Politécnico do Porto
Instituto Superior de Engenharia do Porto

U=RIsolve

Implementação do Teorema da Sobreposição

Guilherme Zenha

LEEC
Licenciatura em Engenharia Eletrotécnica e de Computadores

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Julho, 2023

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Projeto/Estágio, do 3º ano, da Licenciatura em Engenharia Eletrotécnica e de Computadores.

Candidato: Guilherme Zenha, Nº 1201398, 1201398@isep.ipp.pt

Orientação Científica: André Rocha, anr@isep.ipp.pt

Coorientação Científica: Francisco Pereira, fdp@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Julho, 2023

Agradecimentos

Agradeço a todos os que contribuíram para um melhor desenvolvimento deste projeto, nomeadamente os professores envolvidos: Mário Alves, André Rocha e Francisco Pereira. Desejo dar especial ênfase ao professor André Rocha, pela disponibilidade e tempo investidos na orientação deste projeto e ajuda em todas as questões ou problemas que surgiram ao longo do seu desenvolvimento.

Gostaria também de agradecer a todos os familiares, amigos e colegas que prestaram todo o apoio possível.

Por fim, gostaria de reconhecer todos os colegas envolvidos no processo de teste do código desenvolvido durante os estágios iniciais.

Todo este apoio foi essencial e os resultados obtidos são também uma reflexão do esforço de todos os envolvidos.

Resumo

Este projeto tem como objetivo criar um módulo para a aplicação *web* U=RIsolve capaz de receber um circuito desenhado na ferramenta QUCS e resolvê-lo usando o Teorema da Sobreposição, dispondo de um sistema de redesenho do mesmo circuito, de maneira responsiva, interativa e didática.

Este iniciou-se com o estudo das componentes técnicas e científicas , necessárias para a sua execução, nomeadamente um estudo extensivo de linguagens de programação utilizadas na criação de aplicações *web*, como o *JavaScript*.

Durante os estágios de desenvolvimento foram desenvolvidos diversos algoritmos, começando pelo *parsing* e tratamento do ficheiro ".sch"do QUCS para um objeto JSON, redesenho desse circuito num *widget* interativo (com a possibilidade de personalização do nível de interatividade, a certo nível), o algoritmo que gera a *netlist* a partir do ficheiro JSON criado anteriormente, a criação de circuitos parciais a partir da substituição de fontes de acordo com o teorema. Integração destes mesmos algoritmos no código existente, modificando e criando novas funções baseadas nas já existentes. Por fim, foi feita a migração de Bootstrap 4 para Bootstrap 5, trazendo novas capacidades para a aplicação das quais foram usadas *toasts* para criar um sistema de alertas dentro da janela de modal.

Palavras-Chave: U=RIsolve, teorema da sobreposição, QUCS, schematic, netlist, parsing, objeto JSON, widget, aplicação web, *JavaScript*.

Abstract

This project aims to create a module for the web application U=RIsolve that is capable of receiving a circuit drawn in the QUCS tool and solving it using the Superposition Theorem, providing a system for redrawing the circuit in a responsive, interactive, and didactic manner.

The project started with an extensive study of the technical and scientific components necessary for its implementation, including a thorough examination of programming languages used in web application development, such as *JavaScript* (JS).

During the development stages, several algorithms were developed. It began with parsing and processing the .sch file from QUCS into a JSON object. The circuit was then redrawn in an interactive widget with customizable levels of interactivity. Another algorithm was created to generate the netlist from the previously created JSON file. Partial circuits were generated by replacing sources according to the theorem. These algorithms were integrated into the existing code, modifying and creating new functions based on the ones already present. Finally, the migration from Bootstrap 4 to Bootstrap 5 was performed, bringing new capabilities to the application, such as using toasts to create an alert system within the modal window.

Keywords: U=RIsolve, Superposition Theorem , QUCS, schematic, netlist, parsing, JSON objet, widget, web aplication, *JavaScript* (JS).

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AC	*Alternating Current*
API	*Application Programming Interface*
BS	Bootstrap
CSS	*Cascading Style Sheets*
DC	*Direct Current*
DOM	*Document Object Model*
ECMA	*European Computer Manufacturers Association*
ES	ECMAScript
FEELE	Fundamentos da Engenharia Eletrotécnica
GUI	*Graphic User Interface*
HTML	*HyperText Markup Language*
IoT	*Internet of Things*
JS	*JavaScript*
JSON	*JavaScript Object Notation*
LEEC	Licenciatura em Engenharia Eletrotécnica e de Computadores
MCM	Método das Correntes nas Malhas
MCR	Método das Correntes nos Ramos
MTN	Método da Tensão nos Nós
PESTA	Projeto / Estágio
QUCS	*Quite Universal Circuit Simulator*
QUCS-S	*Quite Universal Circuit Simulator with SPICE*
RegExp	*Regular Expressions*

RF	*Radio Frequencies*
SASS	*Syntactically Awesome Style Sheets*
SPICE	*Simulation Program with Integrated Circuit Emphasis*
TCIRC	Teoria dos Circuitos
TSP	Teorema da Sobreposição
UI	Interface de Utilizador
W3C	*World Wide Web Consortium*
WWW	*World Wide Web*
XML	*Extensive Markup Language*

Capítulo 1

Introdução

Neste capítulo, estão presentes informações importantes sobre o projeto desenvolvido. É efetuada a contextualização do projeto e dos temas relacionados em prol de uma melhor compreensão do mesmo e das necessidades que ele pretende atender.

Em seguida, é feita uma descrição detalhada do projeto, destacando as suas principais características e funcionalidades.

São apresentados, também, os objetivos e condições que se propõe a atingir no seu estado final. Estes orientaram todas as etapas, de forma clara e objetiva, descrevendo a finalidade do projeto e os resultados esperados.

De forma a auxiliar a organização do projeto, é apresentada também uma calendarização no formato de um diagrama de Gantt, que permite visualizar as atividades e prazos envolvidos no desenvolvimento do projeto.

Por fim, é abordada a organização do relatório, estabelecendo as suas secções e todos os pontos abordados nas mesmas.

Desta forma, o capítulo de introdução fornece uma visão geral do projeto e do relatório em si, ajudando na navegação e compreensão dos seus conteúdos.

1.1 Contextualização

Um simulador de circuitos é um software que permite modelizar e simular o comportamento de circuitos elétricos. Estes são amplamente utilizados para projetar, testar e afinar circuitos durante as fases do desenvolvimento de diversos projetos, tanto em contexto profissional, como académico. Existe uma abundância de opções quanto a escolher o software mais adequado à aplicação pretendida.

Entre eles, o *Quite Universal Circuit Simulator* (QUCS) destaca-se pelas suas funcionalidades avançadas, como:

- Ser capaz de simular circuitos analógicos, digitais ou mistos, circuitos que combinam componentes analógicas e digitais.
- Permitir a criação de modelos de dispositivos eletrónicos personalizados, ou seja, permite simular o comportamento de componentes que não estão presentes na biblioteca padrão.

Para além disso, este *software* destaca-se por ser fácil de usar, multiplataforma, *open-source*, grátis e possuir funcionalidades básicas para o ensino/aprendizagem de análise de circuitos *Direct Current* (DC)/*Alternating Current* (AC) [1].

No entanto, para contextos de ensino ou estudo da resolução de circuitos, este apenas disponibiliza valores "medidos" permitindo apenas a comparação de valores calculados e valores simulados, não mostrando informação sobre o processo de resolução, i.e., como chegar aos valores corretos.

Para suprir esta limitação, temos a aplicação *U=RIsolve*. Esta permite, a partir do modelo do circuito, gerado no QUCS, acompanhar a resolução do mesmo, de maneira interativa e didática. Antes do desenvolvimento deste projeto, a aplicação possibilita a resolução do circuito por três métodos diferentes: Método da Tensão nos Nós (MTN), Método das Correntes nos Ramos (MCR) e Método das Correntes nas Malhas (MCM).

Utilizando estes métodos, apenas é possível fazer análise de circuitos cujas fontes operem à mesma frequência. Surge então o Teorema da Sobreposição (TSP), um dos teoremas que permite a análise de circuitos elétricos. Este permite simplificar a análise de circuitos com múltiplas fontes de tensão e/ou corrente. Estabelece que a contribuição de cada fonte de energia pode ser calculada separadamente e, posteriormente, somada algebricamente para se obter o resultado final, permitindo a análise de circuitos AC com fontes a trabalhar em diferentes frequências, contudo esta deve ser feita no domínio dos tempos ao invés de no domínio dos complexos.

1.2 Descrição do Projeto

Dadas as vantagens e utilidade do TSP, este é uma mais valia numa ferramenta como o *U=RIsolve*, este projeto, a partir das bases existentes, pretende implementar este teorema, contribuindo para os objetivos centrais da mesma.

1.2.1 Objetivos

Assim, o principal objetivo deste projeto é desenvolver uma aplicação web que:

- Gere a metodologia e explicação, passo a passo, da implementação do TSP em circuitos elétricos a partir de um circuito desenhado no simulador QUCS e exportação da mesma como JSON ou Tex, com possibilidade de abrir diretamente no Overleaf;
- Permite que o utilizador escolha a ordem de análise das fontes e métodos de resolução para cada sub-análise.
- Apresente o raciocínio e resultados de forma lógica e visualmente agradável;
- Seja responsiva, adaptando-se a ecrãs de diferentes dimensões, com ênfase em dispositivos de menor dimensão, nomeadamente smartphones;
- Receba e processe um ficheiro de esquemático (.sch), gerando a *netlist* necessária para os cálculos;
- Represente o circuito visualmente de maneira interativa.

Em suma, pretende-se desenvolver uma ferramenta de ensino/autoaprendizagem útil aos alunos e professores de Fundamentos da Engenharia Eletrotécnica (FEELE) e Teoria dos Circuitos (TCIRC), possibilitando a preparação de exercícios e verificação da análise teórica pelos professores e alunos.

1.3 Calendarização

Visto que o projeto ocorreu durante um grande período de tempo (19 de fevereiro a 10 de julho), este foi dividido em duas fases, cada uma dividida em várias tarefas

A primeira fase inclui objetivos como o estudo inicial dos contextos científicos e tecnológicos, o *parsing* do ficheiro ".sch"e a construção das primeiras versões do modelo de dados, que na altura foi utilizado para criar as primeiras versões do *widget* de redesenho. Nesta etapa também foram feitos os algoritmos de criação da *netlist* e iniciado o processo de extração de nós e conexões utilizados para depois criar a versão final do modelo de dados. No fim desta etapa, deu-se como concluído o processo do *parsing* e criação da *netlist*. Assim, estava feito a maioria do trabalho de extração de nós e tratamento de dados e de redesenho do circuito.

Mês	Fev	Fev/Mar	Mar	Mar	Mar	Mar	Abr	Abr	Abr	Abr
Objetivo \ Semana	1	2	3	4	5	6	7	8	9	10
Estudo do contexto tecnológico e cientifico										
Instalação de software										
Analíse de exemplo CED										
Analíse do ficheiro SCH										
Parsing do ficheiro SCH e modelo de dados										
Widget de redesenho do circuito										
Pesquisa sobre algoritmos de criação de netlists										
Extração de nós do objeto do esquemático										
Criação da netlist										

Figura 1.1: Calendarização 19 de fevereiro a 29 de abril

A segunda e última etapa engloba o processo de testes final, a redação deste relatório, assim como, desenvolvimento das versões finais dos algoritmos anteriores, a criação dos algoritmos relativos ao TSP e implementação dos algoritmos implementados no código do U=RIsolve.

Mês	Mai	Mai	Mai	Mai	Mai/Jun	Jun	Jun	Jun	Jun	Jul	Jul
Objetivo \ Semana	1	2	3	4	5	6	7	8	9	10	12
Modelos de dados finais											
Algoritmo de geração de circuitos parcelares											
Modal inicial e seleção da ordem e metodo											
Migração Bootstrap5											
Integração do parsing e redesenho no U=Risolve											
Algoritmo de resolução TSP											
Modal de resolução TSP											
Debugging											
Redação do relatório											

Figura 1.2: Calendarização 30 de abril a 10 de julho

1.4 Organização do Relatório

Devido à dimensão do projeto e à quantidade de informação que se pretende transmitir, esta é dividida em segmentos correspondentes aos diversos aspetos e etapas do desenvolvimento do mesmo. O relatório é constituído então por **6** capítulos.

Este primeiro capítulo apresenta o contexto e as motivações por detrás do projeto, assim como, os critérios que este pretende cumprir e o planeamento de todas as etapas do processo do seu desenvolvimento, desde o estudo em preparação até à redação deste relatório.

O capítulo **2** apresenta uma síntese de toda a informação proveniente do estudo efetuado antes de iniciar o desenvolvimento do projeto, abordando as diversas ferramentas usadas e o contexto geral, do ponto de vista tecnológico, nos vários momentos em que foi desenvolvido o projeto.

A seguir, no capítulo **3**, é abordado o tema do TSP com maior profundidade, tratando os critérios para a utilização do mesmo, assim como resolvendo um circuito exemplo, explicando o passo-a-passo e dando dicas relevantes para a resolução de circuitos semelhantes.

No capítulo **4** intitulado Desenvolvimento de *Software*, são sintetizados todos os algoritmos e funções desenvolvidas para utilização durante a execução do programa. Esta é, posteriormente, separada em secções correspondentes a cada fase da execução do programa.

Toda a informação em relação a implementação destes algoritmos está presente no capítulo **5**. Este incluí também alguns pontos que ajudam a uma melhor compreensão da utilização da ferramenta, as escolhas tomadas na representação dos resultados e como o trabalho desenvolvido é incorporado ao já presente.

Por último, o capítulo **6**, das conclusões, apresenta um resumo das principais descobertas e conclusões, algumas notas finais e consolida toda a informação apresentada no corpo do relatório.

Em anexo é possível encontrar, a resolução de alguns circuitos exemplo utilizando o TSP.

Capítulo 2

Estado da arte

O capítulo do estado da arte apresenta o contexto tecnológico, das diversas tecnologias relevantes ao projeto no ponto em que estas se localização durante o seu desenvolvimento. Este auxilia a uma melhor compreensão do estado destas e pretende fornecer também uma melhor compreensão do funcionamento das mesmas.

2.1 Javascript

JavaScript (JS), também oficialmente reconhecido como ECMAScript (ES) uma vez que os seus padrões são criados pela *European Computer Manufacturers Association* (ECMA) [2], é uma linguagem de programação de alto nível amplamente utilizada no desenvolvimento de aplicações web e móveis, jogos, aplicações de *desktop* e muitas outras tecnologias e *frameworks* avançados. Tornou-se uma ferramenta essencial para a criação de páginas web interativas e Interface de Utilizador (UI) responsivas[1], graças à sua capacidade de resposta em tempo real a ações do utilizador, como cliques de botões, preenchimento de formulários, deslocamento de página e outras interações.

Este está constantemente em evolução, o ES especificamente, lança um novo padrão de JS anualmente. A mais recente, no momento de redação deste relatório sendo o ES13 ou ES2022. Este introduz o método ***at()***, que possibilita a indexação de elementos de *strings* e *arrays*, adiciona índices ao objeto gerado pelas *Regular*

[1]Uma página web responsiva, é uma página que se adapta automaticamente a diferentes dispositivos e tamanhos de ecrã

Expressions (RegExp), facilitando a comparação e processamento da posição dos grupos encontrados, traz o método ***hasOwn()*** que funciona de maneira semelhante ao ***hasOwnProperty()*** mas é aplicável a todos os tipos de objeto, ***error.cause*** permite especificar a causa do erro quando é feito o *throw* do mesmo, entre outros. [3]

Versões anteriores, como o ES6, trouxeram também mudanças significativas, com a introdução de novas funcionalidades e *frameworks* que simplificaram a escrita de código, permitindo a criação de aplicações mais complexas e sofisticadas. Algumas dessas novas funcionalidades incluem as *Promises, async/await e arrow functions*, que facilitaram a escrita de código assíncrono e permitiram uma melhor manipulação de eventos. Além disso, tornou-se possível executar programas tanto no lado do cliente quanto no servidor, graças ao *Node.js*, permitindo que os desenvolvedores criem aplicações web de ponta a ponta usando uma única linguagem de programação.[4]

O ecossistema de *frameworks* de *front-end* também tem evoluído rapidamente, com o *React*, *Vue* e *Angular* sendo os principais exemplos. Esses *frameworks* permitem que os desenvolvedores criem interfaces de utilizador reativas e responsivas, tornando as aplicações web modernas mais atraentes e fáceis de usar.

O JS também é cada vez mais usado em outras áreas, como desenvolvimento de aplicações móveis híbridas com o *React Native* e o *Ionic*, e no desenvolvimento de aplicações de *desktop* com o *Electron*. Além disso, vem ganhando importância em áreas de tecnologias emergentes, como a *Internet of Things* (IoT) e a realidade virtual, onde é usado para controlar dispositivos e criar experiências interativas. [5]

Em suma, o JS é uma linguagem de programação flexível e poderosa que continua a evoluir e a expandir o seu alcance em diferentes áreas da tecnologia.

2.1.1 JSON

Na representação e armazenamento de dados gerados e utilizados por programas baseados em JS, e também de outras linguagens de programação, é o JSON, uma vez que é uma forma de representar informações em texto simples, organizado no formato de um objeto JS, pares chave-valor em que o valor em si pode representar um conjunto de pares, possibilitando tantos níveis quando necessário. Ainda que algumas práticas de uso do mesmo sejam aprimoradas ao longo dos anos, este não sofreu grandes mudanças ao seu formato.

2.1.2 jQuery

Além do JS padrão, existem muitas bibliotecas que podem ajudar os utilizadores a simplificar e acelerar o processo de desenvolvimento de aplicações web. Uma das bibliotecas mais populares é o jQuery.js, um conjunto de recursos de alto nível, que

permite manipular e interagir com elementos *HyperText Markup Language* (HTML), animações e muito mais, de uma maneira mais fácil e eficiente.

É bastante utilizado pela sua simplicidade e facilidade de utilização, permitindo aos programadores escrever código menos extenso e mais legível. Oferece, também, uma ampla gama de *plugins*, que expandem a sua funcionalidade e permitem que os programadores adicionem recursos complexos nas suas aplicações, como animações avançadas, gráficos, galerias de fotos, formulários avançados e muito mais.

Possui uma grande comunidade de programadores, o que significa que há muitos recursos e tutoriais disponíveis na Internet, para ajudar programadores iniciantes e experientes. Além disso, muitos frameworks modernos, como o *Angular* e o *React*, têm integração com o jQuery.js, o que significa que pode facilmente usar a biblioteca em conjunto com esses.

Ou seja, o jQuery.js é uma biblioteca JS popular que oferece muitos recursos de alto nível para os programadores, tornando mais fácil e rápido o processo de desenvolvimento de aplicações web.

2.1.3 jQueryUI

Com o crescimento de jQuery, surgiram diversas bibliotecas que constroem em cima do mesmo para expandir as suas capacidades. Uma destas é o jQueryUI, que inclui componentes de UI, como menus, caixas de dialogo, calendários e muito mais.

Este oferece um conjunto de ferramentas poderosas para criar UI complexas de maneira fácil e rápida, assim como, temas personalizáveis, que o programador pode escolher e combinar com as suas preferências de design.

Tal como o jQuery, possui uma grande comunidade o que torna encontrar tutoriais e outros materiais úteis, fácil.

2.2 HTML

No centro do desenvolvimento web, está presente o HTML, a principal linguagem da *World Wide Web* (WWW). Este permite criar páginas web, de forma sucinta e facilmente interpretável, fazendo uso de *tags* e atributos, controlando a sua formatação e organização. Não é considerado uma linguagem de programação, mas sim uma linguagem de marcação, um conjunto de sinais e códigos integrados em um texto ou grupo de dados, que define o formato de representação do mesmo, sendo reconhecido pela *World Wide Web Consortium* (W3C)[6].

