Quantensimulations-framework – QSFW

Wiki zur GUI

HOCHSCHULE NORDHAUSEN
University of Applied Sciences

Autoren:

David Bochmann

Laura Komma

Betreuer:

Prof. Dr. Frank-Michael Dittes

Datum: 20. November 2023

Inhaltsverzeichnis

1. Voraussetzungen

Diese Anwendung dient der einfachen Erstellung von Abfolgen von Operationen (Gates) auf einer beliebigen Anzahl von Qubits, dies wird im folgenden Text als „Schaltung“ oder „Schaltplan“ bezeichnet. Die Anwendung bietet eine einfache und benutzerfreundliche Oberfläche, sodass sehr einfach Schaltungen entworfen und berechnet werden können.

Die Anwendung wurde in Python entwickelt. Um die Anwendung möglichst fehlerfrei starten und anwenden zu können, ist mindestens die Version *3.9+* empfehlenswert.
Für die Benutzerbobfläche ist die Libary *„pyqt5“* verwendet worden. Für den Simulator werden die Libaries *„numpy“* und *„shunting-yard“* benötigt. Wenn diese nicht vorhanden sind, können diese entsprechend über *„pip“* nachinstalliert werden.

2. Einstieg in die Anwendung

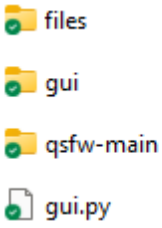
Nach dem Entpacken des zip-Files ist folgende Ordnerstruktur zu vorzufinden.

Unter dem Verzeichnis *„files“* werden alle mit der Benutzeroberfläche erstellten Schaltungen in Form einer Textdatei abgespeichert. Diese können mittels Button oder via Terminal an den Simulator übergeben und schließlich berechnet werden.
In dem Verzeichnis *„gui“* befinden sich alle zusätzlichen Dateien der Oberfläche, welche bspw. für die Verarbeitung der Eingaben oder die Erstellung der Textdateien notwendig sind.

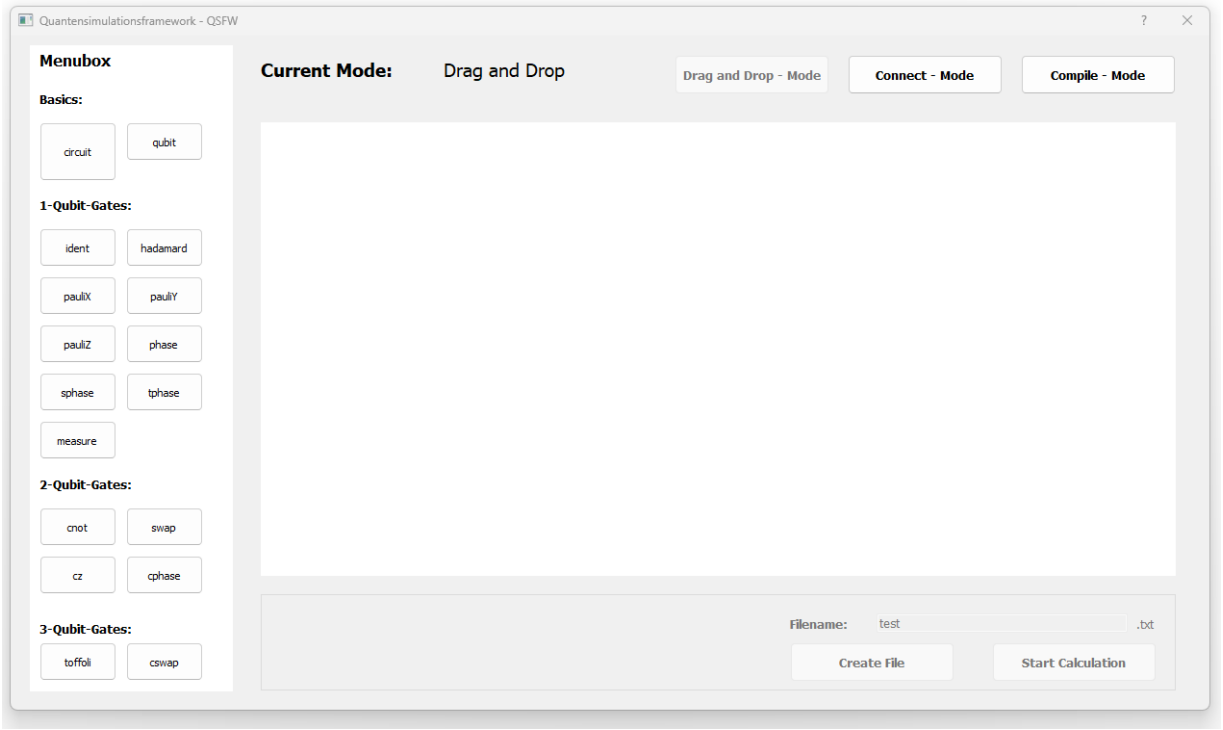
Unter folgendem Link ist, kann der Simulator selbst heruntergeladen werden.

