Politécnico do Porto
Instituto Superior de Engenharia do Porto

U=RIsolve ScopeAcademy

Simulador de osciloscópio para integração no editor de circuitos da U=RIsolve Academy

Óscar Gonçalo da Silva Almeida Pinheiro

LEEC
Licenciatura em Engenharia Eletrotécnica e de Computadores

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Setembro, 2025

Este relatório foi realizado no âmbito dos requisitos que constam da Ficha de Unidade Curricular de Projeto/Estágio, do 3º ano, da Licenciatura em Engenharia Eletrotécnica e de Computadores.

Candidato: Óscar Gonçalo da Silva Almeida Pinheiro, Nº 1171424,
1171424@isep.ipp.pt

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

Coorientação Científica: João Ferreira, jfa@isep.ipp.pt

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

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

Setembro, 2025

"That day, for no particular reason, I decided to go for a little run.
So I ran to the end of the road.
And when I got there, I thought maybe I'd run to the end of town."

-Tom Hanks, Forrest Gump

Agradecimentos

Uma palavra de agradecimento a todas as pessoas que, direta ou indiretamente, contribuíram para a realização deste projeto.

Em primeiro lugar, quero agradecer ao meu orientador, Professor André Rocha, pela oportunidade de integrar este projeto e pelo apoio e atenção dispensada, não apenas durante o seu desenvolvimento, mas em todos os momentos de aprendizagem ao longo da minha licenciatura.

De igual forma, agradeço profundamente ao meu coorientador, Engenheiro João Ferreira, pela sua disponibilidade e apoio permanente, tanto em questões técnicas como pessoais, e por todo o conhecimento que me transmitiu desde o início até à conclusão deste trabalho.

Por fim, deixo uma palavra especial de gratidão à minha família, que constitui um verdadeiro pilar e que certamente tornou esta jornada mais leve e enriquecedora, especialmente pelo facto de ter sido partilhada.

A todos, o meu muito obrigado.

Resumo

Este projeto consiste no desenvolvimento de um simulador de osciloscópio digital em ambiente *web*, com a finalidade de ser integrado no editor de circuitos da plataforma *U=RIsolve Academy*.

O objetivo da aplicação passou por criar uma interface simples e intuitiva, não idêntica a nenhum osciloscópio real específico, mas que possuísse as funcionalidades essenciais e comuns aos equipamentos reais – em particular, aqueles que os alunos da Licenciatura em Engenharia Eletrotécnica e de Computadores (LEEC) do Instituto Superior de Engenharia do Porto (ISEP) têm contacto. Adicionalmente, o interesse foi assegurar que o simulador estivesse devidamente preparado para receber dados semelhantes aos gerados pelo editor de circuitos, e que estes servissem de *input* para ele se comportasse de acordo com o esperado num osciloscópio real.

A aplicação desenvolvida está apta para receber e processar qualquer função matemática, à semelhança de uma calculadora gráfica. Assim, sempre que seja possível traduzir um sinal elétrico numa expressão matemática, o osciloscópio terá a capacidade de o representar adequadamente. Esta flexibilidade, aliada a uma *User Interface* (UI) e *User Experience* (UX) intuitiva, constitui a principal proposta desta aplicação.

No desenvolvimento foram utilizadas tecnologias como *Node.js*, em conjunto com a linguagem *JavaScript* (JS), bem como *HyperText Markup Language* (HTML) e *Cascading Style Sheets* (CSS). Adicionalmente, recorreu-se à ferramenta *Docker* para a contentorização do projeto, assim como outras bibliotecas e ferramentas complementares associadas às linguagens mencionadas.

Palavras-Chave: Simulador de osciloscópio; Formas de onda; Aplicação *web*; Aprendizagem remota; *U=RIsolve Academy*; *JavaScript*.

Abstract

This project consists of the development of a digital oscilloscope simulator in a web-based environment, intended to be integrated into the circuit editor of the U=RIsolve Academy platform.

The application's primary objective was to create a simple and intuitive interface, not identical to any specific real oscilloscope, who consolidates the essential and common functionalities found in real devices – particularly those familiar to students of LEEC at ISEP. Additionally, the goal was to ensure that the simulator was suitably prepared to receive data identical to those generated by the circuit editor, allowing this data to serve as input and enabling the simulator to behave as expected from a real device.