Para a utilização do mesmo, é necessário redigir um documento de extensão **.html** ou **.htm**, constituído por *tags* em conjunto com o conteúdo da página, no formato, *tag* de marcação de abertura (por exemplo, ' '), seguida do conteúdo dessa *tag* e da *tag* de marcação de fecho (por exemplo, ' '). No interior da *tag* de abertura é possível inserir os atributos desse elemento do HTML, como o *id*, *class*, *style*. Também permite a inserção de outros atributos específicos ao tipo de elemento ou atributos próprios à página, que servem um propósito no contexto do JS presente namesma ou em conjunto com *frameworks* como Bootstrap (BS). [6]

Originalmente, o HTML possuía dezoito *tags*. Desde então, cada nova versão trouxe novas *tags* e atributos, sendo a versão mais recente sendo o HTML5.[6]. Esta traz novas *tags*, como '*nav*', *header* e *footer*, *tags* de suporte nativo de áudio e vídeo, *figure* e *figcaption*, *mark*, entre outros.

Atualmente, com o HTML5 surgiram também ferramentas como os *Web Components*, que possibilitam a criação de componentes personalizados e reutilizáveis. Estes possuem três funcionalidades principais:

- ***Custom Elements*** possibilita a criação de *tags* personalizadas com comportamentos e funcionalidades próprios, seja a lógica do próprio comportamento, estilo ou interações;
- ***Shadow Document Object Model (DOM)*** permite encapsular a estrutura HTML, estilos e comportamentos de um elemento personalizado. Ou seja, utilizando esta ferramenta, é possível criar componentes que são isolados do restante do documento, evitando conflitos de estilos e garantindo que o comportamento interno do componente não seja afetado pelo estilo ou comportamento externo;
- ***HTML Templates*** são uma forma de definir modelos de marcação, com possibilidade de os clonar e repetir, criando estruturas reutilizáveis que podem ser, posteriormente, preenchidas dinamicamente com JS.

Novas funcionalidades do HTML facilitaram também a criação de novas *Application Programming Interface* (API). Assim surgiram ferramentas como *Canvas*, *Web Storage*, *Web Workers*, *WebSockets*, *History* e *Drag and Drop*.[7]

2.3 CSS

Em conjunto com JS que é responsável por criar páginas web dinâmicas e HTML que é responsável por fornecer elementos textuais e determinar a estrutura da mesma, frequentemente, são utilizadas também *Cascading Style Sheets* (CSS), que contribuem para a estilização da página, controlando as cores, espaçamento, *layout*, animações e muitos outros aspetos estéticos e estruturais da página. Estas três sendo fundamentais no desenvolvimento *front-end*.

A versão mais recente é intitulada CSS3, trazendo uma ampla gama de recursos e melhorias de ferramentas já existentes em versões anteriores, dentro destas temos seletores mais avançados, animações, transações, sombras, gradientes, *flexbox* e *grid layouts*, entre outros.

Dos quais, os mais relevantes:

- ***Grid Layout*** introduz a possibilidade de criar *designs* de página complexos e responsivos através de um sistema de grelha bidimensional flexível que permite posicionar e organizar elementos de forma precisa, juntamente com as *media queries*, permite satisfazer a crescente necessidade de compatibilidade com dispositivos móveis;
- ***Media Queries*** tornam possível aplicar diferentes regras dependendo das dimensões do ecrã do dispositivo;
- **Variáveis**, valores reutilizáveis que podem ser aplicados a várias propriedades, simplificando a manutenção do código, ao permitir a atualização rápida e fácil de estilos em todo o site;
- ***Flexbox***, um modelo de *layout* flexível que simplifica a criação de *designs* responsivos e flexíveis. Permite o alinhamento e distribuição de elementos em um contentor, independentemente do tamanho ou da ordem dos elementos.

Assim como JS e HTML, o CSS possuí uma comunidade dedicada, então ao mesmo tempo que continua a evoluir e novas funcionalidades e técnicas são desenvolvidas, novas ferramentas não nativas e *frameworks* surgem e evoluem constantemente e criam ecossistemas que trabalham em conjunto para melhorar a estilização e a apresentação das páginas web.

Comunidade esta que promove a colaboração e a troca de ideias por meio de fóruns, grupos de discussão, conferências e comunidades online. Isso permite que os seus utilizadores compartilhem as suas experiências, tirem dúvidas e aprendam uns com os outros. Permitindo que qualquer programador iniciante tenha a possibilidade de obter as capacidades necessárias para eficazmente trabalhar com CSS.

Essa colaboração entre os seus membros resulta na criação e no aprimoramento de recursos e ferramentas adicionais. Uma forma de recursos bastante popular são

os pré-processadores como *Sass* e *Less*, que permitem escrever CSS de maneira mais eficiente e produtiva, e frameworks como BS, *Foundation*, *Tailwind CSS*, que fornecem estilos, componentes e *layouts* pré-estilizados, acelerando o processo de desenvolvimento. Estes possuindo documentação extensiva e fácil de compreender.

Em prol de facilitar a organização, a compilação e a otimização do código, existem também ferramentas de automação, como *task runners* , por exemplo, *Grunt* e *Gulp* e bundlers como *webpack* e *Parcel*.

Esses ecossistemas e ferramentas ajudam os desenvolvedores a melhorar a produtividade, a criar projetos mais consistentes e a lidar com os desafios enfrentados no desenvolvimento CSS.

2.3.1 Bootstrap

De todas as *frameworks*, otimizam e completam o CSS, o BS é um dos mais relevantes para este projeto, pois foi uma ferramenta bastante útil durante o seu desenvolvimento, sendo responsável por grande parte dos estilos presentes nas páginas desenvolvidas. Uma vez que, assim como mencionado anteriormente, este disponibiliza uma grande variedade de classes pré-estilizados com nomes intuitivos e componentes JS na forma de *plugins* como *Carousel*, *Modal*, *Navs and tabs*, o que garante um melhor fluxo de trabalho ao criar elementos web responsivos e interativos. Possibilita ainda, a personalização do estilo geral do site, permitindo a criação de uma identidade visual única por uso de temas e variáveis.

Este possuí documentação bastante extensiva,completa e de fácil compreensão.

Uma das ferramentas chave desta, é o seu sistema de grelha, introduzido na versão BS4, que se adapta a tamanhos e formatos de ecrã.. Este é desenvolvido com uma abordagem *mobile-first*, construindo as suas classes para funcionarem em ecrãs mais pequenas e depois expandindo-as para ecrãs maiores, utilizando *media queries* e outras funcionalidades do CSS, garantindo a sua responsividade [8].

A versão mais recente desta *framework*, o BS5, torna opcional o uso das grelhas integradas no próprio CSS ao invés de *flexbox*, assim como introduz novas classes que permitem ainda maior controlo sobre o mesmo. Esta versão deixa de ser dependente do jQuery, e, assim como deixa de suportar versões mais antigas dos vários browsers, deixa de suportar *Internet Explorer 10* e *11*, uma vez que a *Microsoft* tem concentrado os seus esforços em desenvolver o *Microsoft Edge*, que adotou o *chromium*. Esta alteração teve o objetivo de poder fornecer ferramentas mais modernas que o anterior não seria capaz de suportar, como propriedades CSS personalizadas, pelo uso de variáveis. O BS expandiu também a sua paleta de cores, introduzindo diversos novos tons. Relativamente aos formulários, onde em versões como o BS4 e anteriores era utilizado o visual predefinido de cada *browser*, nesta nova versão é introduzido um visual próprio, de forma a manter consistência, mas mantendo a otimização do processo de gerar formulários utilizando a *framework*.

Uma nova ferramenta que é bastante relevante é a nova *utility* API, esta permite que seja criados utilitários através de *Syntactically Awesome Style Sheets* (SASS), para modificar ou remover os predefinidos do próprio BS.

Esta versão introduz também novos componentes como *offcanvas*, um novo acordeão que inclui ícones, *placeholders* que permitem mostrar o *layout* da página enquanto o UI ainda está a ser renderizado, suporte de *right-to-left*, a criação do BS *icons*, que dispoem de mais de mil icons em formato SVG, um novo logótipo, etc.[9]

2.4 QUCS

Como mencionado no capítulo anterior, o QUCS é um simulador de circuitos elétricos fácil de usar, *open-source*, multiplataforma e grátis.

Recebeu a sua última atualização em 2017, intitulada QUCS0.0.19, que eliminou alguns dos problemas presentes em versões anteriores e adicionou funcionalidades como a possibilidade de gravar esquemáticos como imagem através de linha de comando, pesquisa de ficheiros na *dock* de componentes, um menu contendo os projetos recentemente abertos, assim como, melhorou o gestor de projetos, entre outras melhorias. [10].

Com base neste simulador surgiram algumas ferramentas que adicionam novas funcionalidades ou alteram ferramentas existentes para uma melhor usabilidade. Assim como, foram criados programas com algum mérito e relevância por si próprios, como o Qucsator, Qucsconv, Qucstrans e Qucsfilter. Sendo o QUCS a *Graphic User Interface* (GUI) que engloba todos estes juntamente com outros como o Qucsedit, um editor de texto simples e o Qucshelp que proporciona o sistema de ajuda.

2.4.1 Qucsator

O Qucsator é a componente, baseado em linha de comando, do QUCS verdadeiramente responsável por efetuar a simulação dos circuitos a partir da sua *netlist*, sendo desenvolvida maioritariamente para uso internamente dentro do funcionamento do mesmo, porém possibilitando o seu uso por outras aplicações.

2.4.2 Qucsconv

Outra componente integrada e desenvolvida em simultâneo com o QUCS é o Qucsconv, responsável por fazer a conversão entre diferentes formatos de ficheiro, como .SCH, *netlists* e outros ficheiros de dados representantes de circuitos elétricos, de diferentes simuladores como *Simulation Program with Integrated Circuit Emphasis (SPICE)* e *Verilog-A*. Ou seja, é responsável por possibilitar a interpretação de ficheiros gerados por outros *softwares* de simulação e exportação dos circuitos gerados no QUCS para serem usados nos mesmos.

2.4.3 QUCS-S

Com o propósito de introduzir núcleos de simulação de circuitos SPICE na GUI do QUCS, surgiu o *Quite Universal Circuit Simulator with SPICE* (QUCS-S). Um *spin-off* do GUI original atualizado até a data de redação deste relatório, que não só possuí compatibilidade com o simulador próprio do QUCS, o Qucsator, como integra:

- ***Ngspice***, aquele que é recomendado pelos desenvolvedores do uma vez que é compatível com grande parte dos modelos industriais de SPICE e graças à sua performance.
- ***XYCE***, um simulador mais recente, compatível com SPICE, que assim com o Qucsator, possuí ferramentas de simulação e análise de circuitos *Radio Frequencies* (RF).
- ***SpiceOpus***, um simulador especializado em otimizar ciclos. Destacando-se pela sua eficiência.

Este disponibiliza, para além de todas as vantagens que vêm com o uso de modelos SPICE, também de todo o conjunto de funções e ferramentas disponíveis individualmente em cada núcleo.[11]

2.4.4 Módulos desenvolvidos no ISEP

No âmbito de Projeto / Estágio (PESTA) da Licenciatura em Engenharia Eletrotécnica e de Computadores (LEEC), foram desenvolvidos alguns módulos para integração no QUCS, estes incluem:

- Um ohmímetro, funcional em AC como medidor de impedâncias, que permite medir a resistência equivalente de conjuntos de resistências;
- Um wattímetro, funcional em AC como medidor de potências ativas, reativas e aparentes, tensão, corrente e fator de potência;
- Diagramas temporais e fasoriais em AC;

2.5 U=RIsolve

Como centro deste projeto temos o U=RIsolve, como mencionado no capitulo anterior, é uma ferramenta útil para resolução de circuitos, de maneira didática.

Acedendo `https://urisolve.pt/app/`, deparamo-nos com a página inicial da ferramenta.

Figura 2.1: Página inicial do U=RIsolve

Esta primeira secção apresenta todos os principais elementos que são utilizados durante a sua utilização normal:

1. A área de submissão de ficheiros, onde o utilizador deve submeter os ficheiros necessários.

2. A barra de navegação que por sua vez, contém atalhos para diferentes secções da página, o botão para acessar a funcionalidade do *login* e o controlo da linguagem da página.

3. As instruções de uso da ferramenta.

4. Botões referentes aos métodos de resolução disponíveis.

De modo a utilizar o U=RIsolve, o primeiro passo é submeter um circuito, na forma de uma *netlist*, obtida após a sua simulação no QUCS e, opcionalmente, uma imagem do circuito para que esta seja apresentada na resolução do mesmo. É possível, ainda, optar por utilizar um dos circuitos disponibilizados na secção intitulada "Exemplos", através da barra de navegação ou navegando manualmente para a mesma.

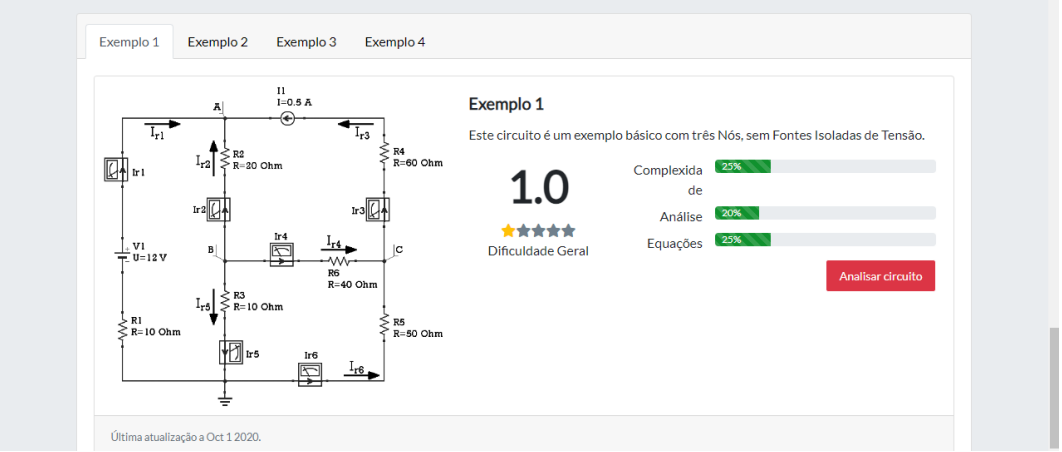


Figura 2.2: Secção de exemplos do U=RIsolve

Nesta estão presentes quatro circuitos, com variação na sua dificuldade de resolução, pressionando "Analisar Circuito", automaticamente, será feita a submissão da *netlist* e imagem do circuito selecionado.

Após a submissão é possível verificar que o ficheiro foi submetido através da lista dos ficheiros submetidos presente na área de submissão juntamente com, se submetida uma imagem do circuito, uma miniatura da mesma.

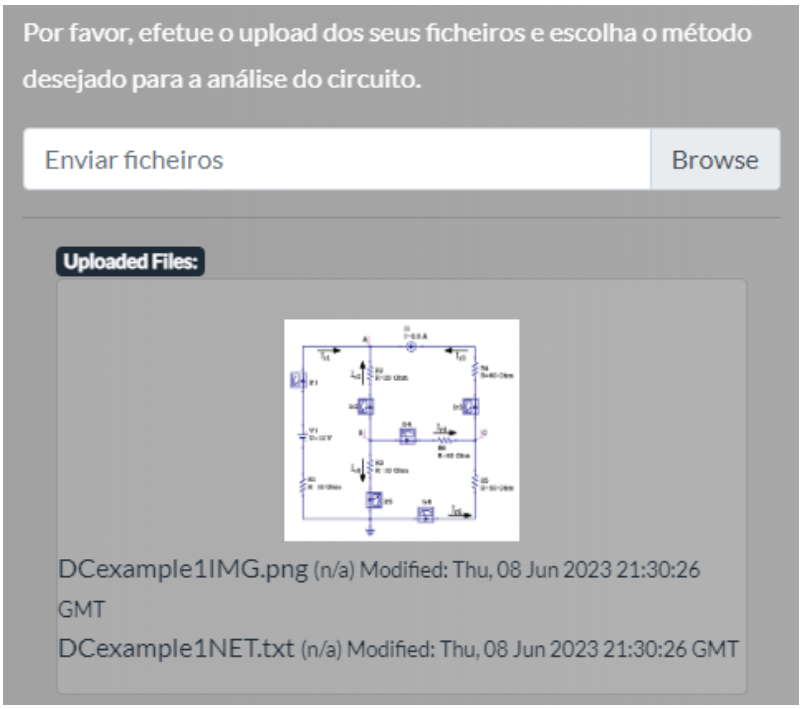


Figura 2.3: Submissão de ficheiro no U=RIsolve

O passo seguinte é selecionar o método de resolução utilizando um dos botões e, assim que estiver concluída a análise do circuito, é apresentada toda a informação

relevante à sua análise e resolução, juntamente com os resultados obtidos numa janela modal[2], como a demonstrada na figura seguinte.

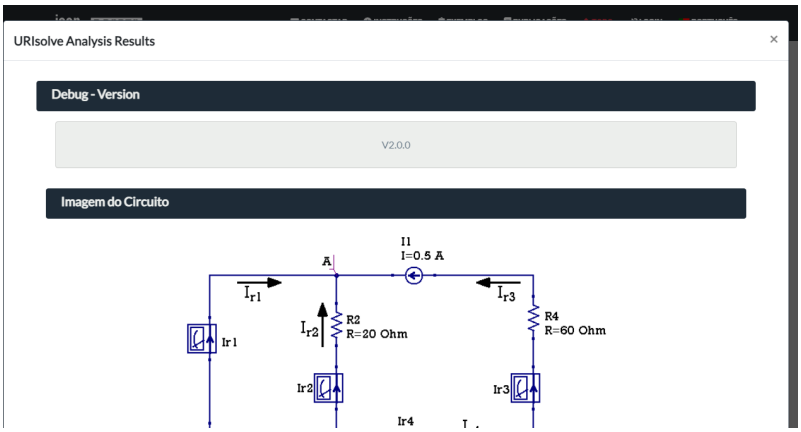


Figura 2.4: Janela modal do U=RIsolve

Os métodos disponíveis são MTN[12, 1, 13], MCM[14, 15] e MCR[16, 17], estando o último na sua versão *beta*.

[2]Uma janela de modal é uma caixa de diálogo que aparece sobreposta à página principal, exigindo interação do utilizador antes de continuar a interagir com a aplicação, geralmente apresentando um fundo escurecido ou desfocado. É usada para exibir informações relevantes ou solicitar ações específicas no contexto da aplicação.

Capítulo 3

Fundamentos Teóricos

O Teorema da Sobreposição é bastante útil como método de análise de circuitos. Então neste capitulo, será abordado mais a fundo em prol da valorização e melhor compreensão de toda a informação que o segue neste relatório.

3.1 Definição do Teorema da Sobreposição

O TSP baseia-se no conceito de que, tendo um circuito linear, composto por apenas elementos lineares cuja curva característica de tensão-corrente é uma reta em regime permanente, é possível fazer a sua análise operando segundo as leis da linearidade, incluindo a propriedade da aditividade[18]:

$$f(x_1 + x_2) = f(x_1) + f(x_2)$$

Ou seja, o resultado total do efeito dos elementos presentes é equivalente à soma do efeito (ou contribuição) de todos os elementos. Aplicando esta propriedade ao contexto relevante, podemos concluir que, num circuito, os efeitos nas correntes e tensões são devidos ao somatório das contribuições individuais de cada fonte. Sendo a contribuição de cada fonte sobre uma grandeza equivalente ao valor dessa grandeza quando apenas essa fonte está presente ou à diferença do valor da grandeza uma vez retirada essa fonte.

