



# Summer Fellowship Report

## On

Integrate Pspice-kicad and LTspice-kicad converters in eSim

## Submitted by

Sangavi GR
MSc. Software Systems
PSG College of Technology Coimbatore

## Under the guidance of

Prof. Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

**Date:** December 11, 2023

# Acknowledgment

I want to express my heartfelt gratitude to those who have played a crucial role in making my FOSSEE fellowship experience truly memorable. Prof. Kannan M. Moudgalya, your mentorship has been a guiding light, fostering in me a sense of purpose and a continuous curiosity for learning. I genuinely appreciate your unwavering support and motivation, which have been instrumental in shaping my perspective.

To mentors Mr. Sumanto Kar and Mr. Rahul Paknikar, I want to extend my sincere thanks for your dedication in providing insightful guidance and a clear path during challenging times. The supportive and collaborative atmosphere you've cultivated has created a space conducive to growth and exploration.

A special thanks to Project manager Mrs. Usha Vishwanathan for their collaborative spirit, valuable contributions, and unique insights that have brought depth to my fellowship journey. Your openness to share expertise has served as a catalyst for overcoming obstacles and reaching milestones.

This appreciation extends to the entire eSim team and fellow friends, whose collective knowledge have made the fellowship environment not only productive but also enjoyable. Looking ahead, I am excited about applying the wealth of knowledge gained during this fellowship to make meaningful contributions to both my personal and professional pursuits. This experience has not only enhanced my skill set but has also instilled in me a sense of responsibility to contribute positively to our society.

# Contents

# Chapter 1

# Introduction

FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools in academia and research. The FOSSEE project is part of the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India. ESim is one of the project promoted by FOSSEE.

eSim, previously recognized as Oscad/FreeEDA, represents a cutting-edge and open-source Electronic Design Automation (EDA) tool tailored for circuit design, simulation, analysis, and PCB design. This comprehensive platform seamlessly integrates various free and open-source software, including KiCad, Ngspice, GHDL, OpenModelica, Verilator, Makerchip, and SkyWater SKY130 PDK, all released under the GNU General Public License.

What sets eSim apart is its ability to provide functionalities and user-friendly features comparable to proprietary software for tasks like schematic creation, simulation, and PCB designall without the substantial costs associated with licensing. This makes eSim a cost-effective alternative, particularly beneficial for educational institutions and small to medium enterprises (SMEs). Serving as a practical substitute for widely-used commercial tools such as OrCAD, PSpice, LTspice, Xpedition, and HSPICE, eSim offers an affordable yet powerful solution for those involved in circuit design and simulation.

# Chapter 2

# Problem Statement

Integrate Pspice-kicad and LTspice-kicad converters in eSim

## 2.1 KiCad

KiCad is a free and open-source electronics design automation (EDA) suite with features like schematic capture, circuit simulation, PCB layout,etc. KiCad's user-friendly interface and continuous updates make it a powerful and accessible tool for electronic design.

## 2.2 PSpice

PSpice serves as a comprehensive virtual SPICE simulation environment, featuring an extensive model library. PSPICE libraries are .slb files and schematics are .sch files.

## 2.3 LTspice

LTspice is a powerful circuit simulation program enabling users to create, explore, and evaluate the performance of circuit designs. It serves as a versatile tool for drafting, probing, and analyzing circuit behavior, facilitating an efficient and effective simulation environment. LTSPICE libraries are .asy files and schematics are .asc files.

## 2.4 Objective

The user can browse Pspice or LTspice schematic and convert it into KiCad schematic also can convert Pspice or LTspice libray into KiCad library.

## 2.5 Advantages

Seamless conversion between PSpice or LTspice and KiCad promotes interoperability, allowing users to leverage the strengths of each tool within a unified environment. Users can benefit from an efficient workflow by seamlessly transitioning between PSpice, LTspice, and KiCad, reducing the need for manual adjustments and enhancing productivity. The integration minimizes redundant efforts by enabling users to directly utilize existing PSpice or LTspice designs and libraries in KiCad, optimizing resource utilization.

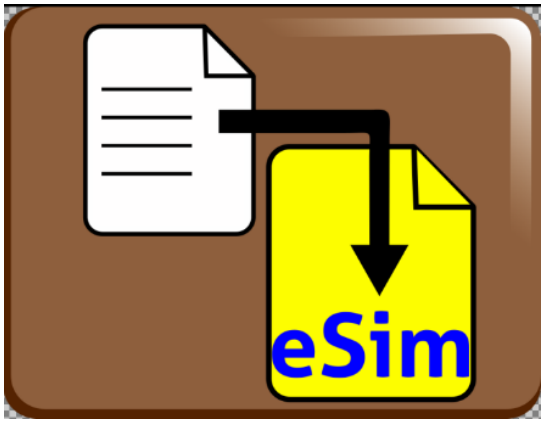
# Chapter 3

# Schematics Converter

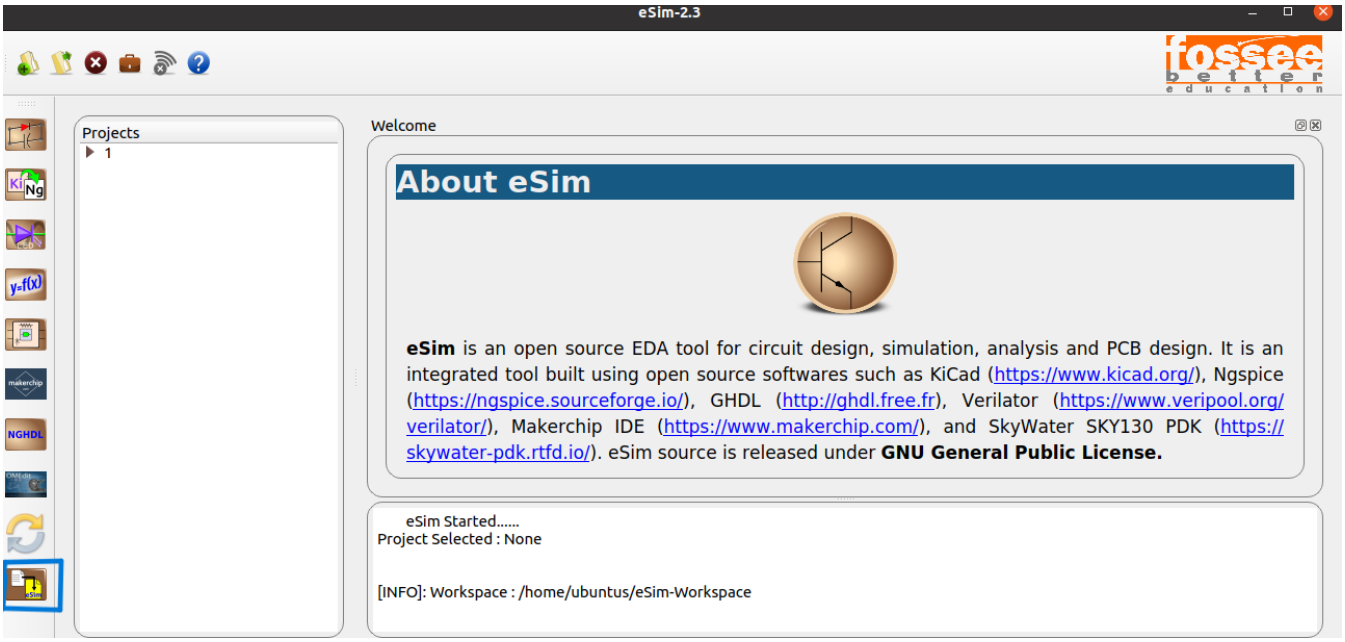
An icon is added to the left toolbar of the eSim application when clicked opens the schematic converter in the dockarea.

## 3.1 Tool Bar

Schematic converter icon is added to the left tool bar of eSim application.



### 3.1.1 Image




### 3.1.2 Code

```
def initToolBar(self):
    self.conToeSim = QtWidgets.QAction(
        QtGui.QIcon(init_path + 'images/icon.png'),
        '<b>Schematics converter</b>', self )
    self.conToeSim.triggered.connect(self.open_conToeSim)

    # Adding Action Widget to tool bar
    self.lefttoolbar.addAction(self.conToeSim)
```

## 3.2 Dock Area

When the schematic converter icon is clicked it opens the converter in dockarea.

**CODE**

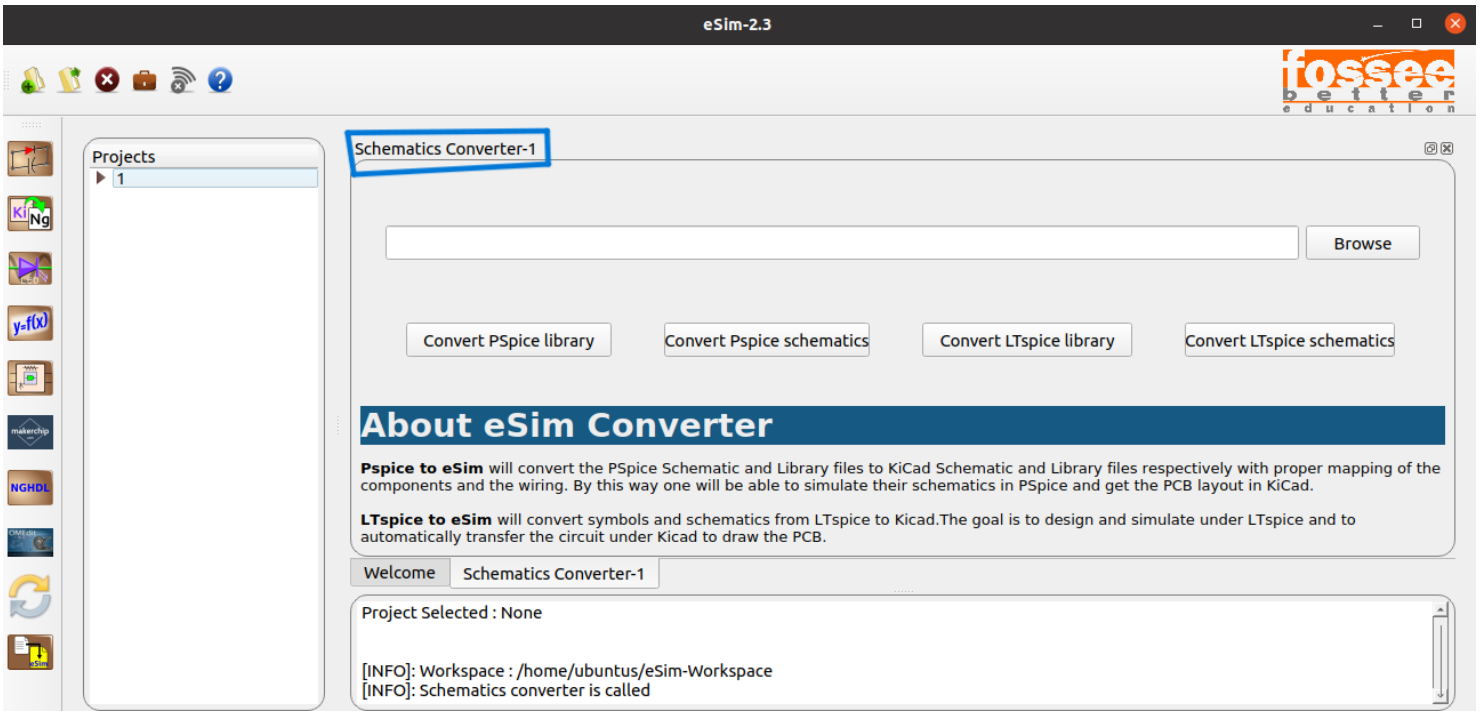
```
def eSimConverter(self):
        """This function creates a widget for eSimConverter."""
        global count

        dockName = 'Schematics Converter-'

        self.eConWidget = QtWidgets.QWidget()
        self.eConLayout = QVBoxLayout()  # QVBoxLayout for the main layout

        file_path_layout = QHBoxLayout()  # QHBoxLayout for file path line
        lib_path_layout = QHBoxLayout()

        file_path_text_box = QLineEdit()
        file_path_text_box.setFixedHeight(30)
        file_path_text_box.setFixedWidth(800)
```

```
file_path_layout.setAlignment(Qt.AlignCenter)
file_path_layout.addWidget(file_path_text_box)

browse_button = QPushButton("Browse")
browse_button.setFixedSize(100, 30)
browse_button.clicked.connect(lambda: browse_path(self,file_path_text_box))
file_path_layout.addWidget(browse_button)

self.eConLayout.addLayout(file_path_layout)  # Add file path layout to main layout

button_layout = QHBoxLayout()  # QHBoxLayout for the buttons

self.pspice_converter = PspiceConverter(self)
self.ltspice_converter = LTspiceConverter(self)
self.pspiceLib_converter = PspiceLibConverter(self)
self.ltspiceLib_converter = LTspiceLibConverter(self)

upload_button2 = QPushButton("Convert PSpice library")
upload_button2.setFixedSize(180, 30)
upload_button2.clicked.connect(lambda: self.pspiceLib_converter.upload_file_Pspice
(file_path_text_box.text()))
button_layout.addWidget(upload_button2)

upload_button1 = QPushButton("Convert Pspice schematics")
upload_button1.setFixedSize(180, 30)
upload_button1.clicked.connect(lambda: self.pspice_converter.upload_file_Pspice
(file_path_text_box.text()))
button_layout.addWidget(upload_button1)

upload_button3 = QPushButton("Convert LTspice library")
upload_button3.setFixedSize(184, 30)
upload_button3.clicked.connect(
lambda: self.ltspiceLib_converter.upload_file_LTspice
(file_path_text_box.text()))
button_layout.addWidget(upload_button3)

upload_button = QPushButton("Convert LTspice schematics")
upload_button.setFixedSize(184, 30)
upload_button.clicked.connect(
lambda: self.ltspice_converter.upload_file_LTspice
(file_path_text_box.text()))
button_layout.addWidget(upload_button)

self.eConLayout.addLayout(button_layout)

self.eConWidget.setLayout(self.eConLayout)

# Add the description HTML content
description_html = """
    <html>
```

```
        <head>
            <style>
                body {
                    font-family: sans-serif;
                    margin: 0px;
                    padding: 0px;
                    background-color: white;
                    border: 4px solid  black;
                    font-size: 10pt; /* Adjust the font size as needed */
                }

                h1{
                    font-weight: bold;
                    font-size: 9pt;
                    color: #eeeeee;
                    padding: 10px;
                    background-color: #165982;
                    border: 4px outset  #0E324B;
                }
            </style>
        </head>

        <body>
            <h1>About eSim Converter</h1>
            <p>
                <b>Pspice to eSim </b> will convert the PSpice Schematic and
                Library files to KiCad Schematic and
                Library files respectively with proper mapping
                of the components and the wiring. By this way one
                will be able to simulate their schematics
                in PSpice and get the PCB layout in KiCad.</b>
                <br/><br/>
                <b>LTspice to eSim </b> will convert symbols
                and schematics from LTspice to Kicad.The goal is to design and
                simulate under LTspice and to automatically
                transfer the circuit under Kicad to draw the PCB.</b>
            </p>
        </body>
    </html>
"""

self.description_label = QLabel()
self.description_label.setFixedHeight(160)
self.description_label.setFixedWidth(950)
self.description_label.setAlignment(Qt.AlignBottom)
self.description_label.setWordWrap(True)
self.description_label.setText(description_html)
# Add the description label to the layout
self.eConLayout.addWidget(self.description_label)
```

```
        self.eConWidget.setLayout(self.eConLayout)

        dock[dockName + str(count)] = QtWidgets.QDockWidget(dockName + str(count))
        dock[dockName + str(count)].setWidget(self.eConWidget)
        self.addDockWidget(QtCore.Qt.TopDockWidgetArea, dock[dockName + str(count)])
        self.tabifyDockWidget(dock['Welcome'], dock[dockName + str(count)])

        # CSS
        dock[dockName + str(count)].setStyleSheet(" \
            .QWidget { border-radius: 15px; border: 1px solid gray;\
                padding: 5px; width: 200px; height: 150px;  } \
            ")

        dock[dockName + str(count)].setVisible(True)
        dock[dockName + str(count)].setFocus()
        dock[dockName + str(count)].raise_()

        count = count + 1
```

### 3.2.1 Image




### 3.2.2 Code

```
def open_conToeSim(self):
        print("Function : Schematics converter")
        self.obj_appconfig.print_info('Schematics converter is called')
        self.obj_Mainview.obj_dockarea.eSimConverter()
```

## 3.3 Components

There is a browse button, text box, description of converter and four buttons to convert Pspice or LTspice schematic or library into KiCad.

### 3.3.1 Browse And Text box

This functionality employs PyQt5 to create a file dialog that enables users to choose files of any type. The dialog is configured to filter and recognize files with extensions like *.sch, *.asc, *.slb, and *.asy. Upon execution, the dialog prompts users to select a file, and the chosen file's path is captured. If a file is selected, its path is then set as the text content of a designated text box. In essence, this implementation provides an intuitive file selection feature, enhancing user interaction within a PyQt5 application.