The developed application is capable of receiving and processing any mathematical function, similar to a graphing calculator. Therefore, whenever it is possible to translate an electrical signal into a mathematical expression, the oscilloscope will be able to represent it appropriately. This flexibility, combined with an intuitive UI and UX, constitutes the main proposition of this application.

During development, technologies such as Node.js, together with JS, HTML, and CSS were employed. Additionally, the Docker tool was used for project containerization, together with complementary libraries and tools related to these mentioned languages.

Keywords: *Oscilloscope simulator; Waveforms; Web-based application; E-learning; U=RIsolve Academy; JavaScript.*

Índice

Lista de Figuras **ix**

Lista de Tabelas **xi**

Listagens **xiii**

Lista de Acrónimos **xv**

1 Introdução **1**
1.1 Contextualização 1
1.2 Descrição do Projeto 2
1.2.1 Requisitos 3
1.2.2 Objetivos 3
1.3 Planeamento 4
1.3.1 Metodologia 4
1.3.2 Calendarização 5
1.4 Organização do Relatório 6

2 Enquadramento científico **7**
2.1 Estudo de circuitos elétricos 7
2.1.1 Do ensino presencial ao remoto 8
2.1.2 Importância dos instrumentos de medição 9
2.2 Caracterização do osciloscópio 10
2.2.1 Tipos de osciloscópio 11
2.2.2 Instrumentação real vs. virtual 12
2.2.3 Graus de virtualização 13
2.2.4 Simulador web vs. local 14

3 Estado da Arte **15**
3.1 Projetos desenvolvidos pela equipa *U=RIsolve* 15
3.1.1 De *U=RIsolve* até *U=RIsolve Academy* 15
3.1.2 U=RIsolve Academy - Circuit Editor 18
3.1.3 Oscilloscope Interface 19
3.1.4 Outros projetos 21

3.2 Editores de circuito com simuladores integrados 21
3.2.1 QUCS 21
3.2.2 PSpice 22
3.2.3 Falstad 22
3.2.4 Tinkercad 23
3.2.5 DCACLab 24
3.2.6 Análise comparativa das tecnologias 25
3.3 Simuladores de osciloscópio independentes 25
3.4 Estado do Software 27
3.4.1 Tecnologias Back-end 27
3.4.2 Tecnologias Front-end 28
3.4.3 Tecnologias Utilitárias 30

4 Arquitetura e Implementação **31**
4.1 Arquitetura da aplicação 31
4.1.1 Visão geral 31
4.2 Tecnologias e ferramentas utilizadas 33
4.2.1 Tecnologias de desenvolvimento 33
4.2.2 Ferramentas paralelas ao desenvolvimento 33
4.2.3 Justificação das escolhas tecnológicas 34
4.3 Estrutura de Pastas e Ficheiros 35
4.4 Etapas de desenvolvimento 37
4.4.1 Design da solução (UI/UX) 37
4.4.2 Formas de onda 39
4.4.3 Abordagem de implementação 41
4.5 Desafios, soluções e otimização 46

5 Funcionalidades e Utilização **49**
5.1 Interface do simulador 49
5.2 Funcionalidades 50
5.2.1 Canais 50
5.2.2 Escalas e off-sets 52
5.2.3 Trigger 53
5.2.4 Auto set 54
5.2.5 Operações matemáticas 55
5.2.6 Medidas 55
5.2.7 Cursores 56
5.2.8 Modo XY 57
5.2.9 Outras funcionalidades 58

6 Conclusão **61**
6.1 Avaliação global 61
6.2 Limitações e bugs 62
6.3 Trabalho Futuro 63
6.3.1 Integração com o editor de circuitos 63
6.3.2 Novas funcionalidades 64

Referências **65**

Anexo A Imagens Auxiliares **69**
A.1 Diagrama de Gantt inicialmente proposto 69
A.2 Mapa detalhado da Interface gráfica do utilizador (GUI) do osciloscópio 69

Lista de Figuras

1.1 Diagrama de Gantt final do projeto 5

2.1 Exemplo de representação do sinal no ecrã do osciloscópio 10
2.2 Exemplos de osciloscópios "reais" 11
2.3 Exemplos de osciloscópios "virtuais" 12