Então os passos de análise de um circuito pelo TSP, seriam:

1. Obter os circuitos parcelares. Para tal, todas as fontes cuja contribuição não se pretende incluir, devem ser substituídas pela sua impedância interna, no caso de fontes consideradas ideais, substituem-se por um curto-circuito ou um circuito-aberto, garantindo tensão ou corrente nula, dependendo se são fontes de tensão ou corrente, respetivamente;

2. Efetuar a análise da cada circuito parcelar, calculando as correntes e tensão pretendidas;

3. Somar algebricamente as contribuições das fontes, obtendo as correntes e tensões reais.

3.2 Aplicações relevantes

Analisar um circuito elétrico pode, por vezes, ser bastante desafiador, devido aos diferentes níveis de dificuldade introduzidos por diversos fatores. Um desses fatores é o número de fontes presentes no mesmo, utilizando o TSP, é possível lidar com esse problema ao simplificar circuitos complexos em vários circuitos mais simples. Para circuitos AC surge, ainda, a mais valia de possibilitar a análise de circuitos com diversas fontes com frequências diferentes, com o auxilio da Transformada de Steinmetz, o que não é possível nos métodos implementados anteriormente em que a frequência deve ser a mesma entre todas as fontes do circuito, evitando que estas operem em planos complexos distintos, o que teria impacto nas suas reactância/impedância devido à sua dependência do valor da frequência.

Ainda que este teorema introduza vantagens, para o seu uso existem mais alguns critérios:

- Evidentemente, o circuito deve possuir duas ou mais fontes de corrente ou tensão;

- Como mencionado anteriormente o circuito deve ser linear;

- As variáveis do sistema devem ser individuais, isto é, as fontes devem ser independentes umas das outras;

- O circuito deve permitir que as fontes sejam isoladas umas das outras, ou seja, todas as suas fontes devem ser independentes, para que seja possível determinar os seus efeitos individuais.

E para circuitos AC a esta lista adiciona-se ainda o facto de que o sistema deve ser não só linear como invariável no tempo, ou seja, a sua resposta deve se manter a mesma ao longo do tempo;

3.3 Exemplo ilustrativo

Tendo, por exemplo, o circuito seguinte, representado na figura 3.1 pretende-se calcular a tensão aos terminais da resistência R5:

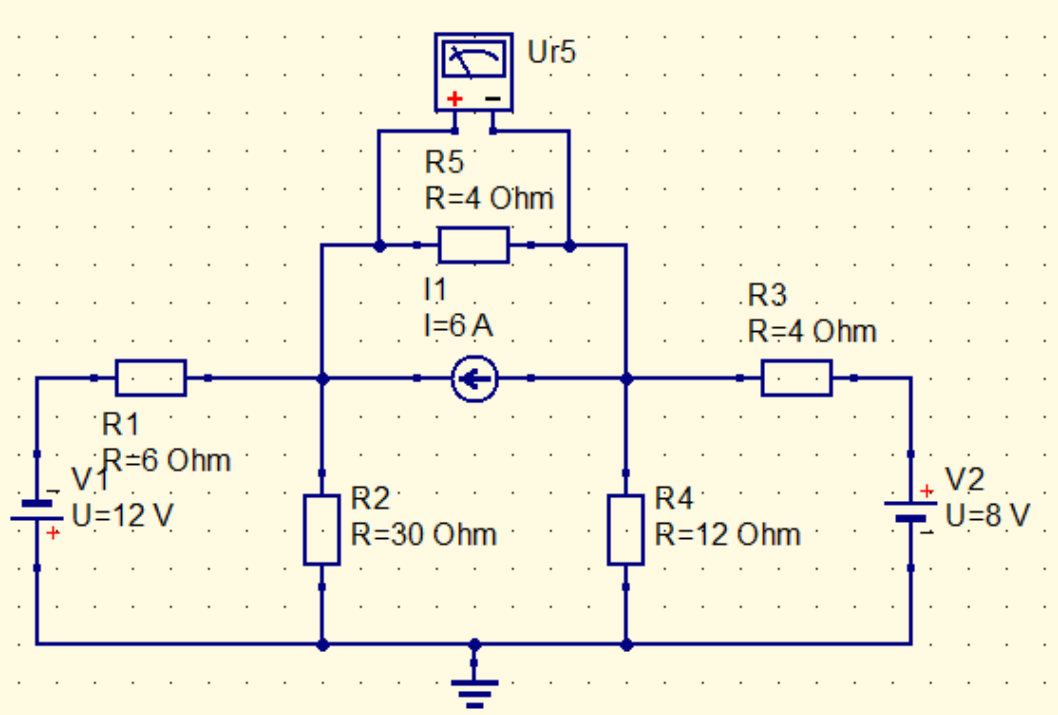


Figura 3.1: Circuito exemplo

Este é linear e possuí três fontes, V1, V2 e I1 pelo que é possível fazer a sua análise utilizando o TSP.

Seguindo os passos delineados anteriormente:

3.3.1 Contribuição de V1

1. Removendo V2 e I1, obtemos o circuito parcelar, seguinte:

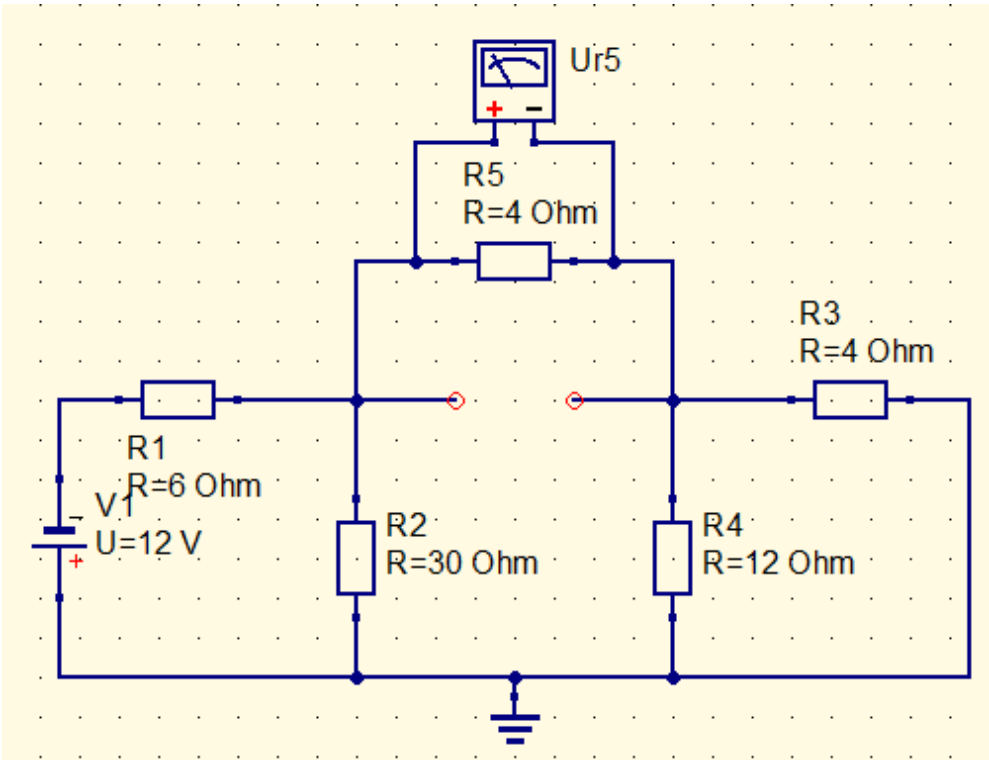


Figura 3.2: Circuito exemplo parcelar de V1

2. Analisar o circuito, calculando o valor da tensão em R5: $V_{R5} = -3,33$ V

3.3.2 Contribuição de V2

1. Removendo V1 e I1, obtemos o circuito parcelar seguinte;

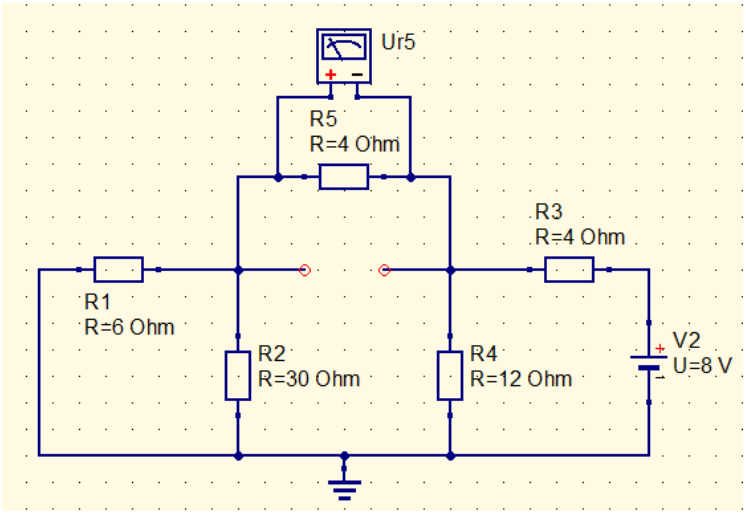


Figura 3.3: Circuito exemplo parcelar de V2

2. Analisar o circuito, calculando o valor da tensão em R5: $V_{R5} = -2$ V

3.3.3 Contribuição de I1

1. Removendo V1 e V2, obtemos o circuito parcelar seguinte;

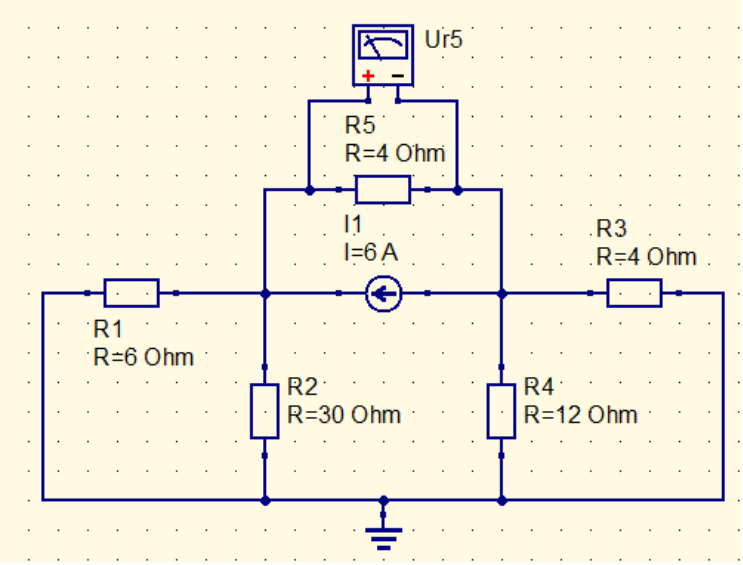


Figura 3.4: Circuito exemplo parcelar de I1

2. Analisar o circuito, calculando o valor da tensão em R5: $V_{R5} = 16$ V

3. Por questões didáticas, para sintetização das contribuições de cada fonte é recomendado utilizar uma tabela de contribuições como a representada na tabela abaixo (tabela 3.1):

Fonte	V_{R5} (V)
V1	-3,33
V2	-2
I1	16

Tabela 3.1: Tabela de contribuições exemplo

Por fim, somam-se as contribuições dos circuitos parcelares, para obter a tensão real.

$$V_{R5} = -3,33 - 2 + 16 = 10,7 \text{ V}$$

Capítulo 4

Desenvolvimento de Software

Durante o decorrer do projeto, vários algoritmos foram desenvolvidos para desempenhar diversas funções. Neste capítulo, estes vão ser delineados e explicados no contexto da estrutura do *software* criado.

Para auxiliar a compreensão, estes vão ser agrupados consoante a etapa do processo à qual pertencem, de maneira a manter uma estrutura mais sintética.

4.1 Parsing

O primeiro passo da etapa de desenvolvimento deste projeto foi a criação de um algoritmo de *parsing* do ficheiro ".sch" gerado pelo QUCS, gerando um ficheiro JSON que organiza toda a informação contida no mesmo, de maneira fácil de aceder e manipular, assim como, outras informações calculadas e deduzidas através das já existentes.

Este ficheiro contém bastante informação sobre o circuito. Para além de conter a informação presente na *netlist* relativamente aos componentes presentes, este possuí também informação concernente à representação do mesmo visualmente, como os dados da grelha, janela de visualização, coordenadas de todos os elementos envolvidos, dados sobre a criação do esquemático, como o título, autor, data e revisão, os diagramas, *wires* e outros elementos não incluidos na *netlist*. Assim, o uso deste ficheiro torna possível, posteriormente, gerar a representação visual do circuito, assim como, a *netlist* necessária para a análise deste, como demonstrado no diagrama da Figura 4.1.

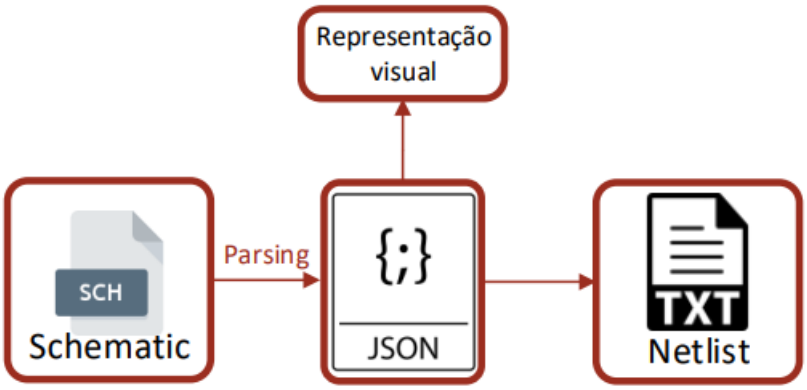


Figura 4.1: Diagrama de blocos do processamento do ficheiro .sch

4.1.1 O ficheiro .sch

O primeiro passo para criar este algoritmo, evidentemente, é perceber como está organizada a informação presente no ficheiro.

Este, tem uma estrutura bastante semelhante a de código em *Extensive Markup Language* (XML), que por sua vez está estruturado de maneira semelhante a HTML. Isto é, os dados são divididos em secções que são delineadas por etiquetas de marcação de abertura e de fecho, porém no ficheiro do esquemático, esta regra não se aplica aos dados em si, que simplesmente são delimitados por caracteres «"e »"e por uma nova linha.

Consideremos, como exemplo, o circuito utilizado anteriormente. Na Figura 4.2, temos o mesmo desenhado no QUCS, assim como o controlador de simulação e tabela que contém o valor "medido"pelo voltímetro Pr1, que corresponde à tensão calculada anteriormente.

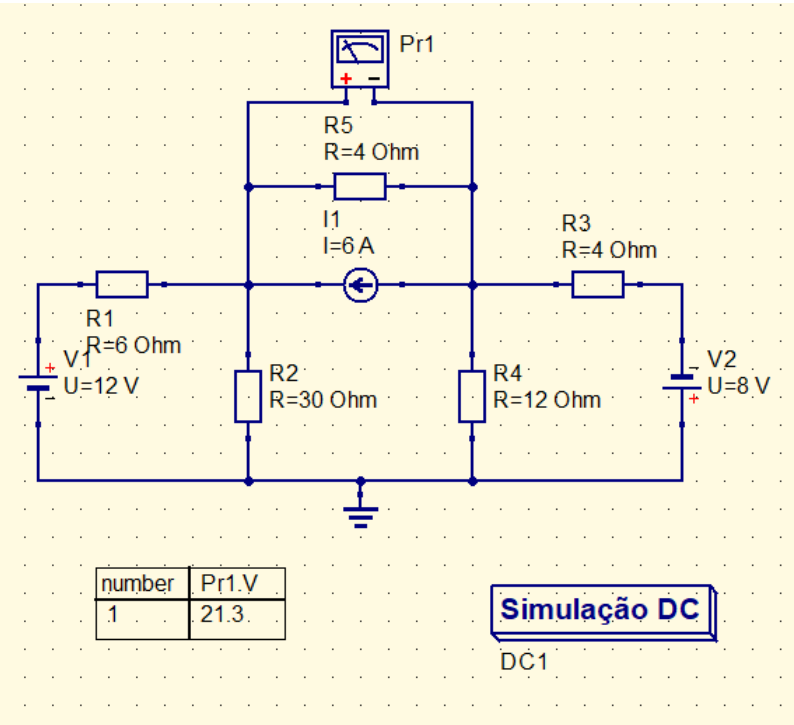


Figura 4.2: Circuito exemplo no QUCS

Desenhando este exato circuito no QUCS e salvando o ficheiro do esquemático num dado diretório do computador o ficheiro ".sch" correspondente é gerado. Este ficheiro encontra-se codificado em texto, pelo que é possível abri-lo com um editor de texto e verificar o seu conteúdo. Analisando a sua estrutura é possível verificar que este inicia com uma etiqueta de titulo, para a versão utilizada, "***<Qucs Schematic 0.0.19>***", seguida de **seis** secções: [19]

1. ***Properties***: este contém informações relevantes como a *View* e a *Grid*, assim como outras informações do documento do esquemático.

```
<Properties>
   <View=150,15,1095,558,1,0,0>
  <Grid=10,10,1>
  <DataSet=exemplo relatorio.dat>
  <DataDisplay=exemplo relatorio.dpl>
  <OpenDisplay=1>
  <Script=exemplo relatorio.m>
  <RunScript=0>
  <showFrame=0>
  <FrameText0=Título>
  <FrameText1=Autor:>
  <FrameText2=Data:>
  <FrameText3=Revisão:>
</Properties>
```

(a) ***View***: tem a estrutura, *<View=x1,y1,x2,y2,scale,xpos,ypos>*, em que os primeiros quatro números representam as coordenadas dos pontos referentes ao canto superior esquerdo e inferior direito do esquemático, *scale* representa a escala com que este é representado no QUCS e os últimos dois representam as coordenadas do canto superior esquerdo da janela de visualização dentro do mesmo.

(b) ***Grid***: demonstrado na forma, *<Grid=x,y,on>*, em que x e y representam o espaçamento horizontal e vertial da grelha, respetivamente e *on* representa o seu estado, 1 para quando é mostrada ou 0 para o oposto.

2. ***Symbol***: contém informação referente à representação de ficheiros embutidos no esquemático, por exemplo circuitos internos, este não sendo relevante para este contexto.

3. ***Components***: contém a informação de todos os componentes do circuito, incluindo controladores de simulação. Todos os tipos de componente relevantes vão ser abordados individualmente de seguida, devido à importância da compreensão da sua estrutura.

```
<Components>
  <GND * 1 420 390 0 0 0 0>
  <Idc I1 1 420 240 -26 -56 0 2 "6 A"1 "0 Ohm"0>
  <R R1 1 250 240 -26 15 0 0 "6 Ohm"1 "26.85"0 "0.0"0 "0.0"0 "26.85"0 "european"0>
  <R R2 1 340 320 15 -26 0 1 "30 Ohm"1 "26.85"0 "0.0"0 "0.0"0 "26.85"0 "european"0>
  <R R4 1 500 320 15 -26 1 3 "12 Ohm"1 "26.85"0 "0.0"0 "0.0"0 "26.85"0 "european"0>
  <R R3 1 590 240 -26 -53 0 2 "4 Ohm"1 "26.85"0 "0.0"0 "0.0"0 "26.85"0 "european"0>
  <R R5 1 420 170 -26 -53 0 2 "4 Ohm"1 "26.85"0 "0.0"0 "0.0"0 "26.85"0 "european"0>
  <VProbe Pr1 1 420 90 28 -31 0 0 "0 Ohm"0>
  <.DC DC1 1 520 460 0 40 0 0 "26.85"0 "0.001"0 "1 pA"0 "1 uV"0 "no"0 "150"0 "no"0 "none"0 "CroutLU"0>
  <Vdc V1 1 190 310 18 -26 0 1 "12 V"1 "0 Ohm"0>
  <Vdc V2 1 650 310 18 -26 1 3 "8 V"1 "0 Ohm"0>
</Components>
```

4. ***Wires***: contém a informação de todos os segmentos de reta que constituem as conexões do circuito.

```
<Wires>
  <280 240 340 240  0 0 0 >
  <340 240 340 290  0 0 0 >
  <500 240 560 240  0 0 0 >
  <500 240 500 290  0 0 0 >
  <450 240 500 240  0 0 0 >
  <340 240 390 240  0 0 0 >
  <340 170 390 170  0 0 0 >
  <340 170 340 240  0 0 0 >
  <450 170 500 170  0 0 0 >
  <500 170 500 240  0 0 0 >
  <500 350 500 380  0 0 0 >
  <340 380 420 380  0 0 0 >
  <340 350 340 380  0 0 0 >
  <420 380 500 380  0 0 0 >
  <420 380 420 390  0 0 0 >
  <340 110 410 110  0 0 0 >
  <340 110 340 170  0 0 0 >
  <430 110 500 110  0 0 0 >
  <500 110 500 170  0 0 0 >
  <190 380 340 380  0 0 0 >
  <190 240 220 240  0 0 0 >
  <620 240 650 240  0 0 0 >
  <650 240 650 280  0 0 0 >
  <650 340 650 380  0 0 0 >
  <500 380 650 380  0 0 0 >
  <190 340 190 380  0 0 0 >
  <190 240 190 280  0 0 0 >
</Wires>
```

Estes tem a estrutura *<x1 y1 x2 y2 "label"xlabel ylabel dlabel "node set">*, os primeiros quatro números representando as coordenadas do ponto inicial e final do segmento, *label* o texto da identificação presente no mesmo, *xlabel* e *ylabel* representam as coordenadas do mesmo, *dlabel* a distância entre o ponto onde está conectada a linha que liga o fio à sua identificação e o ponto de inicio do mesmo e por fim *nodeset* representa informações extra relevantes para a simulação do circuito como a tensão inicial do nó.

