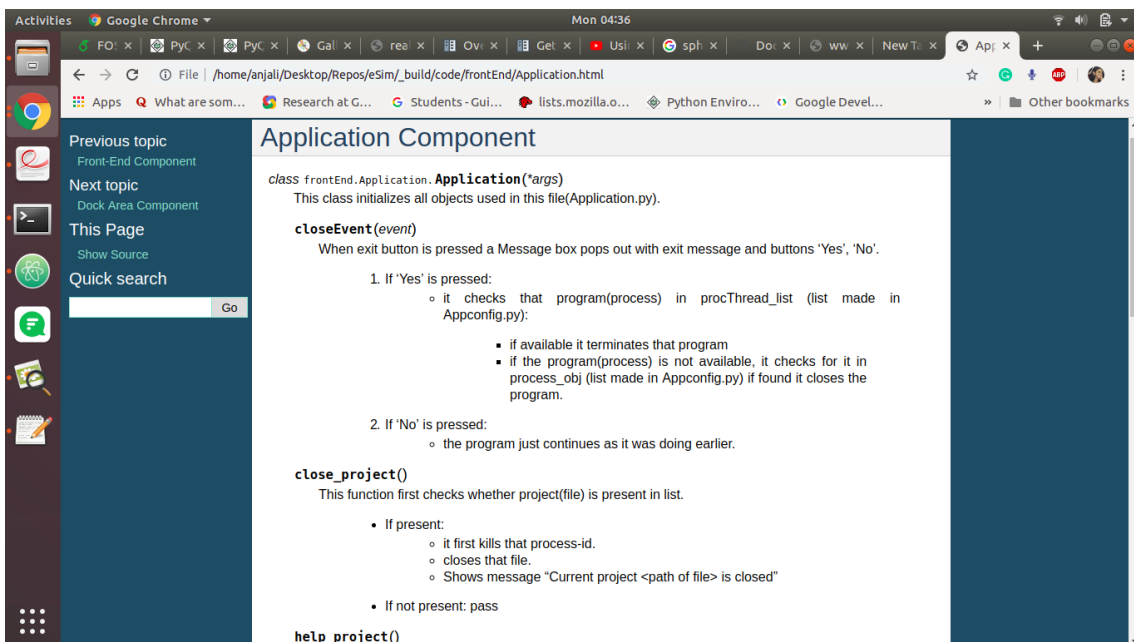


Figure 4.2: Autogenerated Documentation

## 4.2 Read The Docs

Read The Docs handles the building, versioning and even hosting your documentation online. Features-

- It can directly work from the github repo of your code, so if you push any new code, with further documentation, it will automatically build it for you  
*Need to activate webhook on github to allow Read The Docs access.*
- Also provides free hosting under `projName.readthedocs.io`
- Many other features such as redirection, versioning, etc.

Also can set advanced setting, to autogenerate documentation like in Sphinx, though this requires [packaging](#) your python application to provide a module structure for RTD(Read The Docs) to install virtually and document.

Hence, `setup.py` file was added so as to pack the python application and provide an overview to install it virtually and hence enable RTD to document it, the `setup.py` file can be found [here](#) Also can look at all the builds that took place, the exact commands and the bugs if the build failed.

**Github Issue-** [Add Sphinx documentation and host on Read The Docs](#)

The screenshot shows the Read the Docs interface for a project named 'esimTest'. At the top, there is a navigation bar with 'Read the Docs' on the left and a user profile 'nilshah98' on the right. Below the navigation bar, the project name 'esimTest' is displayed, along with a 'View Docs' button. A series of tabs are visible: 'Overview', 'Downloads', 'Search', 'Builds' (which is the active tab), 'Versions', and 'Admin'. The main content area shows details for 'Build #9273027', which was completed on June 25, 2019, at 7:45 a.m. and took 53 seconds. A green badge indicates 'Build completed'. Below this, there is a paragraph of text explaining how to configure documentation builds using a `.readthedocs.yml` file. At the bottom, a series of terminal commands are listed in a light gray box, showing the steps to clone the repository, checkout the sphinx branch, clean the directory, and run pip install commands in a virtual environment.

Read the Docs nilshah98

Projects > **esimTest** View Docs

Overview Downloads Search **Builds** Versions Admin

**Build #9273027** Completed June 25, 2019. 7:45 a.m.  
Build took 53 seconds  
[View docs](#)  
[View raw](#)

**Build completed**

Configure your documentation builds! Adding a `.readthedocs.yml` file to your project is the recommended way to configure your documentation builds. You can declare dependencies, set up submodules, and many other great features.

```
git clone --no-single-branch --depth 50 https://github.com/nilshah98/eSim.git .
git checkout --force origin/sphinx
git clean -d -f -f
python3.7 -mvirtualenv --no-site-packages --no-download /home/docs/checkouts/readthedocs.c
/home/docs/checkouts/readthedocs.org/user_builds/esimtest/envs/latest/bin/python -m pip ir
/home/docs/checkouts/readthedocs.org/user_builds/esimtest/envs/latest/bin/python -m pip ir
/home/docs/checkouts/readthedocs.org/user_builds/esimtest/envs/latest/bin/python -m pip ir
```

Figure 4.3: RTD Build Settings

The screenshot shows the Read the Docs (RTD) interface. At the top, there is a dark header with the text "Read the Docs" on the left and a user profile "nilshah98" on the right. Below the header, the page title is "Projects > esimTest". A green "View Docs" button is visible on the right. A navigation bar contains buttons for "Overview", "Downloads", "Search", "Builds", "Versions", and "Admin".

The main content area is titled "Advanced Settings" and is part of a "Settings" menu. The "Advanced Settings" option is selected. The settings are organized into a "Global settings" section, which includes:

- Default version\***: A dropdown menu set to "latest". Below it, a note states: "The version of your project that / redirects to".
- Default branch**: A dropdown menu set to "sphinx". Below it, a note states: "What branch 'latest' points to. Leave empty to use the default value for your VCS (eg. trunk or master)."
- Privacy Level\***: A dropdown menu set to "Public". Below it, a note states: "Level of privacy that you want on the repository. Protected means public but not in listings."

Figure 4.4: RTD Advanced Settings



# Chapter 5

## Bugs

### 5.1 Default Component Loading

- eSim has an added feature that loads default components, if a circuit has been loaded and simulated earlier
- The above is possible since it stores the previous values if a file labelled as ‘ModelName\_Previous\_Values.xml’ and also in ‘.json’ format
- Under models there are various default models placed and if not filled, left empty
- In this case there is **x1** but in different models it can go to **x12** or **q23** and be more than one digit
- Earlier code used to compare just the first letter and the first digit and load the wrong component

```
1 ...
2 models:{},
3 subcircuit:{
4     "x1":["/path/to/subcircuit/x1"],
5     "x12":["/path/to/subcircuit/x12"]
6 }
7 ...
```

Listing 5.1: Previous Values

Files updated - DeviceModel.py, SubcircuitTab.py

**Github Issue-**[Wrong default components are shown from previous\\_vals files](#)

## 5.2 TreeWidget UI

- Between porting from Python2 to 3, and fixing the pep8 issues, tree Widget was broken in the process.
- Broken and fixed screenshots are attached below
- Corresponding CSS was fixed at `ProjectExplorer.py` and the necessary files added
- This UI has an extra `vline` ie. vertical line as compared to the current eSim-1.1.2 UI, can be fixed if need be.