3.1 Versão original da aplicação U=RIsolve 16
3.2 Exemplo de pergunta num *quizz* do U=RIsolve Academy 17
3.3 Protótipo da página Inicial do *framework* U=RIsolve Academy . . . 18
3.4 Exemplo de circuito usando o editor de circuitos U=RIsolve Academy 19
3.5 *Oscilloscope Interface* representando dois sinais 20
3.6 Representação de um sinal *Pulse Width Modulation* (PWM) no osciloscópio do simulador TinkercAd 23
3.7 Comportamento de um circuito retificador de meia-onda representado no osciloscópio do simulador DCACLab 24

4.1 Visão geral da arquitetura da aplicação 32
4.2 Evolução entre Wireframe inicial (a) e Interface final (b) 38
4.3 Processo de interpolação do D3.js 41
4.4 Lógica de funcionamento do simulador 45
4.5 Exemplos de sinais gerados no simulador de osciloscópio 46

5.1 Interface final do simulador 50
5.2 Botão de canal ligado e desligado 50
5.3 Representação do CH1 ativo 51
5.4 Tipos de acoplamento 52
5.5 Comportamentos dos knobs 53
5.6 Exemplo de variações de trigger 54
5.7 Processo de arranjo dos sinais através do autoset 54
5.8 Representação da onda *math* resultado da subtração de dois canais . 55
5.9 Painel de medidas sobre o ecrã do osciloscópio 56
5.10 Utilização dos cursores no osciloscópio 56
5.11 Representação do Modo XY 57
5.12 Método elíptico 58

5.13 Maximização e minimização da interface 59

6.1 Prótotipo de Editor de ciruitos com osciloscópio integrado 63

A.1 Diagrama de Gantt incial do projeto 69
A.2 Mapa completo da Interface gráfica do utilizador (GUI) do osciloscópio 70

Lista de Tabelas

3.1 Comparação entre tecnologias para simulação e edição de circuitos . 26
3.2 Tecnologias back-end comuns no desenvolvimento web 28
3.3 Tecnologias front-end comuns e relevantes para o projeto 29

Listagens

4.1 Esquema em árvore da organização do software 35
4.2 Desenho da forma de onda com D3.js 40
4.3 Caracterização do Canal 1 antes da geração do dataset 42

Lista de Acrónimos

AC	*Alternating Current*
API	*Application Programming Interface*
CAD	*Computer-Aided Design*
CSS	*Cascading Style Sheets*
CSV	*Comma-separated values*
D3.js	*Data-Driven Documents*
DC	*Direct Current*
DEE	Departamento de Engenharia Electrotécnica
DOM	*Document Object Model*
GUI	Interface gráfica do utilizador
HTML	*HyperText Markup Language*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
ISEP	Instituto Superior de Engenharia do Porto
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 das Tensões nos Nós
npm	*Node Package Manager*
PESTA	Projeto/Estágio

PSpice	*Personal Simulation Program with Integrated Circuit Emphasis*
PWM	*Pulse Width Modulation*
Qucs	*Quite Universal Circuit Simulator*
REST	*Representational State Transfer*
SOTA	*State of the Art*
SPICE	*Simulation Program with Integrated Circuit Emphasis*
SVG	*Scalable Vector Graphics*
TCIRC	Teoria dos Circuitos
UC	Unidade Curricular
UI	*User Interface*
UX	*User Experience*
VS Code	*Visual Studio Code*

Capítulo 1

Introdução

Neste relatório é apresentado o trabalho desenvolvido no âmbito da unidade curricular de Projeto/Estágio (PESTA) da Licenciatura em Engenharia Eletrotécnica e de Computadores (LEEC) do Instituto Superior de Engenharia do Porto (ISEP).

Este capítulo tem início com a contextualização do trabalho desenvolvido, seguindo-se a descrição da aplicação, dos respetivos requisitos e objetivos, bem como da metodologia e calendarização adotadas ao longo das diferentes etapas do seu desenvolvimento. Por fim, apresenta-se a estrutura dos capítulos que compõem o presente relatório.

1.1 Contextualização

O ensino e o trabalho à distância tornaram-se uma realidade incontornável, levando empresas, trabalhadores, alunos, escolas e instituições a adotarem políticas cada vez mais remotas no que se refere à troca e transmissão de informação. As universidades e os institutos politécnicos não são exceção, sendo, por isso, importante garantir a capacidade de oferecer um ensino extra presencial.

