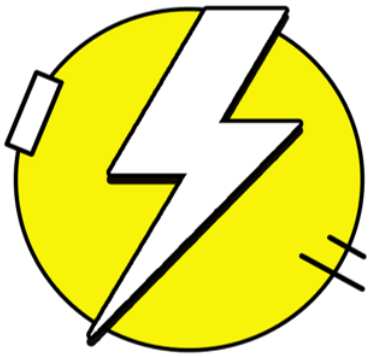
Hochschule Esslingen
University of Applied Sciences

Campus Esslingen Flandernstraße

Projekt Softwaretechnik Sommersemester 2017

BlitzEdit



Christian Gärtner

Matrikelnr.: 752664

chgait01@hs-esslingen.de

David Schick

Matrikelnr.: 750838

dascit12@hs-esslingen.de

Nico Pfaff

Matrikelnr.: 751854

nipfit00@hs-esslingen.de

Marcel Weller

Matrikelnr.: 752369

maweit06@hs-esslingen.de

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Team und Projekt

1.1 Das Team

Das Team besteht aus 4 Mitgliedern aus dem Studiengang Softwaretechnik und Medieninformatik an der Hochschule Esslingen:

Christian Gärtner, Medieninformatik

Nico Pfaff, Medieninformatik

David Schick, Softwaretechnik

Marcel Weller, Softwaretechnik

1.2 Das Projekt

Das Projekt ist Inhalt der Veranstaltung „Projekt Softwaretechnik“ und entsteht im Auftrag und unter der Betreuung der IT-Designers GmbH.

Gegenstand des Projekts ist die Entwicklung eines Editors für elektrische Schaltpläne.

Darunter wird eine grafische Darstellung verstanden, die komplexe Schaltungen vereinfacht und somit zur Planung und Wartung einer solchen dient. Elektrische Bauelemente, wie zum Beispiel Spannungsquellen oder Widerstände, werden durch vordefinierte abstrakte Symbole dargestellt und durch Leitungen logisch miteinander verbunden (s. Abb. 1).

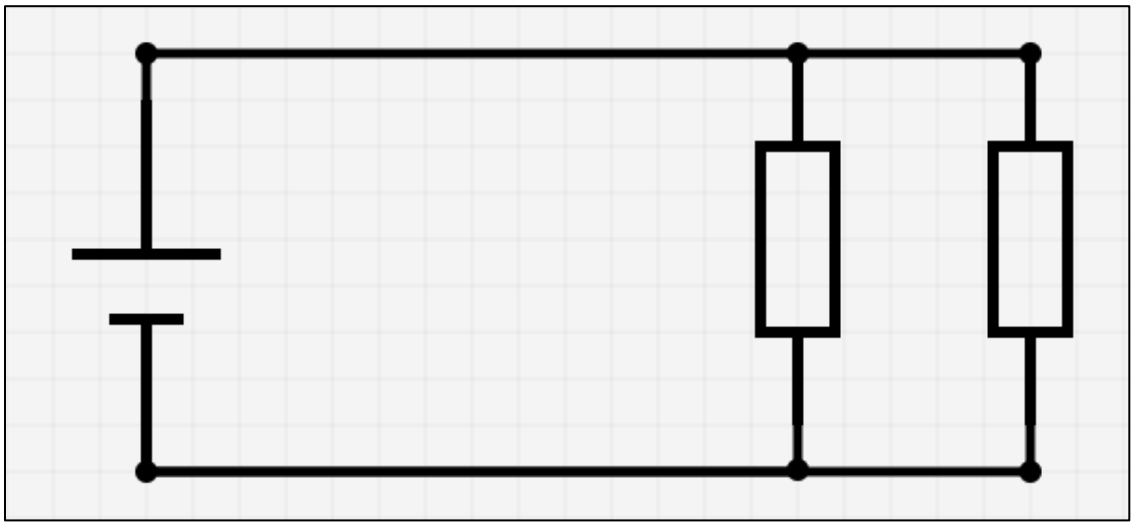


Abbildung 1 - Beispiel eines einfachen Schaltplans (Erstellt in BlitzEdit)

2 Projektmanagement

Da das Team nur aus 4 Mitgliedern besteht, organisieren wir uns nicht gemäß einem vorgegebenen Modell. Entscheidungen werden im Konsens getroffen und jeder hat bei allen Themen gleiches Mitspracherecht.

Gemäß der gewählten Spezialisierung unseres Studiengangs (Medieninformatik oder Softwaretechnik) teilten wir uns in zwei Aufgabenbereiche auf. Der Übergang zwischen den Bereichen ist aber fließend und nicht klar definiert, sodass man auch in beiden Feldern tätig sein kann. Sie dienen also nur als Orientierung.

Zur Verteilung der Aufgaben innerhalb der Bereiche nutzen wir die Onlineplattform „Trello“, in der man Aufgabenkarten erstellen und sich oder anderen Teammitgliedern zuweisen kann. Somit weiß immer jeder Bescheid, was die anderen machen und kann sich dementsprechend anpassen.

„Trello“ konnten wir auch mit „GitHub“ verlinken, unserer Versionierungsplattform, was das Arbeiten mit dieser erleichtert. Auf „GitHub“ wird unser Projekt als Open Source Projekt bereitgestellt. Jedes Teammitglied kann darauf zugreifen und über das Erstellen von sogenannten „Branches“ eine isolierte Entwicklungsumgebung für eine bestimmte Aufgabe erstellen. Nach Fertigstellung der Aufgabe wird sie dem Gesamtsystem hinzugefügt und gilt als abgeschlossen.

3 Marktanalyse

Es besteht bereits ein ausgeprägter Markt für Schaltplaneditoren, welcher jedoch weniger breit gefächert ist und daher nur eine feste Zielgruppe anspricht. Diese setzt sich aus Hobbyanwendern, Studenten oder professionellen Entwicklern in der Industrie zusammen.

Im Folgenden werden die drei auf dem Markt präsenten Editoren „Elwi“, „SPlan“ und „Eagle“ analysiert. „SPlan“ und „Eagle“ sind professionelle, meist in der Industrie angewandte Editoren, während „Elwi“ ein Freeware Tool ist.