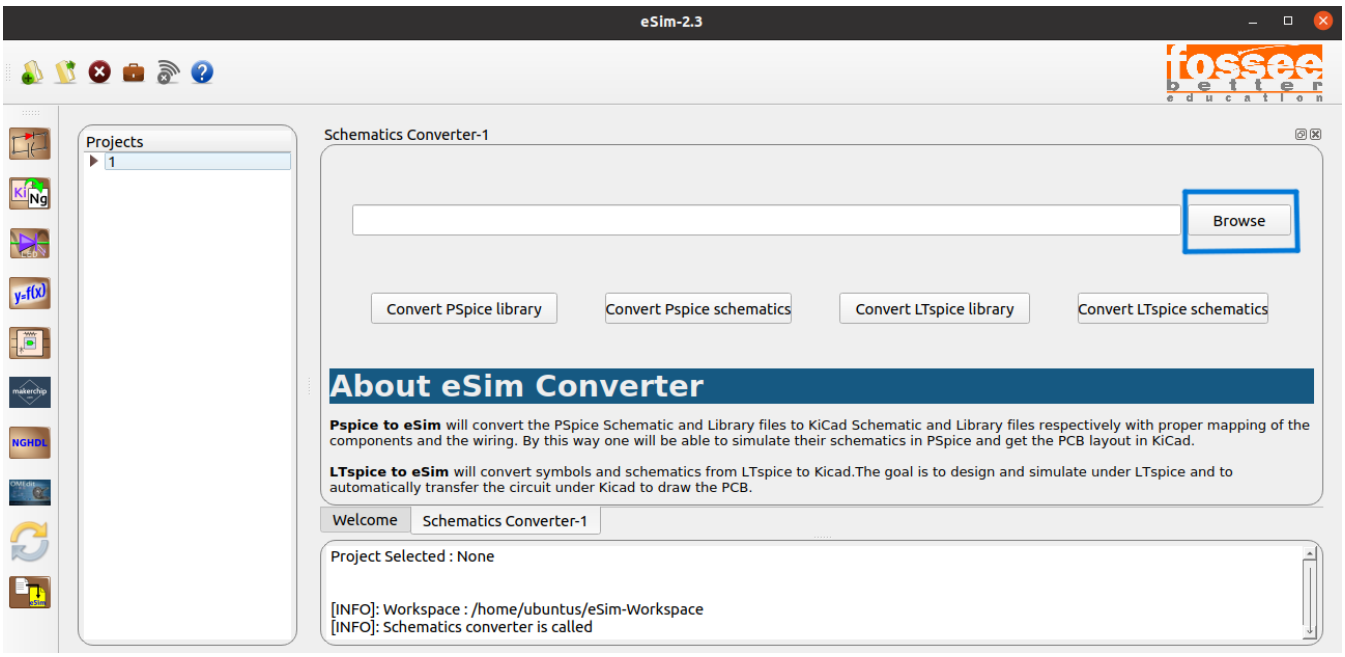
#### Input Output

**Input:**
**Browse Button:** Allows the user to navigate the file system and select a file (schematic file in this case) for conversion. **Text Box:** Displays the path or name of the selected file, providing visual confirmation to the user.

**Output:**
**Selected File Path:** The path of the chosen schematic file is captured from the browse button. **Displayed File Name:** The text box shows the name or path of the selected file, aiding user awareness.

In summary, the browse button and text box work in tandem, allowing users to choose a schematic file through the browse button, with the selected file's path or name then displayed in the text box.

#### Image




#### Code

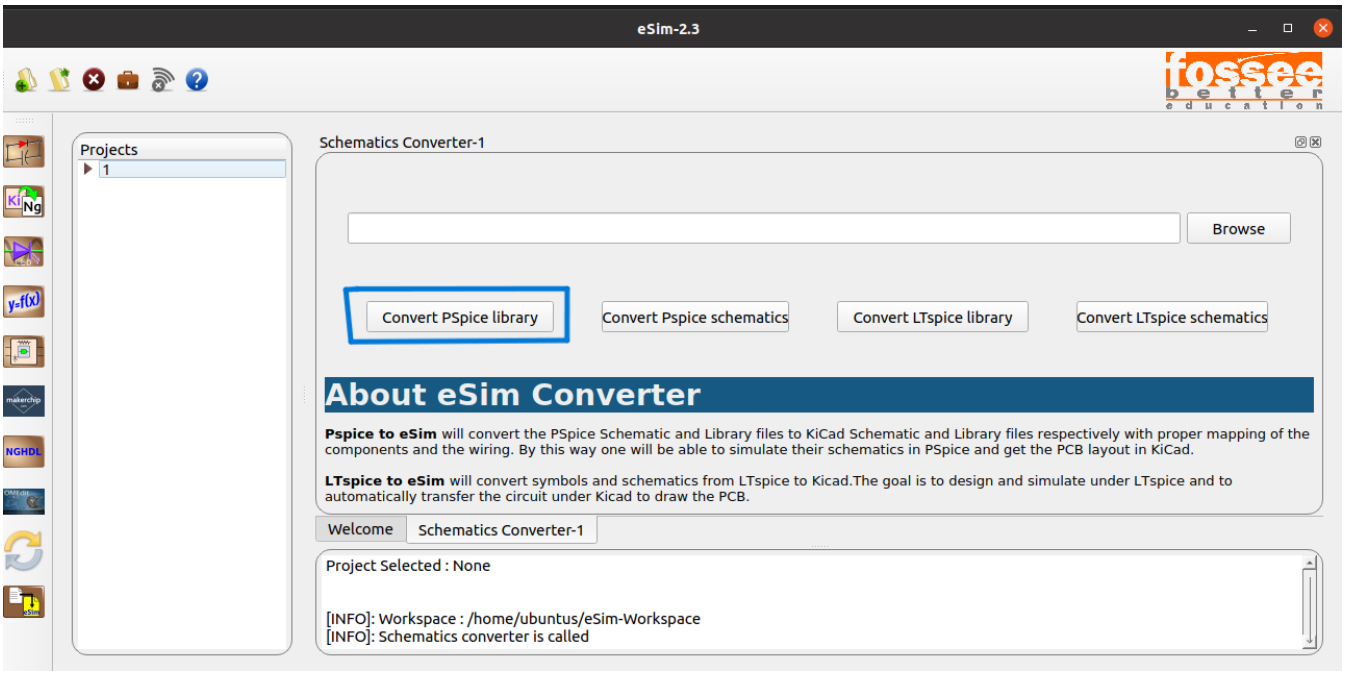
```
from PyQt5.QtWidgets import QFileDialog
```

```
def browse_path(self, text_box):
    file_dialog = QFileDialog()  # a dialog that allows the user to select files or directo
    file_dialog.setFileMode(QFileDialog.AnyFile)
    # Include all supported extensions
    file_dialog.setNameFilter("Supported Files (*.sch *.asc *.slb *.asy);;ASY Files (*.asy)
    file_dialog.exec_()  # Execute the dialog

    selected_files = file_dialog.selectedFiles()  # Get the selected file(s)
    if selected_files:
        text_box.setText(selected_files[0])
```

### 3.3.2 Convert PSpice Library

A .slb file is browsed and this button is clicked .

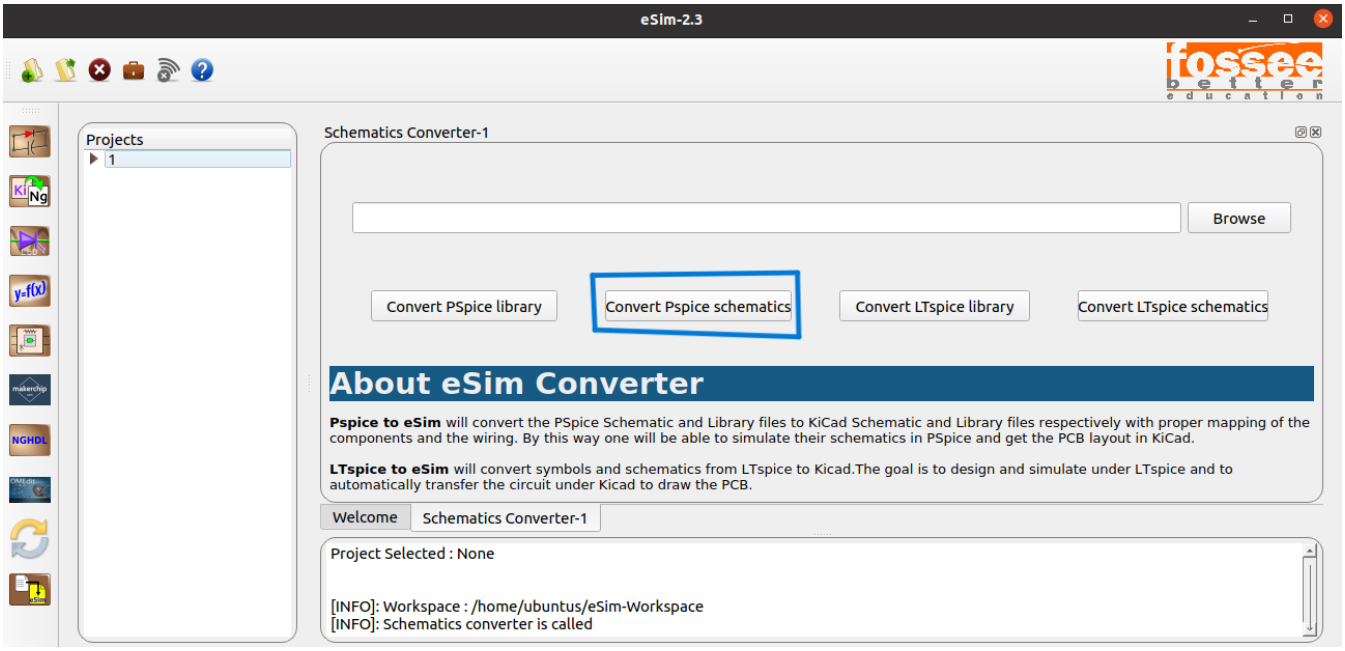
**Image**




### 3.3.3 Convert PSpice Schematics

A .sch file is browsed and this button is clicked .

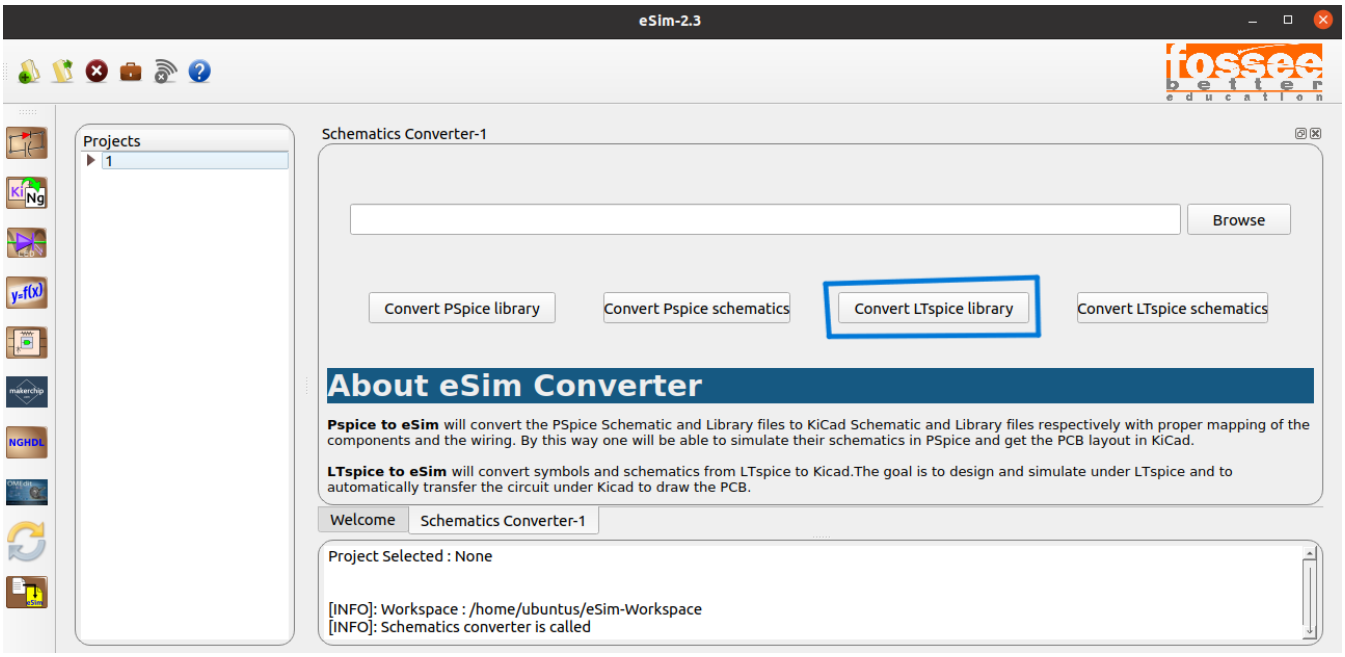
**Image**




### 3.3.4 Convert LTSpice Library

A .asy file is browsed and this button is clicked .

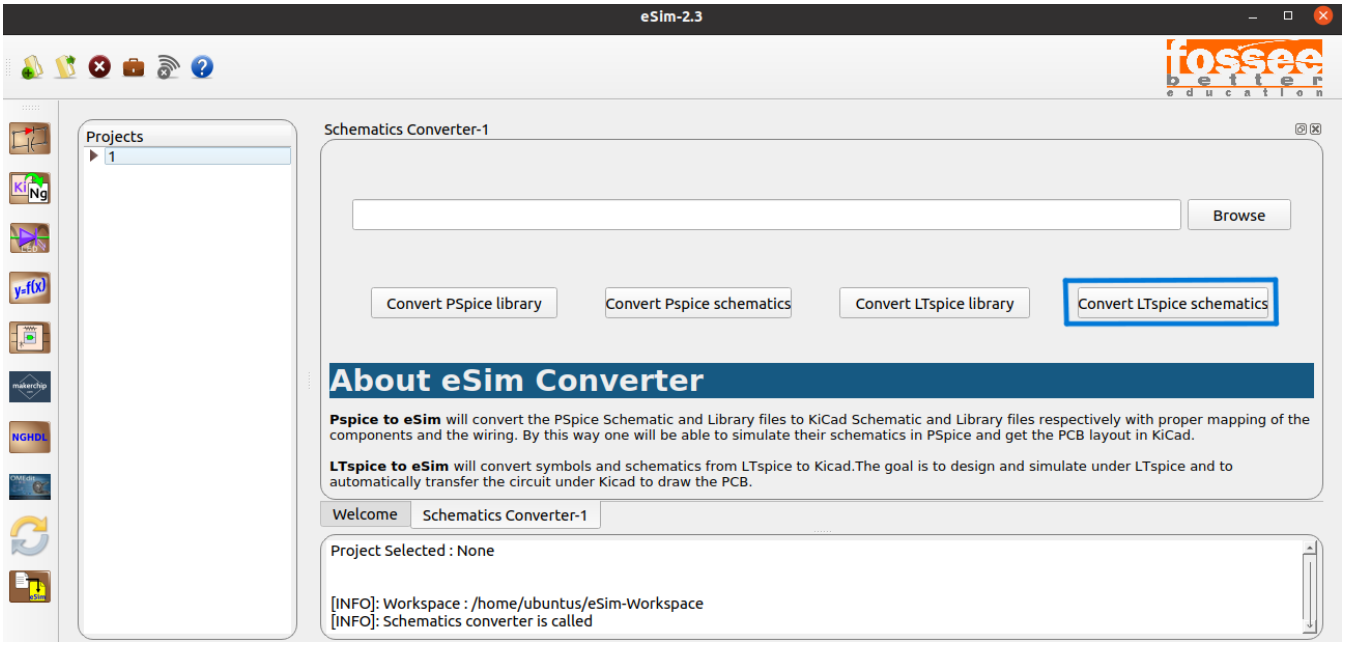
**Image**




### 3.3.5 Convert LTSpice Schematics

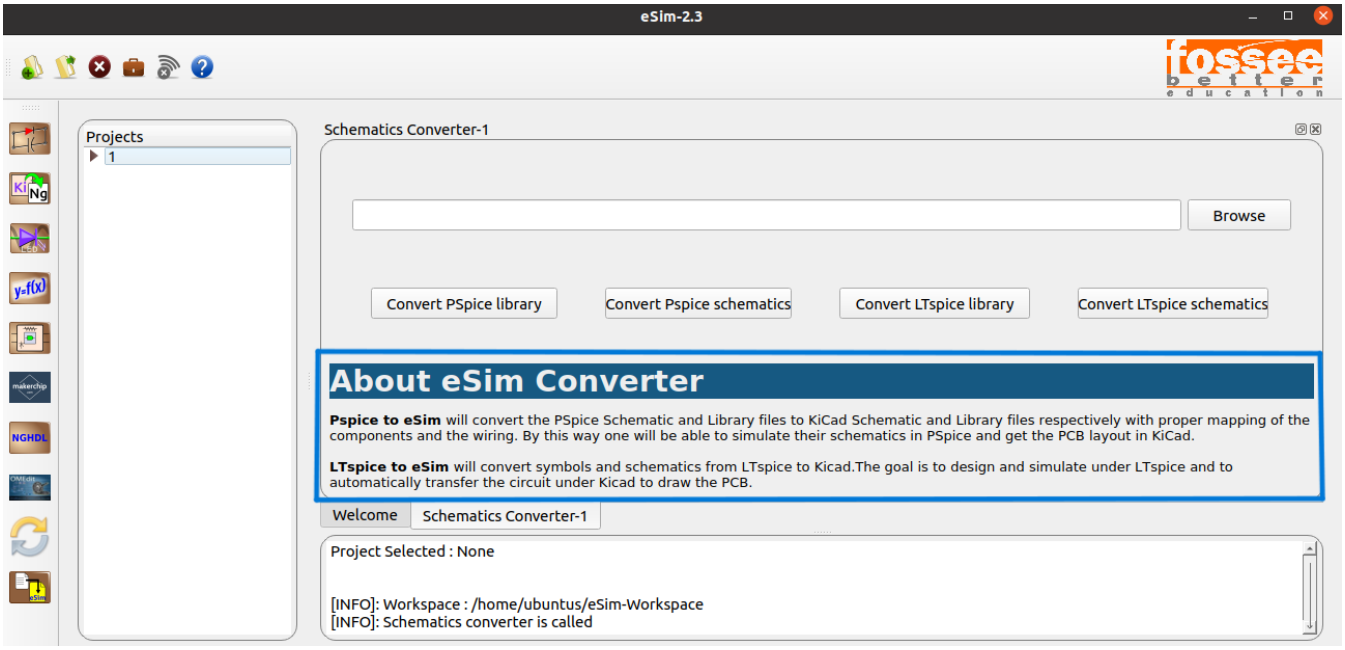
A .asc file is browsed and this button is clicked .