https://github.com/jonasjelonek/qsfw

Dies ist notwendig, um die entsprechende Schaltung auch berechnen zu können. Nach dem Download und dem Entpacken ist der Ordner diesem Verzeichnis hinzuzufügen, sodass das Gesamtverzeichnis folgende Struktur aufweist:



Die Datei *„gui.py“* ist der Einstiegspunkt für die Anwendung. Diese Datei kann entweder im Terminal oder mittels bevorzugten Entwicklungseditor, wie bspw. *Spyder* oder *VS Code,* gestartet werden. Im Anschluss daran öffnet sich eine graphische Oberfläche, die wie folgt aussieht.

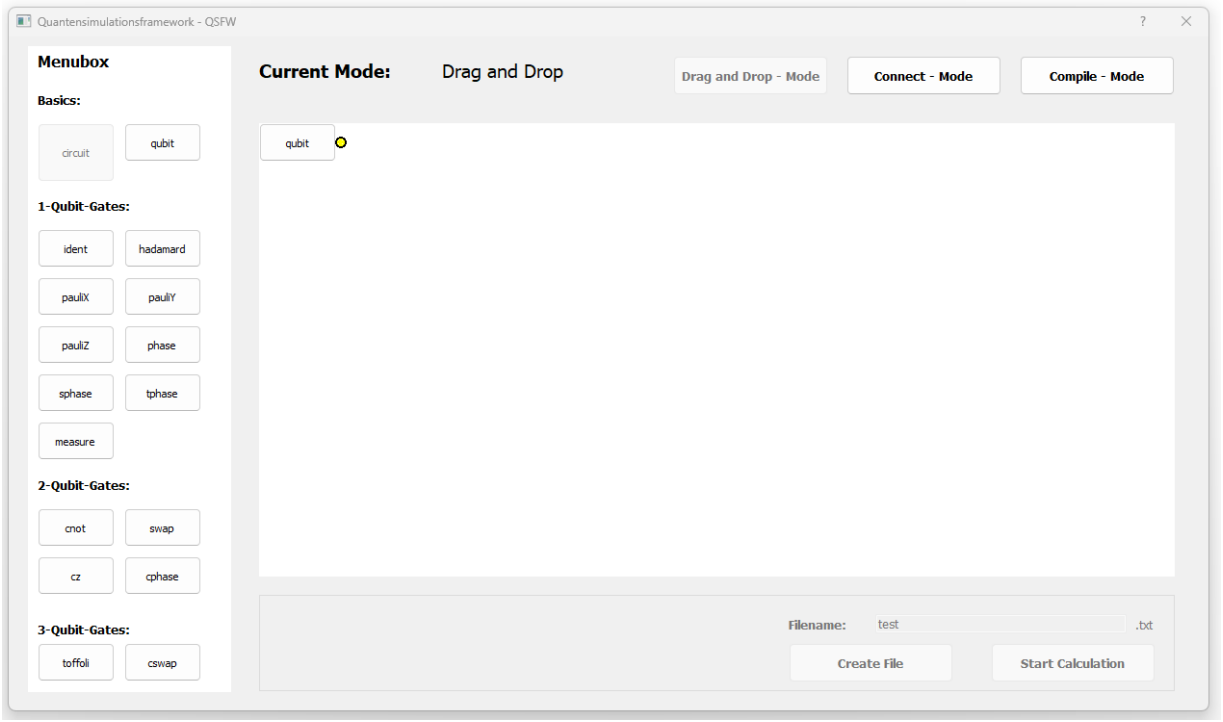


Auf der linken Seite ist eine Menübox zu sehen, welche alle möglichen Funktionen und Bestandteile einer Schaltung enthält. Die Voraussetzung für einen funktionstüchtigen Schaltplan ist dabei das vorhanden sein von mindestens einem Basic-Baustein (*„circuit“* oder *„qubit“*).
Im oberen Teil der Oberfläche ist eine „Mode“-Anzeige zu sehen, welchen den aktuellen Modus beschreibt, in dem sich die Anwendung befindet. Über die entsprechenden Buttons, welche rechts neben der Aufschrift sind, kann dieser gewechselt werden.
In der Mitte ist eine große weiße Fläche zu sehen. In dieser werden die Schaltungen aufgebaut und dargestellt.
Unterhalb der weißen Fläche befindet sich ein Kästchen, dieses wird nur im *„Compile“-Mode* freigeschaltet. Hier kann zum einen die entsprechende Textdatei erstellt und an den Simulator übergeben werden, zum anderen werden hier aber auch Statusmeldungen angezeigt.

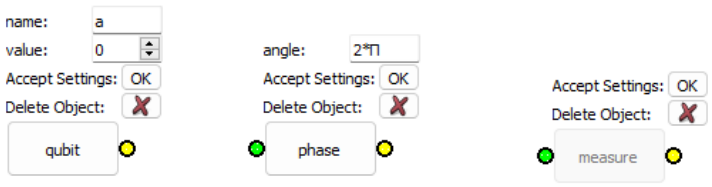
3. Verwendung

3.1 „Drag and Drop“-Mode

Beim Anklicken eines Bausteins der Menübox erscheint dieser auf der weißen Schaltoberfläche. Dieser kann via *„Drag and Drop“*-Verfahren beliebig auf der Oberfläche verschoben und positioniert werden, sofern der aktuelle Modus *„Drag and Drop“* ist.



Zudem können in diesem Zustand auch Einstellungen an den jeweiligen Bausteinen vorgenommen werden. Durch einfaches Anklicken des jeweiligen Elementes erscheint das jeweilige Einstellungsmenü. Je nach Art des Bausteins können die Bezeichnungen, Werte oder Winkel angepasst werden. Außerdem ist es möglich, in diesem Einstellungsmenü ein Element wieder zu löschen.