Die Editoren unterscheiden sich abhängig vom Preis im Angebot der Funktionen. Die größten Defizite sind bei allen Editoren gleich und lassen sich in der Handhabung und Bedienung erkennen:

Das Bedienfenster ist mit Informationen und Funktionen überladen. Außerdem ist die Oberfläche, sowie die Bedienung veraltet. Moderne Bedienelemente wie Drag & Drop fehlen häufig und lassen damit die Handhabung schwerfällig wirken. Die Bedienung ist nicht intuitiv.

Weiterhin braucht der Anwender im Vergleich zu modernen Anwendungen lange, um sich mit der Anwendung vertraut zu machen, das Bedienkonzept zu verstehen und alle Funktionen zu entdecken.

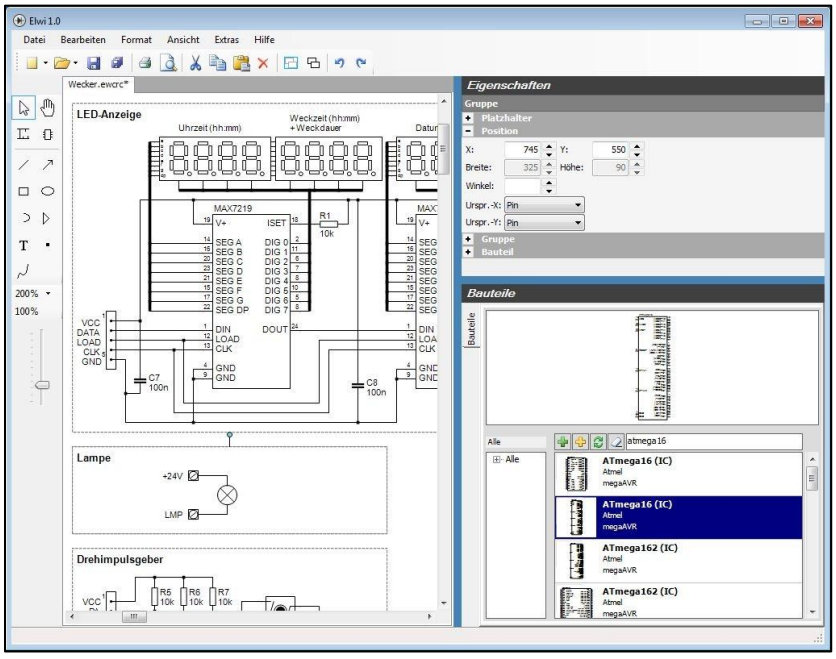


Abbildung 2 - „Elwi“

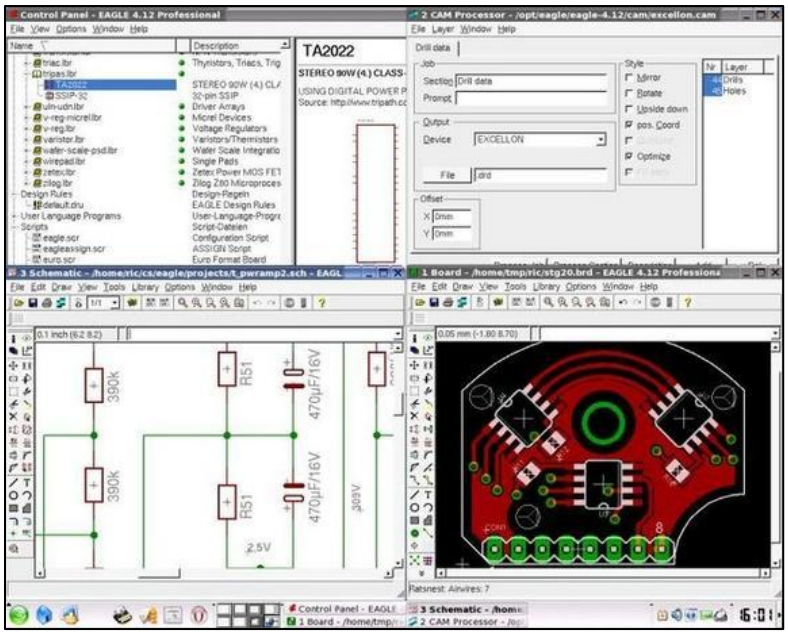


Abbildung 3 - „Eagle"

Um diese Defizite zu verbessern orientiert sich unser Schaltplaneditor „BlitzEdit" an „Microsoft Visio Professional". Dadurch wird dem Nutzer eine moderne, übersichtliche und vor allem intuitive Anwendung zur Verfügung gestellt.