Image

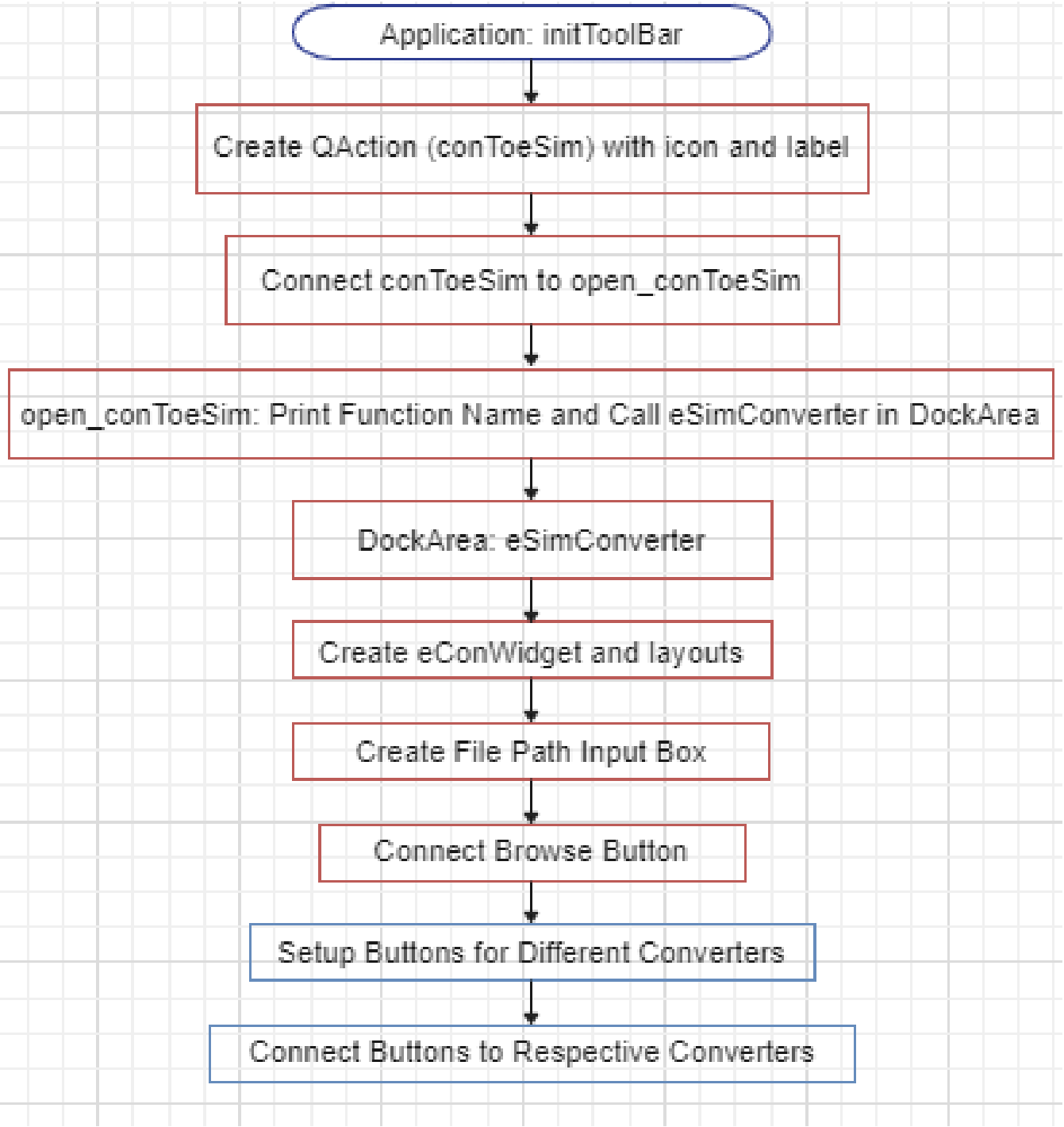



### 3.3.6 Description

Image



## 3.4 Flowchart



# Chapter 4

# Pspice-kicad

This chapter is about conversion of Pspice library and schematics to KiCad.

## 4.1 Conversion of Pspice Library

Pspice library is a .slb file this has to be converted into .lib file of KiCad. This is done by libConverter.py

### 4.1.1 libConverter.py

This Python class, 'PspiceLibConverter', is designed to handle the conversion of PSpice library files (.slb) to a custom format using an external parser script ('libParser.py'). The class is integrated with a PyQt5-based GUI application.

**Functionality**

1.Initialization:
- The class is initialized with a reference to the parent GUI ('parent') to interact with the main application.

2.Conversion Method ('convert'):
- Takes the file path of a PSpice library file as input.
- Checks if the file is not empty.
- Constructs the full path to the external parser script ('libParser.py') and executes it using the 'subprocess.run()' method.
- Displays success or error messages in a QMessageBox based on the outcome of the conversion process.

3.Upload Method ('upload_file_Pspice'):
- Takes the file path of a PSpice library file as input.
- Checks for spaces in the file path and shows a warning message if spaces are present.
- Verifies that the file has a ".slb" extension for compatibility.
- Calls the 'convert' method if the conditions are met, initiating the conversion process.
- Displays appropriate warning messages in QMessageBox for invalid file paths or file types.

The class integrates user-friendly QMessageBox notifications to inform
the user about the status of the conversion process and

handles scenarios such as empty files, invalid file paths, and unsupported file types.
It ensures that the conversion process is initiated only when the provided conditions are met.

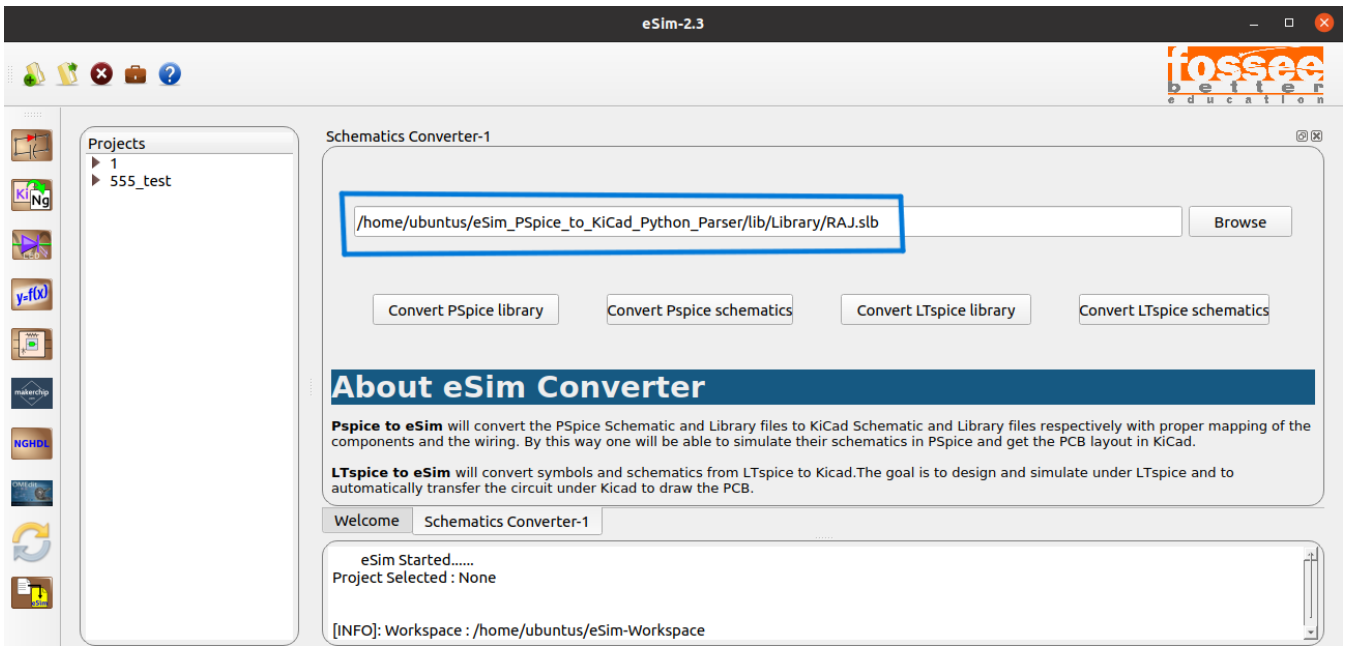
### Code

```
import os
import subprocess
from PyQt5.QtWidgets import QMessageBox

class PspiceLibConverter:
    def __init__(self, parent):
        self.parent = parent

    def convert(self, file_path):

        # Get the base name of the file without the extension
        filename = os.path.splitext(os.path.basename(file_path))[0]
        conPath = os.path.dirname(file_path)

        # Checks if the file is not empty
        if os.path.getsize(file_path) > 0:
            # Get the absolute path of the current script's directory
            script_dir = os.path.dirname(os.path.abspath(__file__))

            # Define the relative path to parser.py from the current script's directory
            relative_parser_path = "schematic_converters/lib/PythonLib"

            # Construct the full path to libParser.py
            parser_path = os.path.join(script_dir, relative_parser_path)
            print(parser_path)
            command = f"cd {parser_path} ; python3 libParser.py {file_path}"
            print(f"cd {parser_path} ; python3 libParser.py {file_path}")
            try:
                subprocess.run(command, shell=True, check=True)
                # Message box with the conversion success message
                msg_box = QMessageBox()
                msg_box.setIcon(QMessageBox.Information)
                msg_box.setWindowTitle("Conversion Successful")
                msg_box.setText("The file has been converted successfully.")
                msg_box.exec()
                print("Conversion of Pspice library is Successful")

            except subprocess.CalledProcessError as e:
                print("Error:", e)
        else:
            print("File is empty. Cannot perform conversion.")
            # A message box indicating that the file is empty
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Warning)
            msg_box.setWindowTitle("Empty File")
```

```
            msg_box.setText("The selected file is empty.
            Conversion cannot be performed.")
            msg_box.setStandardButtons(QMessageBox.Ok)
            msg_box.exec_()

    def upload_file_Pspice(self, file_path):
        if file_path:
            # Check if the file path contains spaces
            if ' ' in file_path:
                # Show a message box indicating that spaces are not allowed
                msg_box = QMessageBox()
                msg_box.setIcon(QMessageBox.Warning)
                msg_box.setWindowTitle("Invalid File Path")
                msg_box.setText("Spaces are not allowed in the file path.")
                msg_box.setStandardButtons(QMessageBox.Ok)
                msg_box.exec_()
                return

            if ".slb" in file_path:
                print(file_path)
                self.convert(file_path)
            else:
                msg_box = QMessageBox()
                msg_box.setIcon(QMessageBox.Warning)
                msg_box.setWindowTitle("Invalid File Path")
                msg_box.setText("Only .slb file can be converted.")
                msg_box.setStandardButtons(QMessageBox.Ok)
                msg_box.exec_()
                return

        else:
            print("No file selected.")

            # Message box indicating that no file is selected
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Warning)
            msg_box.setWindowTitle("No File Selected")
            msg_box.setText("Please select a file before uploading.")
            msg_box.setStandardButtons(QMessageBox.Ok)
            msg_box.exec_()
```

### 4.1.2 Input and Output

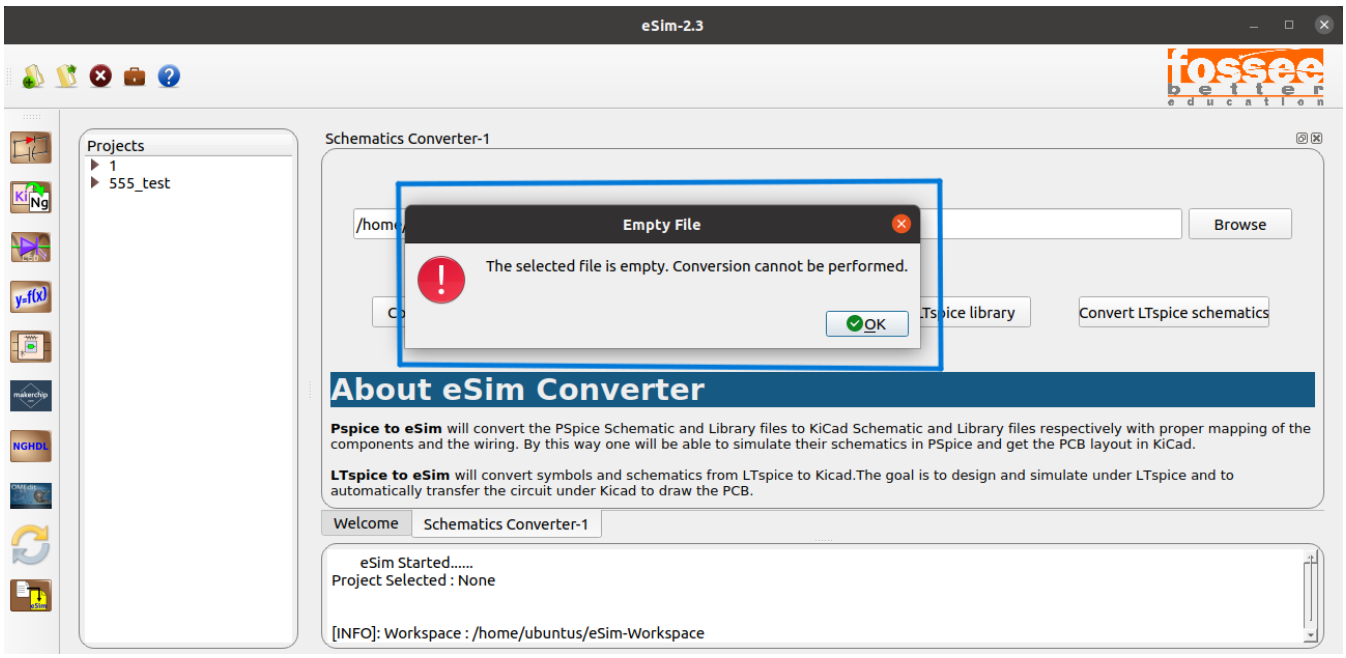
**Input:**
The main input for this code is the path to a PSpice library file (with a ".slb" extension) provided as a command-line argument. The file is expected to be in a non-empty state for the conversion to proceed.