5. ***Diagrams***: contém todos os gráficos e tabelas presentes no esquemático. No caso do circuito exemplo a tabela que contém Pr1. Este não é utilizado no contexto deste projeto.

```
<Diagrams>
  <Tab 231 493 135 51 3 #c0c0c0 1 00 1 0 1 1 1 0 1 1 1 0 1 1 315 0 225    >
    <"Pr1.V"#0000ff 0 3 1 0 0>
  </Tab>
</Diagrams>
```

6. ***Paintings***: contém todos os desenhos existentes no esquemático, este pode incluir setas, linhas, texto, elipses, retângulos, etc. No contexto, deste projeto, estes não são relevantes.

Relativamente aos componentes anteriormente mencionados, todos seguem a seguinte fórmula: *<type name active x y xtext ytext mirrorX rotate* em que:

- ***type*** - o tipo de componente, i.e.: R para resistência, L para bobine, etc.;
- ***name*** - a referência do componente no circuito, i.e.: R1, R2, Vi, Vo, etc.;
- ***active*** - sinaliza se o componente está ativo, ou seja, se é usado na simulação. No contexto do projeto é utilizada esta flag para indicar se será utilizado. nos cálculos e representações;
- ***x*** e ***y*** - coordenadas do centro do componente;
- ***xtext*** e ***ytext*** - coordenadas da *label* do componente relativa ao seu centro;
- ***mirrorX*** - 1 se está espelhado no eixo do x, 0 se não;
- ***rotate*** - número de rotações no sentido anti-horário, ou seja, 0 corresponde a 0º, 1 a 90º, etc.

Estas propriedades são seguidas pelas propriedades próprias ao tipo de componente, no formato *"valor" visível* em que são representados o valor e se este está visível na *label* do componente.[19]

Parâmetros dos componentes

Os parâmetros de todos os componentes, except a terra, estão presentes na Tabela 4.1[1]:

Tabela 4.1: Tabela de parâmetros dos componentes

Propriedade	Formato	Valor padrão	Descrição
R - Resistência			
R	valor unidade	50 Ohm	Valor da resistência
Temp	valor	26.85	Temperatura de simulação (ºC)
TC1	valor	0.0	Coeficiente de temperatura de 1ª ordem
TC2	valor	0.0	Coeficiente de temperatura de 2ª ordem
Tnom	valor	26.85	Temperatura de extração dos parâmetros
Symbol	US/european	US/european	Símbolo no esquemático

[1]Para a versão do Qucs com os módulos desenvolvidos por alunos do DEE, os componentes de fonte e Amperímetro possuem ainda o parâmetro da impedância interna.

Propriedade	Formato	Valor padrão	Descrição
C - Condensador			
C	valor unidade	1 pF	Valor da capacitância
V	valor	-	Tensão inicial
Symbol	neutral/polar	neutral/polar	Símbolo no esquemático
L - Bobine			
L	valor unidade	1 nH	Valor da indutância
I	valor	-	Corrente inicial
Vdc - Fonte de tensão em corrente contínua			
U	valor unidade	1 V	Valor da tensão
Idc - Fonte de corrente em corrente contínua			
I	valor unidade	1 mA	Valor da corrente
Vac - Fonte de tensão em corrente alternada			
U	valor unidade	1 V	Valor da tensão de pico
f	valor unidade	1 GHz	Valor da frequência
Phase	valor	0	Fase inicial em graus
Theta	valor	0	Fator de amortecimento
Iac - Fonte de corrente em corrente alternada			
I	valor unidade	1 mA	Valor da corrente de pico
f	valor unidade	1 GHz	Valor da frequência
Phase	valor	0	Fase inicial em graus
Theta	valor	0	Fator de amortecimento
IProbe . Amperímetro			
Ri	valor unidade	0 Ohm	Valor da impedância interna
VProbe - Voltímetro			
Ri	valor unidade	0 Ohm	Valor da impedância interna

GND - Terra

O componente terra não possui propriedades e a sua referência é sempre "*".

4.1.2 Modelo de dados

Após analisar os dados disponíveis foi necessário arquitetar um modelo de dados, idealizado para organizá-los e quais outros dados serão necessários para completar o mesmo. Este foi baseado no modelo desenvolvido pelo colega Pedro Morim, durante o desenvolvimento do seu projeto *Circuit Editor*[20], no âmbito da unidade curricular de PESTA.

Objeto *Schematic*

A Tabela 4.2 sumariza a constituição do modelo de dados do objeto, biunívoco[2] ao esquemático, gerado durante o processo do *parsing*.

Class: Schematic			
Nível 0	**Nível 1**	**Nível 2**	**Descrição**
qucs-version	-	-	Versão do qucs
properties	view	x1	Coordenadas do canto superior esquerdo do esquemático
		y1	
		x2	Coordenadas do canto inferior direito do esquemático
		y2	
		scale	Escala do esquemático
		xpos	Coordenadas do canto superior esquerdo da janela
		ypos	
	grid	x	Espaçamento horizontal da grelha
		y	Espaçamento vertical da grelha
		on	Estado da grelha
components	-	-	Array de componentes
connections	-	-	Array de conexões
nodes	-	-	Array de nós

Tabela 4.2: Modelo de dados do esquemático

Objeto *component*

O vetor de componentes é constituído por objetos correspondentes a cada componente do circuito, gerado segundo com a disposição mostrada na Tabela 4.3.

[2]Que corresponde única e exclusivamente ao circuito e vice-versa.

Class: Component			
Nível 0	**Nível 1**	**Nível 2**	**Descrição**
id	-	-	ID único
type	-	-	Tipo do componente
active	-	-	Estado do componente
name	value	-	Referência do componente
	position	x	Coordenadas do nome em relação ao componente
		y	
value	value	-	Valor
	unit	-	
	visible	-	Visibilidade na label
position	x	-	Coordenadas do componente
	y	-	
	angle	-	Número de rotações
	mirrorx	-	Espelhamento
symbol	reference	-	Referência do símbolo do componente
	visible	-	Visibilidade na label
frequency	value	-	Frequência
	unit	-	
	visible	-	Visibilidade na label
phase	value	-	Fase
	unit	-	
	visible	-	Visibilidade na label
properties	impedance	value	Impedância
		unit	
		visible	
	temperature	value	Temperatura de simulação
		visible	
	tc1	value	Coeficiente de temperatura de 1ª ordem
		visible	
	tc2	value	Coeficiente de temperatura de 2ª ordem
		visible	
	tnom	value	Temperatura nominal
		visible	
	initialValue	value	Valor inicial de tensão/corrente
		visible	
	damping	value	Fator de amortecimento
		visible	
port (array)	net	-	A connection a que está conectado
	position	x	Posição do port
		y	
	connections (array)	component	id do component conectado
		port	Id do port 0/1
		wire	Id do wire conectado
		point	Ponto begin/end

Objeto *connection*

O vetor de conexões é constituído por objetos de conexão com a disposição mostrada na Tabela 4.4

<table>
<tr><th colspan="5">Class: Connection</th></tr>
<tr><th>Nível 0</th><th>Nível 1</th><th>Nível 2</th><th>Nível 3</th><th>Descrição</th></tr>
<tr><td>id</td><td>-</td><td>-</td><td>-</td><td>ID único</td></tr>
<tr><td rowspan="17">wires (array)</td><td>id</td><td>-</td><td>-</td><td>ID único do wire</td></tr>
<tr><td rowspan="6">begin</td><td>x</td><td>-</td><td rowspan="2">Coordenadas do ponto</td></tr>
<tr><td>y</td><td>-</td></tr>
<tr><td rowspan="2">connectedWires (array)</td><td>wire</td><td rowspan="2">Propriedades do ponto</td></tr>
<tr><td>point</td></tr>
<tr><td rowspan="3">connectedPorts (array)</td><td>component</td><td rowspan="3">Propriedades do porto</td></tr>
<tr><td>port</td></tr>
<tr><td>position</td></tr>
<tr><td rowspan="6">end</td><td>x</td><td>-</td><td rowspan="2">Coordenadas do ponto</td></tr>
<tr><td>y</td><td>-</td></tr>
<tr><td rowspan="2">connectedWires (array)</td><td>wire</td><td rowspan="2">Propriedades do ponto</td></tr>
<tr><td>point</td></tr>
<tr><td rowspan="3">connectedPorts (array)</td><td>component</td><td rowspan="3">Propriedades do porto</td></tr>
<tr><td>port</td></tr>
<tr><td>position</td></tr>
<tr><td rowspan="4">label</td><td>text</td><td>-</td><td rowspan="4">Propriedades da etiqueta</td></tr>
<tr><td rowspan="2">position</td><td>x</td></tr>
<tr><td>y</td></tr>
<tr><td>distance</td><td>-</td></tr>
<tr><td>node_set</td><td>-</td><td>-</td><td></td></tr>
<tr><td>wire</td><td>-</td><td>-</td><td>Fio a que pertence</td></tr>
<tr><td rowspan="5">ports (array)</td><td>component</td><td>-</td><td>-</td><td rowspan="5">Portos conectados</td></tr>
<tr><td>port</td><td>-</td><td>-</td></tr>
<tr><td rowspan="3">position</td><td>x</td><td>-</td></tr>
<tr><td>y</td><td>-</td></tr>
</table>

Tabela 4.4: Modelo de dados das conexões

Objeto *Node*

O vetor dos nós é constituído por objetos de nó com a disposição mostrada na Tabela 4.5

Class: Node		
Nível 0	**Nível 1**	**Descrição**
id	-	ID único
position	x	Posição do nó
	y	
name	-	Referência do nó
connection (array)	component	Propriedades dos componentes conectados
	port	
	wire	Propriedades dos *wires* conectadas
	point	

Tabela 4.5: Modelo de dados dos nós

4.1.3 Tratamento de dados

Com o modelo de dados delineado é possível fazer o *parsing* do ficheiro ".sch"e criar o ficheiro JSON. Porém, é ainda necessário trabalhá-lo para obter alguns dos dados que ainda estão em falta. Para isso foram criados outros algoritmos que calculam e deduzem os mesmos a partir dos já existentes, como por exemplo, cálculo da posição dos portos, ou conexões entre elementos.

Obter coordenadas dos portos

Após o *parsing* e de serem resolvidos os dados relativos às propriedades do esquemático, são criados os objetos dos componentes. Durante este processo são também calculadas as coordenadas dos seus portos.

Criando um simples circuito com o componente com ambos os seus portos conectados e analisando as coordenadas no ficheiro .sch gerado, (como demonstrado na Figura 4.3), podemos concluir que, para componentes como a resistência, condensador, bobine, amperímetro e as fontes de corrente e tensão AC e DC, a distância entre o centro do componente e cada porto individualmente é de 30 pixeis.

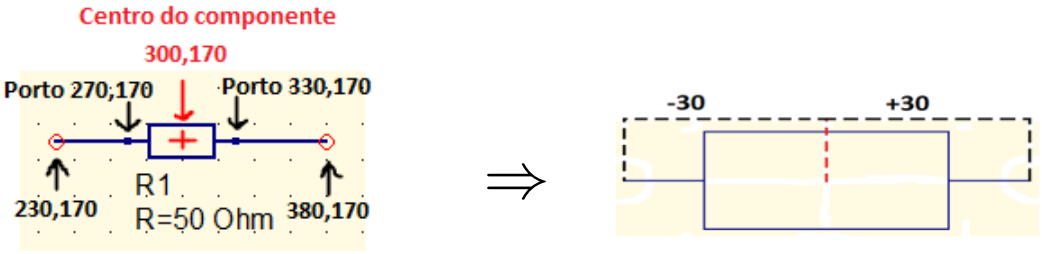


Figura 4.3: Análise das coordenadas de componentes do tipo R, C, L, IProbe, Vdc, Vac, Idc ou Iac

Utilizando o mesmo método, conclui-se que, para o componente terra, a coordenada do componente corresponde à coordenada do seu porto, como demonstrado na Figura 4.4.

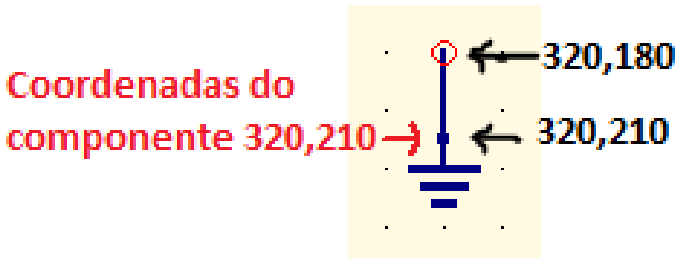


Figura 4.4: análise das coordenadas no componente terra

Por fim, pela análise representada na Figura 4.5, determinamos que o ponto médio entre os dois portos esta a 20 pixeis do centro e a 10 pixeis de cada porto.

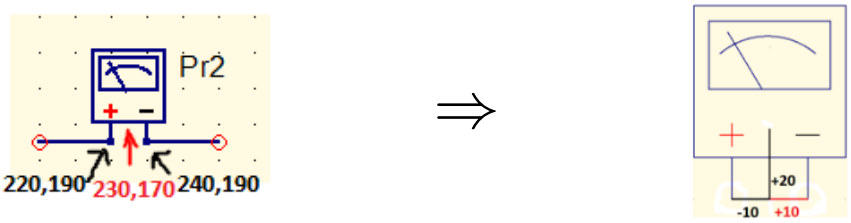


Figura 4.5: Análise das coordenadas num voltímetro

Por fim, considerando a rotação de cada componente, obteve-se o algoritmo demonstrado no diagrama presente na Figura 4.6.

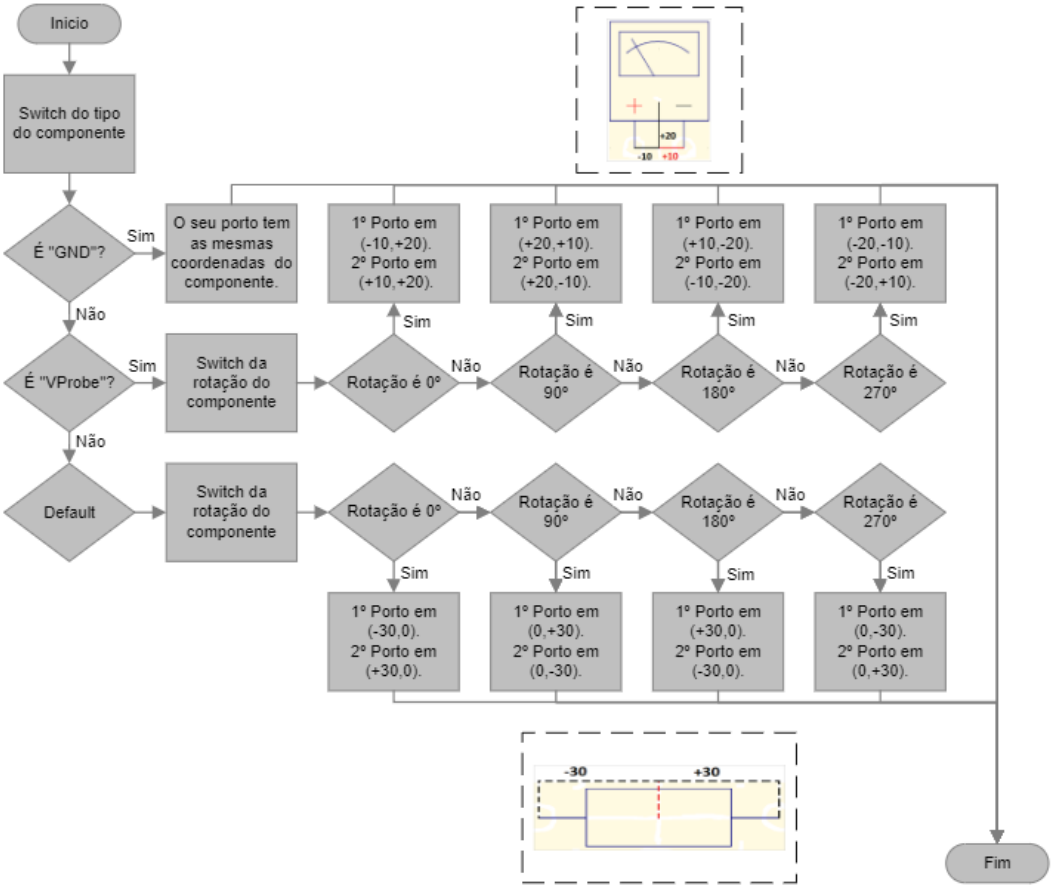


Figura 4.6: Fluxograma do algoritmo de cálculo das coordenadas dos portos

Determinar conexões, nós e fios

Assim que sejam gerados todos os vetores contendo os componentes e segmentos dos *wires*, com os dados dos portos dos componentes disponíveis, são deduzidos os dados relativos às conexões e nós. Na Figura 4.7 encontram-se os blocos da arquitetura do algoritmo que comanda todo este processo.

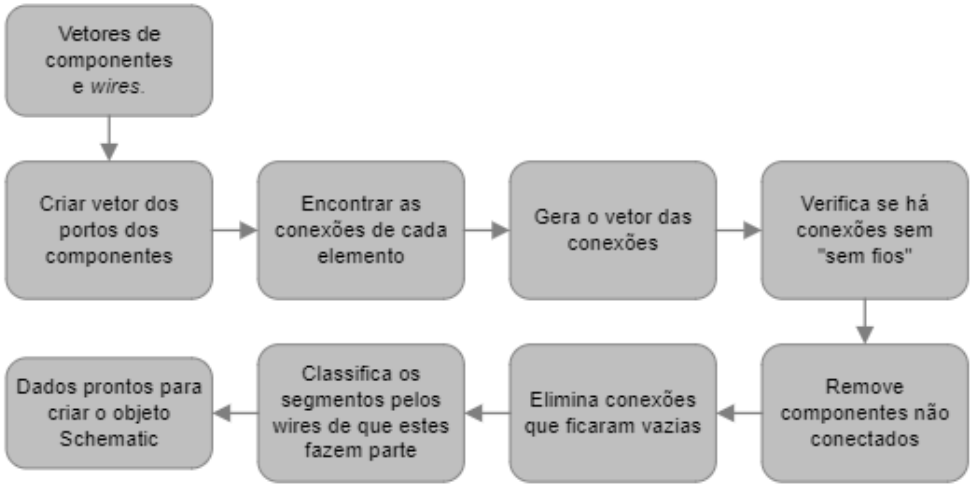


Figura 4.7: Diagrama de blocos do tratamento de dados

Obter conexões

Para encontrar as conexões dos elementos que constituem o circuito é utilizado o algoritmo que está descrito, de maneira simplificada, abaixo.