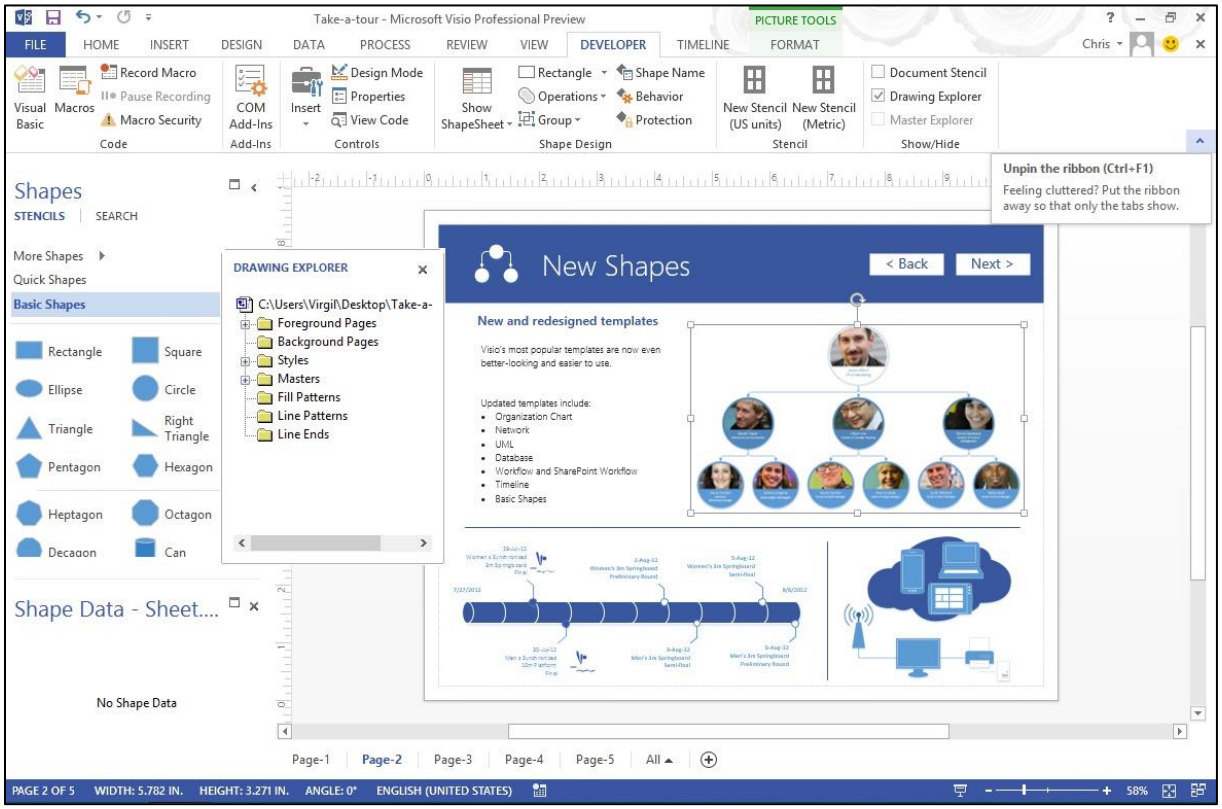


Abbildung 4 - Microsoft Visio Professional

4 Anforderungen

Aus Kundengesprächen und der Marktanalyse ergeben sich folgende Anforderungen an das Produkt:

3.1 Funktionale Anforderungen

Die Funktionalen Anforderungen beschreiben, was das Produkt bewerkstelligen muss.

3.1.1 Schaltplan

- Das System muss es ermöglichen, Schaltpläne zu erstellen, verändern und zu speichern bzw. zu laden.
- Die Schaltpläne bestehen aus Leitungen und über diese verbundenen Bauteile.
- Die Leitungen müssen die Bauteile über rechtwinklig, wenn notwendig auch über 45 Grad, verlaufende, automatisch platzierte Leiterbahnen verbinden.
- Das System muss mehrere Schaltpläne parallel zueinander öffnen und bearbeiten können.

3.1.2 Bauteile

- Sie müssen frei bewegbar sein, Leitungen müssen sinnvoll mitverschoben werden.
- Die Bauteile müssen in 45 Grad Winkeln drehbar sein.
- Man muss sie kopieren und einfügen können (auch über Tabs hinweg).
- Sie verfügen über beliebig viele Connectoren, an denen die Leitungen anknüpfen.
- Jeder Connector hat eine relative Position zum Bauteil.
- Jeder Connector kann verlängert und wieder verkleinert werden.
- Jedes Bauteil besitzt eine Grafik im SVG-Format.

3.1.3 Speichern

- Sowohl die Schaltpläne als auch die Bauteile müssen im XML-Format gespeichert werden.
- Um einen Schaltplan zu speichern, muss:
    - die Position der Bauteile gespeichert werden,
    - die Rotation der Bauteile gespeichert werden,
    - die Verbindungen zwischen den Connectoren der Bauteile gespeichert werden.
- Um ein Bauteil zu speichern, muss:
    - die relative Position der Connectoren gespeichert werden,
    - der Typ des Bauteils benannt werden,
    - die Grafik, die dem Bauteil zugeordnet ist, gespeichert werden.
- Es muss feststellbar sein, ob ein Teil des Schaltplans seit der letzten Sitzung verändert wurde. Ist dies der Fall muss der Benutzer darüber in Kenntnis gesetzt werden.

3.1.4 Laden

- Zur Laufzeit des Systems müssen neue Bauteile erstellt und geladen werden können.
- Bestehende Bauteile müssen durch Änderungen an den XML-Dateien verändert werden können.

3.2 Nicht-Funktionale Anforderungen

Die Nicht-Funktionalen Anforderungen beschreiben, wie das Produkt funktionieren soll.

3.2.1 Design der Anwendung

- Das Design des Produkts soll an das der Microsoft Office Produkte angelehnt sein.
- Die Menüleiste der Anwendung soll eine klare Struktur aufweisen.
- Das Design aller Fenster soll einheitlich gestaltet werden (zum Beispiel Popups).
- Das Anwendungsdesign soll dem Benutzer einen cleanen Look vermitteln.
- Im Editorfeld soll es möglich sein ein Gitternetz anzeigen zu lassen.

3.2.2 Bedienung

- Das Produkt soll über Maus und Tastatur bedienbar sein.
- Die Bedienung der Oberfläche soll intuitiv geschehen.
- Der Benutzer soll die Anwendung aber auch lediglich mit der Maus steuern können.
- Das Produkt soll es dem Benutzer ermöglichen, die meisten bekannten Tastenkürzel benutzen zu können.

3.2.3 Code

- Die Anwendung und der Schaltplan sollen sicher sein vor (unbeabsichtigter) Manipulation durch den Benutzer.
- Die Anwendung soll leicht erweiterbar sein, d.h. neue Bauteile können ohne großen Aufwand hinzugefügt werden.
- Der Programmcode soll verständlich geschrieben werden und gut kommentiert sein.
- Die Wartung von Programmcode soll möglichst einfach sein.

3.2.4 Bedienen der Bauteile

- Die Grafiken von Bauteilen sollen stets gut erkennbar und genormten Vorgaben entsprechen.
- Bauteile sollen per Drag-and-Drop eingefügt werden können.
- Die Auswahl der Bauteile soll intuitiv erfolgen.
- Wählt der Benutzer ein Objekt aus, so wird dieses mittels einer Umrahmung markiert.
- Man soll mehrere Bauteile auf einmal markieren und bearbeiten können.
- Die Leitungen der markierten Bauteile sollen farblich hervorgehoben werden.
- Die Bauteile sollen sich auf Wunsch des Benutzers an einem Raster ausrichten.
- Beim Auswählen eines Connectors soll dieser sich farblich von allen anderen hervorheben. Bei Letzteren soll klar erkennbar sein, ob sie schon mit dem Ausgewählten verbunden sind oder nicht.