The added piece of code-

```
1     self.treewidget.setStyleSheet(" \
2         QTreeView::branch:has-siblings:!adjoins-item { \
3         border-image: url(../images/vline.png) 0;} \
4         QTreeView::branch:has-siblings:adjoins-item { \
5         border-image: url(../images/branch-more.png) 0; } \
6         QTreeView::branch:!has-children:!has-siblings:adjoins-item
7     { \
8         border-image: url(../images/branch-end.png) 0; } \
9         QTreeView::branch:has-children:!has-siblings:closed, \
10        QTreeView::branch:closed:has-children:has-siblings { \
11        border-image: none; \
12        image: url(../images/branch-closed.png); } \
13        QTreeView::branch:open:has-children:!has-siblings, \
14        QTreeView::branch:open:has-children:has-siblings { \
15        border-image: none; \
16        image: url(../images/branch-open.png); } \
    ")
```

Listing 5.2: Fixed CSS Tree-Widget

**Github Issue-**[ProjectExplorer treeWidget UI](#)

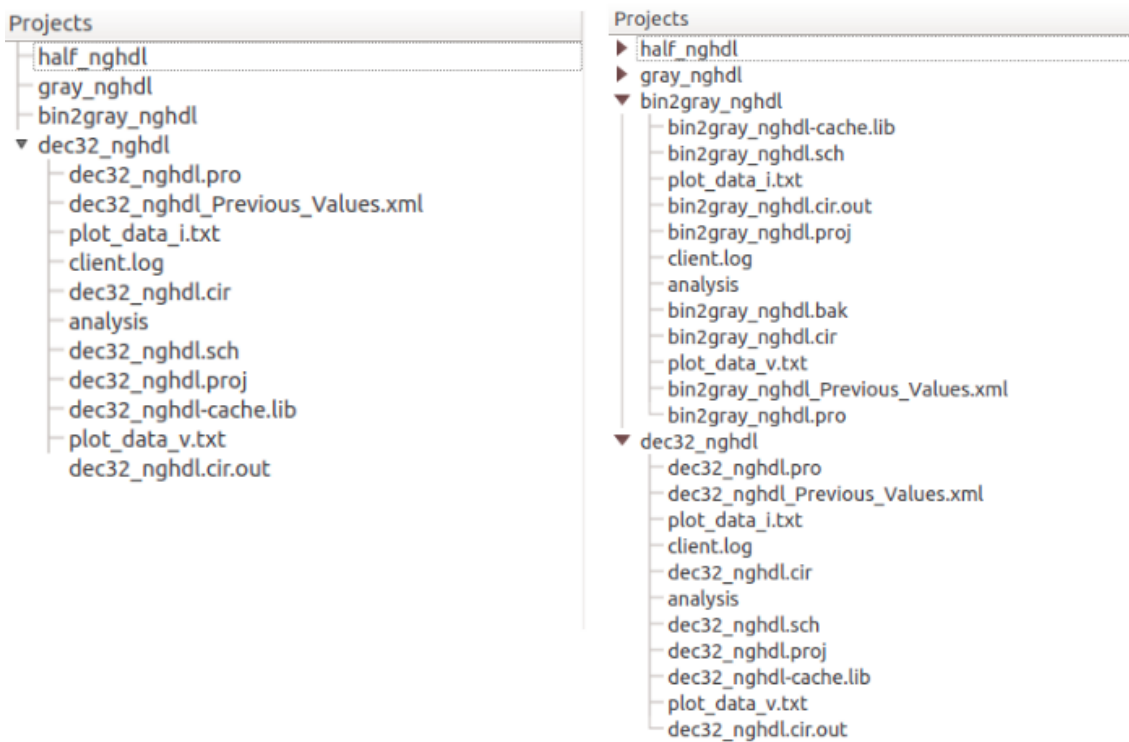


Figure 5.1: Tree Widget Bug and Fixed

## 5.3 Model Editor Bug

- Since Python3 Strings have unicode support and hence in PyQt 4.6+ for Python3, QString class is depreciated
- The same can be found [here](#)
- Hence, the issue here was that QString was being used to parse a string, hence would throw an error and the application would break down, and changes weren't written to file
- The same was fixed by using the 'str' data type in Python3

```
1 #Bugged
2 QtCore.QString("Parameters;Values").split(";")
3
4 #Fixed
5 ("Parameters;Values").split(";")
```

Listing 5.3: Fixed Code

**Github Issue-**[Edited values for models, don't reflect in files](#)

## 5.4 Python Plotting

- eSim generates a ‘.cir.out‘ file for ‘ngspice‘ to parse through and simulate
- At the end of the simulation, there’s a call to write data to ‘plot\_v.txt‘ and ‘plot\_i.txt‘ so that plotting can be achieved using ‘matplotlib‘ in Python as well
- Now the issue here is that, ‘ngspice‘ can take random amount of time to complete simulation of given circuit and it’s not guaranteed that data is written to ‘plot‘ files or not
- Earlier developers had added a kludge by sleeping the thread for 2 seconds, and if the simulation is still not finished, then python plotting won’t work
- The work around here can be to use [Synchronous Event API](#) provided by PyQt, but it requires the whole ngspice to be closed, so that event can be emitted, but we require python plots and ngspice plots side by side
- The work around we came to finally was [polling](#), we polled and checked whether new data was created. Polling frequency kept at a low 0.2 per second, so that the program doesn’t crash
- Added a fail safe as well

```
1 currTime = time.time()
2 count = 0
3 while True:
4     try:
5         st = os.stat(os.path.join(self.projDir, "plot_data_i.txt"))
6         if st.st_mtime >= currTime:
7             break
8
9     except Exception:
10        pass
11
12    time.sleep(0.2)
13
14 # Fail Safe ==>
15    count += 1
16    if count >= 100:
17
18        raise Exception("ngspice taking too long, check netlist file")
```

Listing 5.4: Fixed Code

[Github Issue-Python plotting issue](#)

## 5.5 Device Model Tab

- When porting from Python2 to Python3, `String` data type has an added attribute of `__iter__` which was not the case in Python2
- So, the earlier check used to parse through `NgSpice Model` parameters failed here, and it displayed input box for each letter as it iterated through it
- The fix was to check for type along with the attribute