No contexto de áreas como a eletrónica e a eletrotécnica — em particular no estudo de circuitos elétricos — o contacto físico com componentes e instrumentos relacionados é fundamental tanto para a aprendizagem como para a prática profissional. Deste modo, torna-se pertinente dispor de ferramentas e tecnologias que permitam, de forma remota, simular com elevada eficiência a realidade laboratorial,

garantindo não apenas a reprodução fiel dos fenómenos, mas também oferecendo funcionalidades adicionais inerentes ao ambiente virtual e computacional.

No que diz respeito aos instrumentos de medição, estes são inseparáveis da análise de circuitos elétricos e acompanham, desde o início, o percurso formativo de qualquer estudante nestas áreas científicas. Com o surgimento de inúmeras tecnologias e plataformas digitais para a edição de circuitos elétricos, tornou-se igualmente relevante virtualizar instrumentos de medição — nomeadamente os osciloscópios — de forma a possibilitar que os estudantes mantenham um contacto mais frequente com ferramentas essenciais ao seu desenvolvimento académico.

A partir de um circuito elétrico, a possibilidade de visualizar, num determinado ponto, o comportamento temporal de um sinal permite identificar inúmeras características do circuito em questão e, eventualmente, solucionar problemas. Tal facto evidencia a razão pela qual o osciloscópio é uma ferramenta fundamental e insubstituível para engenheiros, investigadores e profissionais desta área e justifica o porquê de tantas vezes ser alvo de investimentos e projetos científicos.

Em anos anteriores, no âmbito desta unidade curricular, foram desenvolvidos outros simuladores de osciloscópio com objetivos em parte semelhantes e resultados interessantes, especialmente no que diz respeito à reprodução fiel do funcionamento de um osciloscópio real. Estes projetos serão abordados mais adiante neste relatório. Contudo, o interesse em retomar esta temática reside na criação de um osciloscópio que não funcione apenas de forma independente, mas que tem como finalidade a sua integração num editor de circuitos — mais concretamente o da plataforma *web U=RIsolve Academy*, a qual também será detalhada posteriormente.

1.2 Descrição do Projeto

A aplicação desenvolvida visa, primeiramente, proporcionar aos alunos uma maior oportunidade de contacto com o osciloscópio para além do contacto físico obtido nas aulas laboratoriais. Ao mesmo tempo, procura facilitar o processo de aprendizagem através das uso da maioria das funcionalidades comuns de um osciloscópio de bancada — sobretudo aquelas utilizadas nas unidades curriculares introdutórias ao estudo de circuitos elétricos.

Para isso, a aplicação apresenta uma interface amigável, capaz de proporcionar, num primeiro contacto, uma experiência de aprendizagem menos exigente e, eventualmente, mais motivadora. Isto porque, considerando alguns aparelhos reais de bancada, devido à sua complexidade, a curva de aprendizagem pode revelar-se relativamente elevada.

Tendo em vista a integração futura no editor de circuitos, teve-se em atenção que o simulador desenvolvido estivesse *a priori* adaptado para trabalhar sobre um formato de dados compatível com essa integração. Paralelamente, foram consideradas

funcionalidades que apenas um instrumento virtual pode disponibilizar, abrangendo características de flexibilidade e adaptação tanto à interface do editor como às necessidades específicas de cada utilizador.

1.2.1 Requisitos

Considerando que este projeto se insere exclusivamente na vertente de *software*, não foram exigidos nem necessários requisitos de *hardware*, sendo que, em termos de materiais físicos utilizados, apenas apenas houve a necessidade de um computador pessoal com características adequadas e, pontualmente, contacto com um osciloscópio real. Neste sentido, não houve necessidade de quaisquer gastos monetários para a sua execução.

No que diz respeito ao conhecimento teórico e prático prévio, não foi necessária uma mestria em nenhum conceito específico. Contudo, o domínio de noções básicas de circuitos elétricos, bem como alguma familiaridade com o instrumento de medição em causa, revelou-se importante para agilizar o processo de desenvolvimento e assegurar a fiabilidade dos resultados obtidos.

Relativamente à programação, em particular no âmbito do desenvolvimento *web*, a situação foi semelhante. Embora uma base prévia de programação tenha facilitado a adaptação e o ritmo de trabalho, não foi exigida experiência avançada. Como a maioria das tecnologias utilizadas — especialmente *JavaScript* (JS) — era novidade, foi necessário realizar um estudo prévio antes do início efetivo do desenvolvimento da aplicação, complementado naturalmente por um estudo contínuo ao longo de todo o processo de implementação.