**Output:** The code performs the following checks and generates outputs accordingly:

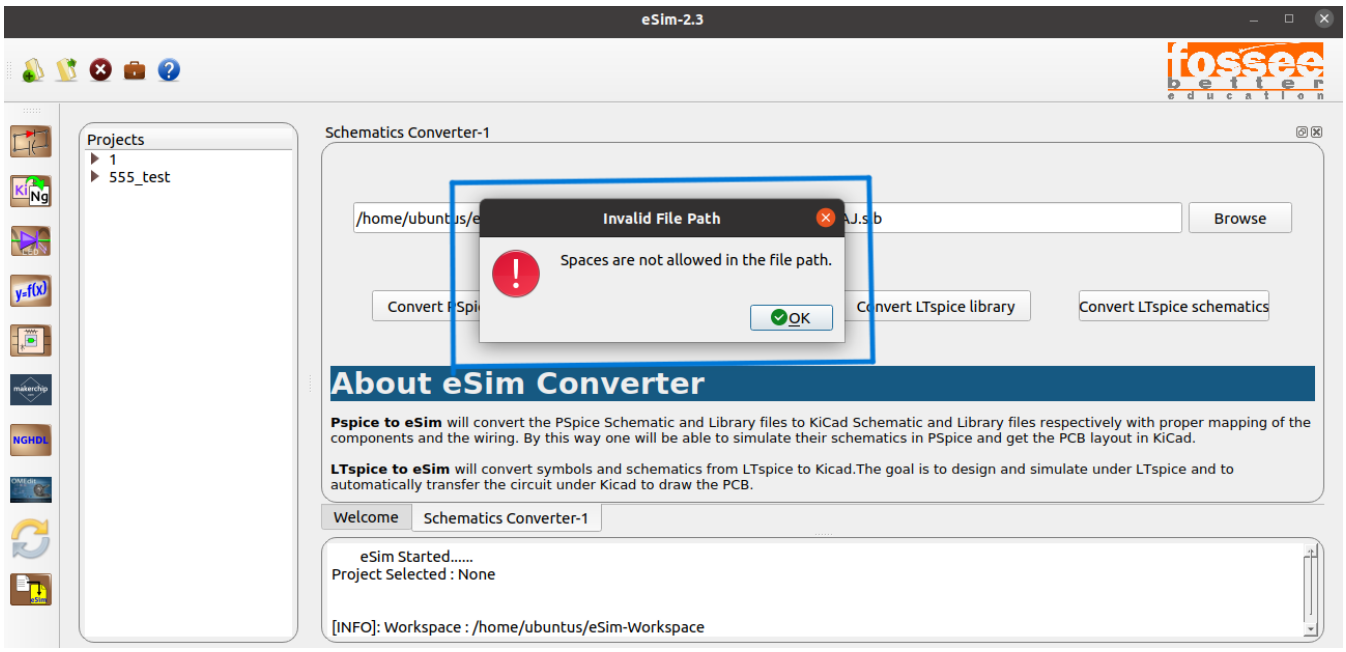
**File Empty Check:** If the specified PSpice library file is empty:

**Output:** Display a QMessageBox warning stating "The selected file is empty. Conversion cannot be performed."



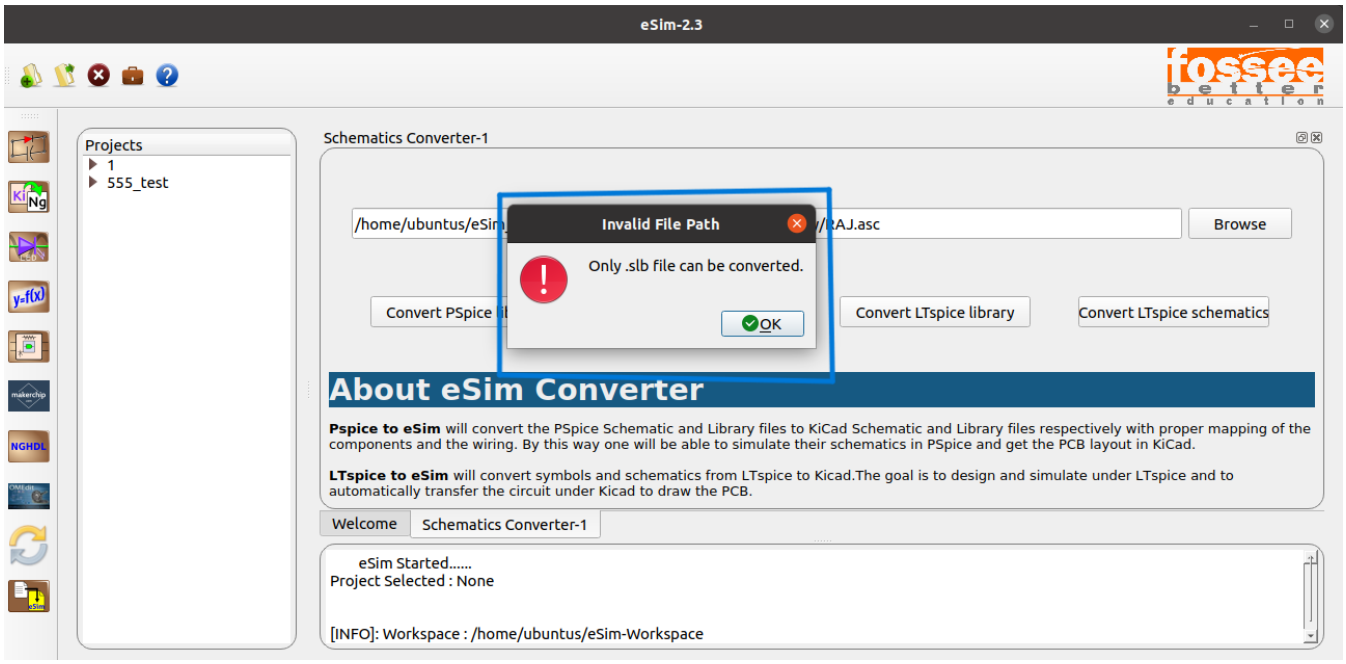

**File Path Check:** If there are spaces in the file path:

**Output:** Display a QMessageBox warning stating "Spaces are not allowed in the file path."




**File Type Check:** If the file does not have a ".slb" extension:

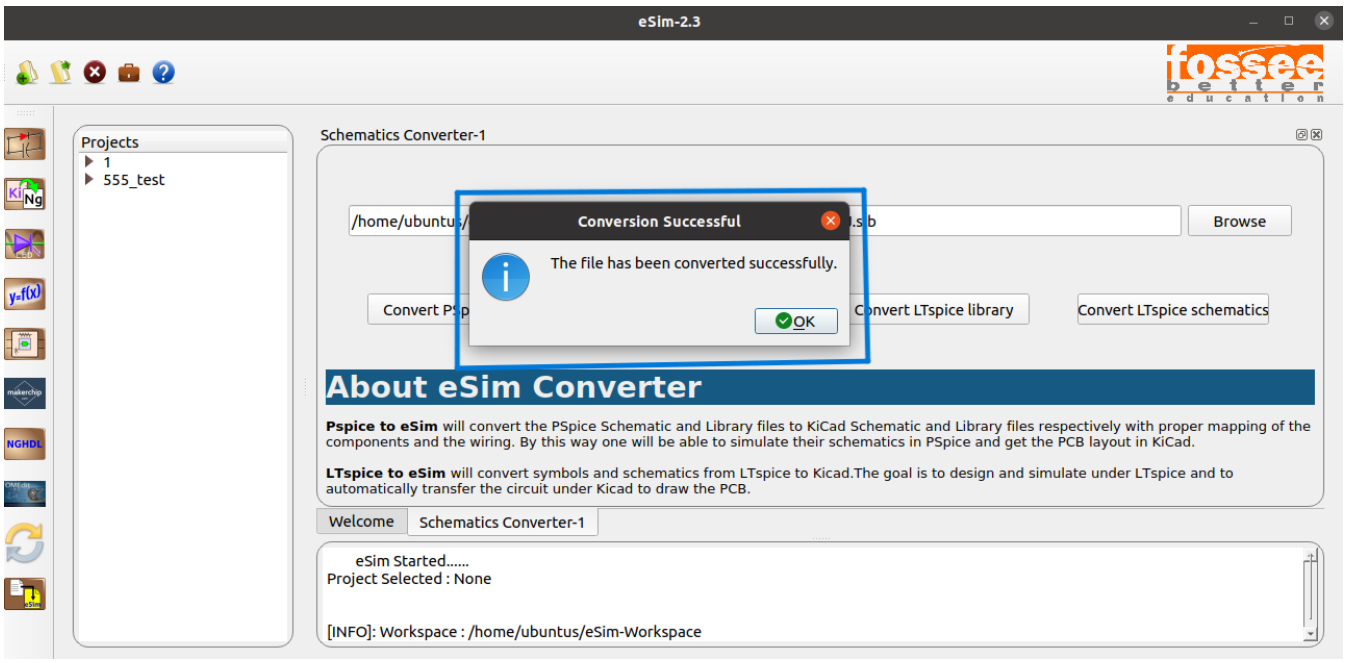
**Output:** Display a QMessageBox warning stating "Only .slb file can be converted."




**Conversion:** If the above checks pass and the file is non-empty, has no spaces in the path, and has a ".slb" extension: Execute an external script (libParser.py) for conversion.

**<u>If the conversion is successful:</u>**

**Output:** Display a QMessageBox information stating "The file has been converted successfully." And the .lib file will be created.




**If an error occurs during conversion:**

**Output:** Print an error message to the console.

## 4.2 Conversion of Pspice Schematic

Pspice schematic is a .sch file this has to be converted into KiCad schematic. This is done by pspiceToKicad.py

### 4.2.1 pspiceToKicad.py

This Python class, PspiceConverter, is designed to facilitate the conversion of PSpice schematic files (.sch) to a custom format using an external parser script (parser.py). The class is part of a larger PyQt5-based GUI application.

**Functionality**

**1.Initialization:**
- The class is initialized with a reference to the parent GUI (parent) to establish communication with the main application.

**Input:** Takes the parent GUI as a reference (parent) to interact with the main application.
**Output:** None.

**2. get_workspace_directory Method:**
-Obtains the workspace directory from a hidden folder ( /.esim) and a workspace file (workspace.txt).
-Returns the workspace directory if found; otherwise, returns None.

**Input:** None.
**Output:** Returns the workspace directory (string) if found, otherwise returns None.

**3. convert Method:**
-Takes the file path of a PSpice schematic file as input.
-Checks if the file is not empty.
-Constructs the full path to the external parser script (parser.py) and executes it using the subprocess.run() method.
-Displays a QMessageBox with a success message if the conversion is successful.
-Copies the converted file to the workspace directory if available and notifies the user.
-Handles errors during the conversion process and displays appropriate messages.

**Input:** file_path: File path (string) of the PSpice schematic file to be converted.
**Output:** None. Displays messages using QMessageBox to inform the user about the conversion status.

**4. upload_file_Pspice Method:**
-Takes the file path of a PSpice schematic file as input.
-Checks for spaces in the file path and displays a warning message if spaces are present.
-Verifies that the file has a ".sch" extension for compatibility. -Calls the convert method if conditions are met, initiating the conversion process.
-Displays warning messages in QMessageBox for invalid file paths or unsupported file types.

**Input:** file_path: File path (string) of the PSpice schematic file to be uploaded.
**Output:** None. Displays messages using QMessageBox to inform the user about the upload status.

**5. merge_copytree Function:**
-Helper function for merging the converted file into the eSim workspace directory.
-Creates a folder in the workspace directory corresponding to the converted file.
-Recursively copies the contents of the converted file to the workspace directory.
-Displays messages regarding the success or failure of folder creation and file copying.

**Input:** src: Source directory (string) to be copied.
dst: Destination directory (string) where the contents will be copied.
filename: Name of the file being converted (string).
**Output:** None. Displays messages regarding the success or failure of folder creation and file copying.

**6. User Interaction:**
-Utilizes QMessageBox to inform the user about the status of the conversion process, empty files, invalid file paths, and successful additions to the project explorer.
-Allows the user to open the project manually after successful conversion.

**7. Integration with Project Explorer:**
-Integrates with the project explorer (ProjectExplorer class from frontEnd module) by adding the converted file under the project explorer if the workspace directory is found.

The class follows a modular and user-friendly design, providing clear feedback to the user

through QMessageBox. Incorporates error handling to gracefully manage situations like empty files or conversion errors. Ensures that the conversion process is only initiated when conditions for compatibility are met, enhancing robustness.

**Code**

```
import os
import subprocess
import shutil
from PyQt5.QtWidgets import QMessageBox
from frontEnd import ProjectExplorer

class PspiceConverter:
    def __init__(self, parent):
        self.parent = parent

    def get_workspace_directory(self):
        # Path to the hidden folder and the workspace file
        hidden_folder_path = os.path.join(os.path.expanduser('~'), '.esim')
        workspace_file_path = os.path.join(hidden_folder_path, 'workspace.txt')

        # Check if the hidden folder and the workspace file exist
        if os.path.exists(hidden_folder_path) and os.path.exists(workspace_file_path):
            # Read the workspace directory from the workspace.txt file
            with open(workspace_file_path, 'r') as file:
            # Remove any leading/trailing whitespaces
                workspace_directory = file.read().strip()
            # Split the string by spaces and select the last element
            workspace_directory = workspace_directory.split()[-1]
            return workspace_directory

# Return None if the hidden folder or the workspace file is not found
        return None

    def convert(self, file_path):
        # Get the base name of the file without the extension
        filename = os.path.splitext(os.path.basename(file_path))[0]
        conPath = os.path.dirname(file_path)

        # Checks if the file is not empty
        if os.path.getsize(file_path) > 0:
            # Get the absolute path of the current script's directory
            script_dir = os.path.dirname(os.path.abspath(__file__))

            # Define the relative path to parser.py from the current script's directory
            relative_parser_path = "schematic_converters/lib/PythonLib"

            # Construct the full path to parser.py
            parser_path = os.path.join(script_dir, relative_parser_path)
            command = f"cd {parser_path} && python3 parser.py {file_path}
```

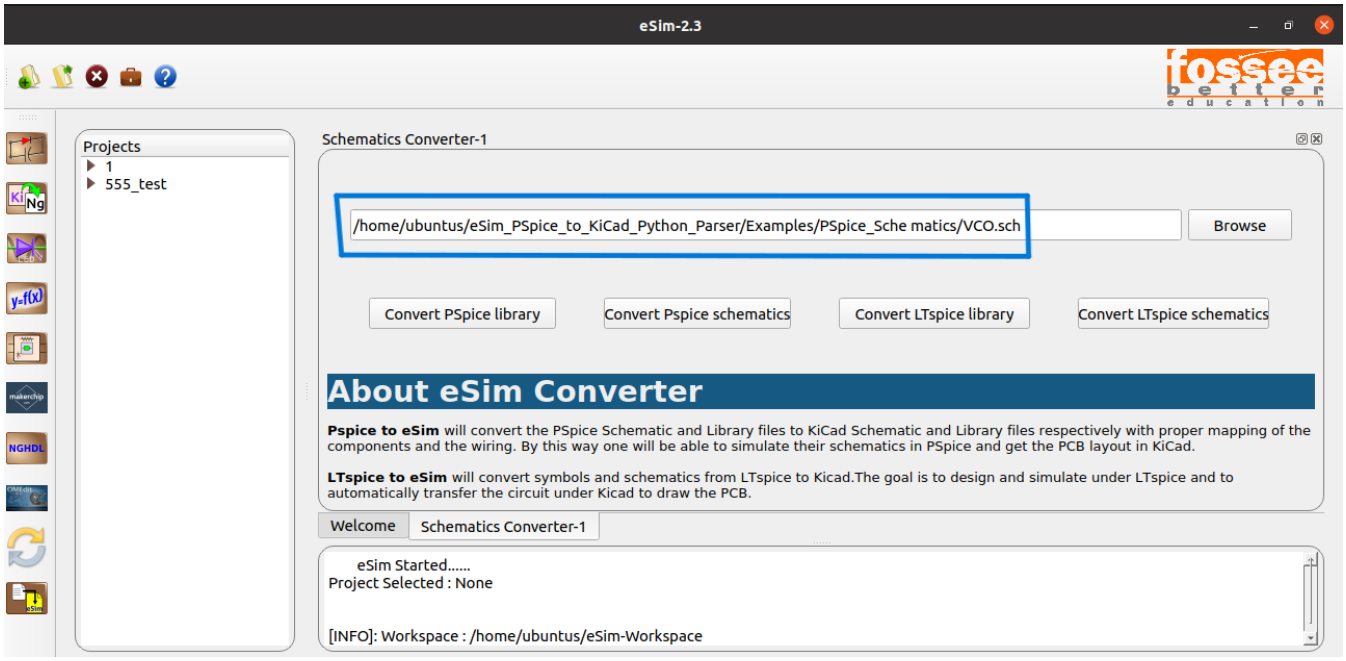
```
                {conPath}/{filename}"
        try:
            subprocess.run(command, shell=True, check=True)
            # Message box with the conversion success message
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Information)
            msg_box.setWindowTitle("Conversion Successful")
            newFile = str(conPath + "/" + filename)
            workspace_directory = self.get_workspace_directory()


            if workspace_directory:
                    print(f"Workspace directory found: {workspace_directory}")
                    merge_copytree(newFile, workspace_directory, filename)
                    msg_box.setText(f"The file has been converted successfully.
                    Saved in {workspace_directory}.  Open the Project manually.")
                    print("File added under the project explorer.")
            else:
                    print("Workspace directory not found.")
            result = msg_box.exec_()
            print("Conversion of Pspice to eSim schematic Successful")

        except subprocess.CalledProcessError as e:
            print("Error:", e)
    else:
        print("File is empty. Cannot perform conversion.")
        # A message box indicating that the file is empty
        msg_box = QMessageBox()
        msg_box.setIcon(QMessageBox.Warning)
        msg_box.setWindowTitle("Empty File")
        msg_box.setText("The selected file is empty. Conversion cannot be performed.")
        msg_box.setStandardButtons(QMessageBox.Ok)
        msg_box.exec_()

def upload_file_Pspice(self, file_path):
    if file_path:
        # Check if the file path contains spaces
        if ' ' in file_path:
            # Show a message box indicating that spaces are not allowed
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Warning)
            msg_box.setWindowTitle("Invalid File Path")
            msg_box.setText("Spaces are not allowed in the file path.")
            msg_box.setStandardButtons(QMessageBox.Ok)
            msg_box.exec_()
            return

        if ".sch" in file_path:
            print(file_path)
            self.convert(file_path)
```

```
            else:
                msg_box = QMessageBox()
                msg_box.setIcon(QMessageBox.Warning)
                msg_box.setWindowTitle("Invalid File Path")
                msg_box.setText("Only .sch file can be converted.")
                msg_box.setStandardButtons(QMessageBox.Ok)
                msg_box.exec_()
                return

        else:
            print("No file selected.")

            # Message box indicating that no file is selected
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Warning)
            msg_box.setWindowTitle("No File Selected")
            msg_box.setText("Please select a file before uploading.")
            msg_box.setStandardButtons(QMessageBox.Ok)
            msg_box.exec_()

def merge_copytree(src, dst, filename):
    if not os.path.exists(dst):
        os.makedirs(dst)

    folder_path = f"{dst}/{filename}" # Folder to be created in eSim-Workspace

    # Create the folder
    try:
        os.makedirs(folder_path)
        print(f"Folder created at {folder_path}")
    except OSError as error:
        print(f"Folder creation failed: {error}")

    for item in os.listdir(src):
        src_item = os.path.join(src, item)
        dst_item = os.path.join(folder_path, item)

        if os.path.isdir(src_item):
            merge_copytree(src_item, dst_item, filename)
        else:
            if not os.path.exists(dst_item) or os.stat(src_item).st_mtime
            > os.stat(dst_item).st_mtime:
                shutil.copy2(src_item, dst_item)
```