3.2.5 Beschriftung

- Die Anwendung soll in deutscher Sprache entwickelt und designed werden.
- Sämtliche Texte der Anwendung sollen in der gleichen Schriftart verfasst sein.
- Im Falle von Fehlermeldungen sollen diese informativ und verständlich verfasst sein.
- Die Beschriftung von Bauteilen soll stets lesbar und deutlich sein. (z.B. nicht verpixelt)

3.2.6 Bibliothek

- Das Programm soll einen Bereich haben, in dem die verfügbaren Bauteile mit Beschriftung dargestellt sind.
- Bauteile sollen in Gruppen, sprich Bibliotheken, organisiert sein.
- Der Benutzer soll Bauteile bestehenden Bibliotheken hinzufügen oder neue Bibliotheken importieren können.

3.2.7 Sonstiges

- In einem Hilfefenster soll der Benutzer Informationen beziehungsweise eine Einführung zu der Bedienung des Programms erhalten.
- In einem About Fenster sollen Informationen über das Entwicklerteam und andere Mitwirkende bereitgestellt werden.

5 Lösungen

5.1 XML als Speicherformat

In diesem Kapitel wird ein sogenannter XML Parser vorgestellt. Um zu verstehen, wie dieser funktioniert, werden simple XML Grundlagen benötigt. Diese kann man sich unter anderem mit weiteren Informationen auf der Tutorial-Seite von w3schools aneignen (https://www.w3schools.com/xml/).

5.1.1 XML Parser

Bei dem verwendeten XML Parser handelt es sich um eine selbstgeschriebene Klasse, die Dateien ausliest und mit Hilfe von Stringalgorithmik analysiert. Dabei wird vor allem Wert auf die Attributwerte der XML Elemente gelegt. Der Parser ist dabei präventiv gestaltet, das heißt bei auftretenden Fehlern im XML Format bricht er in der Regel den Auslesevorgang für das einzelne Element oder aber für die gesamte XML Datei ab. Dies wurde eingebaut um Folgefehler und unvorhersehbares Verhalten des Schaltplaneditors durch falsche Änderungen an Dateien durch den Benutzer zu vermeiden. Als Zusatzfunktion enthält der Parser die Möglichkeit Änderungen an Dateien mittels Hash-Codes zu verfolgen. Anzumerken ist, dass es zum derzeitigen Stand nicht möglich ist, einzelne Elemente oder Zeilen auszuklammern, der Parser wird sie trotzdem auslesen.

5.1.2 Speichern von Schaltplänen

Das Speichern von Schaltplänen und Auswählen der Zieldatei erfolgt über den typischen betriebssystemabhängigen File Browser. Wird anstatt einer neuen Datei eine bereits Existierende ausgewählt, wird diese nach einer Bestätigung komplett überschrieben. Eine XML Schaltplandatei besteht aus der XML Deklaration und dem Root Element <Circuit>. Das Root Element enthält sämtliche Schaltplankomponenten und Verbindungen zwischen diesen. Die Speicherung von Werten der Komponenten erfolgt dabei als Attribut der XML Komponenten im typischen XML Stil, property=“value“. Ein Schaltplan kann dabei eine beliebig große Anzahl an Komponenten beinhalten, da jede Komponente in sich geschlossen ist und lediglich Verbindungen die IDs anderer Komponenten verwenden.

5.1.3 Manipulation von Schaltplandateien

Eine gespeicherte Komponente hat den folgenden typischen Aufbau:

```
<component id="12" x="750" y="650" rot="0" type="Kondensator" hash="-1455387425">
        <child comp="12" conn="10;11;" hash="1418336891"/>
        <connector id="10" x="750" y="825" rot="0" relX="0" relY="100" hash="-1756379682">
                <connection conn1="10" conn2="6" hash="975596792"/>
        </connector>
        <connector id="11" x="750" y="550" rot="180" relX="0" relY="-100" hash="-812529944">
                <connection conn1="11" conn2="2" hash="-831857675"/>
        </connector>
</component>
```

Jeder der Werte kann durch den Benutzer mit einem simplen Texteditor geändert werden. Jedoch ist zu beachten, dass keine Änderungen an den IDs durchgeführt werden sollten.

Das Component-Element enthält sämtliche Informationen über den Typ der Komponente und der Position auf dem Schaltplan. Die Zuordnung von Verbindungsstücken (connector) mit der Komponente erfolgt über das Kind Element. Verbindungsstücke enthalten die ursprüngliche relative Position und die aktuelle Position auf dem Schaltplan. Verbindungen werden über das Connection Element erstellt. Dabei wird die ID von zwei zu verbindenden Verbindungsstücken angegeben.

Werden manuelle Änderungen durchgeführt und in BlitzEdit geladen, wird dieses anhand der hash Attribute der Elemente erkannt und bei fehlerhaften Änderungen das geänderte Element ausgegeben. Diese Überprüfung kann deaktiviert werden durch Entfernen des „circuithash“ Elementes am Anfang vom Root Element.

5.1.4 Laden von Schaltplänen

Das Laden erfolgt, wie beim Speichern, mittels File Browser. Sollte eine Änderung der Schaltplandatei durch den Benutzer durchgeführt worden sein, muss dieser den Ladevorgang bestätigen, da nicht feststellbar ist, ob die Datei noch kompatibel mit dem Parser ist. Akzeptiert der Benutzer, versucht der Parser die Datei zu lesen und zu analysieren. Tritt ein Fehler auf, wie zum Beispiel das fehlende Einbinden einer Komponente in die aktuellen Bibliotheken, wird eine Meldung als Pop-up angezeigt und der Ladevorgang unterbrochen.