Algorithm 1 Encontrar conexões nos elementos do circuito

```
1: for cada segmento s sem conexões no circuito do
2:     for cada ponta p do segmento s do EncontrarConexões(p)
3: procedure EncontrarConexões(p)
4:     Tendo as coordenadas c, de p
5:     for cada ponto q sem conexões encontradas no circuito do
6:         Tendo as coordenadas d, de q
7:         if c for igual a d then
8:             Estabelecer conexão entre p e q
9:             if q for a ponta de um segmento then
10:                Encontrar a outra ponta de q, r
11:                EncontrarConexões(r)
12:    if p não tem conexões then RemoverFiosSoltos(p)
13:    else
14:        if p tem 2 ou mais conexões then
15:            Adiciona um nó real ao circuito
16: procedure RemoverFiosSoltos(p)
17:    Remover ponto p do circuito
18:    RemoverFiosSoltos(p)
19:    for cada ponto q com conexão a p no circuito do
20:        Remover conexão entre q e p
21:        if p era a única conexão de q then
22:            RemoverFiosSoltos(q)
```

No final desta etapa, após reunir uma lista de todos os segmentos que compõem a conexão e todos os portos conectados à mesma, serão adicionados os dados relacionados a cada ponto de conexão e criados os nós correspondentes, quando forem identificados três ou mais pontos com coordenadas idênticas.

Com estes dados é então criado o objeto *Connection* (segundo o modelo de dados dimensionado, Tabela 4.4), atribuindo-lhe uma referência consoante o algoritmo presente no diagrama abaixo (4.8). Isto é relevante visto que estes nomes serão depois usados na *netlist* que deve manter a identificação dos nós feita no QUCS.

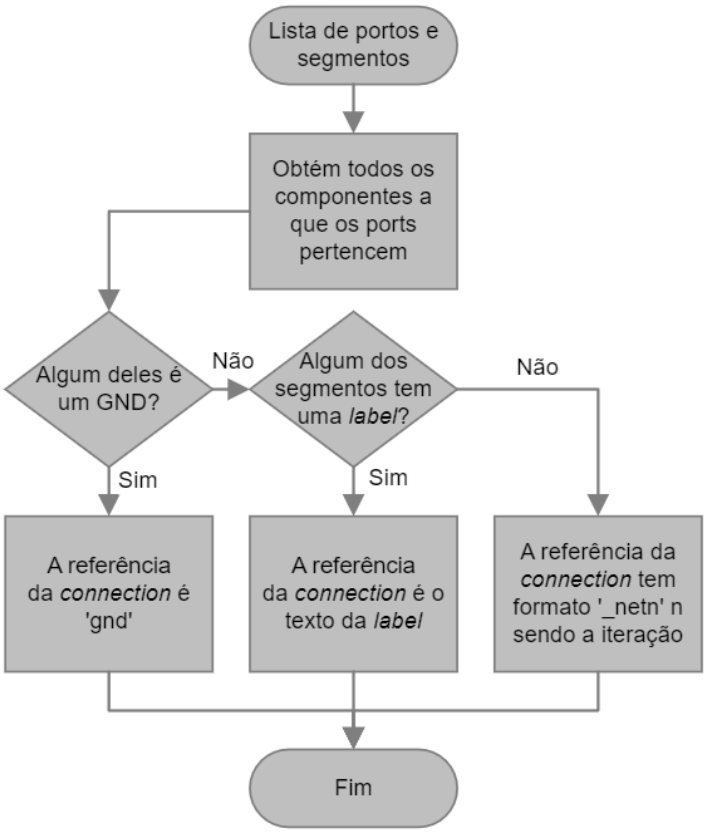


Figura 4.8: Fluxograma do algoritmo que determina a referência da conexão

Encontrar conexões sem fios

De seguida, são procurados portos que ainda permaneçam sem conexões. Verifica-se se estes se encontram conectados entre eles, criando nós quando necessário. Se no fim desta verificação não for encontrada qualquer conexão, o componente é removido e é corrido o algoritmo seguinte.

Remover elementos não conectados

Este recebe um componente ou segmento e remove-o das conexões de todos os outros elementos, através das suas próprias conexões,

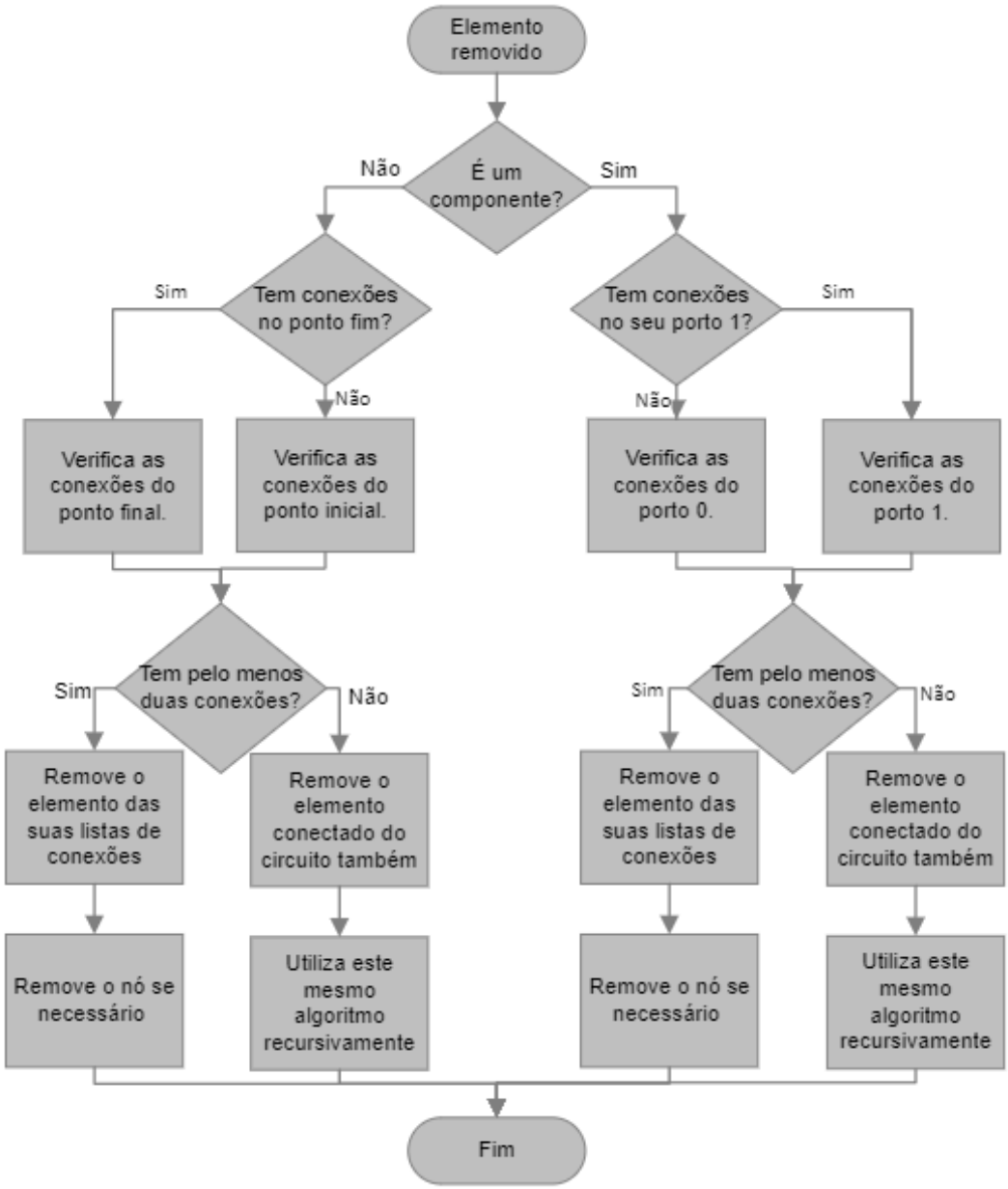


Figura 4.9: Fluxograma do algoritmo de remoção de ramos não conectados

Identificar fios

O passo final no tratamento dos dados é agrupar os segmentos em fios. Considerou-se um fio como um conjunto de segmentos que ligam portos de componentes ou nós a outros portos e/ou nós, como demonstrado na Figura 4.10. Classificou-se ainda, como uma conexão, um grupo de fios, e por vezes nós, que conectam dois ou mais portos de componentes, como demonstrado na Figura4.11.

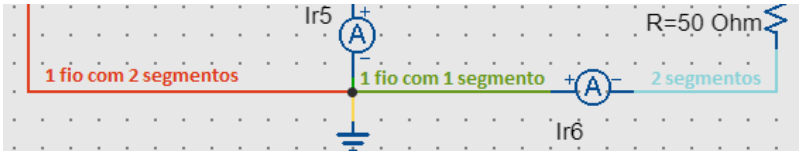


Figura 4.10: Representação visual dos fios com vários segmentos

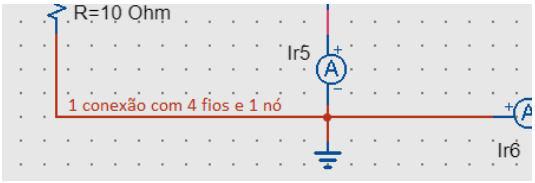


Figura 4.11: Representação visual de uma conexão

Para isso foi criado o algoritmo descrito pelo fluxograma representado na Figura 4.12.

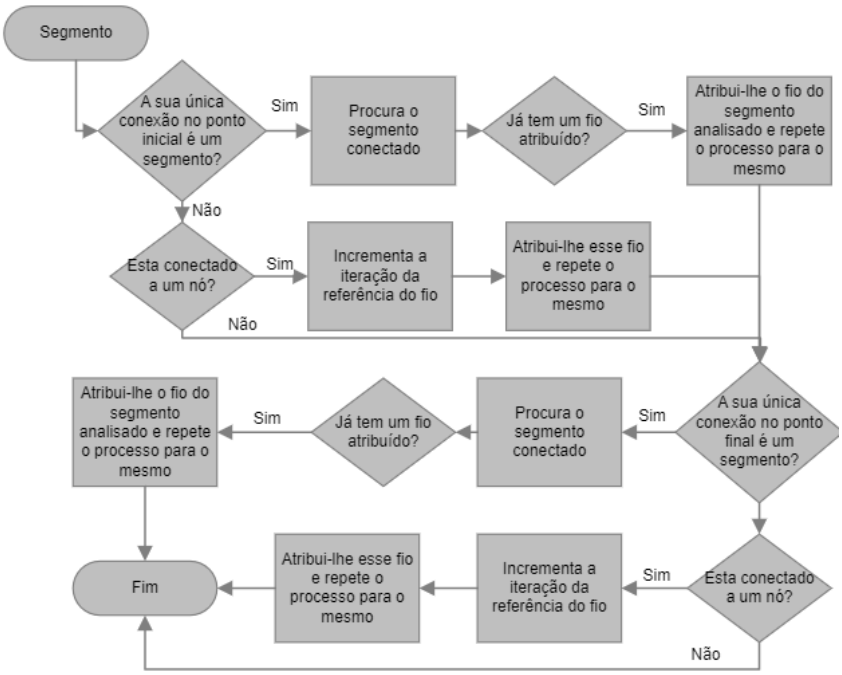


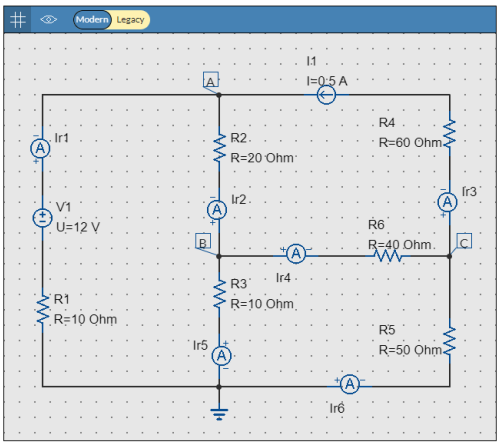
Figura 4.12: Fluxograma do algoritmo de identificação de fios

Recortar janela de visualização

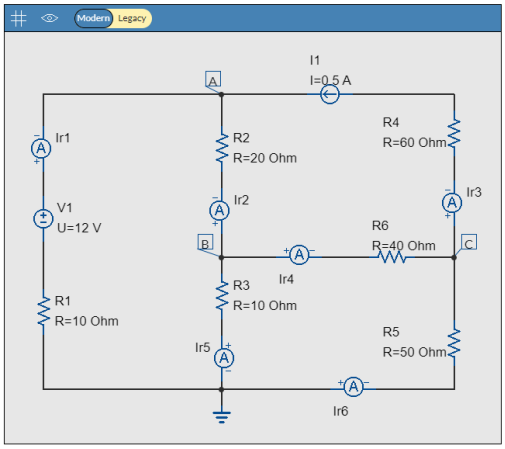
Um outro algoritmo desenvolvido que manipula o objeto do esquemático, influencia o processo de representação do circuito através do *widget*. Este determina os pontos de maior e menor coordenada x, e y, e reduz a janela de visualização para a mínima possível, mantendo um determinado espaçamento.

4.2 *Widget* de redesenho do circuito

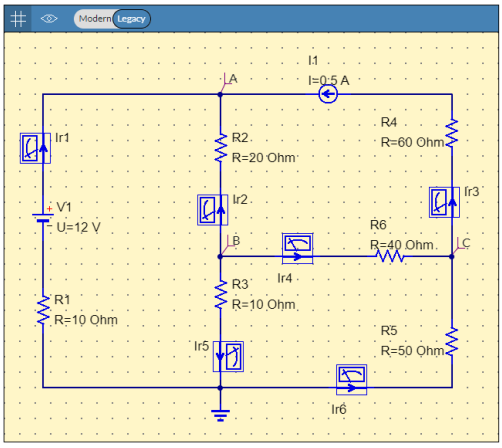
No âmbito da representação visual de circuitos, através do modelo de dados desenvolvido foi criado um *widget*, que não só representa o circuito, como é interativo. Este é constituído pela área de desenho, onde é desenhado o circuito, com todos os seus componentes, conexões, nós e as suas repectivas *labels*, bem como pela barra de ferramentas onde estão dispostos, o botão de *toggle* da grelha, o botão de *toggle* da visualização dos fios e o botão deslizante, que permite alterar o grafismo do widget. Existem dois temas gráficos: "*Modern*", o grafismo original do *widget* e "*Legacy*", o grafismo do QUCS, proporcionando assim uma gama de opções que o utilizador pode personalizar a seu gosto.



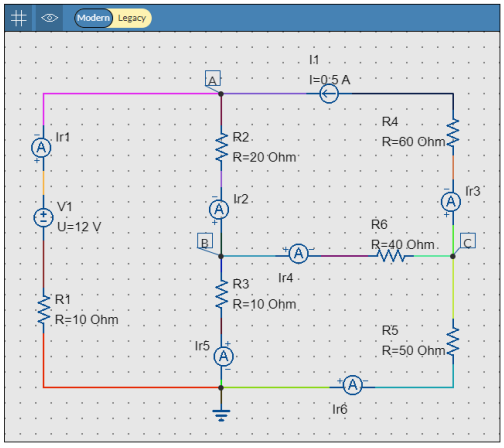
(a) Grafismo base



(b) Representação sem grelha



(c) Grafismo *Legacy*



(d) Representação dos fios do circuito

Figura 4.13: *Widget* de redesenho

O processo de criação do circuito pode ser, também, dividido em etapas, uma vez que este desenha os elementos por partes:

4.2.1 Dimensionamento da área de desenho

O processo de dimensionamento da área de desenho ocorre após a verificação da integridade do ficheiro *Schematic* e do elemento DOM passado para a função. A partir das propriedades da *View* do esquemático, retira as coordenadas dos cantos superior esquerdo e inferior direito e calcula a largura e altura da área de desenho, e aplica essas dimensão ao elemento em que se pretende desenhar o circuito, assim como, adiciona um novo contentor para conter todos os elementos do mesmo.

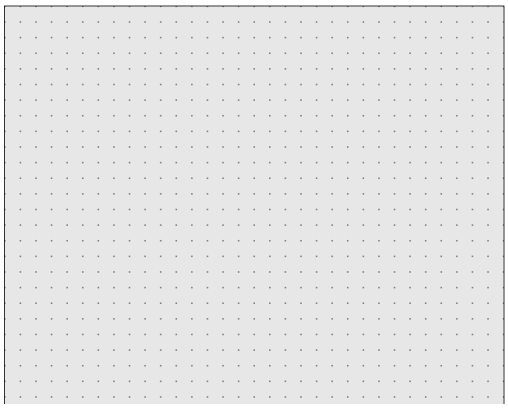


Figura 4.14: *Widget* de redesenho após a primeira etapa

4.2.2 Barra de ferramentas

A seguir é adicionada a barra de ferramentas e todos os seus botões.

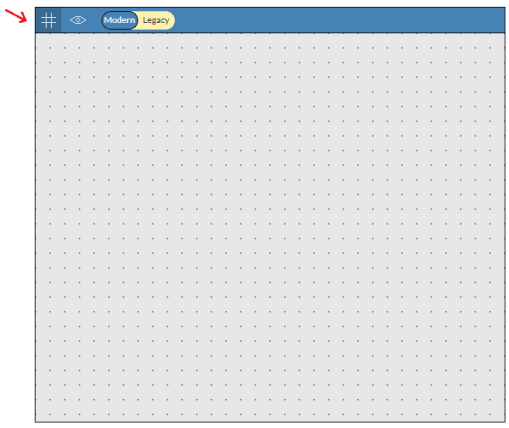


Figura 4.15: *Widget* de redesenho com barra de ferramentas

4.2.3 Desenho dos componentes

Em seguida são adicionados os componentes e as suas *labels*, utilizando um contentor e grelhas CSS para organizar os elementos que constituem cada componente, o símbolo e os seus portos, e classes para atribuir o seu aspeto e respetivos ícones.

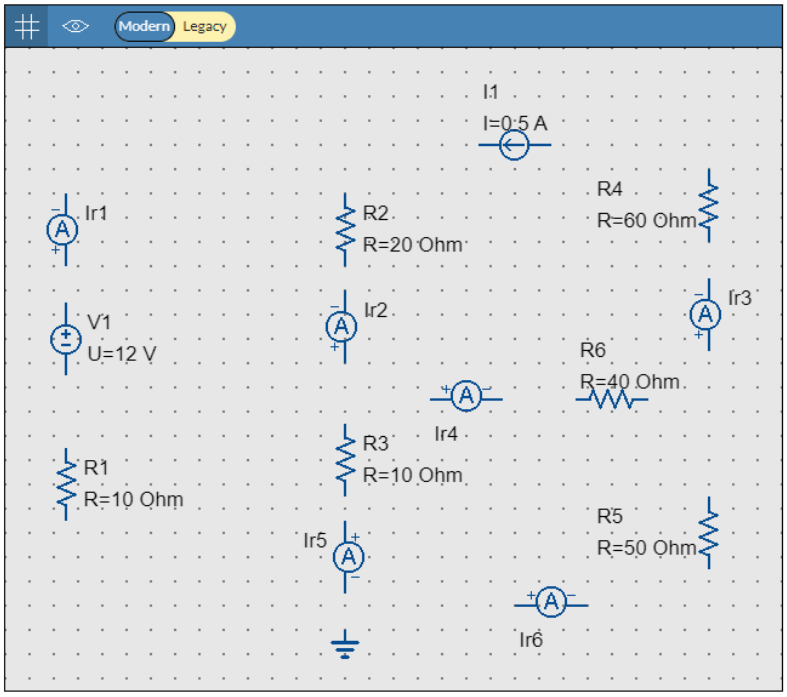


Figura 4.16: *Widget* de redesenho com componentes

4.2.4 *Labels* escondidas

Juntamente com cada componente, é adicionada uma *label*, visível no circuito em todos os momentos. Esta contém apenas os dados marcados como visíveis e pode ser arrastada livremente dentro da àrea de desenho. Para além disso, foi desenvolvida uma *label* escondida que aparece apenas quando se passa o cursor do rato por cima do componente, mostrando todas as propriedades do componente.

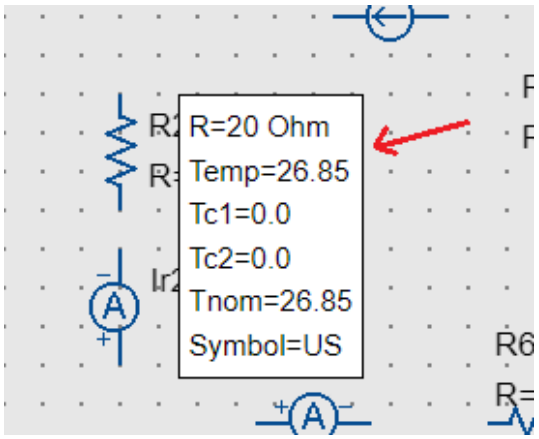


Figura 4.17: *Label* escondida

4.2.5 Menu de propriedades

O menu de propriedades surge com o evento *"double click"* e é mostrado no ecrã em sobreposição do esquemático. Permite alterar valores dessas propriedades e a sua visibilidade na *label*.