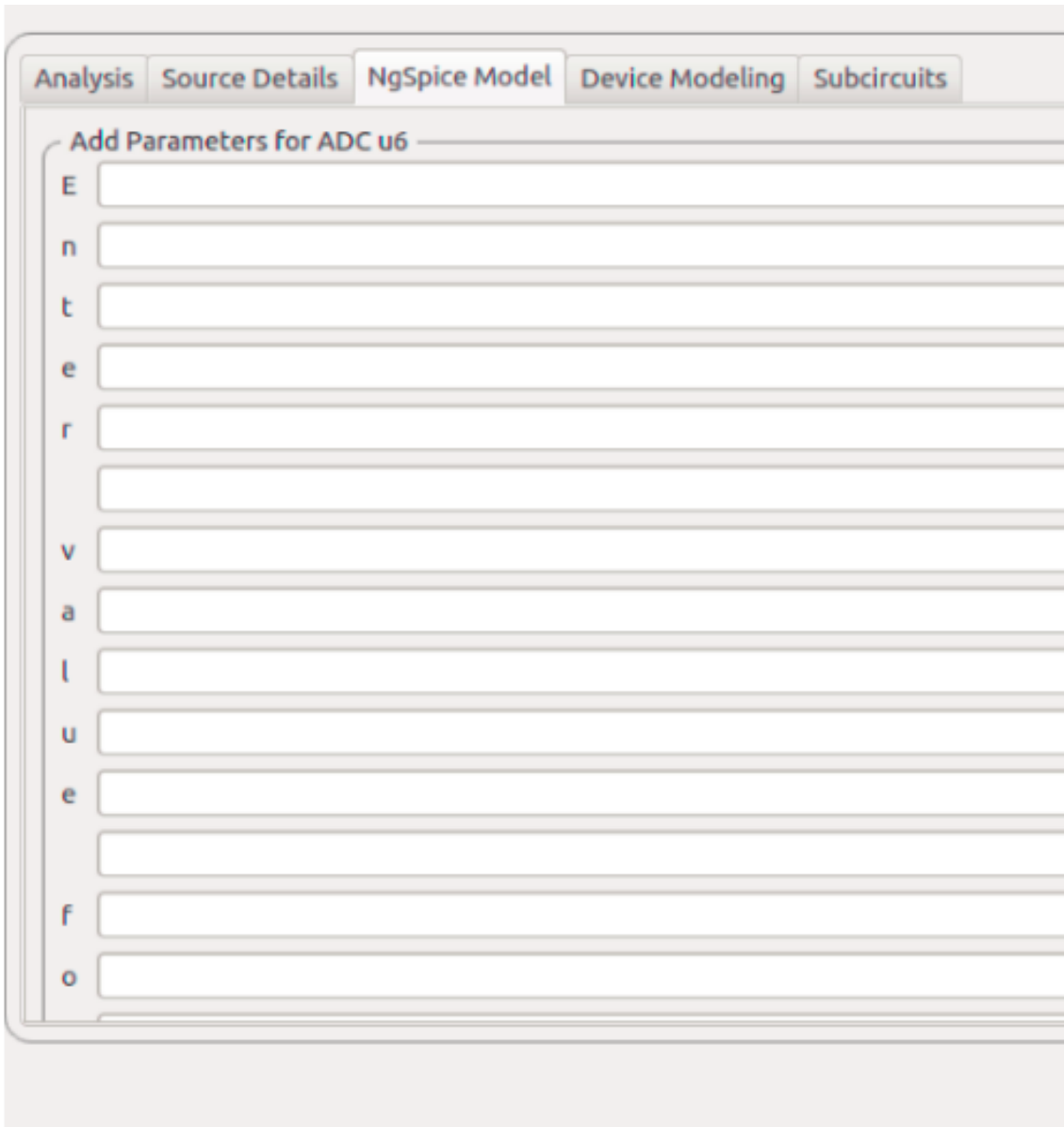


Figure 5.2: Device Model Bug

Fixed Code-

```
1 if type(value) is not str and hasattr(value, '__iter__'):
2 ...
```

**Github Issue-**[Fix NgSpice Model tab under kicadToNgspice](#)

# Chapter 6

## Additional Features

### 6.1 Upload Subcircuit

By the end of internship we added some feature in eSim one among them was uploading subcircuit option.

It has following steps:

- Click on subcircuit button in left-menu bar.
- Click on Upload Subcircuit option.
- Enter the name of subcircuit folder.
- Select the .sub file to copy in folder created.
- A folder with that file will be created inside Subcircuit folder, if the file has right format otherwise folder will not be generated inside Subcircuit folder.
- **Correct file format is:**
  - File should start with **.subckt filename**
  - File should end with **.end filename**

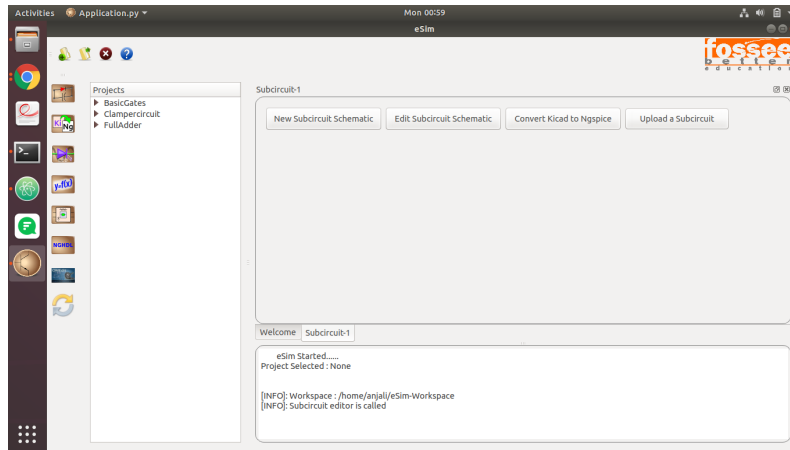


Figure 6.1: Upload Subcircuit

## 6.2 Rename Project

We also added renaming project option in project explorer area.

- For the projects those are inside the project explorer area, one can rename them but it has following conditions:
  - Name should not be same as earlier one.
  - Name should be unique, same project should not exist.
  - Name should not contain space between them.

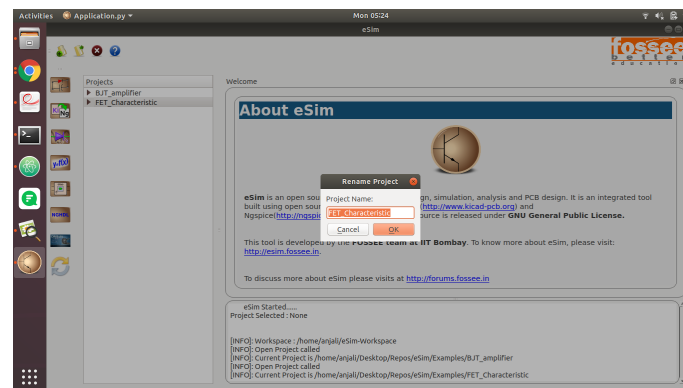


Figure 6.2: Rename Project

## 6.3 Logging

Currently there are print statements distributed all around the code to help debug the program in case of problems, which has made the terminal output all the more confusing, one solution to this is logging. Also, in case of an error from a user, which is not solvable via the **Fossee Forum**, they can mail their log files for the developers to have a look at. Logging opens up a world of whole new features such as-



- Logging to terminal or file
- Rotating, Modifying, Appending to existing logs
- Showcasing exact function call, from where, and on which line
- Set log output format
- Set date time

Exhaustive documentation for python logger can be found [here](#)

**NOTE:** This feature hasn't been merged into the Fossee eSim code yet, the issue and the corresponding feature can be found here- [Github Issue-Logging feature implemented](#)

## 6.4 Change lib format

Lib format generation was changed from multiple lines to single line to assist the user in easier handling of lib files.