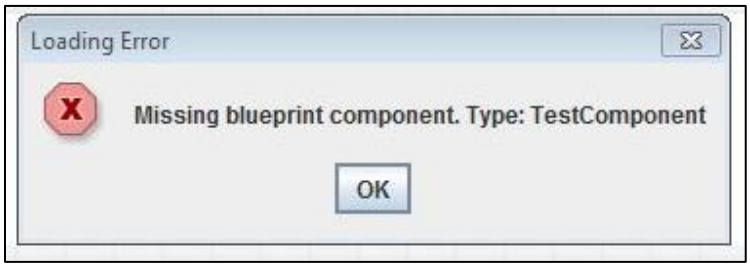


Abbildung 5 - Fehlermeldung für fehlende Komponente

Es können auch Änderungen an der Schaltplandatei durchgeführt werden während diese in BlitzEdit geöffnet ist. Der geänderte Schaltplan kann über die Reload (Shortcut: F5) Funktion einfach erneut geladen werden.

Abbildung 6 - Fehlermeldung für Änderungen in der Schaltplandatei

5.1.5 Komponenten im XML Format

Die Deklaration einer Komponente in XML hat das folgende Format:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Bauteil Deklaration: Spannungsquelle -->
<component type="Spannungsquelle">
        <svg path="img/Spannungsquelle.svg"/>
        <connector x="0" y="100" rot="0"/>
        <connector x="0" y="-100" rot="180"/>
        <property name="Spannungsquelle" type="decimal" unit="volt"/>
</component>
```

Eine Komponente besteht aus einer beliebigen Anzahl von Verbindungsstücken, einem SVG Bild und einer beliebigen Anzahl an Eigenschaften, wie z.B. einem Feld mit der Einheit Volt für eine Spannungsquelle. Die Pfadangabe zu einem SVG Bild erfolgt dabei relativ, sodass auch ganze Ordner einfach als Bibliotheken zwischen verschiedenen Benutzern und Computern ausgetauscht werden können. Verbindungsstücke bestehen aus Positionsattributen, die die relative Position zu der Komponente beschreiben, und ihre relative Rotation. Eigenschaften bestehen aus einem Namen, einem Datentyp (string, integral, decimal) und einer Einheit (Ohm, Farad, Volt, Ampere, Henry, Watt, Coulomb, Hertz, Windungen).

5.1.6 Modifikation von Komponenten

Um das Aussehen einer Komponente zu verändern muss das SVG Bild geändert werden. Dies kann man auch mittels eines einfachen Texteditors durchführen. Der SVG Renderer in BlitzEdit unterstützt bisher nur rechteckige Elemente und Rahmen. Um Verbindungsstücke oder Eigenschaften hinzuzufügen, können vorhandene kopiert und abgeändert oder neu geschrieben werden. Dabei ist jedoch auf die Syntax und Tippfehler zu achten.

5.1.7 Erstellen von Komponenten

Um eine neue Komponente zu erstellen, kann die bereits existierende Vorlagen Komponente im „test“ Ordner verwendet werden. Dazu muss diese an den gewünschten Zielort kopiert und der Name der Datei und der Komponente geändert werden. Schließlich muss lediglich der Pfad zu dem SVG Bild angegeben und wenn nötig Verbindungsstücke und Eigenschaften hinzugefügt werden. Wurde die Komponente erfolgreich erstellt, kann diese über die Import Funktion in BlitzEdit hinzugefügt und verwendet werden. (Bild mit Import?)

5.2 Corepackage

Das Corepackage implementiert die Strukturen und grundlegenden Funktionen der Schaltpläne sowie der Bauteile, die in den Schaltplänen abgelegt werden. Darüber hinaus werden Werkzeuge bereitgestellt, die den Zugriff auf die Bibliotheken gewährleisten, die Vorlagen für die Bauteile speichern. Im Folgenden werden die wichtigsten Klassen des Corepackage und deren Features vorgestellt.

5.2.1 Circuit