### 4.2.2 Input and Output

**Input:**
The primary input for the PspiceConverter class is the file path to a PSpice schematic file (with a ".sch" extension) provided as an argument, typically through user interaction in a GUI. The file is

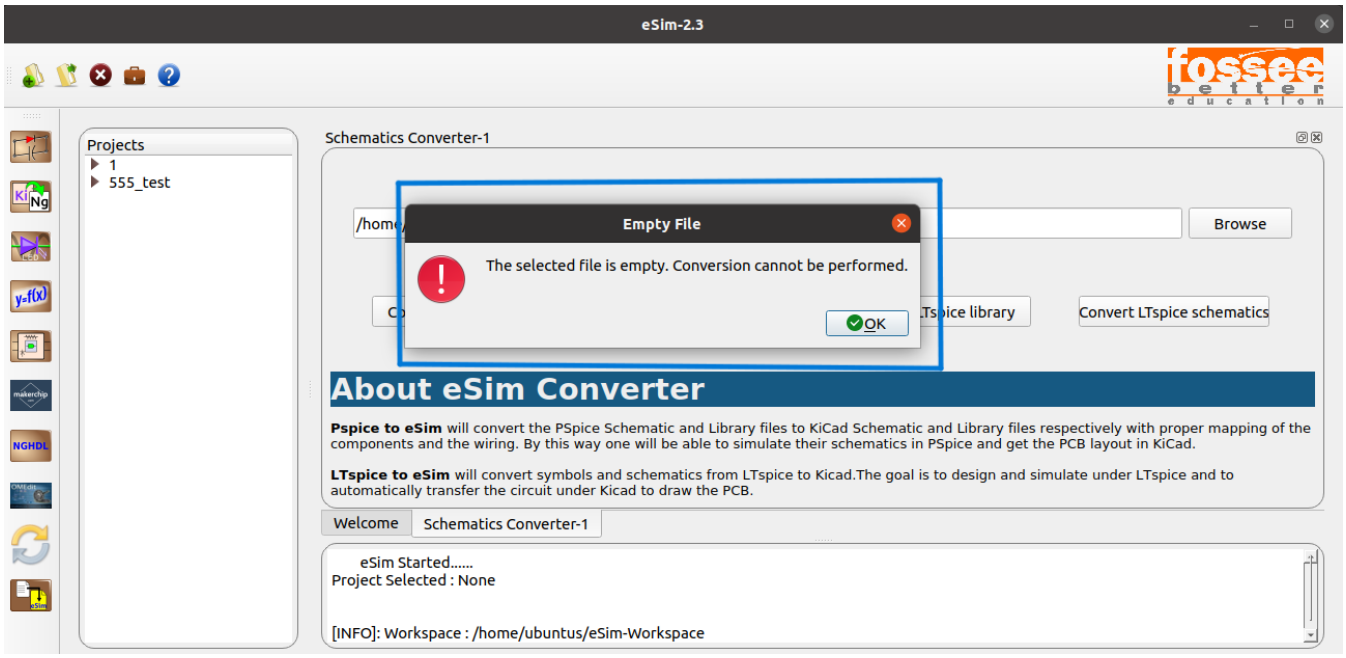
expected to be non-empty for the conversion process to proceed.



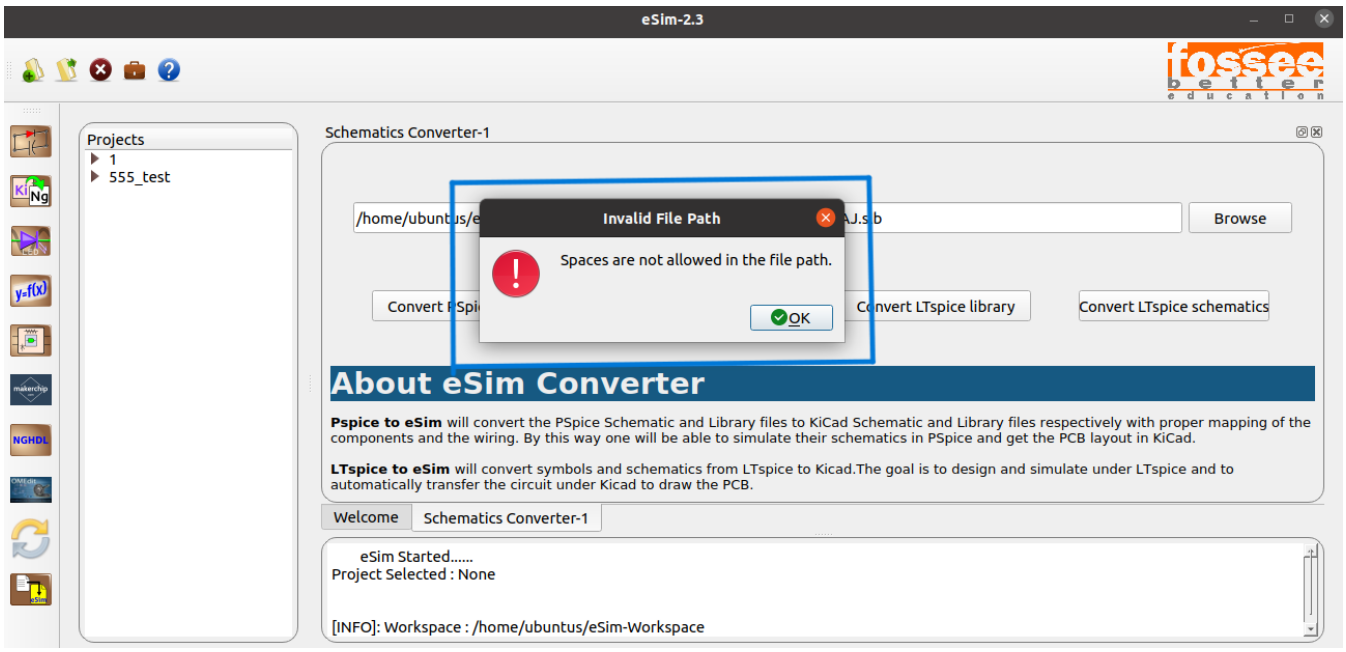

**Output:**

The code performs the following checks and generates outputs accordingly:

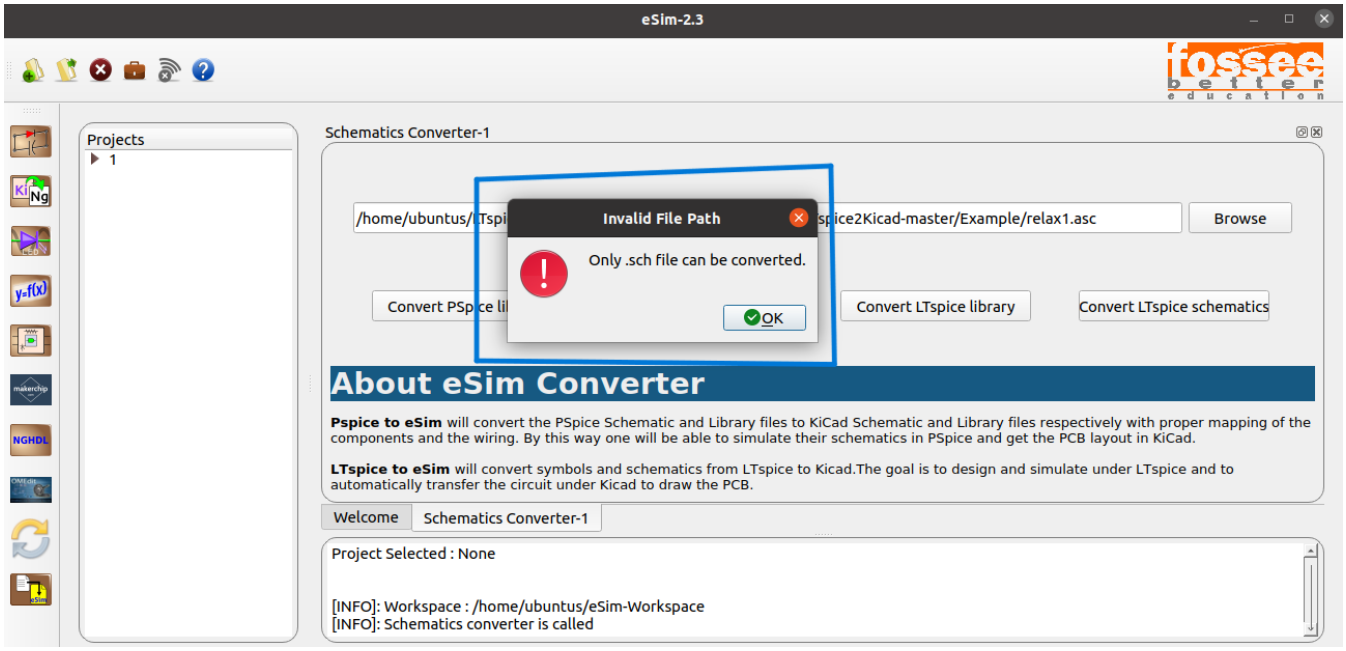
**File Empty Check:** If the specified PSpice schematic file is empty:
**Output:** Display a QMessageBox warning stating "The selected file is empty. Conversion cannot be performed."




**File Path Check:** If there are spaces in the file path:
**Output:** Display a QMessageBox warning stating "Spaces are not allowed in the file path."

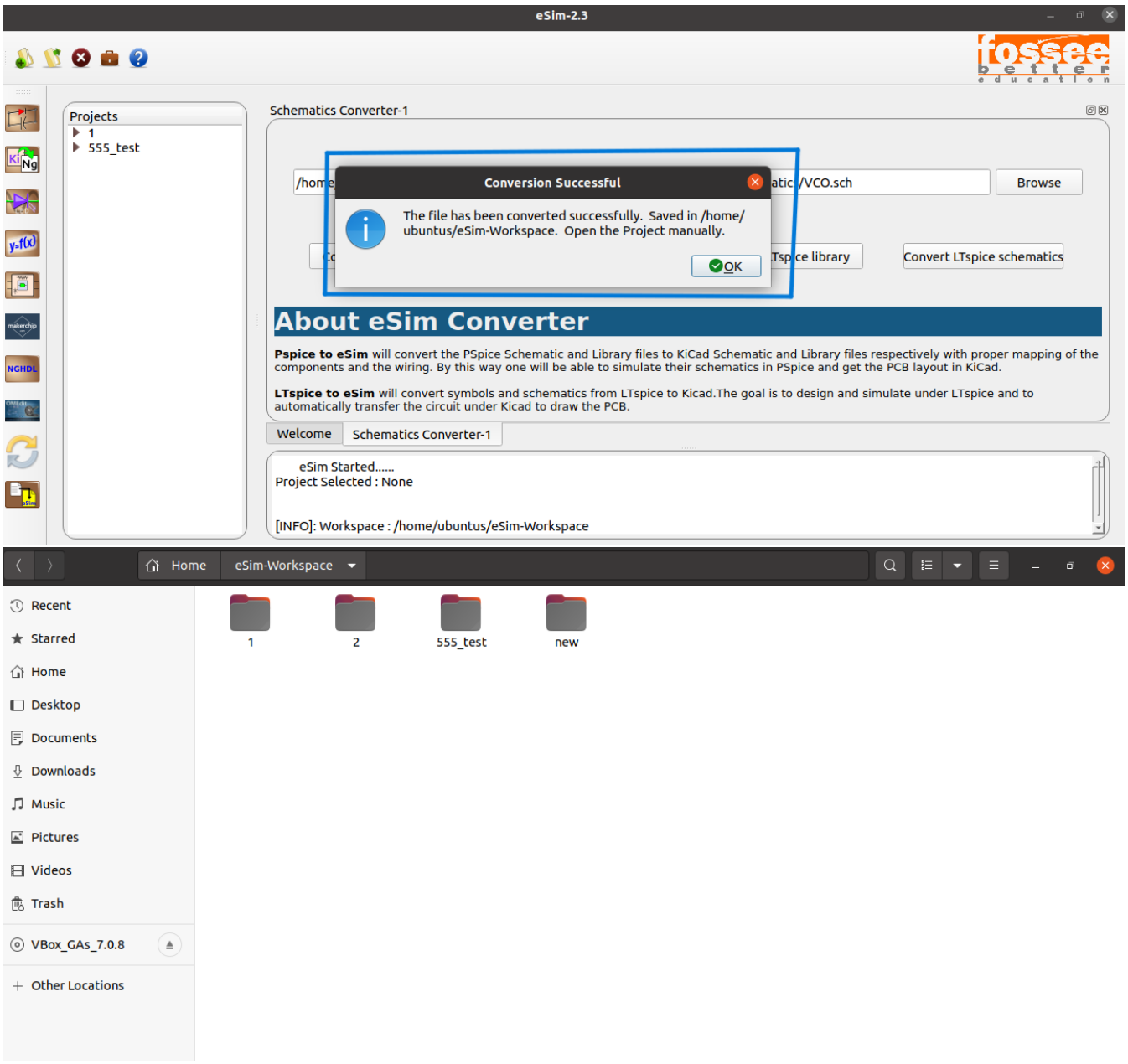



**File Type Check:** If the file does not have a ".sch" extension:
**Output:** Display a QMessageBox warning stating "Only .sch file can be converted."




**Conversion:** If the above checks pass and the file is non-empty, has no spaces in the path, and has a ".sch" extension:
Execute an external script (parser.py) for conversion.

**If the conversion is successful:**
**Output:** Display a QMessageBox information stating "The file has been converted successfully."
The converted file is then added under the workSpace if the workspace directory is found.




**If an error occurs during conversion:**

**Output:** Print an error message to the console.

# Chapter 5

# LTspice-kicad

This chapter is about conversion of LTspice library and schematics to KiCad.

## 5.1 Conversion of LTspice Library

LTspice library is a .asy file this has to be converted into .lib file of KiCad. This is done by LTspiceLibConverter.py

### 5.1.1 LTspiceLibConverter.py

The LTspiceLibConverter Python class is designed to handle the conversion of LTspice library files to a custom format using an external parser script (lib_LTspice2Kicad.py). The class is integrated with a PyQt5-based GUI application.

**Functionality**

**1.Initialization:**
- The class is initialized with a reference to the parent GUI (parent) to interact with the main application.

**2.Conversion Method ('convert'):**
- Takes the file path of an LTspice library file as input.
-Checks if the file is not empty.
-Constructs the full path to the external parser script (lib_LTspice2Kicad.py) and executes it using the subprocess.run() method.
-Displays success or error messages in a QMessageBox based on the outcome of the conversion process.

**3.Upload Method ('upload_file_LTspice):**
- Takes the file path of an LTspice library file as input. -Checks for spaces in the file path and shows a warning message if spaces are present.
-Verifies that the file has a ".asy" extension for compatibility. -Calls the convert method if the conditions are met, initiating the conversion process.
-Displays appropriate warning messages in QMessageBox for invalid file paths or file types.

The class integrates user-friendly QMessageBox notifications to inform the user about the status of the conversion process and handles scenarios such as empty files, invalid file paths, and

unsupported file types. It ensures that the conversion process is initiated only when the provided conditions are met.