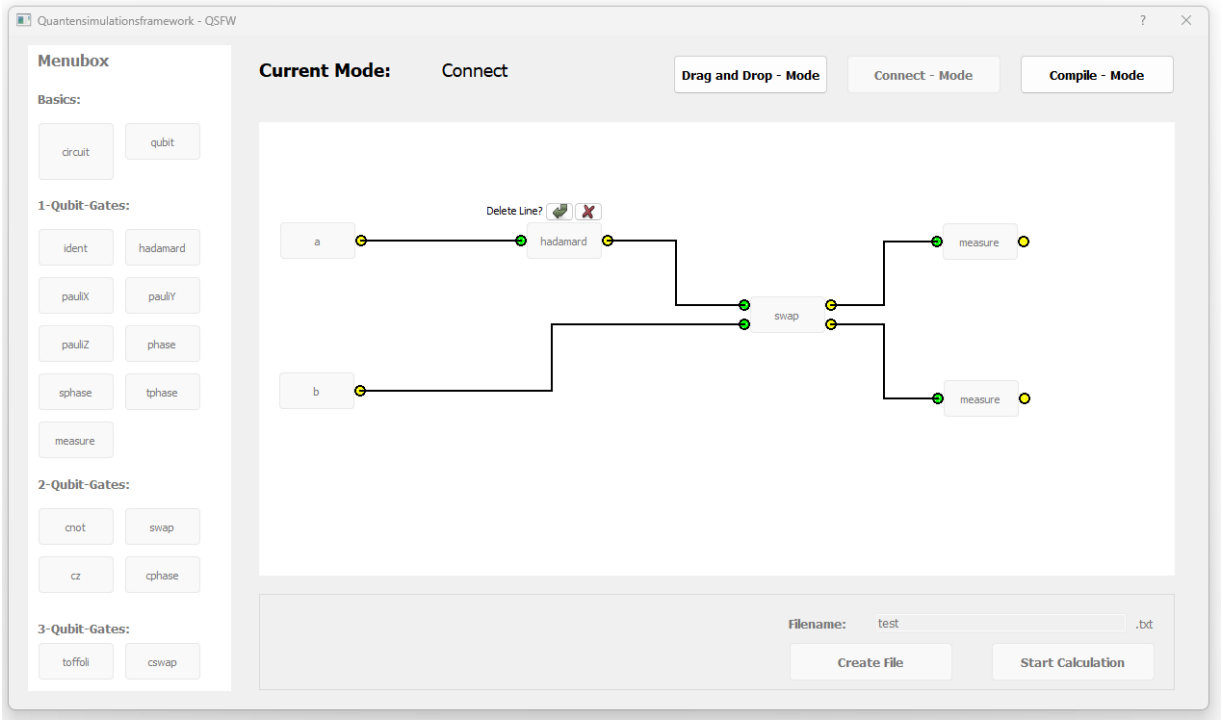


Wie bereits erwähnt, müssen in jeder Schaltung immer Basic-Elemente enthalten sein, damit die fehlerfreie Datei-Erstellung und anschließende Berechnung funktionieren kann. Dabei ist es möglich **n viele** *„qubit“* oder **einen** *„circuit“* einzufügen.
Bei den Qubits können die Bezeichnung und der Startwert eingestellt werden. Beim Circuit können nur **n viele** Ausgänge eingestellt werden (max. 5 Ausgänge), wobei jeder Ausgang ein eigenständiges Qubits mit dem Startwert **|0>** darstellt.

3.2 „Connect“-Mode

Wenn alle Elemente eingestellt und auf der Oberfläche positioniert sind, kann in den *„Connect“-Mode* gewechselt werden. Hier werden nun die Elemente deaktiviert und vor dem Verschieben geschützt. Durch Klicken auf den Output eines Elementes (gelb, rechtsseitig) und den Input (grün, linksseitig) eines anderen Elementes werden diese beiden Elemente miteinander verbunden und in Zusammenhang gebracht. Somit kann der „Schaltplan“ erstellt werden. Dabei sollte auch darauf geachtet werden, dass zuerst der Output und dann der Input angeklickt wird, da es sonst unter Umständen zu Fehlern kommt.