Die Circuit-Klasse implementiert die Schaltpläne, die mit dem Editor erstellt werden. Sie enthält einen dynamischen Array, der die Bauteile des Schaltplans speichert, und regelt den Zugriff auf diese Bauteile. Die wichtigsten Methoden fragen Bauteile anhand ihrer Eigenschaften ab. Bauteile können entweder über ihren Typ oder über ihre Position ausgewählt werden, darüber hinaus gibt es die Möglichkeit, mehrere Bauteile in einem bestimmten Bereich auszuwählen. Neben den Bauteilen kann die Circuit-Klasse auch die Position der Leitungen liefern, die die Bauteile miteinander verbinden, damit diese von den Klassen des Applicationpackage abgerufen werden können. Die Schaltpläne werden im XML-Format gespeichert, damit sie vom Nutzer auch manuell bearbeitet werden können.

5.2.2 Element

Die Element-Klasse ist die Basisklasse für alle Elemente, die in einen Schaltplan abgelegt werden können. Neben einer Position beinhaltet ein Element unter anderem eine Enumeration, die beschreibt, wie das Element momentan angewählt ist. Dadurch wird es möglich, Elemente kontextabhängig in unterschiedlicher Weise darstellen zu können.

5.2.3 Component

Die Component-Klasse bildet die Bauteile in den Schaltplänen ab. Sie stellt Methoden zur Verfügung, mit denen die Bauteile verschoben und rotiert werden können. Das Rotieren der Elemente ist prinzipiell in 1° Schritten möglich, implementiert wurde es aus Usability-Gründen aber nur in 45° Schritten. Die Bauteile erhalten Typ Namen, die die Art des Bauteils spezifizieren. Zu jedem Bauteil werden in der Component Klasse die Connectoren gespeichert, die das Bauteil mit anderen Bauteilen verbindet. Dies ist nötig, um das Bauteil korrekt darzustellen und um die Connectoren richtig verschieben zu können. Des Weiteren ist eine „draw“-Methode vorhanden, die die Darstellung des Bauteils spezifiziert. Zum einen delegiert diese auf die „draw“-Methoden der Connectoren, zum anderen wird eine SVG-Datei eingebunden, damit die Darstellung konfigurierbar bleibt und ohne Neukompilierung des Programms geändert werden kann.

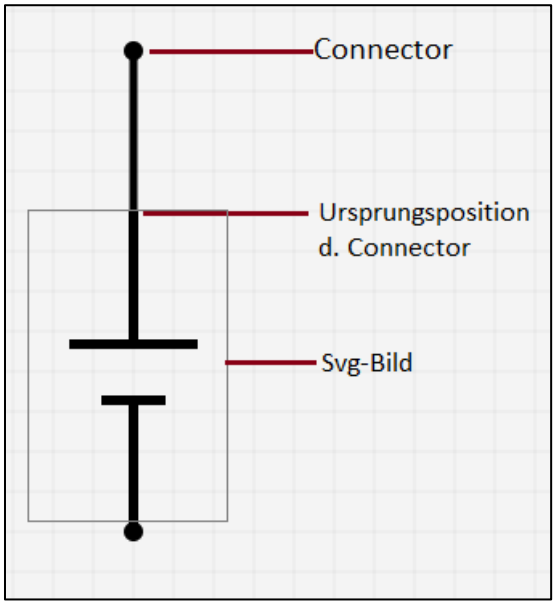


Abbildung 7 - Aufbau einer Komponente

Durch die Implementierung als Vektorgrafik lassen sich die Bauteile ohne Qualitätsverlust vergrößert darstellen und sehr ressourcenschonend rotieren. Um Speicherplatz zu sparen wird lediglich der Dateipfad auf die SVG-Datei gespeichert. Die Erscheinung des gesamten Bauteils mit der Position seiner Connectoren und dem zu verwendeten SVG-Bild wird in einer XML-Datei definiert, die sich im "Blueprints" Ordner befindet. Dadurch lassen sich während der Laufzeit neue Bauteile in das Programm einbinden.

5.2.4 Connector

Der Connector ist das Verbindungsstück zwischen den Bauteilen. An jedem Connector können beliebig viele Leitungen anliegen, die mit anderen Connectoren verbunden sind. Sie sind an das Bauteil gebunden, zu dem sie gehören, darüber hinaus lässt sich aber auch ihre relative Position zum Bauteil verändern. Dies geschieht auf einer geraden Linie zum Bauteil hin, um die Symmetrie des Bauteils nicht zu verändern. Dadurch ist es möglich, die Position der Leitungen über ihre Endpunkte zu verändern, damit sie zum Beispiel nicht durch ein anderes Bauteil hindurchlaufen. Um den Umgang mit den Connectoren bei unübersichtlichen Schaltplänen zu erleichtern, wurden mehrere Hervorhebungen realisiert. So wird, wenn man einen Connector anwählt, der Connector selbst blau dargestellt. Alle Connectoren, mit denen er verbunden ist, sind grün dargestellt, die, mit denen er nicht verbunden ist, werden rot gezeichnet.