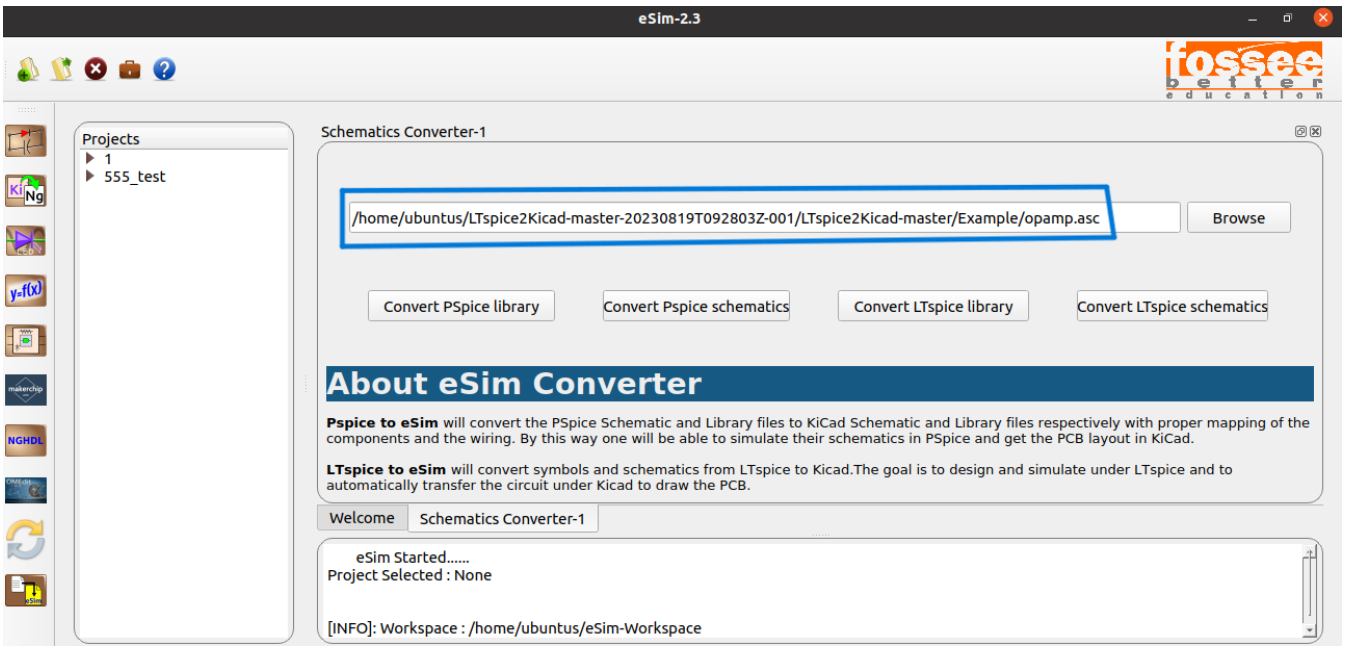
### Code

```
import os
import subprocess
from PyQt5.QtWidgets import QMessageBox

class LTspiceLibConverter:
    def __init__(self, parent):
        self.parent = parent

    def convert(self, file_path):

        # Get the base name of the file without the extension
        filename = os.path.splitext(os.path.basename(file_path))[0]
        conPath = os.path.dirname(file_path)

        # Checks if the file is not empty
        if os.path.getsize(file_path) > 0:
            # Get the absolute path of the current script's directory
            script_dir = os.path.dirname(os.path.abspath(__file__))

            # Define the relative path to parser.py from the current script's directory
            relative_parser_path = "LTSpiceToKiCadConverter/src/Ubuntu"

            # Construct the full path to libParser.py
            parser_path = os.path.join(script_dir, relative_parser_path)
            print(parser_path)
            command = f"cd {parser_path} ; python3 lib_LTspice2Kicad.py {file_path}"
            print(f"cd {parser_path} ; python3 lib_LTspice2Kicad.py {file_path}")
            try:
                subprocess.run(command, shell=True, check=True)
                # Message box with the conversion success message
                msg_box = QMessageBox()
                msg_box.setIcon(QMessageBox.Information)
                msg_box.setWindowTitle("Conversion Successful")
                msg_box.setText("The file has been converted successfully.")
                msg_box.exec()
                print("Conversion of LTspice library is Successful")

            except subprocess.CalledProcessError as e:
                print("Error:", e)
        else:
            print("File is empty. Cannot perform conversion.")
            # A message box indicating that the file is empty
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Warning)
            msg_box.setWindowTitle("Empty File")
```

```
            msg_box.setText("The selected file is empty. Conversion cannot be performed.")
            msg_box.setStandardButtons(QMessageBox.Ok)
            msg_box.exec_()

    def upload_file_LTspice(self, file_path):
        if file_path:
            # Check if the file path contains spaces
            if ' ' in file_path:
                # Show a message box indicating that spaces are not allowed
                msg_box = QMessageBox()
                msg_box.setIcon(QMessageBox.Warning)
                msg_box.setWindowTitle("Invalid File Path")
                msg_box.setText("Spaces are not allowed in the file path.")
                msg_box.setStandardButtons(QMessageBox.Ok)
                msg_box.exec_()
                return

            if ".asy" in file_path:
                print(file_path)
                self.convert(file_path)
            else:
                msg_box = QMessageBox()
                msg_box.setIcon(QMessageBox.Warning)
                msg_box.setWindowTitle("Invalid File Path")
                msg_box.setText("Only .asy file can be converted.")
                msg_box.setStandardButtons(QMessageBox.Ok)
                msg_box.exec_()
                return

        else:
            print("No file selected.")

            # Message box indicating that no file is selected
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Warning)
            msg_box.setWindowTitle("No File Selected")
            msg_box.setText("Please select a file before uploading.")
            msg_box.setStandardButtons(QMessageBox.Ok)
            msg_box.exec_()
```

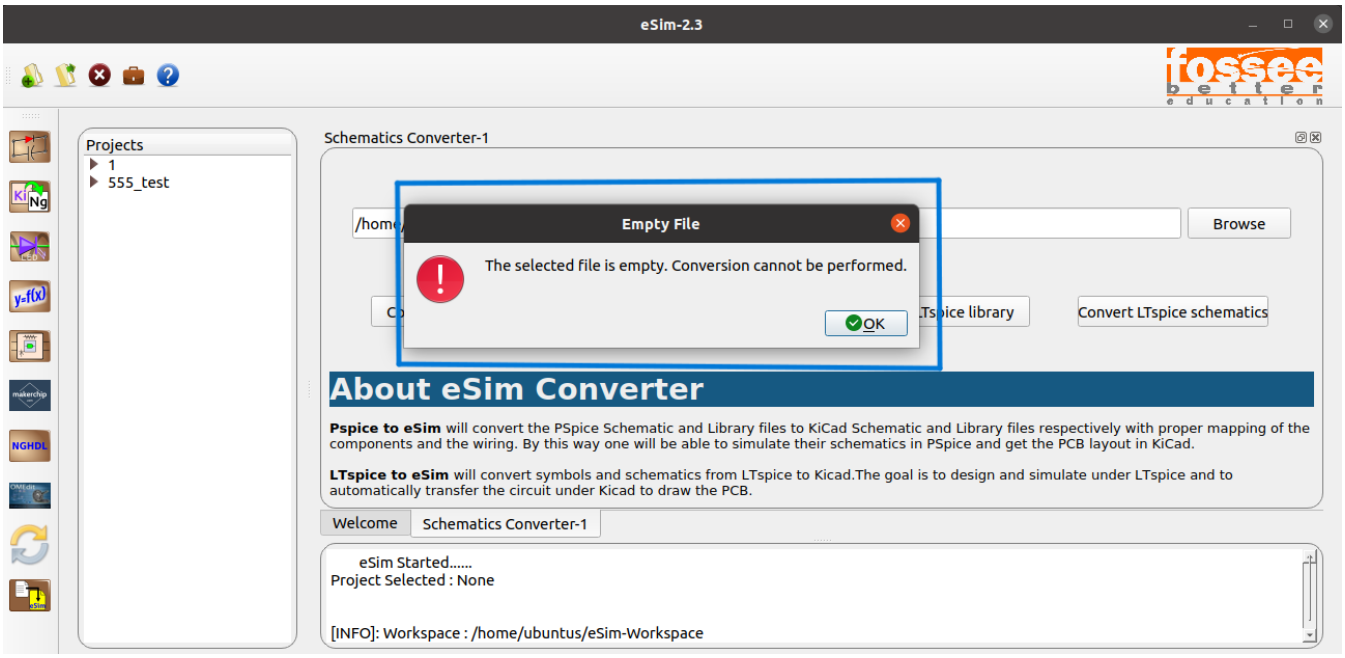
### 5.1.2 Input and Output

**Input:**
The main input for this code is the path to an LTspice library file (with a ".asy" extension) provided as a parameter to the upload_file_LTspice method. The file is expected to be in a non-empty state for the conversion to proceed.

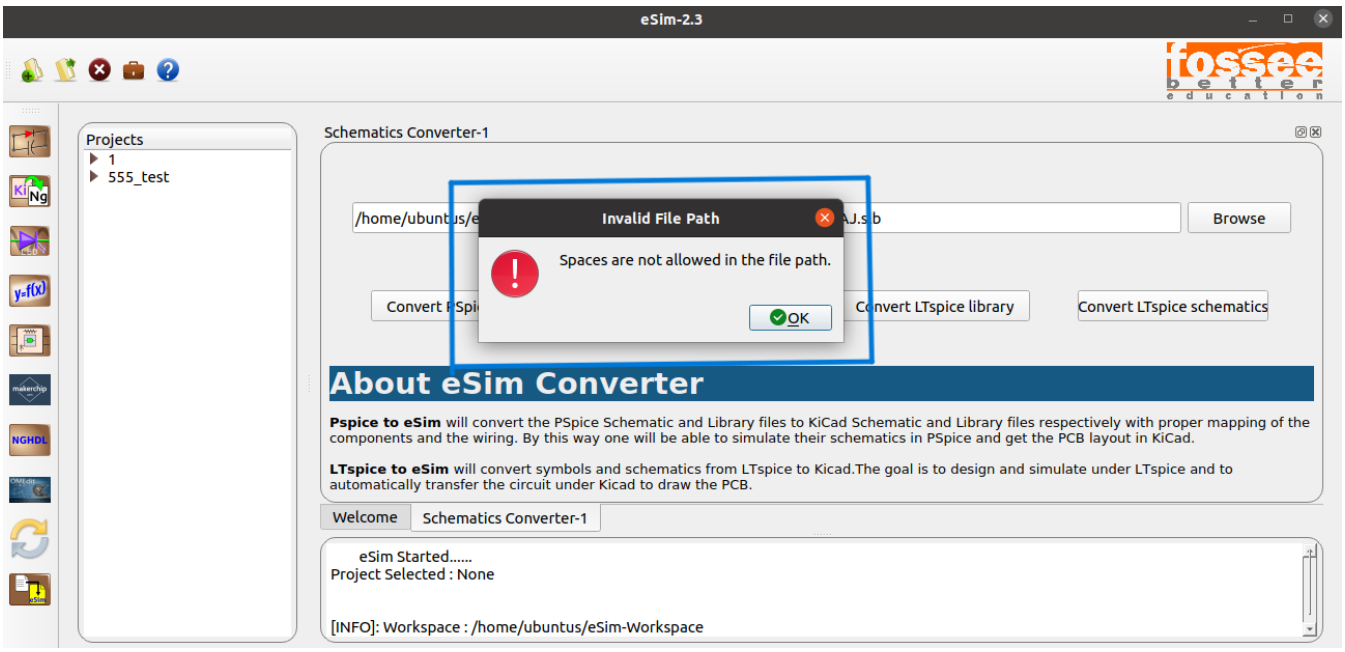



**Output:**
The code performs the following checks and generates outputs accordingly:

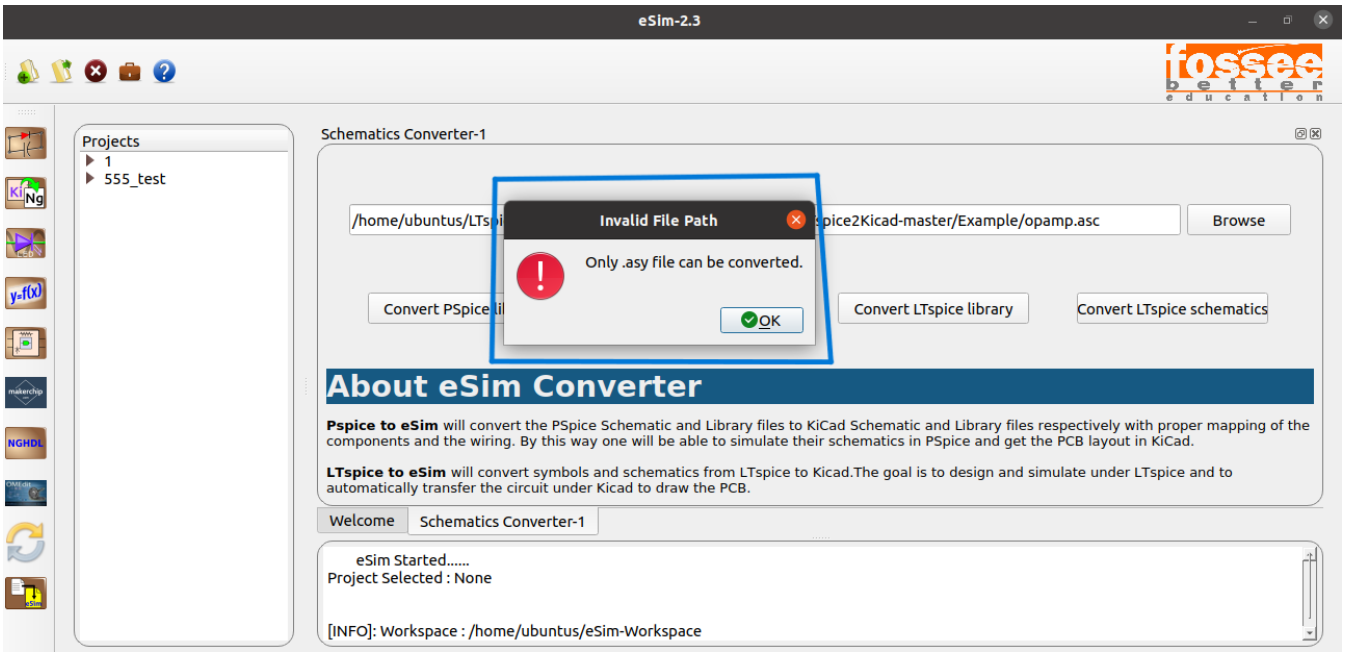
**File Empty Check:** If the specified LTspice library file is empty:
**Output:** Display a QMessageBox warning stating "The selected file is empty. Conversion cannot be performed."




**File Path Check:** If there are spaces in the file path:
**Output:** Display a QMessageBox warning stating "Spaces are not allowed in the file path."

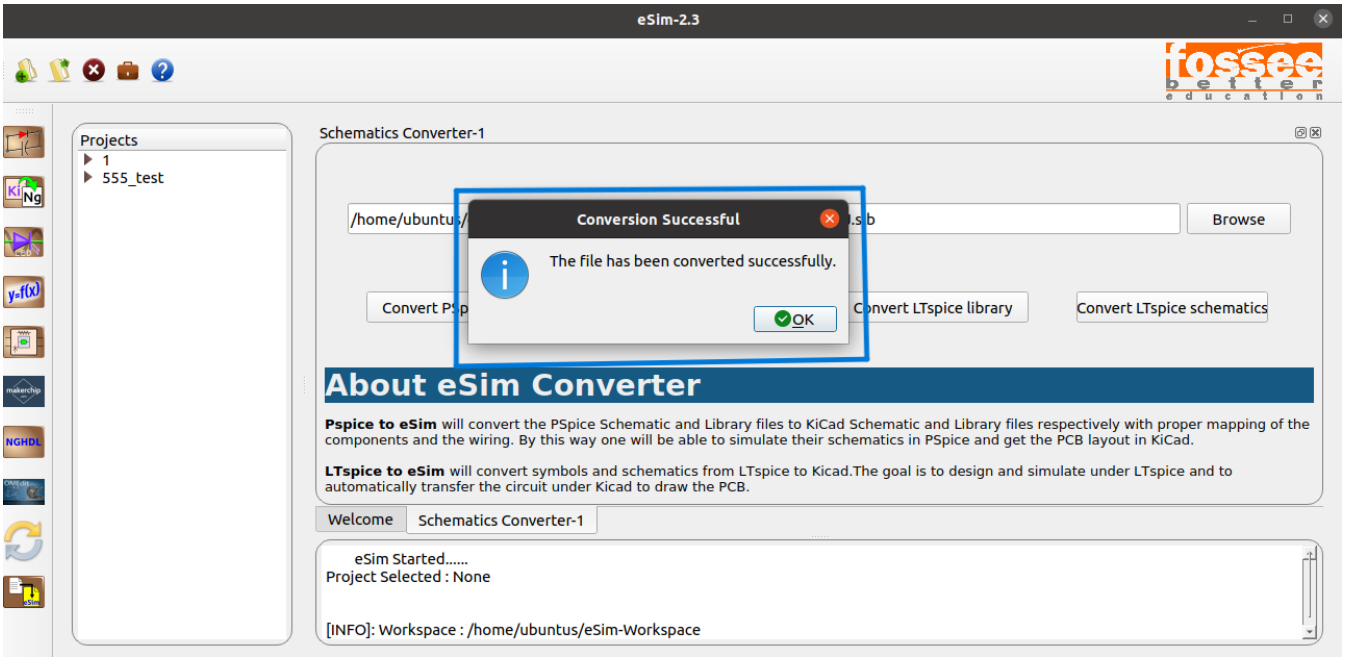



**File Type Check:** If the file does not have a ".asy" extension:
**Output:** Display a QMessageBox warning stating "Only .asy file can be converted."




**Conversion:** If the above checks pass and the file is non-empty, has no spaces in the path, and has a ".asy" extension:
Execute an external script (lib_LTspice2Kicad.py) for conversion.