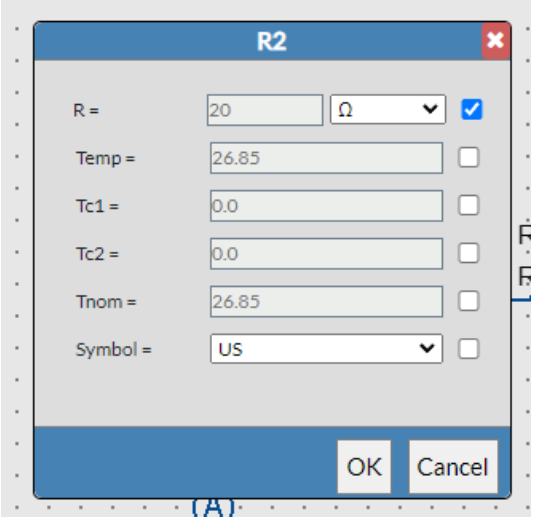
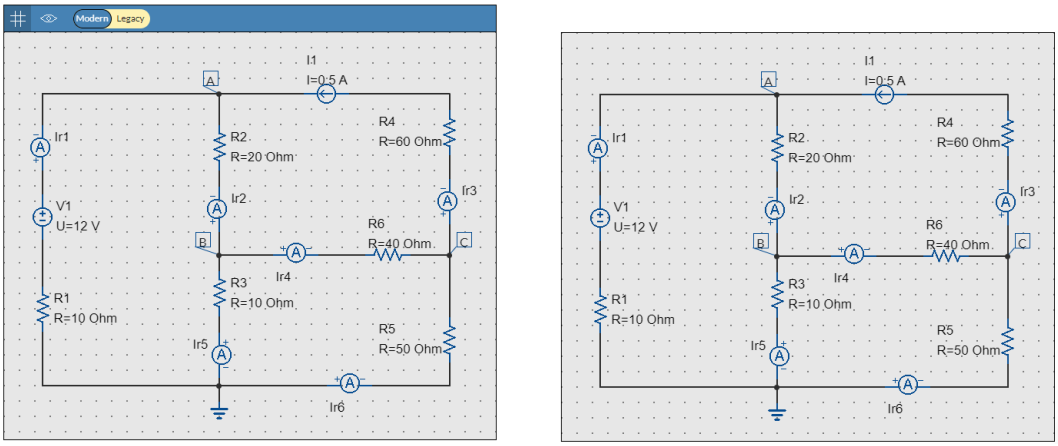


Figura 4.18: Menu de propriedades do componente

4.2.6 Fios, nós e as suas *labels*

Por fim são adicionados todos os segmentos de reta, representando assim as conexões do circuito, são adicionadas as suas *labels*, que também podem ser posicionadas livremente na àrea de desenho, e os nós, obtendo assim o circuito representado na Figura 4.13a.

De maneira a dar a este *widget* uma gama maior de usos, foi ainda adicionada a possibilidade de desabilitar aspetos como a sua interatividade, passando o valor *"false"* para o argumento *"interactive"* o circuito será gerado sem barra de ferramentas e sem possibilidade de mover ou alterar aspetos do mesmo, assim como, sem *labels* escondidas.



(a) Interatividade ativada (b) Interatividade desativada

Figura 4.19: Variação da interatividade

Também é opcional, a possibilidade da alteração das propriedades dos componentes através do menu de propriedades do mesmo, passando o argumento *"mutable"* como *false* para a função que gera o *widget*. No menu passa a ser apenas possível escolher quais são mostradas na *label*, como demonstrado na Figura 4.20.

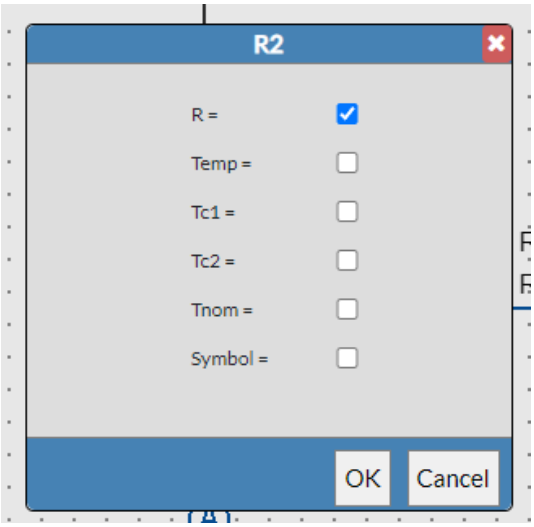


Figura 4.20: Menu de propriedades sem alteração de valores

4.3 Netlist

A *netlist* é um ficheiro de texto que descreve um circuito. Este apresenta informações sobre os componentes que constituem o circuito, as suas propriedades e os nós que os interligam. É utilizado como entrada em programas de simulação de circuitos de maneira a analisar e prever o seu comportamento, podendo ser escritos em formatos como *SPICE, Verilog* e *VHDL*. No QUCS, as *netlists* podem ter formato *SPICE*, fazendo uso do QUCSCONV [21], porém este possuí o seu próprio formato. Um aspeto importante deste é a representação dos nós aos quais o componente está conectado como nó positivo e nó negativo, o que permite verificar a polaridade dos elementos ativos do circuito.

A geração de ficheiros *netlist* tem como fonte de informação o ficheiro JSON.

Para tal, primeiro foi preciso estudar o ficheiro *netlist*. Este começa com uma primeira linha de comentário, linha iniciada com o caractér "#"[22], que contém informação relativa à versão do QUCS usada e o caminho para o ficheiro .sch que a originou, no formato ***# Qucs 0.0.19 .../Exemplo TSP.sch***. As restantes das linhas correspondem aos componentes que constituem o circuito no formato ***tipo:referência noP noN propriedade="valor"***. Por exemplo, a linha:
"R:R1 B C R="50 Ohm"Temp="26.85"Tc1="0.0"Tc2="0.0"Tnom="26.85", corresponde a um componente do tipo 'R', i.e., uma resistência, com referência R1, ligada entre o nó B e o nó C, sendo B o seu nó positivo e C o seu nó negativo, de valor 50 Ohm, temperatura de simulação 26.85ºC, etc.

Este formato é semelhante para todos os componentes, exceto a terra que não é representada neste ficheiro. Na representação de fontes de tensão AC e DC é criado um nó virtual, colocado no lugar do seu nó positivo, e é adicionada a sua resistência interna, como um componente do tipo R ligado entre esse nó virtual e o nó positivo da fonte. Por exemplo, esta fonte de tensão conectada entre o nó B e a terra é representada da seguinte maneira,
Vdc:V2 B_TEMP_V2 gnd U="20 V"Ri="50 Ohm"R:_R_B_TEMP_V2 B_TEMP_V2 B R="50 Ohm"Temp = "26.85"Tc1 = "0.0"Tc2 = "0.0"Tnom = "26.85". Os amperímetros funcionam de maneira semelhante, porém a sua resistência interna é colocada no seu nó negativo.

Assim, para gerar a netlist, basta organizar a informação já presente no objeto JSON no formato demonstrado.

4.4 Circuitos Parcelares

A resolução por TSP tem como base a criação de circuitos parcelares, de maneira a calcular a contribuição das diversas fontes para as grandezas do circuito. Assim é essencial a existência de um algoritmo que seja capaz de criar os ficheiros JSON dos respetivos circuitos parcelares, para que estes possam, posteriormente, ser usados para representação dos mesmos, assim como a criação da *netlist* passada para os métodos de resolução.

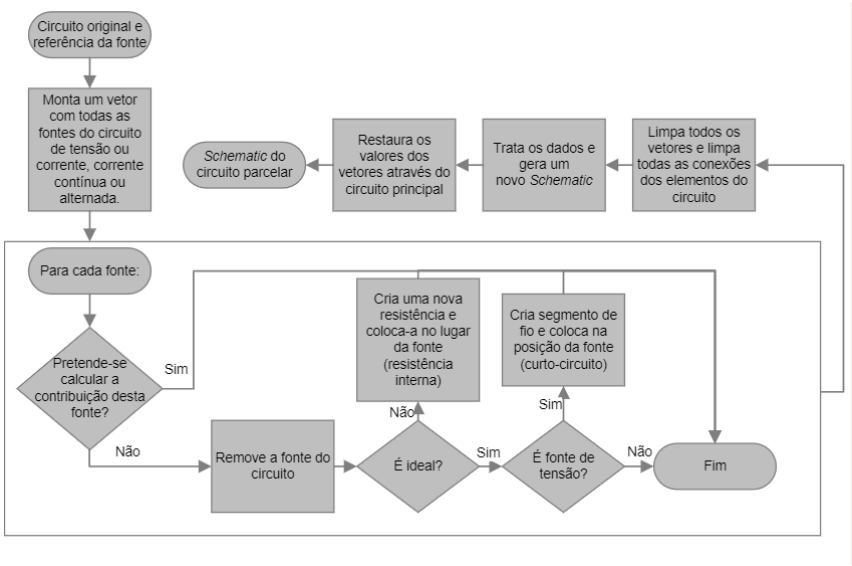
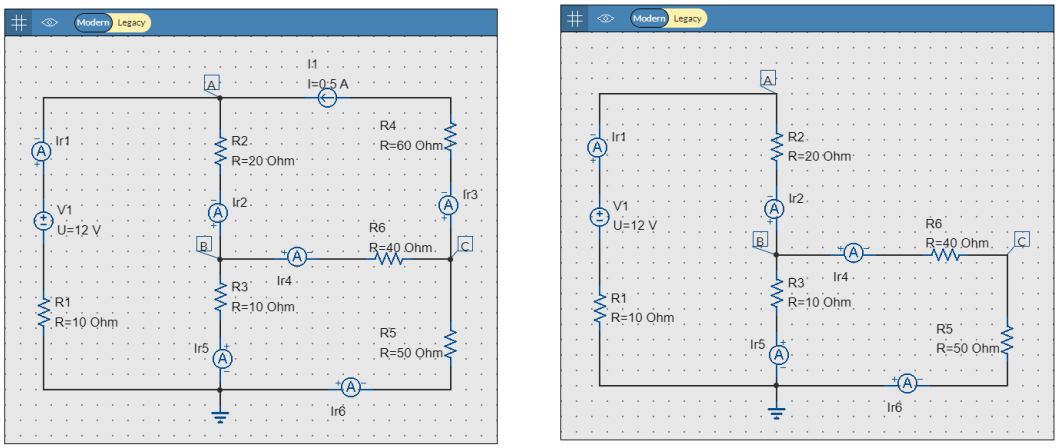


Figura 4.21: Fluxograma do algoritmo que gera o objeto JSON do circuito parcelar

Assim é possível obter circuitos parcelares, como, por exemplo:



(a) Circuito Principal

(b) Circuito parcelar, contribuição de V1

Figura 4.22: Circuito parcelar gerado a partir do algoritmo

4.5 Resolução de circuitos

Com todos os algoritmos auxiliares desenvolvidos, foi então desenvolvido o algoritmo de resolução TSP. Este tem por base os algoritmos anteriormente desenvolvidos por colegas em anos anteriores, não só pelo uso de funções já existentes, como pelo uso de uma estrutura semelhante, por motivos de consistência e reaproveitamento do código já desenvolvido.

Este recebe os dados do circuito, tanto no modelo de dados implementado como o já existente, juntamente com a ordem de resolução e os seus métodos, gera os circuitos parcelares, obtém as contribuições, faz o cálculo da sua soma e adiciona ao ficheiro JSON os dados de resolução, assim como demonstrado na Figura 4.23.

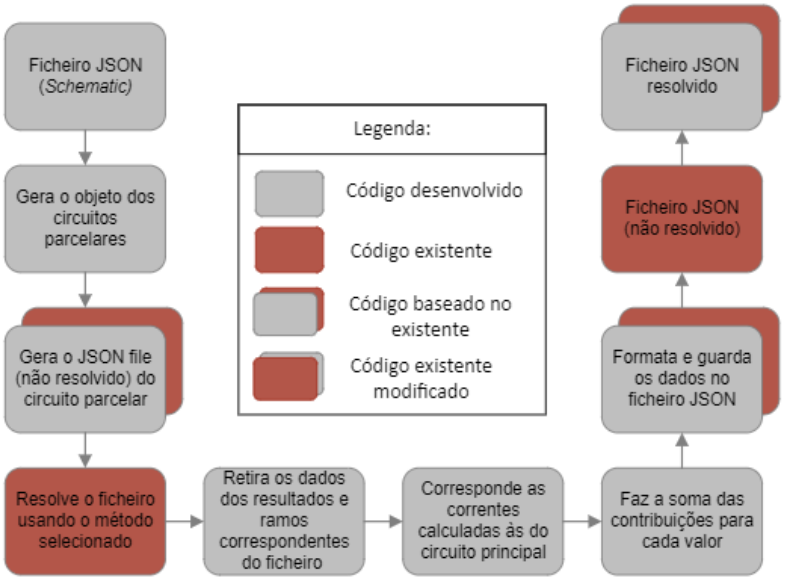


Figura 4.23: Diagrama de blocos do algoritmo do TSP

Capítulo 5

Implementação

Com todos os algoritmos desenvolvidos e funcionais, o último passo do desenvolvimento do projeto foi a implementação dos mesmo no código já existente no repositório do **U=RIsolve** de maneira a adicionar as funcionalidades pretendidas.

Reiterando, então, os objetivos anteriormente delineados, atualmente, da perspetiva do utilizador, este desenha o circuito no QUCS, executa a simulação e guarda a *netlist*, faz a submissão da mesma no site e seleciona o método de resolução, obtendo a resolução e os resultados com a possibilidade de gerar e guardar estes dados no formato do ficheiro JSON resolvido, ficheiro LaTeX via *download* ou Overleaf, ou guardar o ficheiro como PDF.

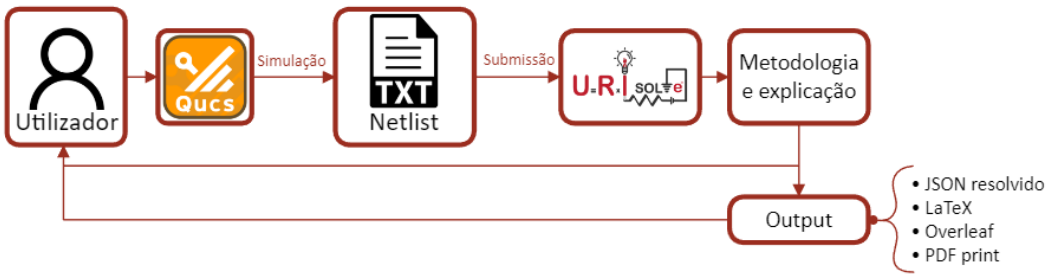


Figura 5.1: Funcionamento atual do ponto de vista do utilizador

Um dos objetivos do projeto é simplificar a experiência do ponto de vista do utilizador, reduzindo o trabalho necessário antes da submissão do ficheiro e abrindo a possibilidade de introdução de novas possíveis funcionalidades. Assim, por utilização

do do ficheiro ".sch", este apenas precisa de desenhar o circuito e submeter o ficheiro remetente ao mesmo.

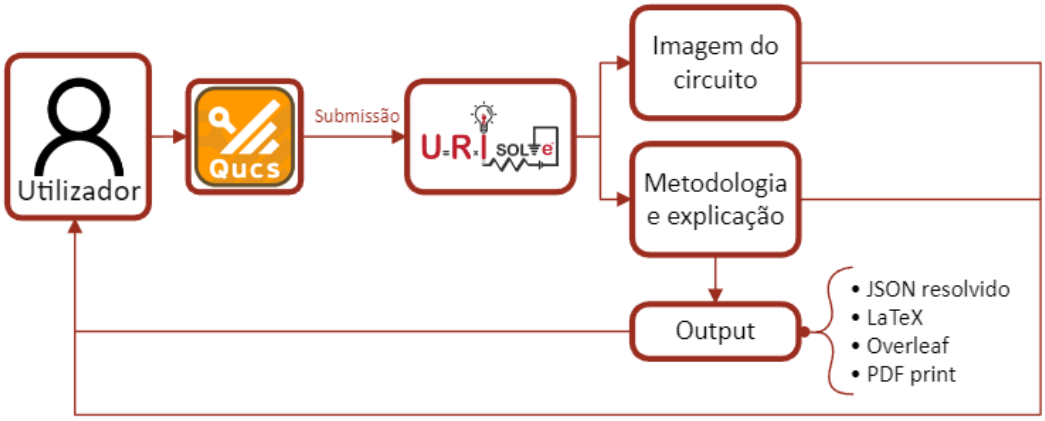


Figura 5.2: Funcionamento objetivo do ponto de vista do utilizador

5.1 Funcionamento por .sch

A primeira etapa da implementação foi alterar o código já existente no repositório da aplicação para funcionar com a submissão do ficheiro .sch ao invés da *netlist* e resolver qualquer incompatibilidade resultante da inclusão dos algoritmos desenvolvidos anteriormente no mesmo. Isto engloba, o processo de carregamento do ficheiro e a resolução utilizando os métodos já existentes, de maneira a poder reutilizar funções existentes.

5.1.1 Carregamento do ficheiro

Foi criado novo código para o processo de carregamento do ficheiro, baseado no já existente, mas ao invés de fazer as verificações para autorizar apenas um ficheiro de *netlist* e, opcionalmente, uma imagem, este passa a aceitar apenas um ficheiro .sch.

5.1.2 Miniatura

Mantém a demonstração da lista de ficheiros, neste caso ficheiro mas substitui a miniatura da imagem, se submetida, passando a ser representado qualquer circuito carregado, pelo uso do *widget* de redesenho com a sua interatividade desativada, como demonstrado na figura 5.3.

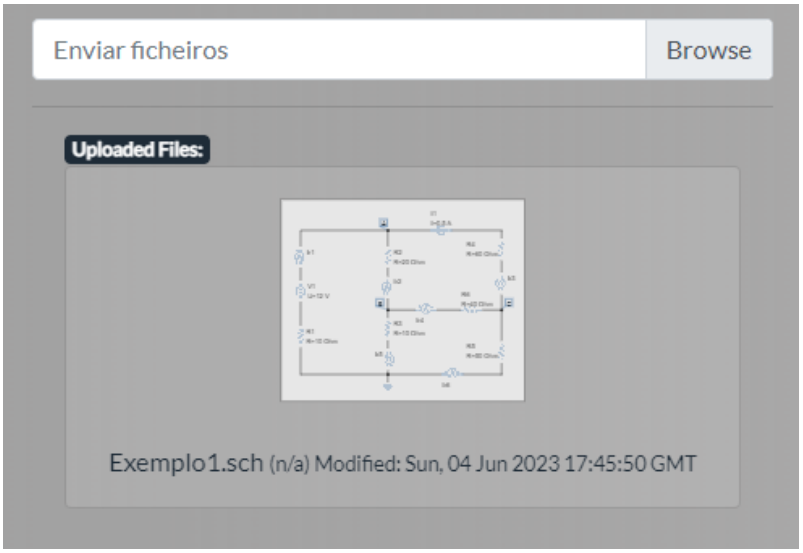


Figura 5.3: Submissão do ficheiro .sch

5.2 Resolução pelo Teorema da Sobreposição

Selecionando o teorema como método de resolução, assim como em outros métodos, é aberta a janela de modal.