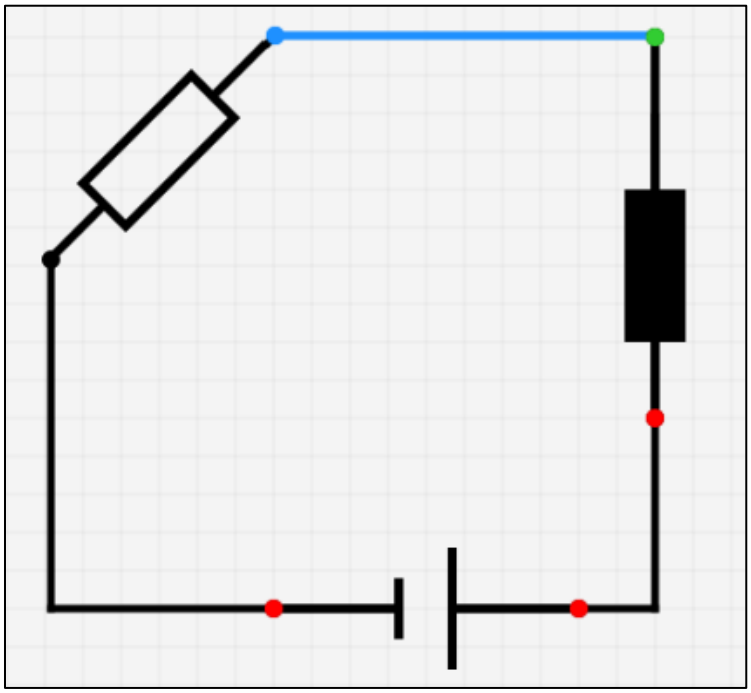


Abbildung 8 - Auswählen eines Connectors

5.2.5 Line

Die Line-Klasse implementiert die Leitungen in den Schaltplänen. Jede Leitung wird nur anhand ihres Anfangs- und Endpunktes definiert. Dadurch lassen sich die Leitungen zwar nicht frei nach Belieben verlegen, aber es wird durch die einfachere Darstellung möglich, sie in Echtzeit zu zeichnen. So kann der Nutzer schon während des Verschiebens der Elemente sehen, wie sich die Position der Leitungen verändern wird. Auch in der Line-Klasse befindet sich eine „draw“-Methode, in der die Leitung gezeichnet wird. Die Leitungen können situationsabhängig in verschiedenen Farben dargestellt werden, um Hervorhebungen bestimmter Leitungen zu realisieren. So werden Leitungen, die direkt mit dem momentan ausgewählten Bauteil oder Connector verbunden sind, blau dargestellt, alle anderen werden schwarz gezeichnet. Dadurch wird gewährleistet, dass der Nutzer beim Legen von Leitungen in unübersichtlichen Schaltplänen die Übersicht behält.

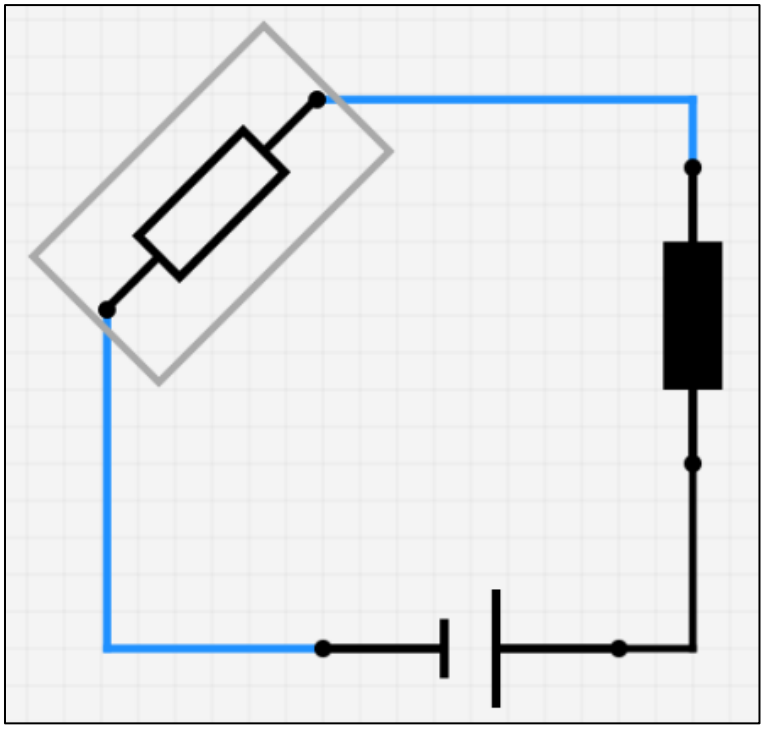


Abbildung 9 - Färbung der Linien

5.3 Oberfläche

Die Oberfläche von BlitzEdit besteht aus drei Fenstern und der Menüleiste, welche allgemeine Funktionalitäten bereitstellt wie Speichern und Laden.

5.3.1 Editorfenster

BlitzEdit unterstützt das Bearbeiten von mehreren Schaltplänen in einem Fenster mit mehreren Tabs. In jedem dieser Tabs kann man durch den Schaltplan navigieren, indem man den Viewport verschiebt oder den Schaltplan vergrößert bzw. verkleinert. Beim Platzieren der Schaltpläne wird ein Drag and Drop Konzept verfolgt. Der Nutzer zieht also eines der Bauteile aus dem Library Fenster in den Schalplan seiner Wahl um es zu platzieren. Die Bauteile können auf Wunsch an einem Gitter ausgerichtet oder frei Hand platziert werden. Die Bauteile können entweder einzeln ausgewählt werden, oder mittels klicken und ziehen in der Mehrfachauswahl. Bauteile, die ausgewählt wurden, werden mit einem grauen Rechteck umrandet. Anschließend können die ausgewählten Bauteile verschoben und rotiert werden, letzteres ist in 45 Grad Schritten möglich. Um das Programm auch alleine mit der Maus benutzen zu können, gibt es ein Rechtsklickmenü, das die Aktionen Kopieren, Einfügen, Löschen und Rotieren enthält.

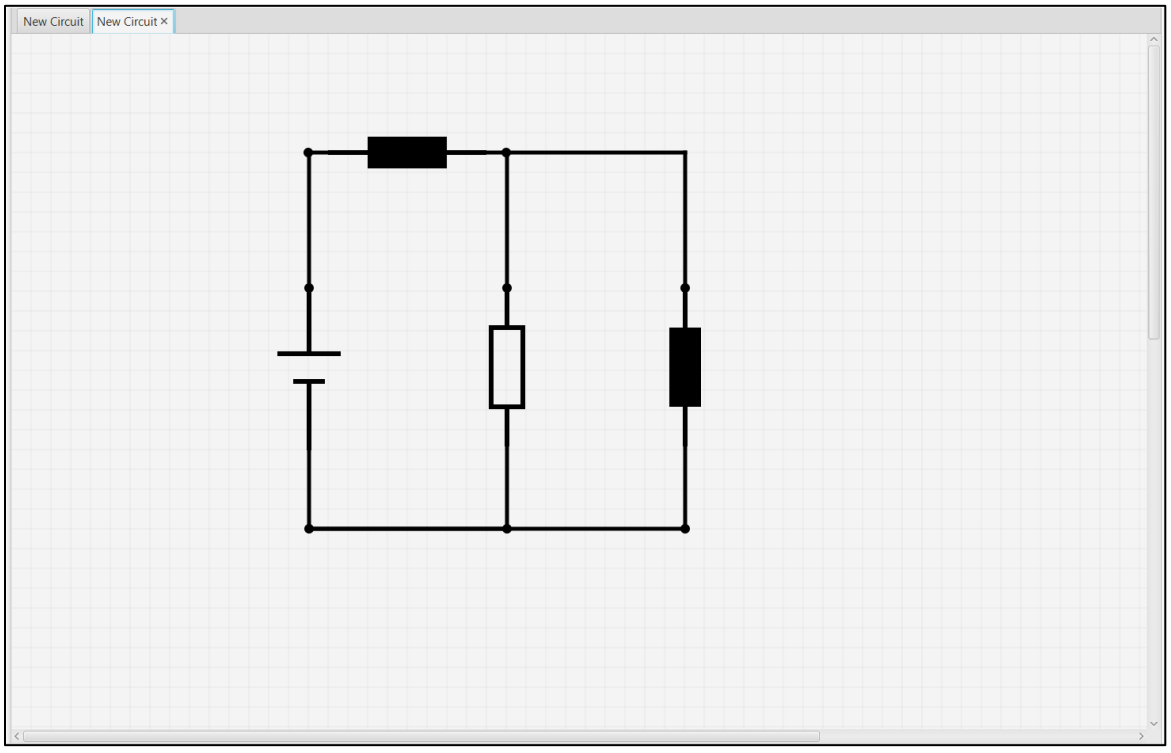


Abbildung 10 - Editorfenster

5.3.2 Bibliothekenfenster

Das Bibliothekenfenster zeigt alle Bauteile an, die in die Schaltpläne platziert werden können und ist an der linken Seite des Hauptfensters ausgerichtet. Wie es der Name schon vermuten lässt, kann man die Bauteile als Bibliotheken einbinden. Dadurch können zum Beispiel Bauteile thematisch geordnet oder verschiedene Notationen von Standartbauteilen verwendet werden. Neben der Grafik des Bauteils wird im Bibliothekenfenster auch der Name der Bauteile angezeigt. So kann der Nutzer auch dann die Bauteile identifizieren, wenn ihm die verwendete Notation unbekannt ist.

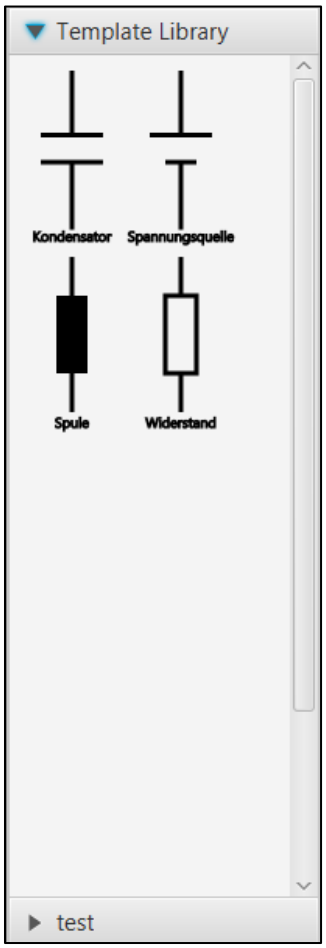


Abbildung 11 - Bibliothekenfenster

5.3.3 Hilfefenster