1.2.2 Objetivos

Tendo em conta o propósito e a descrição da aplicação apresentada anteriormente, foram definidos objetivos de forma a assegurar a qualidade e a fiabilidade dos resultados bem como garantir a compatibilidade com a futura integração na plataforma. Nomeadamente:

1. Desenvolver uma nova interface de osciloscópio, intuitiva e simplificada, para utilização em conjunto com o editor de circuitos;

2. Implementar as funcionalidades essenciais no simulador que reflitam o comportamento de um osciloscópio de bancada;

3. Otimizar a lógica de geração da forma de onda no simulador, garantindo que o sistema esteja devidamente preparado para processar os dados provenientes do editor de circuitos e assegurando a fiabilidade na amostragem e na apresentação dos resultados.

Cada um destes objetivos foi determinante no desenvolvimento da aplicação e serão explorados com maior detalhe na Secção 4.4 referente às etapas de desenvolvimento.

1.3 Planeamento

Para garantir a concretização dos objetivos descritos e, consequentemente, alcançar o resultado final desejado, foi necessário delinear um plano de ação que assegurasse o bom desenvolvimento do trabalho, a sua conclusão com sucesso e, simultaneamente, a minimização de riscos ao longo do processo.

O planeamento será detalhado na Subsecção 1.3.1, seguindo-se a sua representação cronológica, através de um diagrama de *Gantt*, na Subsecção 1.3.2.

1.3.1 Metodologia

O planeamento das tarefas foi dividido em oito etapas distintas, facilitando a organização e a definição de um percurso claro para o trabalho a realizar.

A fase inicial englobou o estudo do contexto do projeto e a configuração do ambiente de trabalho, incluindo a análise das tecnologias base necessárias ao desenvolvimento da aplicação. Seguiu-se o estudo do *State of the Art* (SOTA), que incluiu a análise de *frameworks* e ferramentas disponíveis, com o objetivo de identificar as mais relevantes para a solução proposta, juntamente com a análise de editores de circuitos e simuladores de osciloscópio — abrangendo tanto projetos internos do ISEP como aplicações externas.

Na fase de implementação, deu-se um estudo introdutório dos conceitos e funcionalidades das tecnologias a utilizar. Em paralelo, realizou-se a seleção das ferramentas e *frameworks* mais adequadas, com base na análise previamente efetuada no SOTA.

Foi então elaborado o design da solução, utilizando ferramentas e práticas comuns nas áreas de *User Interface* (UI) e *User Experience* (UX), especialmente no contexto do desenvolvimento de *software front-end*, as quais serão abordadas mais adiante neste relatório.

Posteriormente, iniciou-se o desenvolvimento efetivo da aplicação. Numa primeira fase, foi implementada da interface gráfica do osciloscópio com recurso às tecnologias clássicas *HyperText Markup Language* (HTML) e *Cascading Style Sheets* (CSS), com base no design definido. Em seguida, deu-se a etapa central do projeto: dar "vida" ao simulador, implementando a lógica de funcionamento e conferindo-lhe as suas dinâmicas de interação, utilizando para isso JS e o respetivo *framework Data-Driven Documents* (D3.js) [1]. Por fim, foram realizados os testes e as depurações da aplicação.

A última fase consistiu na redação deste relatório e na preparação da documentação necessária para a entrega e apresentação do projeto.

Embora possa parecer, a metodologia descrita não foi executada de forma estritamente linear. Algumas etapas foram desenvolvidas em paralelo ou até retomadas em fases mais avançadas, como foi o caso do design da solução. Apesar do processo seguir, no geral, uma estrutura sequencial, essa sequência não é necessariamente unidirecional. A interligação entre as várias fases revelou-se, aliás, essencial e benéfica para a concretização do projeto, sobretudo tendo em conta a sua natureza voltada para a experiência do utilizador.

1.3.2 Calendarização

Devido à dimensão e complexidade do projeto desenvolvido, e sobretudo devido à preocupação em entregar um trabalho com a maior qualidade possível, o tempo de realização foi alargado em relação ao inicialmente previsto.