Multi line format-

```
1 .MODEL PowerDiode D(  
2 + Vj=.75  
3 + Nbv1=14.976  
4 + Cjo=175p  
5 + Rs=.25  
6 + Isr=1.859n  
7 + Eg=1.11  
8 )
```

Singe line format-

```
1 .MODEL PowerDiode D( Vj=.75 Nbv1=14.976 Cjo=175p Rs=.25 Isr=1.859n Eg  
=1.11 )
```

ModelEditor.py generates these lib files and is modified.

**Github Issue-**[Change format of .lib files from multiple lines to single lines](#)

# Chapter 7

## Installer

The following approaches has been adopted for making installer for eSim:

### 7.1 Pyinstaller

PyInstaller bundles a Python application and all its dependencies into a single package. The user can run the packaged app without installing a Python interpreter or any modules. PyInstaller supports Python 2.7 and Python 3.4+, and correctly bundles the major Python packages such as numpy, PyQt, Django, wxPython, and others.

The main goal of PyInstaller is to be compatible with 3rd-party packages out-of-the-box. This means that, with PyInstaller, all the required tricks to make external packages work are already integrated within PyInstaller itself so that there is no user intervention required. You'll never be required to look for tricks in wikis and apply custom modification to your files or your setup scripts. As an example, libraries like PyQt, Django or matplotlib are fully supported, without having to handle plugins or external data files manually. The Supported Packages for pyinstaller can be viewed [here](#)

PyInstaller is a normal Python package. You can download the archive from PyPi, but it is easier to install using pip where it is available, for example:

```
1 pip install pyinstaller
```

PyInstaller reads a Python script written by you. It analyzes your code to discover every other module and library your script needs in order to execute. Then it collects copies of all those files – including the active Python interpreter! – and puts them with your script in a single folder, or optionally in a single executable file.

```
1 pyinstaller --onefile --windowed myscript.py
```

You distribute the bundle as a folder or file to other people, and they can execute your program. To your users, the app is self-contained. They do not need to install any particular version of Python or any modules. They do not need to have Python installed at all.

There are some command to make .exe file using pyinstaller-

```
1 pyinstaller myscript.py
```

For windowed mode (In this mode command prompt window will not open while running application. This is done when you don't want to show what actually is going on behind the scene when using the application) , we have a command-

```
1 pyinstaller --windowed myscript.py
```

There are some useful commands that can be used to show all file in one file mode or one directory mode in which all files and libraries will be added in one file and one directory respectively.

```
1 pyinstaller --onefile --windowed myscript.py
```

```
2 pyinstaller --onedir --windowed myscript.py
```

After giving any above commands Pyinstaller generates two folders (*build* and *dist*) and one file named *myscript.spec*. Our *.exe* file will be present inside dist folder. If you want to to some change in installer after making it then by editing *.spec* file it can done. And after making changes we have to run command:

```
1 pyinstaller myscript.spec
```

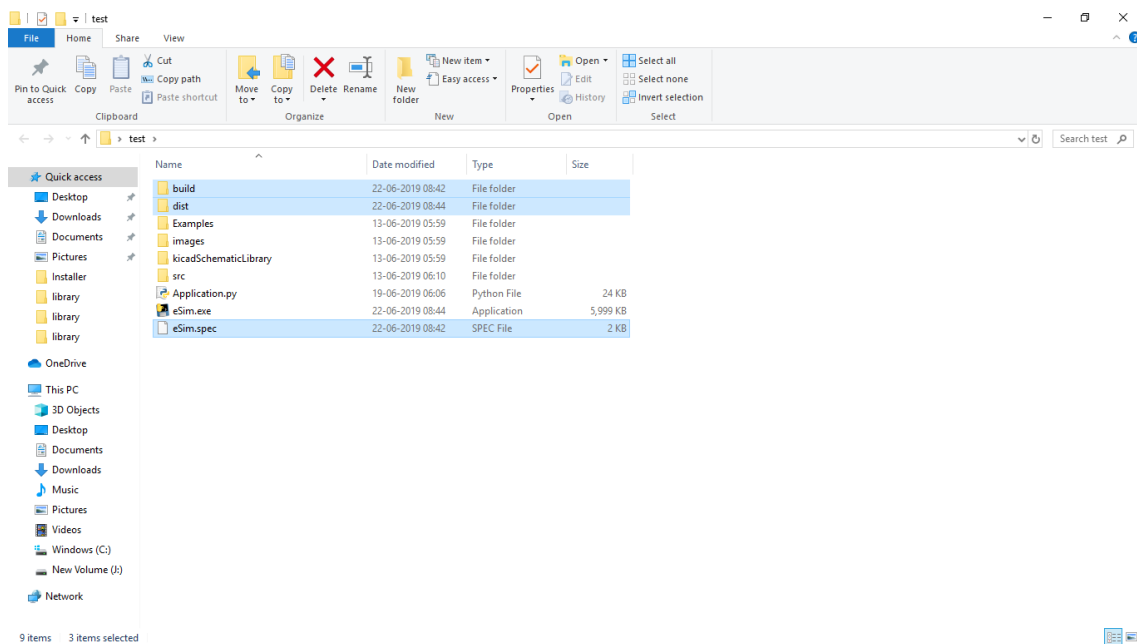


Figure 7.1: Screenshot showing build,dist and .spec file

There are many other useful commands which are available [here](#).

## Advantages of using Pyinstaller

- The most important advantage of pyinstaller is you don't need to install anything, just run the .exe file.
- No need to worry about packages.
- Its easy to use since there is no dealing with installation anymore.

The problems faced by us during making installer using Pyinstaller is listed below:

- For an application whose main file is inside some folder or directory like we have (Application.py), pyinstaller is making output file(.exe file) inside that folder only and if we are trying to make it outside then many functions of application doesn't work.
- The other problem faced is Pyinstaller does not support ngspice library which is used in our program and manually we are not able to give ngspice zip file since we don't know how whether Pyinstaller works and fetch zip files inside it's compressed folder.

The above two problems compelled us to look for other option for installer. So we made installer using NSIS Script which is discussed in next topic.

## 7.2 NSIS

NSIS (Nullsoft Scriptable Install System) is a professional open source system to create Windows installers. It is designed to be as small and flexible as possible and is therefore very suitable for internet distribution.

Being a user's first experience with your product, a stable and reliable installer is an important component of successful software. With NSIS you can create such installers that are capable of doing everything that is needed to setup your software.