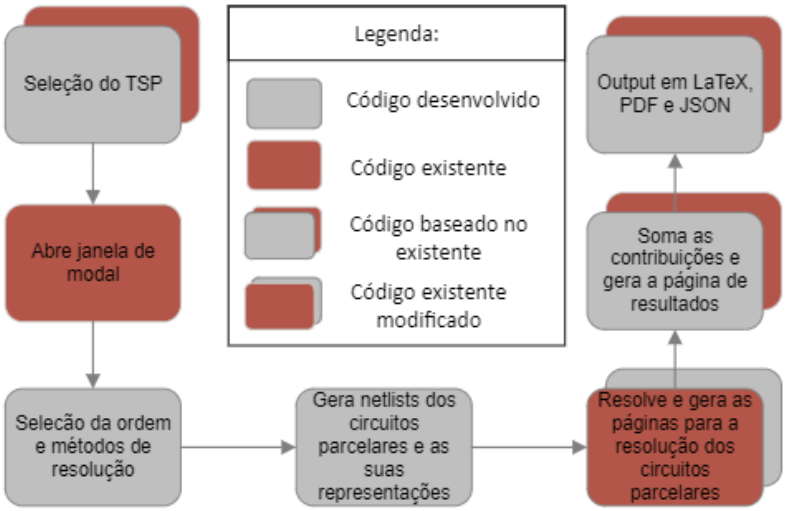


Figura 5.4: Passos de resolução por TSP

5.2.1 Página Inicial

Inicialmente esta janela apenas contém uma página inicial utilizadas para seleção do método e ordem de resolução, visualização do circuito e outras funcionalidades e configurações a fazer antes de resolver o mesmo.

Esta página está dividida em duas secções:

1 - Imagem do circuito

Esta é constituída principalmente por um *widget* de imagem do circuito com interatividade e possibilidade de alteração das propriedades dos componentes. Neste é possível alterar aspetos do circuito, tanto visualmente, como propriedades dos componentes, assim como, verificar o mesmo. Esta secção também é usada em funções da seguinte secção.

2 - Seleção do método de resolução

A secção principal desta página, esta é utilizada para seleção do modo e configuração da resolução.

O processo de seleção e configuração da resolução começa pela escolha entre o **modo automático** (figura 5.5) e **modo manual** (figura 5.6).

Com o modo automático o utilizador opta por não efetuar qualquer configuração do método ou ordem de resolução. Tendo como único passo pressionar o botão "Calcular", para obter a resolução. Esta será feita pela ordem com que as fontes estão disponíveis no ficheiro do esquemático e pelo MCR.

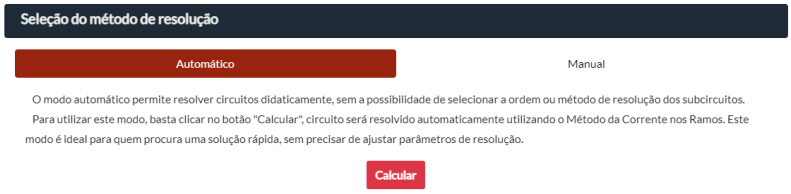


Figura 5.5: Modo Automático

O modo manual, por outro lado, permite a personalização da ordem e método de resolução de cada circuito parcelar individualmente.

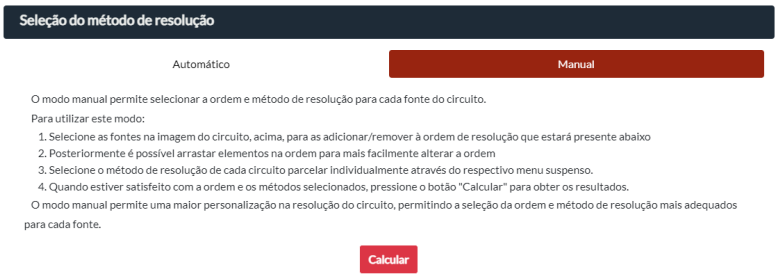


Figura 5.6: Modo Manual

Ao selecionar o mesmo, alguma alterações surgem na secção anterior, como demonstrado na figura seguinte, as fontes do circuito são destacadas e um contador do número de fontes selecionadas passa a ser mostrado.

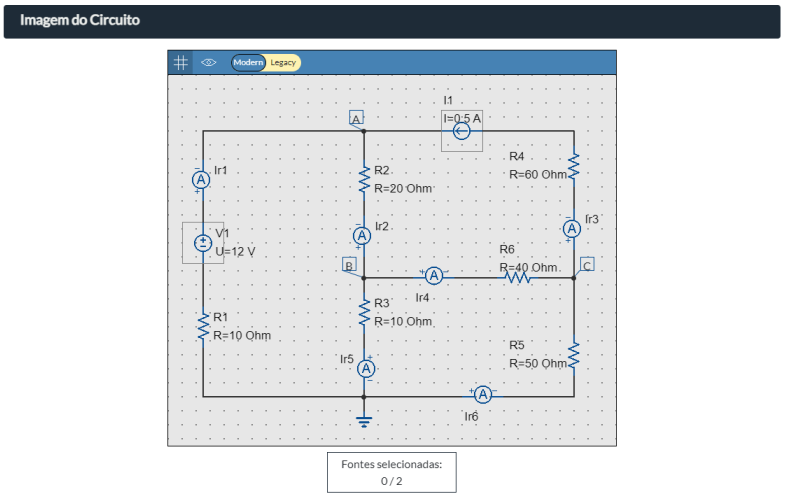
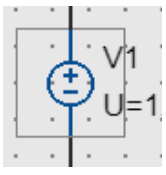
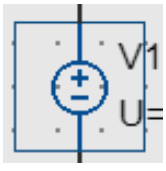


Figura 5.7: Secção da imagem do circuito em modo manual

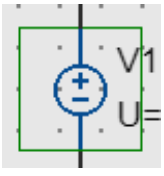
Seguindo os passos de utilização deste modo, o utilizador deve então selecionar as fontes. Esta seleção é feita com um clique na fonte desejada, passando a mesma a ter um contorno verde, ao invés de cinza. A inclusão de fontes incrementa a percentagem de progresso, indicando ao utilizador o estágio em que se encontra.



(a) Fonte selecionável



(b) Fonte *on hover*



(c) Fonte selecionada

Figura 5.8: Seleção de fontes

Ao selecionar uma fonte, esta irá ser adicionada à ordem de resolução na forma de um cartão.

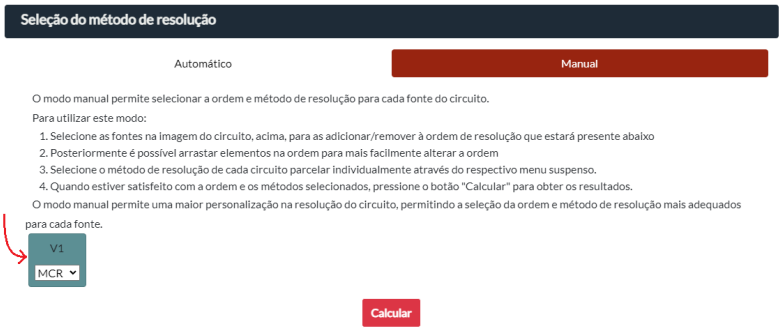


Figura 5.9: Ordem de resolução

Estes cartões são constituídos pela identificação da fonte e o método selecionada que pode ser alterado no menu suspenso disponível.

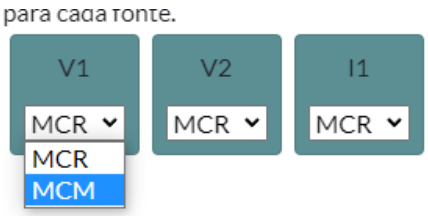


Figura 5.10: Cartão de ordem de seleção

Para evitar ter que remover e adicionar cartões novamente para alterar a ordem selecionada, é possível ainda arrastar estes de maneira a reorganizá-los.

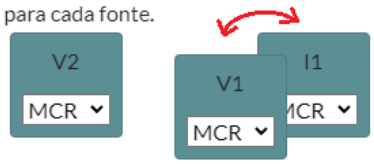


Figura 5.11: Reorganização de cartões

Quando o utilizador se encontrar satisfeito com a ordem selecionada deve então, finalmente, pressionar o botão "Calcular".

Depois de pressionado é sempre possível, consultar esta página para visualizar o circuito, alterar as suas propriedades e alterar a ordem de resolução.

5.2.2 Resolução dos circuitos parcelares

Após pressionar o botão "Calcular", são feitos todos os cálculos necessários utilizando os algoritmos de resolução e representação dos diversos circuitos parcelares.

E são geradas todas as abas de resolução, elementos que permitem alternar entre as diferentes páginas da análise do circuito, relativas à análise individual de cada circuito parcelar e de cálculo e representação de resultados.

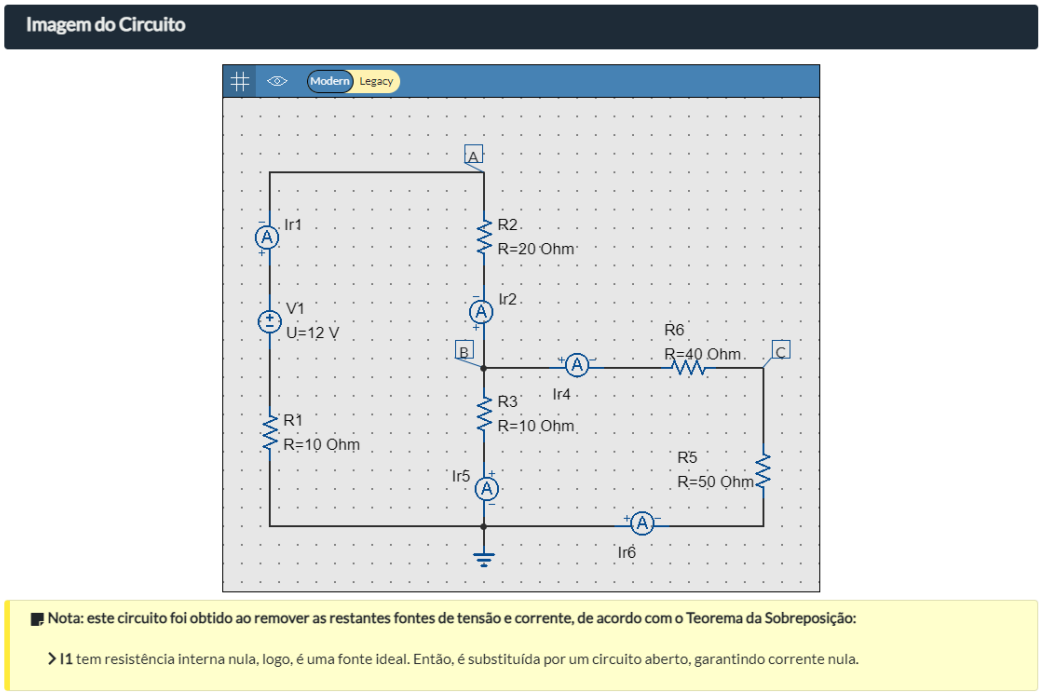


Figura 5.12: Representação do circuito parcelar no modal de resolução

Entrando numa das abas de resolução, é possível ver a metodologia presente quando utilizando o método selecionado, porém, neste, também está presente a imagem do circuito parcelar, interativa mas sem a possibilidade de alteração do circuito. Juntamente com esta representação está presente uma nota com a explicação do processo de remoção das restantes fontes para obter o circuito. No topo da página está também presente uma nova barra de navegação, adicionada abaixo da barra presente na página principal.

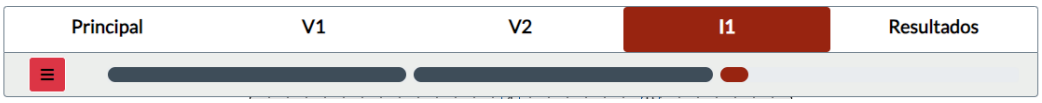


Figura 5.13: Barra de navegação

Esta contém uma barra de progresso para cada resolução, dando as páginas anteriores como concluídas e seguindo o *scroll* da página para demonstrar o progresso na resolução atual, assim como, um índice das secções da resolução colapsável, em que é possível clicar no seu título para navegar para a mesma.

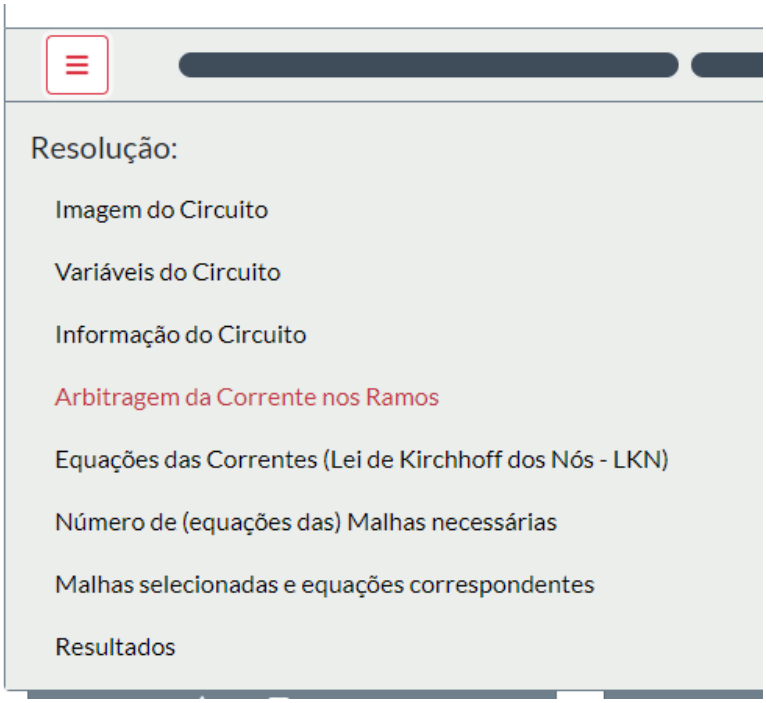


Figura 5.14: Índice de navegação da resolução

5.3 Resultados

Por fim, a ultima página tem toda a informação relativa aos resultados.

Primeiro é sintetizada toda a informação das contribuições calculadas anteriormente, na forma de uma tabela de contribuições.

Tabela de Contribuições

Parâmetro \ Fonte		V1	I1
Ir2	Direção	gnd → B	B → A
	Valor	307.69 mA	-192.31 mA
Ir1	Direção	gnd → B	gnd → A
	Valor	307.69 mA	-307.69 mA
Ir3	Direção	-	C → A
	Valor	0 A	500 mA
Ir5	Direção	B → gnd	B → gnd
	Valor	276.92 mA	-76.92 mA
Ir4	Direção	B → gnd	B → C
	Valor	30.77 mA	269.23 mA
Ir6	Direção	B → gnd	gnd → C
	Valor	30.77 mA	230.77 mA

Figura 5.15: Tabela de contribuições

Esta é seguida da informação das correntes atribuídas ao circuito original, esta secção é gerada por código já existente, mas é útil aqui para verificar a direção das correntes e, caso não tenham sido posicionados amperímetros para todas as correntes, para identificar a qual ramo corresponde cada uma.

Por fim é representada a soma das contribuições e os respetivos resultados.

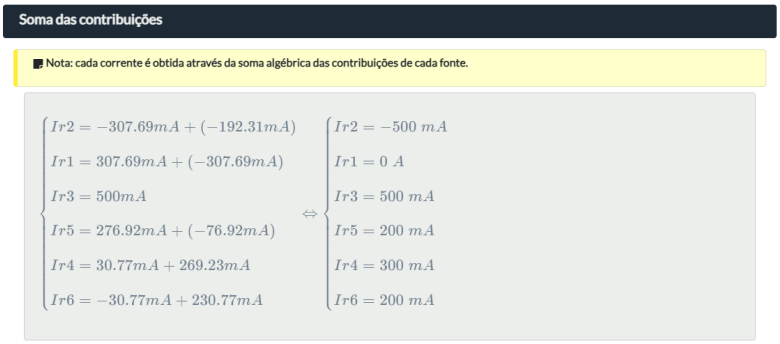


Figura 5.16: Soma das contribuições

5.4 Ficheiros de saída

Assim como nos restantes métodos, é possível introduzir os dados do utilizador para introduzir nos ficheiros JSON e LaTeX, com a possibilidade de abrir o último no *Overleaf*. No anexo A é possível encontrar o ficheiro de saída LaTeX resultante da resolução de um circuito exemplo (Anexo A).

5.5 Outras implementações

Para além da implementação dos algoritmos principais, de modo a cumprir os objetivos do projeto, foram também integradas algumas outras funções secundárias, nomeadamente a migração de BS4 para BS5, o que possibilitou o uso de diversas novas ferramentas, das quais as *toasts*. Estas foram utilizadas para criar um sistema de notificações dentro do modal, como, por exemplo, mensagens de sucesso ou erro de ferramentas integradas no mesmo. Como as demonstradas na figura 5.17

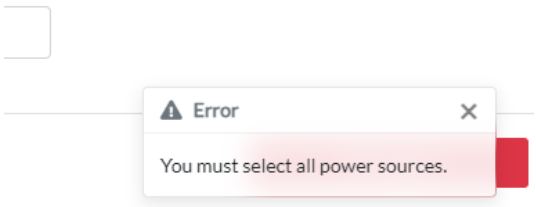


Figura 5.17: Mensagem de erro exemplo

Esta função vem equipada com formato para mensagens de erro, aviso, sucesso e informação. Sendo apenas necessário passar um objeto de formato.

Propriedade	**Descrição**
type	Tipo de mensagem (error, warning, success, info)
title	Titulo da mensagem
body	Corpo da mensagem

Tabela 5.1: Propriedades da mensagem

Estas mensagens desaparecem com o tempo, ou podem ser dispensadas, autorizando o máximo de quatro mensagens de cada vez.

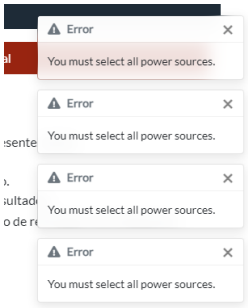


Figura 5.18: *Toast stack*

Capítulo 6

Conclusões

No âmbito deste projeto, foram necessárias competências técnicas nas diversas áreas do desenvolvimento de aplicações *web*, nomeadamente JS e as diversas bibliotecas utilizadas para a criação de páginas dinâmicas, responsivas e interativas. Foram ainda imprescindíveis conhecimentos sobre as diversas técnicas e boas práticas de navegação e criação de HTML. Toda a pesquisa sobre estes temas, foi feita em grande parte nos estágios iniciais de estudo e pesquisa dos assuntos.

Também foram essenciais ao progresso num projeto desta escala e complexidade, capacidades de planeamento, organização, divisão de etapas em etapas mais pequenas e gestão de tempo. Isto impôs alguns desafios, relativamente ao controlo do tempo e do delineamento de um nível de progresso constante e viável ao longo da duração de todo o projeto.

Os principais objetivos do projeto foram cumpridos, com diferentes graus de sucesso, tendo sido concebida uma ferramenta que resolve circuitos desenhados no QUCS pelo TSP, demonstrando a resolução do circuito passo-a-passo e que permite exportar os dados para JSON e LaTeX, com possibilidade de abrir diretamente no *Overleaf*. O projeto desenvolvido permite ainda ao utilizador controlar a ordem e método de analise do circuito, apresentando os resultados de forma lógica e visualmente agradável. Faz ainda uso do ficheiro do esquemático para desenvolver um ficheiro JSON biunívoco ao mesmo, capaz de gerar a *netlist* correspondente, assim como, uma representação visual e interativa do circuito. Porém, ainda que esta seja responsiva na sua maioria, em dispositivos com ecrãs de pequenas dimensões, como *smartphones* e *tablets*, o grau de interatividade fica bastante afetado, apresentando

alguns problemas com funcionalidades *drag and drop*, nomeadamente na reorganização dos cartões na ordem de resolução. A visualização do circuito representado nestes dispositivos não foi resolvida na sua totalidade.

Dito isto, o *parsing* e *widget* de desenho do circuito foram desenvolvidos a um nível bastante satisfatório, ultrapassando o esperado. O último tornando-se uma ferramenta com grande capacidade de crescimento e utilização durante o desenvolvimento deste projeto e outros projetos semelhantes, presentes e futuros.