**<u>If the conversion is successful:</u>**
**Output:** Display a QMessageBox information stating "The file has been converted successfully."
And the .lib file will be created.




**If an error occurs during conversion:**
**Output:** Print an error message to the console.

## 5.2 Conversion of LTspice Schematic

LTspice schematic is a .asc file this has to be converted into KiCad schematic. This is done by ltspiceToKicad.py

### 5.2.1 ltspiceToKicad.py

This Python class, LTspiceConverter, is designed to handle the conversion of LTspice schematic files (.asc) to a custom format using an external parser script (sch_LTspice2Kicad.py). The class is part of a larger PyQt5-based GUI application.

**Functionality**

**1.Initialization:**
- The class is initialized with a reference to the parent GUI (parent) to establish communication with the main application.

**Input:** Takes the parent GUI as a reference (parent) to interact with the main application.
**Output:** None.

**2. get_workspace_directory Method:**
-Obtains the workspace directory from a hidden folder ( /.esim) and a workspace file (workspace.txt).
-Returns the workspace directory if found; otherwise, returns None.

**Input:** None.
**Output:** Returns the workspace directory (string) if found, otherwise returns None.

**3. convert Method:**
-Takes the file path of a LTSpice schematic file as input.
-Checks if the file is not empty.
- Constructs the full path to the external parser script (sch_LTspice2Kicad.py) and executes it using the subprocess.run() method.
-Displays a QMessageBox with a success message if the conversion is successful.
-Copies the converted file to the workspace directory if available and notifies the user.
-Handles errors during the conversion process and displays appropriate messages.

**Input:** file_path: File path (string) of the LTSpice schematic file to be converted.
**Output:** None. Displays messages using QMessageBox to inform the user about the conversion status.

**4. upload_file_LTspice Method:**
-Takes the file path of a LTSpice schematic file as input.
-Checks for spaces in the file path and displays a warning message if spaces are present.
-Verifies that the file has a ".asc" extension for compatibility.
-Calls the convert method if conditions are met, initiating the conversion process.
-Displays warning messages in QMessageBox for invalid file paths or unsupported file types.

**Input:** file_path: File path (string) of the LTSpice schematic file to be uploaded.
**Output:** None. Displays messages using QMessageBox to inform the user about the upload status.

**5. merge_copytree Function:**
-Helper function for merging the converted file into the eSim workspace directory.
-Creates a folder in the workspace directory corresponding to the converted file.
-Recursively copies the contents of the converted file to the workspace directory.
-Displays messages regarding the success or failure of folder creation and file copying.

**Input:**
**src:** Source directory (string) to be copied.
**dst:** Destination directory (string) where the contents will be copied.
**filename:** Name of the file being converted (string).
**Output:** None. Displays messages regarding the success or failure of folder creation and file copying.

**6. User Interaction:**
-Utilizes QMessageBox to inform the user about the status of the conversion process, empty files, invalid file paths, and successful additions to the project explorer.
-Allows the user to open the project manually after successful conversion.

**7. Integration with Project Explorer:**
-Integrates with the project explorer (ProjectExplorer class from frontEnd module) by adding the converted file under the project explorer if the workspace directory is found.

The class follows a modular and user-friendly design, providing clear feedback to the user through QMessageBox.
Incorporates error handling to gracefully manage situations like empty files or conversion errors. Ensures that the conversion process is only initiated when conditions for compatibility are met, enhancing robustness.

### Code

```
import os
import subprocess
import shutil
from PyQt5.QtWidgets import QMessageBox

class LTspiceConverter:
    def __init__(self, parent):
        self.parent = parent

    def get_workspace_directory(self):
        # Path to the hidden folder and the workspace file
        hidden_folder_path = os.path.join(os.path.expanduser('~'), '.esim')
        workspace_file_path = os.path.join(hidden_folder_path, 'workspace.txt')

        # Check if the hidden folder and the workspace file exist
        if os.path.exists(hidden_folder_path) and os.path.exists(workspace_file_path):
            # Read the workspace directory from the workspace.txt file
            with open(workspace_file_path, 'r') as file:
            # Remove any leading/trailing whitespaces
                workspace_directory = file.read().strip()
            # Split the string by spaces and select the last element
            workspace_directory = workspace_directory.split()[-1]
            return workspace_directory

        return None  # Return None if the hidden folder or the workspace file is not found

    def convert(self, file_path):

        # Get the base name of the file without the extension
        filename = os.path.splitext(os.path.basename(file_path))[0]
        conPath = os.path.dirname(file_path)

        # Check if the file is not empty
        if os.path.getsize(file_path) > 0:
            # Get the absolute path of the current script's directory
            script_dir = os.path.dirname(os.path.abspath(__file__))

            # Define the relative path to parser.py from the current script's directory
            # Check the current operating system
            if os.name == 'nt':  # Windows
                relative_parser_path = "LTSpiceToKiCadConverter/src/Windows"
            else:
```

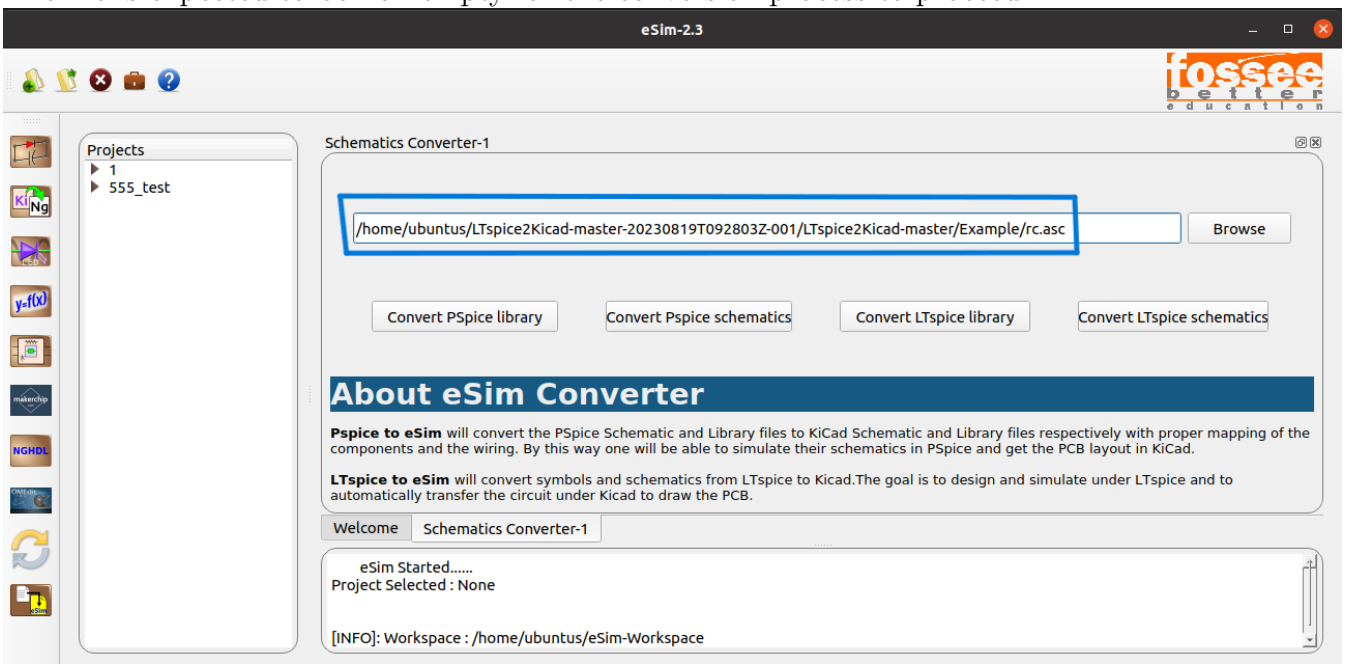
```
            relative_parser_path = "LTSpiceToKiCadConverter/src/Ubuntu"

        # Construct the full path to parser.py
        parser_path = os.path.join(script_dir, relative_parser_path)

        command = f"cd {parser_path} && python3 sch_LTspice2Kicad.py {file_path}"
        try:
            subprocess.run(command, shell=True, check=True)
            # Message box with the conversion success message
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Information)
            msg_box.setWindowTitle("Conversion Successful")
            newFile = str(conPath + "/LTspice_" + filename)
            workspace_directory = self.get_workspace_directory()
            if workspace_directory:
                    print(f"Workspace directory found: {workspace_directory}")
                    merge_copytree(newFile, workspace_directory, filename)
                    msg_box.setText(f"The file has been converted successfully.
                    Saved in {workspace_directory}.  Open the Project manually.")
                    print("File added under the project explorer.")
            else:
                    print("Workspace directory not found.")
            result = msg_box.exec_()
            print("Conversion of LTspice to eSim schematic Successful")


        except subprocess.CalledProcessError as e:
            print("Error:", e)
    else:
        print("File is empty. Cannot perform conversion.")
        # A message box indicating that the file is empty
        msg_box = QMessageBox()
        msg_box.setIcon(QMessageBox.Warning)
        msg_box.setWindowTitle("Empty File")
        msg_box.setText("The selected file is empty. Conversion cannot be performed.")
        msg_box.setStandardButtons(QMessageBox.Ok)
        msg_box.exec_()

def upload_file_LTspice(self, file_path):
    if file_path:
        # Check if the file path contains spaces
        if ' ' in file_path:
            # Show a message box indicating that spaces are not allowed
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Warning)
            msg_box.setWindowTitle("Invalid File Path")
            msg_box.setText("Spaces are not allowed in the file path.")
            msg_box.setStandardButtons(QMessageBox.Ok)
            msg_box.exec_()
            return
```

```
            if ".asc" in file_path:
                print(file_path)
                self.convert(file_path)
            else:
                msg_box = QMessageBox()
                msg_box.setIcon(QMessageBox.Warning)
                msg_box.setWindowTitle("Invalid File Path")
                msg_box.setText("Only .asc file can be converted.")
                msg_box.setStandardButtons(QMessageBox.Ok)
                msg_box.exec_()
                return

        else:
            print("No file selected.")

            # Message box indicating that no file is selected
            msg_box = QMessageBox()
            msg_box.setIcon(QMessageBox.Warning)
            msg_box.setWindowTitle("No File Selected")
            msg_box.setText("Please select a file before uploading.")
            msg_box.setStandardButtons(QMessageBox.Ok)
            msg_box.exec_()

def find_workspace_directory(target_directory_name):
    for root, dirs, files in os.walk("/"):
        if target_directory_name in dirs or target_directory_name in files:
            return os.path.join(root, target_directory_name)
    return None  # Return None if the directory is not found

def merge_copytree(src, dst, filename):
    if not os.path.exists(dst):
        os.makedirs(dst)

    folder_path = f"{dst}/LTspice_{filename}" # Folder to be created in eSim-Workspace

    # Create the folder
    try:
        os.makedirs(folder_path)
        print(f"Folder created at {folder_path}")
    except OSError as error:
        print(f"Folder creation failed: {error}")

    for item in os.listdir(src):
        src_item = os.path.join(src, item)
        dst_item = os.path.join(folder_path, item)

        if os.path.isdir(src_item):
            merge_copytree(src_item, dst_item)
        else:
```

```
if not os.path.exists(dst_item) or os.stat(src_item).st_mtime
> os.stat(dst_item).st_mtime:
    shutil.copy2(src_item, dst_item)
```

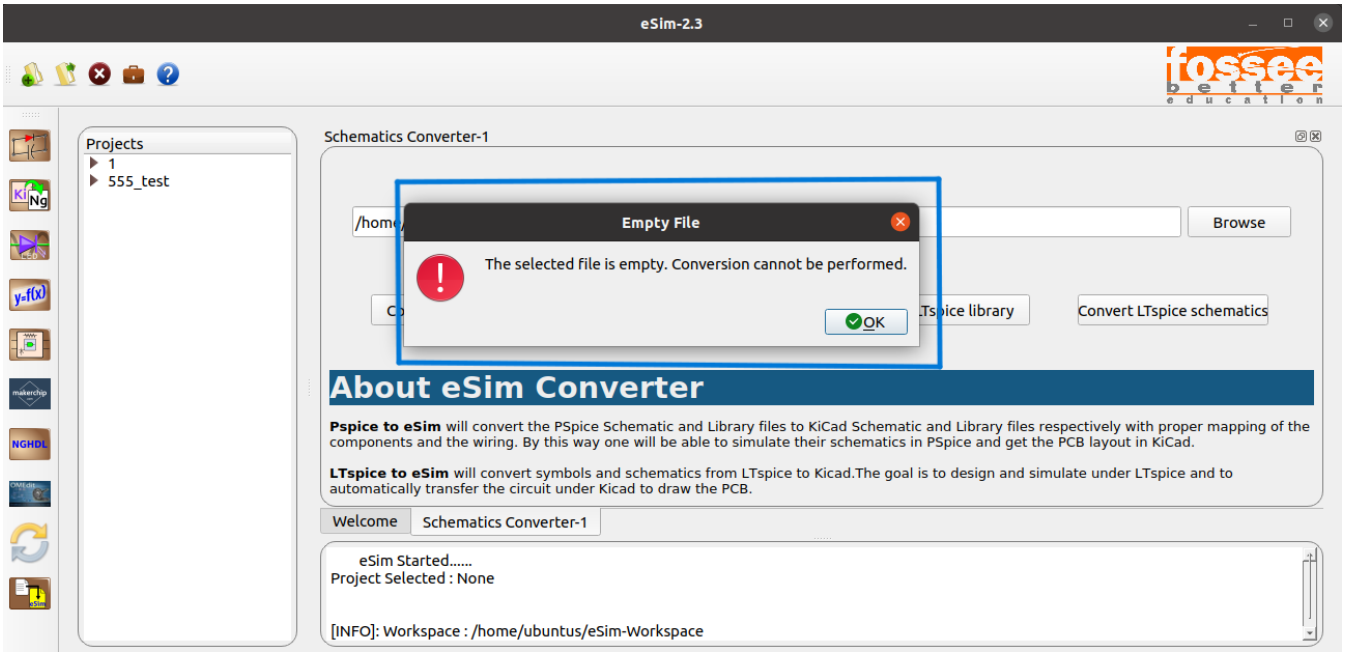
### 5.2.2 Input and Output

**Input:**
The primary input for the LTspiceConverter class is the file path to an LTspice schematic file (with a ".asc" extension), typically provided through user interaction in a GUI.
The file is expected to be non-empty for the conversion process to proceed.