NSIS is script-based and allows you to create the logic to handle even the most complex installation tasks. Many plug-ins and scripts are already available: you can create web installers, communicate with Windows and other software components, install or update shared components and more.

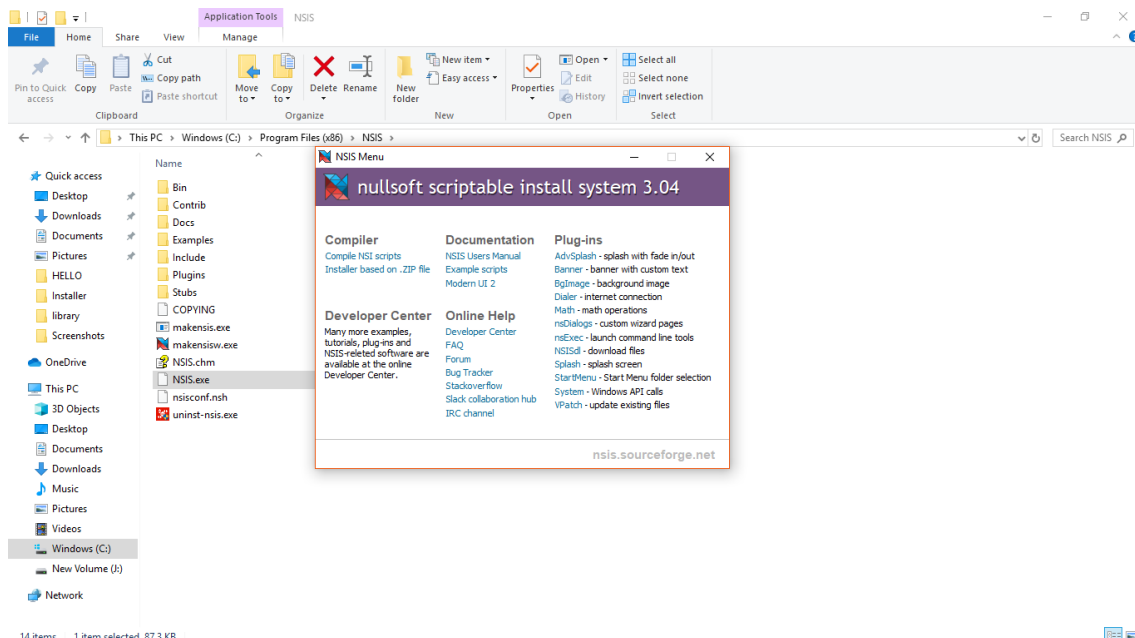
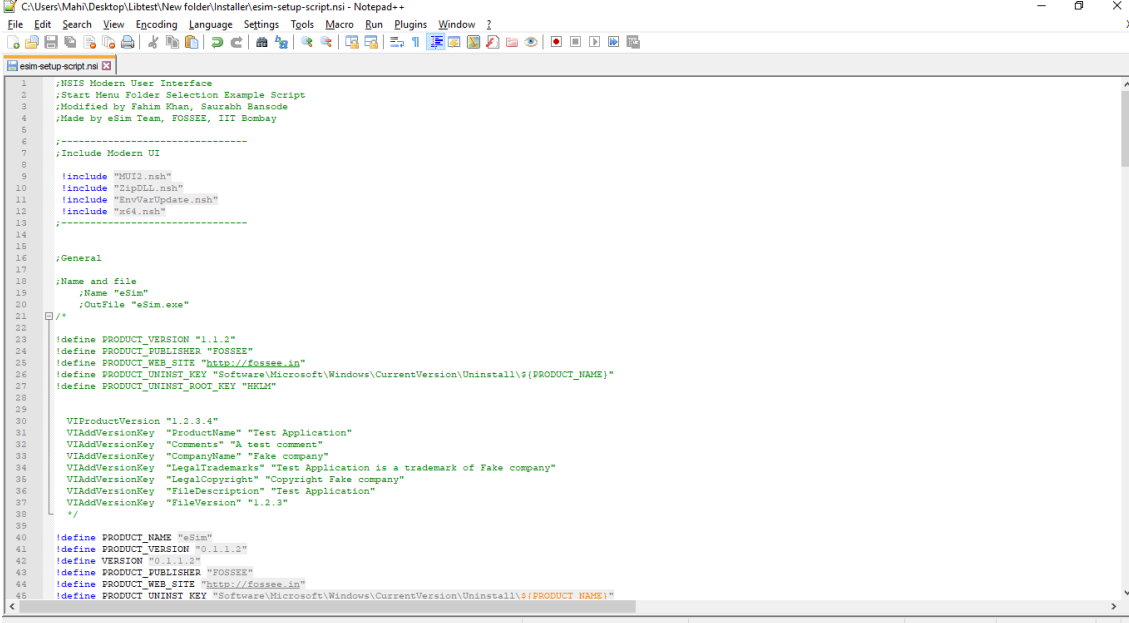


Figure 7.2: NSIS Installer

The NSIS uses a script name as NSIS Script(*.nsi*) which is the main script used for making installer. All the files, dependencies used in our installer is assigned here and user can modify according to his needs. The script is shown below:



```
1 ;NSIS Modern User Interface
2 ;Start Menu Folder Selection Example Script
3 ;Modified by Fahim Khan, Saurabh Bansode
4 ;Made by eSim Team, FOSSEE, IIT Bombay
5
6 ;-----
7 ;Include Modern UI
8
9 #include "MUI2.nsh"
10 #include "EipDLL.nsh"
11 #include "EnvVarUpdate.nsh"
12 #include "x64.nsh"
13 ;-----
14
15 ;General
16
17 ;Name and file
18 ;Name "eSim"
19 ;OutFile "eSim.exe"
20
21 /*
22
23 #define PRODUCT_VERSION "1.1.2"
24 #define PRODUCT_PUBLISHER "FOSSEE"
25 #define PRODUCT_WEB_SITE "http://fossee.in"
26 #define PRODUCT_UNINST_KEY "Software\Microsoft\Windows\CurrentVersion\Uninstall\{PRODUCT_NAME}"
27 #define PRODUCT_UNINST_ROOT_KEY "HKLM"
28
29
30 VIProductVersion "1.2.3.4"
31 VIAddVersionKey "ProductName" "Test Application"
32 VIAddVersionKey "Comments" "A test comment"
33 VIAddVersionKey "CompanyName" "Fake company"
34 VIAddVersionKey "LegalTrademarks" "Test Application is a trademark of Fake company"
35 VIAddVersionKey "LegalCopyright" "Copyright Fake company"
36 VIAddVersionKey "FileDescription" "Test Application"
37 VIAddVersionKey "FileVersion" "1.2.3"
38 */
39
40 #define PRODUCT_NAME "eSim"
41 #define PRODUCT_VERSION "0.1.1.2"
42 #define VERSION "0.1.1.2"
43 #define PRODUCT_PUBLISHER "FOSSEE"
44 #define PRODUCT_WEB_SITE "http://fossee.in"
45 #define PRODUCT_UNINST_KEY "Software\Microsoft\Windows\CurrentVersion\Uninstall\{PRODUCT_NAME}"
```

Figure 7.3: NSIS-Script for making Installer

For more information related to NSIS Scripts, you can visit this [link](#)

### **Features Added**

The following new features have been added with this installer:

- Added Python3 version code.
- Added dependencies required for Python3 version.
  - Python 3.3
  - PyQt4
  - Matplotlib
  - Pyparsing
  - numpy
  - dateutil
  - NgSpice
  - Kicad version 4.0.7
- Added License page in installer.
- Added Uninstaller

## **7.3 Uninstaller**

Uninstaller section is added with our installer file that will remove all the dependencies installed by installer during the time of installation. It will be automatically created inside the folder where user selects to install eSim. Below is the screenshot of uninstaller created.