Das Hilfefenster findet sich an der rechten Seite des Hauptfensters. Es zeigt nützliche Informationen an, die dem Nutzer die Features von BlitzEdit näherbringen. Die Tipps sind zur einfacheren Navigation kategorisiert geordnet, sodass der Nutzer im jeweiligen Kontext stets die richtigen Tipps bekommt.

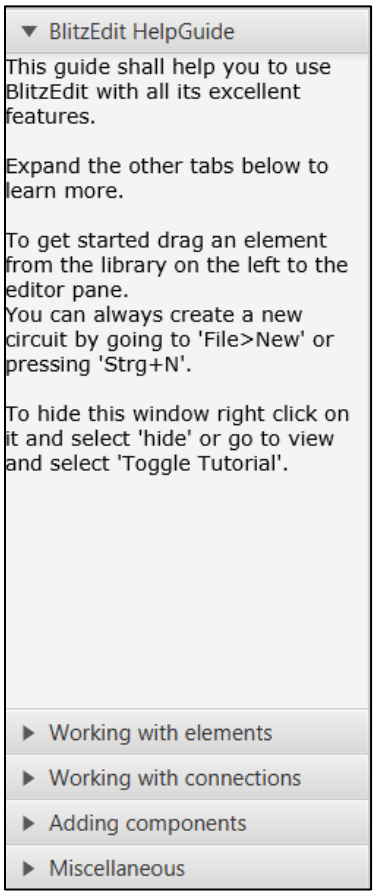


Abbildung 12 - Hilfefenster

6 Ausblick

Das Produkt ist darauf ausgelegt durch verschiedenste Funktionalitäten möglichst einfach erweitert werden zu können. Außerdem ist es als Open Source Projekt auf GitHub für jeden zugänglich und modifizierbar.

6.1 Ideen für Erweiterungen

Im Laufe des Projekts kamen immer wieder neue Ideen auf, wie das Produkt erweitert werden könnte, aber aus verschiedenen Gründen nicht in die nähere Planung einfließen konnten.

6.1.1 Komponenten

Die naheliegendste Erweiterung ist das Hinzufügen von weiteren logischen Schaltplankomponenten zu den Bibliotheken wie zum Beispiel Dioden oder Schalter. Dazu müssen SVG Grafiken der jeweiligen Komponenten erstellt und in entsprechenden XML Dokumenten eingebunden werden. Es ist möglich die neuen Komponenten während der Laufzeit zu importieren oder schon zum Programmstart zu laden.

6.1.2 Eigenschaften

Des Weiteren haben Komponenten oder auch Leitungen in echten elektrischen Schaltplänen bestimmte Eigenschaften. Über ein Eigenschaftenfenster könnten sich beispielsweise die Höhe des Widerstands oder die Stromstärke einer Stromquelle individuell einstellen lassen. Die Werte könnten auch auf dem Schaltplanfenster angezeigt werden und eine realistischere Darstellung ermöglichen. Ein erster Prototyp für das Eigenschaftenfenster ist bereits im Code vorhanden aber findet noch keine Verwendung.

6.1.3 Simulation

Die Eigenschaften sind Voraussetzung für eine realistische Simulation des Schaltplans, mit der veranschaulicht werden soll, ob dieser funktionsfähig ist oder nicht. Benutzer sollen die Software dazu nutzen, Schaltpläne zu planen und zu testen, damit bei der späteren Umsetzung Komplikationen ausgeschlossen werden können. Durch eine Simulation könnte man zum Beispiel erkennen, ob ein Kurzschluss auftritt. Zur Umsetzung dieser Erweiterung ist allerdings ein tieferes elektrotechnisches Verständnis von Nöten.

6.1.4 Interaktive Hilfe

Momentan ist eine Nutzerhilfe implementiert, die einen Überblick über die Funktionalität des Programms gibt. Sie ist aufrufbar über ein Hilfefenster, das sich ein- und ausklappen lässt und verschiedene Tabs mit Informationen enthält. Für die Nutzerfreundlichkeit wäre es sehr von Vorteil, wenn ein passender Tipp angezeigt werden würde, wenn der Nutzer dabei ist, entsprechende Funktionen nutzen zu wollen. Beim Erstmaligen Auswählen eines Connectors könnte beispielsweise ein Hilfefenster erscheinen, das erklärt, wie man zwei Bauteile verbindet.

6.1.5 Design

Etwas einfacher ist die Umsetzung von verschiedenen Designs des Programms um die Nutzerfreundlichkeit zu steigern. Benutzer sollen das Produkt an ihre individuellen Bedürfnisse anpassen können, das Design steht dabei an oberster Stelle. Eine dunkle Benutzeroberfläche kann bei längerem Arbeiten angenehmer sein, als eine sehr helle und grelle. Da dies selbstverständlich eine Geschmacksfrage ist soll der Nutzer selbst entscheiden und jederzeit wechseln können. Für die Implementierung neuer Designs müssten nur CSS Dateien eingebunden werden.