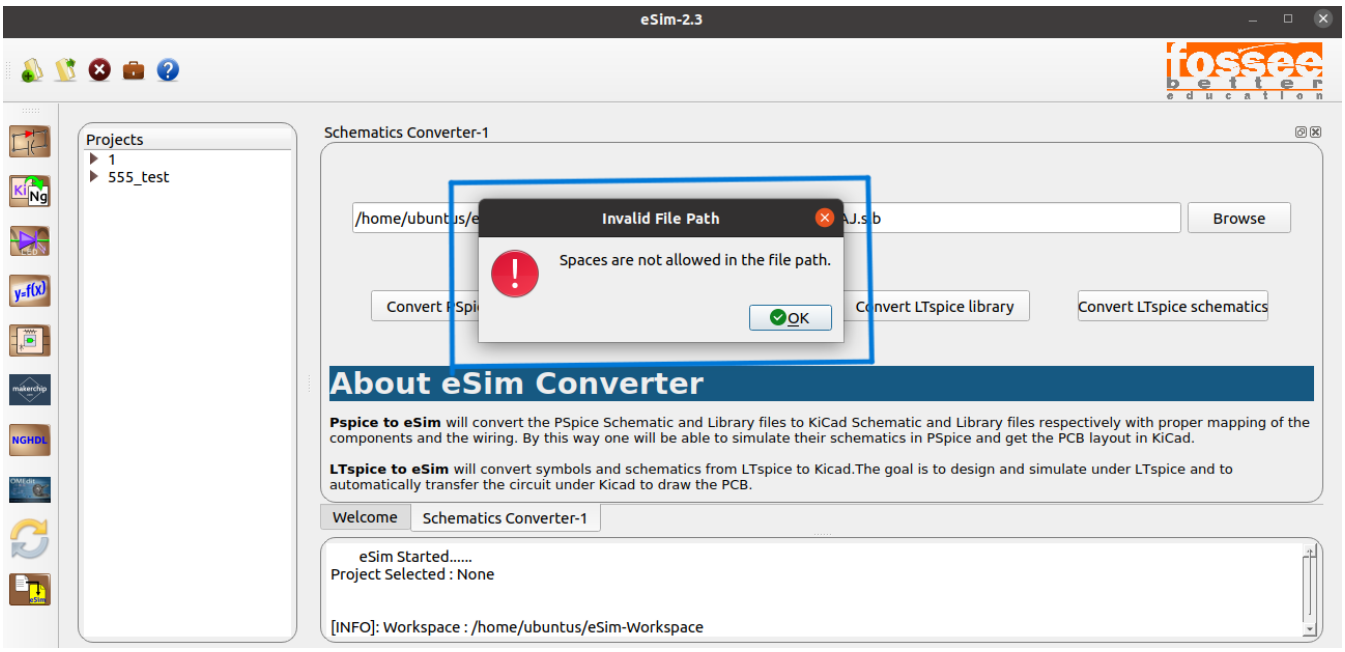



**Output:**
The code performs the following checks and generates outputs accordingly:

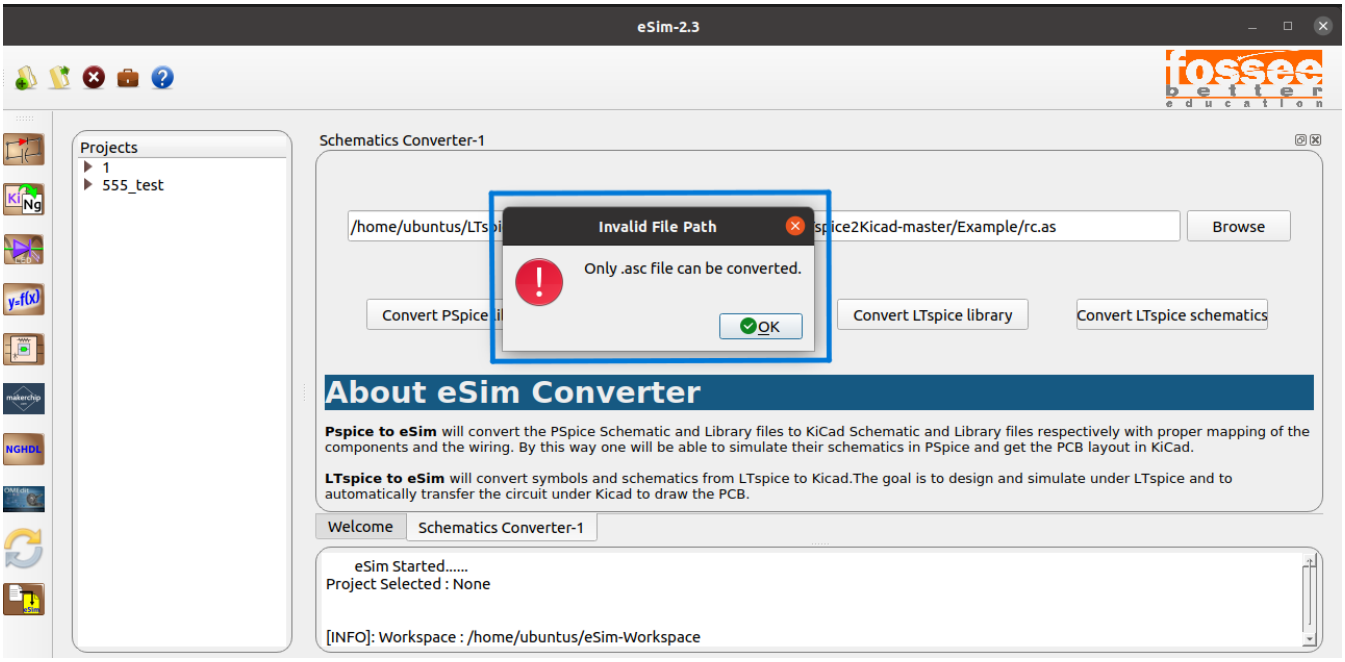
**File Empty Check:** If the specified LTSpice schematic file is empty:
**Output:** Display a QMessageBox warning stating "The selected file is empty. Conversion cannot be performed."




**File Path Check:** If there are spaces in the file path:
**Output:** Display a QMessageBox warning stating "Spaces are not allowed in the file path."



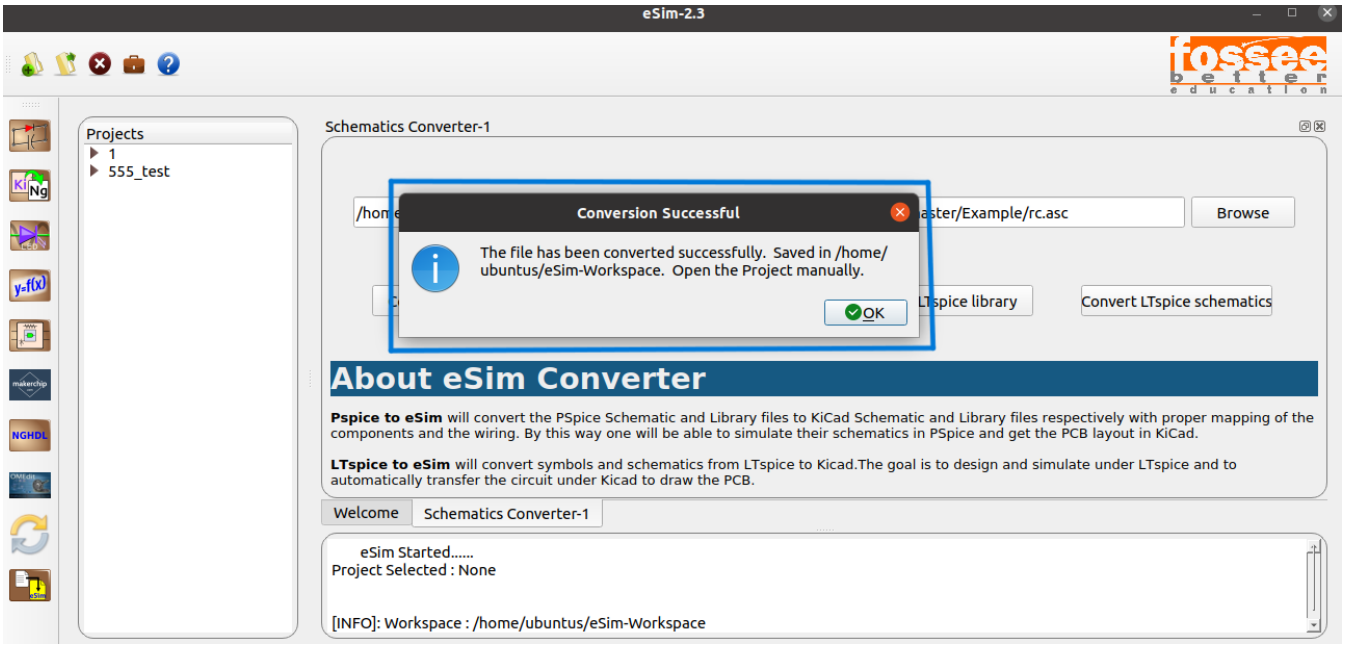

**File Type Check:** If the file does not have a ". asc" extension:
**Output:** Display a QMessageBox warning stating "Only . asc file can be converted."

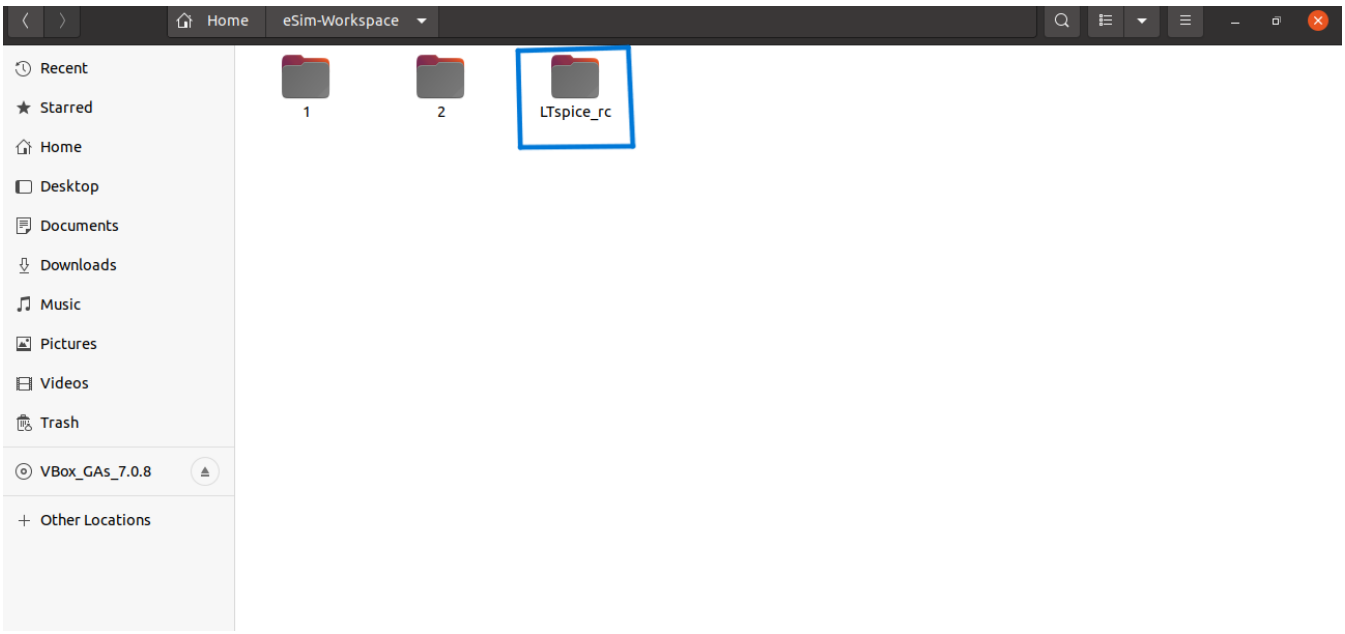



**Conversion:** If the above checks pass and the file is non-empty, has no spaces in the path, and has a ".asc" extension:
Execute an external script (sch_LTspice2Kicad.py) for conversion.

**<u>If the conversion is successful:</u>**

**Output:** Display a QMessageBox information stating "The file has been converted successfully." The converted file is then added under the workspace if the workspace directory is found.






**If an error occurs during conversion:**
**Output:** Print an error message to the console.

The class checks for various conditions (empty file, spaces in the file path, file type) before initiating the conversion, ensuring a smooth and error-free process.

Utilizes QMessageBox for user-friendly notifications, providing clear feedback about the status of the conversion process.
If the conversion is successful, the code attempts to add the converted file under the workspace directory, enhancing user convenience. Incorporates error handling to print detailed error messages to the console in case of conversion failures.

# Chapter 6

# Other Tasks And Challenges

Crash eSim on opening/editing a schematic without any project selection. 241

## 6.1 Subcircuit Builder Method modified to fix #241

In the Subcircuit Builder added a conditional statement to handle the case when a Project is not selected (projDir is None).

### 6.1.1 Approach

Adding a conditional statement to handle the case when projDir is None raising an appropriate error. This will ensure the availability of the project directory when opening or editing a schematic.

## 6.2 LtspiceLibConverter.py

Here lib_ltspice2kicad.py was converting a folder with multiple .asy files.
Thus, had to change lib_ltspice2kicad.py to convert a single .asy file.

### 6.2.1 Code (lib_ltspice2kicad.py)

```
import sys,re,os,codecs

def find_all(a_str, sub):
    start = 0
    while True:
        start = a_str.find(sub, start)
        if start == -1:
            return
        yield start
        start += len(sub)

asy_file = sys.argv[1]

directory = os.path.dirname(asy_file)
comp = [os.path.basename(asy_file)]

indir = directory.split("/")
```

```
out_file = "LTspice_" + indir[len(indir) - 1] + ".lib"
outfl = codecs.open(out_file, "w")
outfl.write("EESchema-LIBRARY Version 2.3\n#encoding utf-8\n#\n")

for component in comp:
    print(component)
    in_file = directory + "/" + component

    infl = codecs.open(in_file, "r")
    lines = infl.readlines()
    infl.close()

    drw_lin = list()
    pin_pos = []
    pin_orient = []
    pin_justif = []
    pin_name = []
    pin_order = []
    pin_off = []

    Value = "Value"
    Value_XY = "0 0"
    Value_orient = "H"
    Value_justif = "L"
    Prefix = ""
    Prefix_XY = "0 0"
    Prefix_orient = "H"
    Prefix_justif = "L"
    Description = ""
    SpiceModel = ""

    for line1 in lines:
        line1 = line1.rstrip('\n')
        line1 = line1.rstrip('\r')

        spc = list(find_all(line1, " "))
        if re.match(r"^SYMATTR Prefix *", line1) is not None:
            Prefix = line1[15:]
        if re.match(r"^WINDOW 0 *", line1) is not None:
            Prefix_XY = str(int(3.125 * int(line1[spc[1]:spc[2]])))
            + " " + str(int(-3.125 * int(line1[spc[2]:spc[3]])))
            Prefix_orient = "H"
            Prefix_justif = line1[spc[3] + 1:spc[3] + 2]
            if Prefix_justif == "V":
                Prefix_orient = "V"
                Prefix_justif = line1[spc[3] + 2:spc[3] + 3]
        if re.match(r"^SYMATTR Value *", line1) is not None:
            Value = line1[14:]
        if re.match(r"^SYMATTR Value2 *", line1) is not None:
            Value = line1[15:]
```

```
    if re.match(r"^WINDOW 3 *", line1) is not None:
        Value_XY = str(int(3.125 * int(line1[spc[1]:spc[2]])))
        + " " + str(int(-3.125 * int(line1[spc[2]:spc[3]])))
        Value_orient = "H"
        Value_justif = line1[spc[3] + 1:spc[3] + 2]
        if Value_justif == "V":
            Value_orient = "V"
            Value_justif = line1[spc[3] + 2:spc[3] + 3]
    if re.match(r"^SYMATTR Description *", line1) is not None:
        Description = line1[19:]
    if re.match(r"^SYMATTR SpiceModel *", line1) is not None:
        SpiceModel = line1[18:]

    if re.match(r"^LINE *", line1) is not None:
        if len(spc) == 5:
            drw_lin.append("P 2 0 0 0 " + str(
                int(3.125 * int(line1[spc[1]:spc[2]]))) + " " + str(
                int(-3.125 * int(line1[spc[2]:spc[3]]))) + " " + str(
                int(3.125 * int(line1[spc[3]:spc[4]]))) + " " + str(
                int(-3.125 * int(line1[spc[4]:]))))
        else:
            drw_lin.append("P 2 0 0 0 " + str(
                int(3.125 * int(line1[spc[1]:spc[2]]))) + " " + str(
                int(-3.125 * int(line1[spc[2]:spc[3]]))) + " " + str(
                int(3.125 * int(line1[spc[3]:spc[4]]))) + " " + str(
                int(-3.125 * int(line1[spc[4]:spc[5]]))))


if Description != "":
    outfl.write("#   " + component[0:len(component) - 4] + "\n")
    outfl.write("# " + Description + "\n")
    outfl.write("# SpiceModel : " + SpiceModel + "\n")
    outfl.write("#\n")
    if (Prefix == "B" or Prefix == "E" or Prefix == "F" or
    Prefix == "G" or Prefix == "H" or Prefix == "I" or Prefix == "V"):
        Pow = "P"
    else:
        Pow = "N"
    if (Prefix == "X" or Prefix == "U"):
        ViewPin = "Y"
    else:
        ViewPin = "N"
    if (component.find("voltage") != -1 or component.find("current") != -1):
        outfl.write("DEF " + component[0:len(component) - 4] + " "
        + Prefix + " 0 1 Y Y 1 F N\n")
    else:
        outfl.write("DEF " + component[0:len(component) - 4] + " "
        + Prefix + " 0 1 N " + ViewPin + " 1 F " + Pow + "\n")
    if ((Prefix_justif == "B") or (Prefix_justif == "T")):
        outfl.write("F0 \"" + Prefix + "\" " + Prefix_XY + " 50 "
```

```
            + Prefix_orient + " V C " + Prefix_justif + "NN\n")
        else:
            outfl.write("F0 \"" + Prefix + "\" " + Prefix_XY + " 50 "
            + Prefix_orient + " V " + Prefix_justif + " CNN\n")

        if ((Value_justif == "B") or (Value_justif == "T")):
           outfl.write("F1 \"" + component[0:len(component)-4] + "\" "
           + Value_XY + " 50 " + Value_orient + " V C " + Value_justif + "NN\n")
        else :
           outfl.write("F1 \"" + component[0:len(component)-4] + "\" "
           + Value_XY + " 50 " + Value_orient + " V " + Value_justif + " CNN\n")

        # the value is transferd to F5 instead of F1 because
        #F1 text should be the component name
        if Value != "Value" :
            if ((Value_justif == "B") or (Value_justif == "T")):
               outfl.write("F5 \"" + Value + "\" " + Value_XY + " 50 "
               + Value_orient + " I C " + Value_justif + "NN\n")
            else :
               outfl.write("F5 \"" + Value + "\" " + Value_XY + " 50 "
               + Value_orient + " I " + Value_justif + " CNN\n")

            if (Pow=="N") : outfl.write("$FPLIST\n " + Prefix + "_*\n$ENDFPLIST\n")

        #DRAWINGS and PINS
        outfl.write("DRAW\n")
        for i in range(0,len(drw_lin)) :
           outfl.write(drw_lin[i] + "\n")

        for i in range(0,len(pin_name)) :
            pinjustif = pin_justif[i]
            if pin_justif[i] == "L" : pinjustif = "R"
            if pin_justif[i] == "R" : pinjustif = "L"
            if pin_justif[i] == "T" : pinjustif = "D"
            if pin_justif[i] == "B" : pinjustif = "U"
            outfl.write("X " + pin_name[i].replace(" ","")
            + " " + pin_order[i] + " " + pin_pos[i] + " 0 "
            + pinjustif + " 50 50 1 1 U\n")

        outfl.write("ENDDRAW\nENDDEF\n#\n")

outfl.close()
```