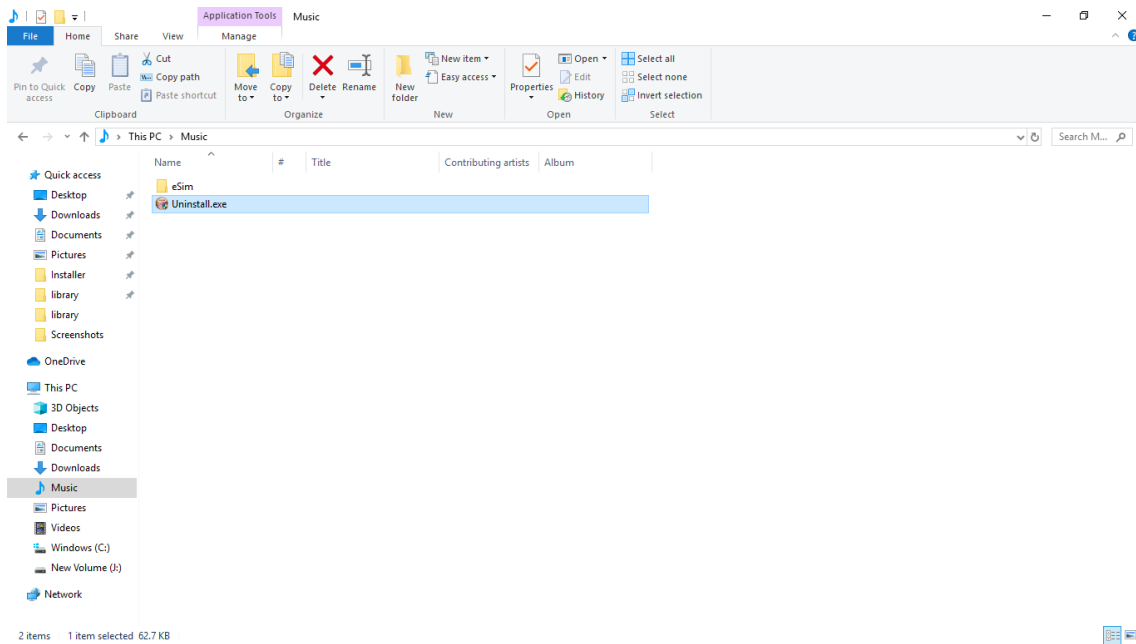


Figure 7.4: Uninstaller

### Bugs Removal

Two bugs came into notice which were not present in old installer but during porting of our code and making new version of installer they somehow came which are listed below:

- eSim kicad library is not getting called in eeschema.
- One bug was present which show error as shown in below figure. It was due to some lines were missing in the eSim\_Pspice.lib and eSim\_subckt.lib inside kicadSchematicLibrary folder.

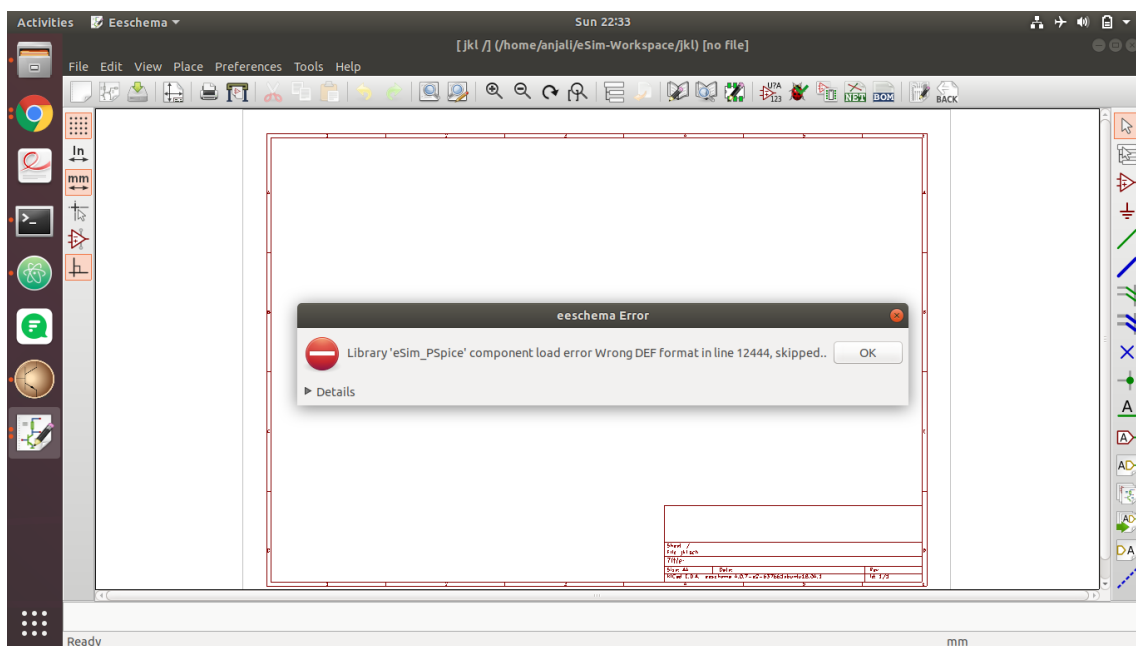
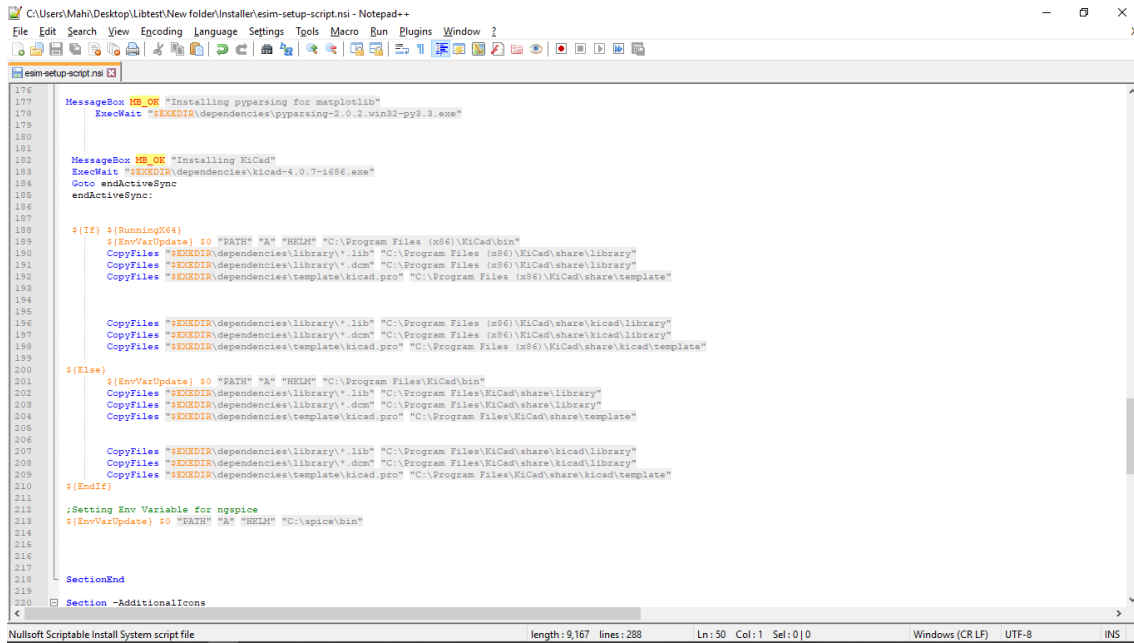