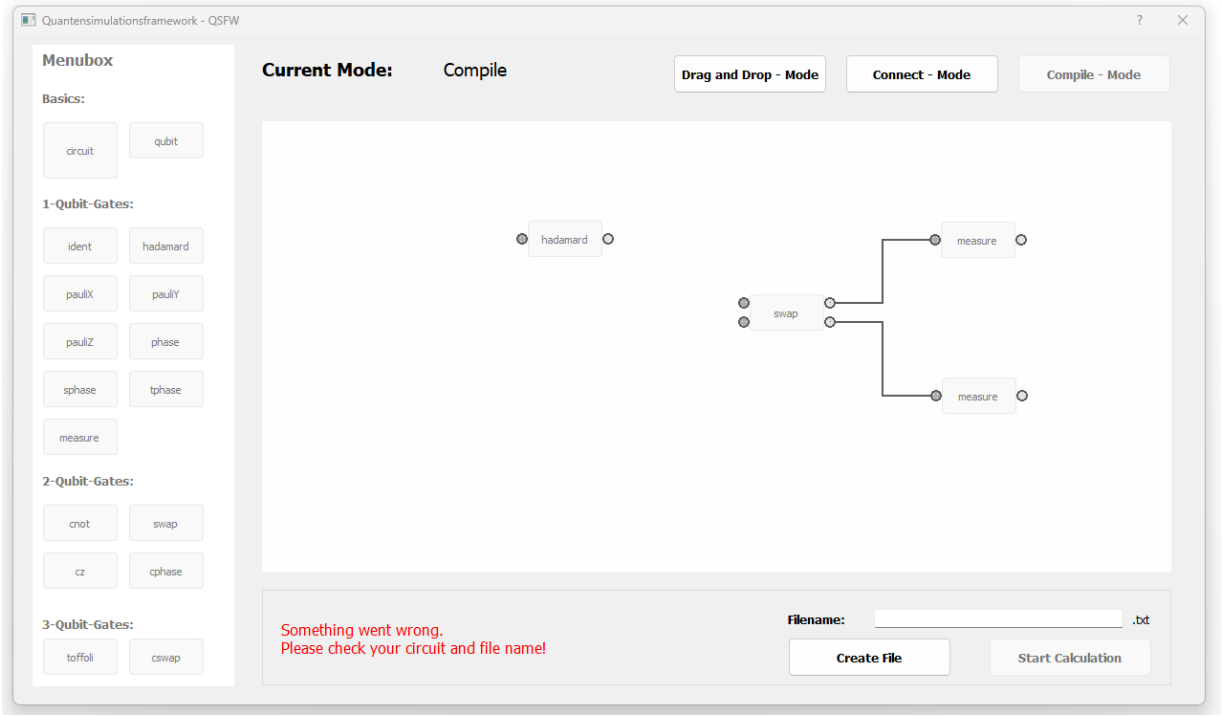
Wenn eine falsche Verbindung gemacht wurde, kann durch den Rechtsklick auf den entsprechenden Input eine Abfrage geöffnet werden, über welche die fehlerhafte Linie wieder gelöscht wird.



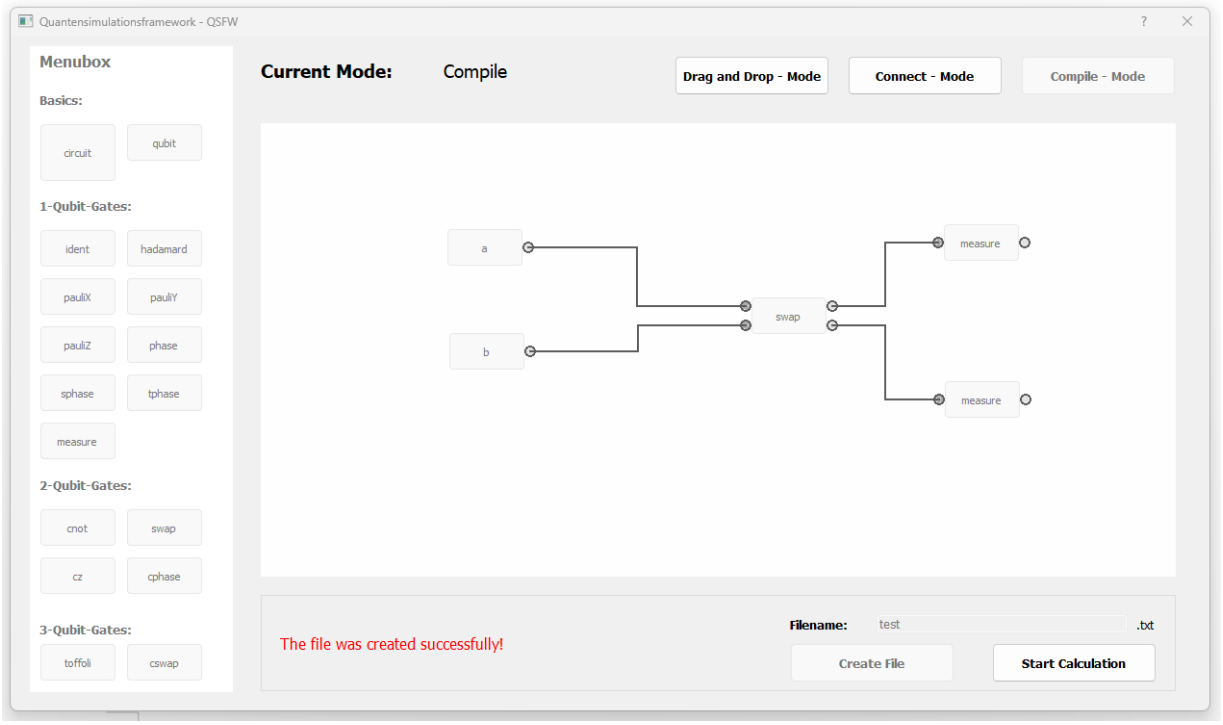
Wenn nach dem Wechsel in diesen Modus noch ein Element fehlt oder die Einstellungen bzw. Anordnung nicht den Wünschen entspricht, kann noch einmal in den *„Drag and Drop“* Modus zurück gegangen werden. Auch dann bleiben die Elemente miteinander verbunden, es sei denn, es wird ein Element aus der Kette gelöscht