6.1 Trabalho Futuro

Vários aspetos deste projeto, podem ainda ser melhorados e ser desenvolvidos além do mesmo. Como, por exemplo, alguns pontos que ficaram por desenvolver:

- Compatibilidade com o MTN; Trabalhar o *widget* de redesenho para ser mais responsivo e possibilitar a sua conversão para um elemento *canvas* e/ou imagem.

Existem outros aspetos técnicos com margem de melhoria, como o controlo de fluxo do código utilizado e aspetos de estilização dos elementos *web* (CSS).

Algumas funcionalidades que foram conceptualizadas mas não postas em práticas foram:

- Uma segunda tabela de contribuições, disponível em todas as páginas, atualizada consoante a página aberta, de maneira a proporcionar apoio na sintetização dos cálculos já efetuados sem mostrar os que ainda não foram efetuados;
- A representação de ramos, correntes, malhas e outros aspetos do circuito, através do *widget* do redesenho;
- A escolha da resistência interna das fontes e/ou aparelhos de medição por botão de rádio.

Referências

[1] L. Sousa, M. Alves, and F. Pereira, "U=RIsolve: Teaching & self-learning the nodal analysis method the easy way." CASHE - Conference on Academic Success in Higher Education, Politécnico do Porto (ISEP/IPP), 2019. Best presentation award. [Citado nas páginas 2 e 17]

[2] M. Foundation, "Javascript | MDN." `https://developer.mozilla.org/docs/Web/JavaScript`, 2023. Accessed April 19, 2023. [Citado na página 7]

[3] J. Virdi, "What's new in es2022?." `https://dev.to/jasmin/whats-new-in-es2022-1de6`, 2022. [Accessed 06-Jun-2023]. [Citado na página 8]

[4] W3schools.com, "JavaScript ES6 — w3schools.com." `https://www.w3schools.com/js/js_es6.asp`. [Citado na página 8]

[5] A. Wysoczański, "JavaScript trends in 2023. State of JavaScript report results — tsh.io." `https://tsh.io/blog/javascript-trends/`, 2023. [Citado na página 8]

[6] A. S., "What is html? hypertext markup language basics explained." `https://www.hostinger.com/tutorials/what-is-html`, May 2023. [Citado na página 10]

[7] I. H. Hickson, "HTML Standard." `https://html.spec.whatwg.org/multipage/`. [Accessed 01-Jun-2023] - Hickson não é atualmente o editor da *HTML specification*, atualmente é o lider técnico do *Flutter*. [Citado na página 10]

[8] B. Mark Otto, Jacob Thornton, "Approach — getbootstrap.com." `https://getbootstrap.com/docs/4.1/extend/approach/`. [Accessed 07-Jun-2023]. [Citado na página 12]

[9] S. Team, S. T. team is committed to highlight the latest products, S. Sen, MicarNet, M. Voelker, Hamidreza, John, Kaito, Iqbol, S. N. Mishra, and et al., "Bootstrap 5 vs bootstrap 4 - what's new and what changed?." `https://superdevresources.com/bootstrap5-vs-bootstrap4-whats-new/`, Oct 2022. [Citado na página 13]

[10] G. "guitorri"Torri, "qucs/NEWS.md at qucs-0.0.19 · Qucs/qucs." `https://github.com/Qucs/qucs/blob/qucs-0.0.19/NEWS.md`, 2017. [Accessed 08-Jun-2023]. [Citado na página 13]

[11] V. "ra3xdh"Kuznetsov, "Qucs-S: Qucs with SPICE." `https://ra3xdh.github.io/`. [Accessed 08-Jun-2023]. [Citado na página 14]

[12] L. Sousa, "U=RIsolve - aplicativo de análise de circuitos elétricos simulados no qucs," 2018. [Citado na página 17]

[13] L. Sousa, *U=RIsolve: a web-based tool for teaching and self-learning the Nodal Voltage Method*. PhD thesis, Instituto Superior de Engenharia do Porto, 2020. [Citado na página 17]

[14] M. Duarte, *APLICAÇÃO PARA GERAÇÃO DA METODOLOGIA/EQUAÇÕES DO MÉTODO DAS CORRENTES NAS MALHAS, A PARTIR DO QUCS*. Relatório pesta, Instituto Superior de Engenharia do Porto, 2019. [Citado na página 17]

[15] A. Pinheiro, *Projeto, implementação e integração do Método das Correntes de Malha na aplicação U=RIsolve*. Relatório de pesta, Instituto Superior de Engenharia do Porto, 2022. [Citado na página 17]

[16] J. Soares, *APLICAÇÃO PARA GERAÇÃO DA METODOLOGIA/EQUAÇÕES DO MÉTODO DAS CORRENTES NOS RAMOS A PARTIR DO QUCS*. Relatório pesta, Instituto Superior de Engenharia do Porto, 2019. [Citado na página 17]

[17] H. Carvalho, *U=RIsolve APP Implementação do Método das Correntes nos Ramos*. Relatório de pesta, Instituto Superior de Engenharia do Porto, 2023. [Citado na página 17]

[18] W. McAllister, "Superposition (article) | Circuit analysis | Khan Academy — khanacademy.org." `https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-dc-circuit-analysis/a/ee-superposition`. [Citado na página 19]

[19] Q. Team, "Schematic File Format Qucs Help latest documentation." `https://qucs-help.readthedocs.io/en/latest/internal.html`, 2014-2017. [Accessed 11-Jun-2023]. [Citado nas páginas 27 e 30]

[20] P. Morim, *Circuit Editor - Design, implementation, and integration into the U=RIsolve web application*. Relatório de pesta, Instituto Superior de Engenharia do Porto, 2023. [Citado na página 31]

[21] M. Brinson, "Qucs - a tutorial - qucs simulation of spice netlists." `https://qucs.sourceforge.net/docs/tutorial/spicetoqucs.pdf`, 2007. [Citado na página 49]

[22] S. Jahn, M. Margraf, V. Habchi, and R. Jacob, "Qucs - technical papers." `https://qucs.sourceforge.net/docs/technical/technical.pdf`, 2003-2005. [Citado na página 49]

Anexo A

Resolução de circuitos exemplo

A.1 Circuito DC resolvido por Método da corrente nas Malhas

A.1.1 Variáveis do Circuito

Ramos [R]	**Nós [N]**	**Fontes de Corrente [C]**	**Fontes Isoladas de Tensão [T]**
R=6	N=4	C=1	T=0

A.1.2 Informação do Circuito

Simulação [AC/DC]	**Frequência [A]**	**Amperímetros [I]**
DC	N / A	6/6

A.1.3 Cálculo das contribuições de cada fonte

Fonte V1

Imagem do Circuito

Identificação da Corrente nos Ramos

Arbitragem da Corrente nos Ramos

Tabela A.1: Lista das Correntes do Circuito e as suas propriedades/-componentes

Reference	Start Node	End Node	Components
Ir5	B	gnd	R3
Ir1	gnd	B	V1, R2, R1
Ir4	B	gnd	R6, R5

Valores das correntes dos ramos

$$\begin{cases} Ir5 = 276.923\ mA \\ Ir1 = 307.692\ mA \\ Ir4 = 30.769\ mA \end{cases}$$

Fonte I1

Imagem do Circuito

Identificação da Corrente nos Ramos

Arbitragem da Corrente nos Ramos

Tabela A.2: Lista das Correntes do Circuito e as suas propriedades/-componentes

Reference	**Start Node**	**End Node**	**Components**
Ir1	gnd	A	R1
Ir2	B	A	R2
Ir3	C	A	I1, R4
Ir5	B	gnd	R3
Ir4	B	C	R6
Ir6	gnd	C	R5

Valores das correntes dos ramos

$$\begin{cases} Ir1 = -307.692\ mA \\ Ir2 = -192.308\ mA \\ Ir3 = 500\ mA \\ Ir5 = -76.923\ mA \\ Ir4 = 269.231\ mA \\ Ir6 = 230.769\ mA \end{cases}$$

A.1.4 Tabela de Contribuições

Parâmetro \ Fonte		**V1**	**I1**
Ir2	Direção	gnd → B	B → A
	Valor	307.692 mA	-192.308 mA
Ir1	Direção	gnd → B	gnd → A
	Valor	307.692 mA	-307.692 mA
Ir3	Direção	-	C → A
	Valor	0 A	500 mA
Ir5	Direção	B → gnd	B → gnd
	Valor	276.923 mA	-76.923 mA
Ir4	Direção	B → gnd	B → C
	Valor	30.769 mA	269.231 mA
Ir6	Direção	B → gnd	gnd → C
	Valor	30.769 mA	230.769 mA

A.1.5 Identificação da Corrente nos Ramos

Arbitragem da Corrente nos Ramos

Tabela A.3: Lista das Correntes do Circuito e as suas propriedades/-componentes

Reference	Start Node	End Node	Components
Ir2	B	A	R2
Ir1	gnd	A	V1, R1
Ir3	C	A	I1, R4
Ir5	B	gnd	R3
Ir4	B	C	R6
Ir6	gnd	C	R5

Valores das correntes dos ramos

$$\begin{cases} Ir2 = -307.692\ mA + (-192.308\ mA) \\ Ir1 = 307.692\ mA + (-307.692\ mA) \\ Ir3 = 500\ mA \\ Ir5 = 276.923\ mA + (-76.923\ mA) \\ Ir4 = 30.769\ mA + 269.231\ mA \\ Ir6 = -30.769\ mA + 230.769\ mA \end{cases}$$

$$\Leftrightarrow \begin{cases} Ir2 = -500\ mA \\ Ir1 = 0\ A \\ Ir3 = 500\ mA \\ Ir5 = 200\ mA \\ Ir4 = 300\ mA \\ Ir6 = 200\ mA \end{cases}$$

Nota: cada corrente é obtida através da soma algébrica das contribuições de cada fonte.

A.1.6 Anexos

A.1.7 Resolução: V1

Variáveis do Circuito

Ramos [R]	**Nós [N]**	**Fontes de Corrente [C]**	**Fontes Isoladas de Tensão [T]**
R=3	N=2	C=0	T=0

Informação do Circuito

Simulação [AC/DC]	**Frequência [A]**	**Amperímetros [I]**
DC	N / A	3/3

Número de malhas Principais e Auxiliares

Malhas Principais

$$M_p = R - (N-1) - C \Leftrightarrow$$
$$M_p = 3 - (2-1) - 0 \Leftrightarrow$$
$$\Leftrightarrow M_p = 2$$

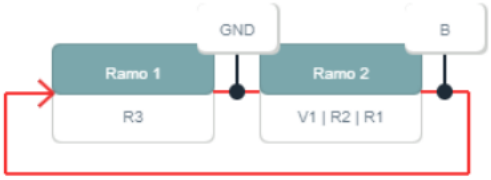
Nota: O número de malhas principais (Mp) calculado corresponde ao número de equações das malhas (do sistema de equações), em que as incógnitas são as correntes (fictícias) das malhas.

Malhas Auxiliares

Nota: O número de malhas auxiliares é o mesmo que o número de fontes de corrente (FC). Cada malha auxiliar passa numa única FC, normalmente escolhendo-se o sentido de circulação da (corrente da) malha igual de forma a alinhar-se no sentido da corrente da FC, assumindo assim o seu próprio valor (positivo).

$$C = 0 \implies M_a = 0$$

Malhas selecionadas (Auxiliares e Principais) e equações correspondentes



Malha 1 - Principal

$$\text{Equation}: \quad I_{Mp1} \ : \ V1 = R3 \cdot (I_{Mp1} + I_{Mp2}) + R1 \cdot (I_{Mp1}) + R2 \cdot (I_{Mp1}) \quad \text{(A.1)}$$

Malha 2 - Principal

$$\text{Equation}: \quad I_{Mp2} \ : \ 0 = R3 \cdot (I_{Mp1} + I_{Mp2}) + R5 \cdot (I_{Mp2}) + R6 \cdot (I_{Mp2}) \quad \text{(A.2)}$$

Sistema de Equações

Equações:

$$\begin{cases} 12 = 10 \cdot (I_{Mp1} + I_{Mp2}) + 10 \cdot (I_{Mp1}) + 20 \cdot (I_{Mp1}) \\ 0 = 10 \cdot (I_{Mp1} + I_{Mp2}) + 50 \cdot (I_{Mp2}) + 40 \cdot (I_{Mp2}) \end{cases}$$

Passos:

Step 1: Initial equation system

$$\begin{cases} V1 = R3 \cdot (I_{Mp1} + I_{Mp2}) + R1 \cdot (I_{Mp1}) + R2 \cdot (I_{Mp1}) \\ 0 = R3 \cdot (I_{Mp1} + I_{Mp2}) + R5 \cdot (I_{Mp2}) + R6 \cdot (I_{Mp2}) \end{cases}$$

Step 2: Substitute the mesh current values

$$\begin{cases} V1 = R3 \cdot (I_{Mp1} + I_{Mp2}) + R1 \cdot (I_{Mp1}) + R2 \cdot (I_{Mp1}) \\ 0 = R3 \cdot (I_{Mp1} + I_{Mp2}) + R5 \cdot (I_{Mp2}) + R6 \cdot (I_{Mp2}) \end{cases}$$

Step 3: Substitute the circuit component values

$$\begin{cases} 12 = 10 \cdot (I_{Mp1} + I_{Mp2}) + 10 \cdot (I_{Mp1}) + 20 \cdot (I_{Mp1}) \\ 0 = 10 \cdot (I_{Mp1} + I_{Mp2}) + 50 \cdot (I_{Mp2}) + 40 \cdot (I_{Mp2}) \end{cases}$$

Valores das correntes de malha

$$\begin{cases} I_{Mp1} = 307,69\ mA \\ \\ I_{Mp2} = -30,77\ mA \end{cases}$$

Identificação da Corrente nos Ramos

Tabela A.4: Lista das Correntes do Circuito e as suas propriedades/-componentes

Reference	**Start Node**	**End Node**	**Components**
Ir5	B	gnd	R3
Ir1	gnd	B	V1, R2, R1
Ir4	B	gnd	R6, R5

Arbitragem da Corrente nos Ramos

Valores das correntes dos ramos

$$\begin{cases} Ir5 = I_{Mp1} + I_{Mp2} \\ Ir1 = I_{Mp1} \\ Ir4 = -I_{Mp2} \end{cases} \Leftrightarrow \begin{cases} Ir5 = 276,92\ mA \\ Ir1 = 307,69\ mA \\ Ir4 = 30,77\ mA \end{cases}$$

Nota: cada corrente no ramo foi obtida através da soma algébrica das correntes de malha que existem nesse ramo.

A.1.8 Resolução: I1

Variáveis do Circuito

Ramos [R]	**Nós [N]**	**Fontes de Corrente [C]**	**Fontes Isoladas de Tensão [T]**
R=6	N=4	C=1	T=0

Informação do Circuito

Simulação [AC/DC]	**Frequência [A]**	**Amperímetros [I]**
DC	N / A	6/6

Número de malhas Principais e Auxiliares

Malhas Principais

$$M_p = R - (N-1) - C \;\Leftrightarrow$$
$$M_p = 6 - (4-1) - 1 \;\Leftrightarrow$$
$$\Leftrightarrow \; M_p = 2$$

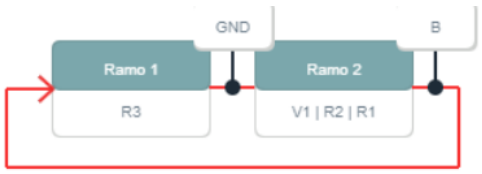
Nota: O número de malhas principais (Mp) calculado corresponde ao número de equações das malhas (do sistema de equações), em que as incógnitas são as correntes (fictícias) das malhas.

Malhas Auxiliares

Nota: O número de malhas auxiliares é o mesmo que o número de fontes de corrente (FC). Cada malha auxiliar passa numa única FC, normalmente escolhendo-se o sentido de circulação da (corrente da) malha igual de forma a alinhar-se no sentido da corrente da FC, assumindo assim o seu próprio valor (positivo).

$$C = 1 \implies M_a = 1$$

Malhas selecionadas (Auxiliares e Principais) e equações correspondentes

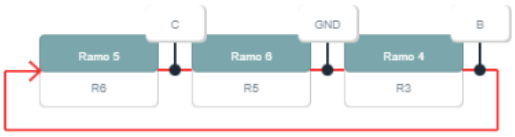


Malha 1 - Auxiliar

$$\text{Value}: \quad I_{Ma1} \ : \ 5\ mA \tag{A.3}$$

Malha 2 - Principal

$$\text{Equation}: \quad I_{Mp1} \ : \ 0 = R2 \cdot (I_{Ma1} + I_{Mp1}) + R3 \cdot (I_{Mp1} - I_{Mp2}) + R1 \cdot (I_{Mp1}) \tag{A.4}$$



Malha 3 - Principal

$$\text{Equation}: \quad I_{Mp2} \ : \ 0 = R6 \cdot (I_{Ma1} + I_{Mp2}) + R5 \cdot (I_{Mp2}) + R3 \cdot (-I_{Mp1} + I_{Mp2}) \quad \text{(A.5)}$$

Sistema de Equações

Equações:

$$\begin{cases} 0 = 20 \cdot (0,5 + I_{Mp1}) + 10 \cdot (I_{Mp1} - I_{Mp2}) + 10 \cdot (I_{Mp1}) \\ 0 = 40 \cdot (0,5 + I_{Mp2}) + 50 \cdot (I_{Mp2}) + 10 \cdot (-I_{Mp1} + I_{Mp2}) \end{cases}$$

Passos:

Step 1: Initial equation system

$$\begin{cases} 0 = R2 \cdot (I_{Ma1} + I_{Mp1}) + R3 \cdot (I_{Mp1} - I_{Mp2}) + R1 \cdot (I_{Mp1}) \\ 0 = R6 \cdot (I_{Ma1} + I_{Mp2}) + R5 \cdot (I_{Mp2}) + R3 \cdot (-I_{Mp1} + I_{Mp2}) \end{cases}$$

Step 2: Substitute the mesh current values

$$\begin{cases} 0 = R2 \cdot (0,5 + I_{Mp1}) + R3 \cdot (I_{Mp1} - I_{Mp2}) + R1 \cdot (I_{Mp1}) \\ 0 = R6 \cdot (0,5 + I_{Mp2}) + R5 \cdot (I_{Mp2}) + R3 \cdot (-I_{Mp1} + I_{Mp2}) \end{cases}$$

Step 3: Substitute the circuit component values

$$\begin{cases} 0 = 20 \cdot (0,5 + I_{Mp1}) + 10 \cdot (I_{Mp1} - I_{Mp2}) + 10 \cdot (I_{Mp1}) \\ 0 = 40 \cdot (0,5 + I_{Mp2}) + 50 \cdot (I_{Mp2}) + 10 \cdot (-I_{Mp1} + I_{Mp2}) \end{cases}$$

Valores das correntes de malha

$$\begin{cases} I_{Ma1} = 500\ mA \\ I_{Mp1} = -307,69\ mA \\ I_{Mp2} = -230,77\ mA \end{cases}$$

Identificação da Corrente nos Ramos

Tabela A.5: Lista das Correntes do Circuito e as suas propriedades/-componentes

Reference	**Start Node**	**End Node**	**Components**
Ir1	gnd	A	R1
Ir2	B	A	R2
Ir3	C	A	I1, R4
Ir5	B	gnd	R3
Ir4	B	C	R6
Ir6	gnd	C	R5

Arbitragem da Corrente nos Ramos

Valores das correntes dos ramos

$$\begin{cases} Ir1 = I_{Mp1} \\ Ir2 = -I_{Ma1} - I_{Mp1} \\ Ir3 = I_{Ma1} \\ Ir5 = I_{Mp1} - I_{Mp2} \\ Ir4 = I_{Ma1} + I_{Mp2} \\ Ir6 = -I_{Mp2} \end{cases} \Leftrightarrow \begin{cases} Ir1 = -307,69\ mA \\ Ir2 = -192,31\ mA \\ Ir3 = 500\ mA \\ Ir5 = -76,92\ mA \\ Ir4 = 269,23\ mA \\ Ir6 = 230,77\ mA \end{cases}$$

Nota: cada corrente no ramo foi obtida através da soma algébrica das correntes de malha que existem nesse ramo.