Figure 7.5: eSim PSpice error



The procedure to remove above bugs is described below:

- The eSim library issue is resolved by writing script to copy library inside kicad folder which is shown below:



```
176
177     MessageBox MB_OK "Installing pyParsing for matplotlib"
178     ExecWait "$EXEDIR\dependencies\pyParsing-2.0.2.win32-py3.3.exe"
179
180
181     MessageBox MB_OK "Installing KiCad"
182     ExecWait "$EXEDIR\dependencies\kicad-4.0.7-1696.exe"
183
184     Goto endActiveSync
185     endActiveSync:
186
187
188     ;[If] ;[RunningX64]
189     ;[EnvVarUpdate] $0 "PATH" "%*" "%HKLM%" "C:\Program Files (x86)\KiCad\bin"
190     CopyFiles "$EXEDIR\dependencies\library*.lib" "C:\Program Files (x86)\KiCad\share\library"
191     CopyFiles "$EXEDIR\dependencies\library*.dcm" "C:\Program Files (x86)\KiCad\share\kicad\library"
192     CopyFiles "$EXEDIR\dependencies\template\kicad.pro" "C:\Program Files (x86)\KiCad\share\template"
193
194
195     CopyFiles "$EXEDIR\dependencies\library*.lib" "C:\Program Files (x86)\KiCad\share\kicad\library"
196     CopyFiles "$EXEDIR\dependencies\library*.dcm" "C:\Program Files (x86)\KiCad\share\kicad\library"
197     CopyFiles "$EXEDIR\dependencies\template\kicad.pro" "C:\Program Files (x86)\KiCad\share\kicad\template"
198
199
200     ;[Else]
201     ;[EnvVarUpdate] $0 "PATH" "%*" "%HKLM%" "C:\Program Files\KiCad\bin"
202     CopyFiles "$EXEDIR\dependencies\library*.lib" "C:\Program Files\KiCad\share\library"
203     CopyFiles "$EXEDIR\dependencies\library*.dcm" "C:\Program Files\KiCad\share\kicad\library"
204     CopyFiles "$EXEDIR\dependencies\template\kicad.pro" "C:\Program Files\KiCad\share\template"
205
206
207     CopyFiles "$EXEDIR\dependencies\library*.lib" "C:\Program Files\KiCad\share\kicad\library"
208     CopyFiles "$EXEDIR\dependencies\library*.dcm" "C:\Program Files\KiCad\share\kicad\library"
209     CopyFiles "$EXEDIR\dependencies\template\kicad.pro" "C:\Program Files\KiCad\share\kicad\template"
210
211     ;[EndIf]
212
213     ;Setting Env Variable for ngspice
214     ;[EnvVarUpdate] $0 "%HKLM%" "%*" "%HKLM%" "C:\spice\bin"
215
216
217
218     SectionEnd
219
220     Section -AdditionalIcons
```

Figure 7.6: Library issue resolving code

- The *eSim PSpice* error is removed by updating the file inside kicadSchematicLibrary folder. Github link is [here](#)

## Virus Checking

The installer made by above script is tested against virus on [online platform](#) which is free and open platform for checking virus in any application or website. For more information visit the website given above. When we have tested our installer on it we got six communities detected virus. When we have worked on it more the number reduced to two as shown in below figure. After verifying it for many times we came to the conclusion that these two are false flags generated by communities. Among the two one false flag is of the NSIS software itself and other one is asking for digital signature that verifies that our software is trusted which is generally used by companies. So both the flags are **False flags**, we can ignore them because we have tested it on Quick heal antivirus which shows there is no virus in installer.