3.3 „Compile“-Mode

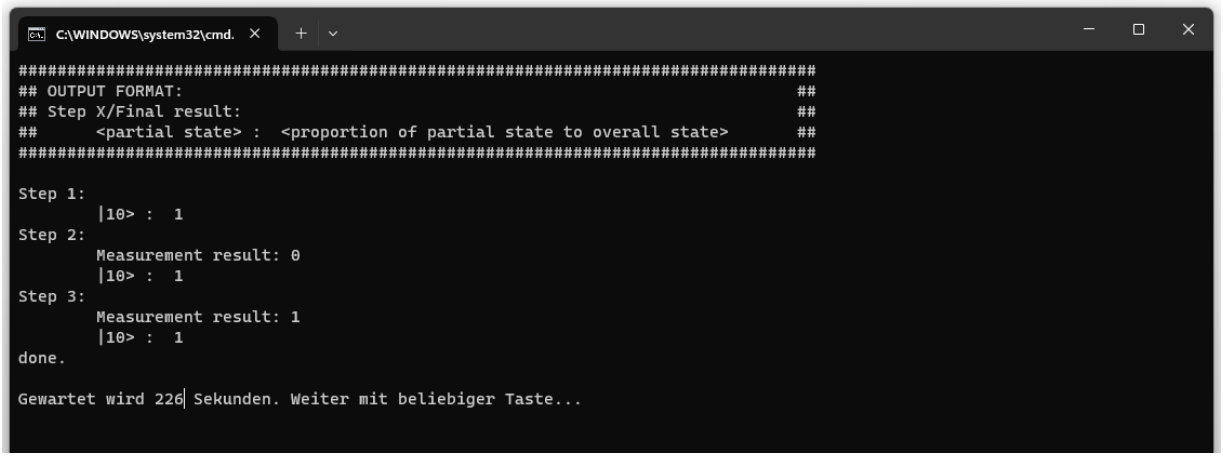
Wenn alle Elemente nach Belieben miteinander verknüpft wurden, kann in den *„Compile“-Mode* gewechselt werden. Hier wird nun der untere Teil der Anwendung aktiviert bzw. freigeschaltet. Zunächst kann eine eineindeutige Bezeichnung für die zu erstellende Textdatei vergeben werden. Standardmäßig wird hier die Bezeichnung *„test“* hinterlegt. Die entsprechende Dateiendung „.txt“ muss nicht mit eingegeben werden, dies wird im Quellcode vorgenommen.
Im Anschluss kann über den *„Create File“-Button* die Datei zu der entworfenen Schaltung erstellt werden. Nun erscheint auf der linken Seite neben diesem Button eine Meldung. Wenn die Erstellung erfolgreich war, wird dies hier angegebenen und der *„Start Calculation“-Button* ist aktiviert.
Wenn im Verlauf der Datei-Erstellung ein Fehler aufgetreten ist, bleibt dieser Button deaktiviert und die Meldung weißt darauf hin.



Bei der Auswahl der Datei-Bezeichnungen ist darauf zu achten, dass keine Bezeichnungen doppelt verwendet werden sollten. Wenn im Unterverzeichnis *„files“* bereits eine Datei mit der selben Bezeichnung existiert, wird diese durch die neue Datei einfach überschrieben. Der Nutzer wird darüber allerdings nicht informiert. Unter Umständen sollte also eine Sicherheitskopie des *„files“*-Verzeichnises gemacht werden.



Nach dem erfolgreichen Erstellen der Datei kann die Berechnung mit dem Simulator über die Oberfläche gestartet werden. Sobald dazu auf den *„Start Calculation“-Button* geklickt wird, öffnet sich unter Windows ein neues Terminal und unter Linux werden die Berechnungsschritte im eigentlichen Terminal, aus dem die Anwendung gestartet wurde, angezeigt.
Im Betriebssystem Windows ist das Terminal max. 5 min geöffnet und kann auch durch Tastendruck vorzeitig geschlossen werden.



Sobald man zur Oberfläche zurückkehrt, kann die Berechnung erneut durchgeführt werden.
Wenn der Modus noch einmal gewechselt wird und somit Änderungen vorgenommen wurden, ist ein Neu-Erstellen der Datei notwendig, bevor die Berechnung wieder gestartet werden kann.

4. Test der Software

Das Programm wurde bisher in den Betriebssystemen *„Windows 10“, „Windows 11“* und *„Linux (Ubuntu)“* mit einfachen Beispielen getestet. Es erstellt dabei zuverlässig nach der gewünschten Syntax die Dateien, sofern die geforderten Regeln beachtet wurden.
Dennoch kommt auch diese Oberfläche an diverse Grenzen. Es sollten nicht zu viele Verschachtelungen vorgenommen werden, da sonst ein Fehler in der Reihenfolge der Gatter auftreten kann. Außerdem sollte darauf geachtet werden, dass vom Output eines Elementes (gelb, rechtsseitig) zum Input (grün, linksseitig) eines anderen Elementes verbunden wird, da sonst Fehler austreten können. Zudem sollten nicht zu viele Aktionen gleichzeitig auf den Elementen ausgeführt werden, wie bspw. Umbenennen, Verschieben, Verbinden, etc., da auch hier Fehler bei der Erkennung des aktuell aktiven Elements und den dazu gehörenden Aktionen auftreten können.

5. Grundaufbau des Programmcodes

Die Oberfläche selbst setzt sich aus insgesamt vier Python-Dateien, einer UI-Datei und einem Bild zusammen.
Den Einstiegspunkt bildet dabei die *„gui.py“*. Hier befindet sich die main-Funktion sowie die gesamte Klasse *„GUI“* welche für die Darstellung der Hauptoberfläche und Verarbeitung der Nutzereingaben zuständig ist.
Im Verzeichnis *„gui“* sind die anderen Dateien zu finden. In der Datei *„dragNdropObject.py“* ist eine Klasse mit der selbigen Bezeichnung. Alle Elemente, die für eine Schaltung erstellt werden, sind Typ dieser Klasse. Diese Objekte besitzen entsprechende Informationen zu den Elementen wie bspw. die zugehörige Kategorie (bspw. *„qubit“* oder *„normalFunction1“*), die Bezeichnung, den Wert, die Anzahl der Inputs bzw. Outputs sowie Position und Größe. Die entsprechenden Zeichnungen der Punkte sowie das aktualisieren der Linien werden ebenfalls über diese Klasse realisiert.
Die Klasse *„Line“,* welche in der gleichnamigen Python-Datei zu finden ist, kümmert sich um die Verbindungen zweier Elemente. Hierüber können die Linien gezeichnet und die zwei Partner einer Verbindung ermittelt werden.
Die Datei und Klasse *„fileGenerator.py“* ist für die syntaktisch korrekte Erstellung der Textdatei zuständig. Hier wird aus einer Liste, welche im Hauptfenster erstellt wurde, die Datei zusammengestellt, die der Simulator lesen und verarbeiten kann.

Um den Grundaufbau des Hauptfensters einfacher umzusetzen, wurde der von PyQt bereitgestellte Designer verwendet. Mit diesem wurde die hier vorhandene *„qsfw.ui“*-Datei erstellt, welche in der Hauptdatei eingelesen wird. Das in diesem Verzeichnis abgelegte Bild *„quanten.png“* soll als Icon der Software dienen. Leider wird es in dem meisten Betriebssystemen nicht dargestellt.