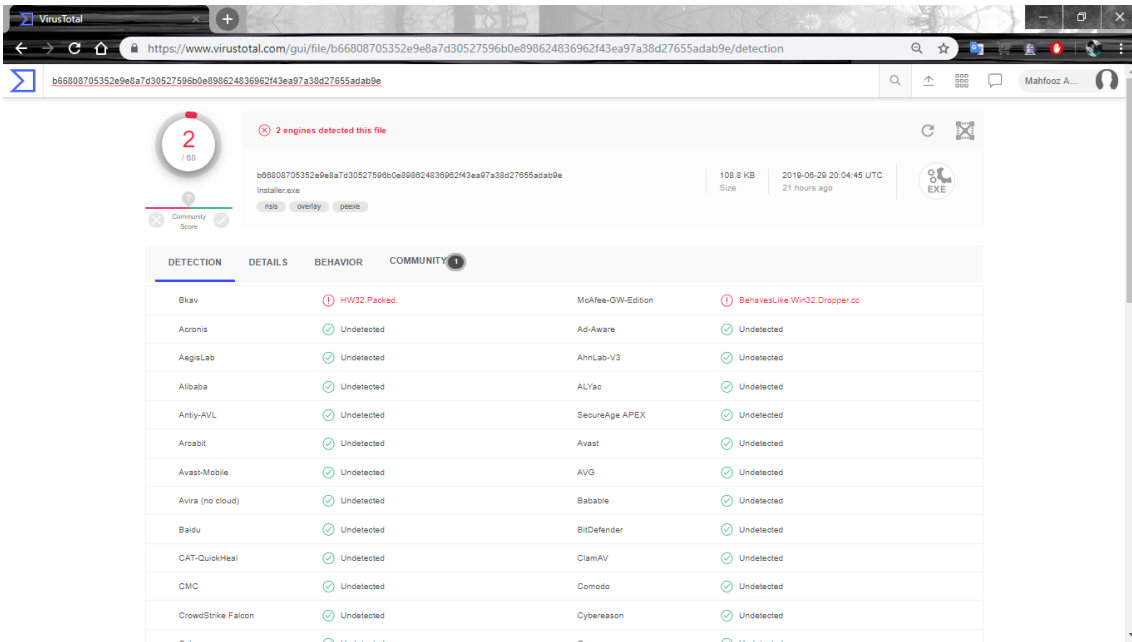


Figure 7.7: Communities showing virus

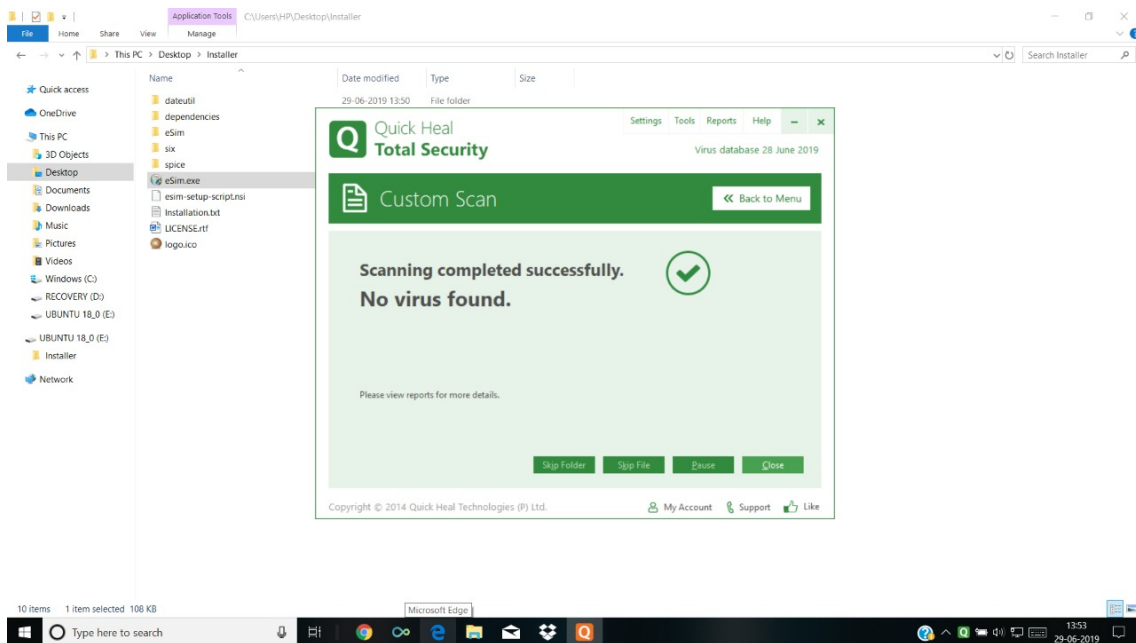


Figure 7.8: Quick Heal Scan

The above screenshot clears that there is no virus in our application. So we have to take those two flags as **False flags**.

## Summary

<b>Week no.</b>	<b>Task accomplished</b>
Week 1	Understanding the project
Week 2	Pep8 and Documentation
Week 3	Documentation
Week 4	Sphinx Format Documentation
Week 5	Additional features
Week 6	Report making and presentation

Table 7.1: Task accomplished by Anjali

<b>Week no.</b>	<b>Task accomplished</b>
Week 1	Understanding the project
Week 2	Pep8 and Documentation
Week 3	Installer (Pyinstaller)
Week 4	Installer (NSIS)
Week 5	Installer (NSIS)
Week 6	Report, Virus removal and bugs removal

Table 7.2: Task accomplished by Mahfooz

<b>Week no.</b>	<b>Task accomplished</b>
Week 1	Understanding the project
Week 2	Pep8 and Documentation
Week 3	Debugging and Read the docs hosting
Week 4	Debugging and logger addition
Week 5 & 6	NGHDL

Table 7.3: Task accomplished by Neel

### **Bugs fixed**

- Default component loading
- Treewidget UI
- Model Editor issue
- Python plotting issue
- Device model view

### **Additional Features**

- Renaming Projects
- Uploading Subcircuit
- Change format of lib files generated for ngspice
- Logger added

Listed below are the links to pull requests and commits-

- **Anjali:**

- <https://github.com/FOSSEE/eSim/pull/104>
- <https://github.com/FOSSEE/eSim/pull/107>
- <https://github.com/FOSSEE/eSim/pull/110>
- <https://github.com/FOSSEE/eSim/pull/88>
- <https://github.com/FOSSEE/eSim/commit/23c03bea24b2ad59b8abc1394c39985928373598>
- <https://github.com/FOSSEE/eSim/commit/880f6eaa02416aacbc732728a91c7025841ea2e5>
- <https://github.com/FOSSEE/eSim/commit/d620109c1255aa4af35aa8f86f0e147b709d9b9c>
- <https://github.com/FOSSEE/eSim/commit/11796de3a3bbb0568d44c18ba6ff3d5e14b7407f>

- **Mahfooz:**

- <https://github.com/FOSSEE/eSim/commit/aa20d92a95166a82f93a19594ede632de0666c8b>
- <https://github.com/FOSSEE/eSim/commit/48196556e90f5566e8d42088f596a5fc7ae8e918>
- <https://github.com/FOSSEE/eSim/commit/90d656da8cc6dfb076fa2b8ec9764e9bf391bda7>
- <https://github.com/FOSSEE/eSim/commit/32132ebfcc90e993acc3a87c4017ef0b00bcd890>
- <https://github.com/FOSSEE/eSim/commit/28d8fd378bf717e8daba8a564708856769f24b98>

- **Neel:**

- <https://github.com/FOSSEE/eSim/pull/109>
- <https://github.com/FOSSEE/eSim/pull/101>
- <https://github.com/FOSSEE/eSim/pull/96>
- <https://github.com/FOSSEE/eSim/pull/95>
- <https://github.com/FOSSEE/eSim/pull/92>
- <https://github.com/FOSSEE/eSim/pull/86>
- <https://github.com/FOSSEE/eSim/pull/85>
- <https://github.com/FOSSEE/eSim/pull/77>

## **Conclusion**

By the end of the internship we learned many new things and got to know more about the things we already did, best example is git, we were so compact in world of git and happy with push, pull, commit. After working on eSim we learned many new commands like rebase, cherry-pick, etc which are very useful in future works. We learned how to co-ordinate in a team and also outside team, how to meet the requirements of user while adding any feature or removing any bug. How to work under given deadline and value of discipline. We also learned about various types of license used for open-source softwares. We learned about making installer for software and faced lot of problem with it, but finally got our way out of it, with continuous efforts and support of mentors.

## Reference

- [https://www.tutorialspoint.com/pyqt/pyqt\\_introduction.htm](https://www.tutorialspoint.com/pyqt/pyqt_introduction.htm)
- <https://www.python.org/dev/peps/pep-0008/#introduction>
- <http://www.sphinx-doc.org/en/master/>
- [https://nsis.sourceforge.io/Main\\_Page](https://nsis.sourceforge.io/Main_Page)
- <https://www.pyinstaller.org/>
- <https://www.virustotal.com/gui/home/upload>