Por este motivo, foram elaborados dois diagramas de *Gantt*: o primeiro corresponde ao planeamento temporal definido no início do projeto e pode ser visto no Anexo A.1; o segundo reflete a execução real das tarefas, mostrando uma distribuição cronológica mais ajustada ao desenvolvimento, e encontra-se representado na Figura 1.1.

Figura 1.1: Diagrama de Gantt final do projeto

1.4 Organização do Relatório

No seguimento deste capítulo introdutório, o relatório esta organizado da seguinte maneira:

O Capítulo 2 apresenta o enquadramento científico da aplicação, procurando situá-la através de uma breve explicação sobre a importância do estudo de circuitos elétricos e dos instrumentos de medição, bem como das limitações e oportunidades desse estudo num contexto virtual face ao real. Inclui ainda uma caracterização do osciloscópio e dos seus principais tipos de arquitetura.

O Capitulo 3 é dedicado ao estado da arte. Neste são analisados os projetos que motivaram o desenvolvimento da aplicação, bem como ferramentas tecnológicas atuais que apresentam semelhanças com a proposta inicial e/ou serviram de referência para o resultado final, além de uma revisão de software existente no contexto *web*.

O Capitulo 4 aborda a implementação da aplicação, descrevendo a arquitetura geral, as ferramentas utilizadas, a organização do software e as etapas de desenvolvimento. São ainda discutidos os principais desafios encontrados ao longo do processo.

O Capitulo 5 apresenta a aplicação do ponto de vista da utilização, caracterizando a interface e destacando as funcionalidades desenvolvidas, acompanhadas de exemplos ilustrativos. Neste capítulo são também identificadas as limitações observadas à data da redação do relatório.

O Capitulo 6 corresponde às conclusões, trazendo uma avaliação global do projeto. São analisados os pontos-chave do desenvolvimento, os desafios enfrentados e é feita referência a possíveis etapas de evolução da aplicação.

Capítulo 2

Enquadramento científico

Fundamentalmente, o simulador de osciloscópio desenvolvido tem o propósito de, juntamente com o editor de circuitos, possibilitar a replicação de um ambiente laboratorial. Portanto, da mesma maneira que, num contexto físico, devemos garantir que o circuito está bem dimensionado, montado e que a comunicação com o osciloscópio é bem feita, o mesmo se aplica neste contexto virtual. Partindo do princípio de que o circuito foi corretamente desenhado no editor e que a comunicação com o simulador ocorre sem erros, este deverá comportar-se à semelhança de um instrumento real, reproduzindo os sinais conforme esperado.

Neste sentido, para percebermos como este simulador se comporta é importante perceber como o instrumento real o faz.

Este capítulo apresenta o enquadramento da aplicação, através de uma explicação sobre a importância do estudo de circuitos elétricos e dos instrumentos de medição no ponto de vista laboratorial face ao virtual. Inclui ainda uma caracterização do osciloscópio e dos seus tipos de arquitetura.

2.1 Estudo de circuitos elétricos

Os circuitos elétricos e os instrumentos de medição são elementos centrais na engenharia eletrotécnica, assim como em diversas áreas relacionadas. Contudo, a forma como estes conteúdos eram estudados e praticados há décadas difere bastante da abordagem atual. Esta transformação resultou não apenas na evolução natural por parte de quem ensina, mas especialmente na necessidade de adaptação por parte

de quem aprende. Sendo que, apesar dos desafios, esta mudança revelou-se uma mais-valia, beneficiando o estudo, a simulação e o teste de circuitos e sistemas, quer em contexto académico ou profissional.

2.1.1 Do ensino presencial ao remoto

O ensino presencial, especialmente nesta área, é de extrema importância. O contacto direto com componentes e equipamentos aliado à aprendizagem, em contexto laboratorial, de metodologias e boas práticas da sua utilização, constitui um elo de ligação fundamental entre o conhecimento teórico e a experiência prática. Esta interação permite desenvolver competências técnicas que apenas o manuseamento real e o contacto físico podem proporcionar.

No entanto, numa realidade de constante evolução tecnológica, meios de comunicação digitais sem fronteiras e, em particular, as transições forçadas como a ocorrida durante a pandemia *COVID-19*, aquilo que poderia eventualmente ser exclusivamente real deixou gradualmente de o ser. Surgiram, assim, inúmeras tecnologias destinadas a apoiar o ensino e a prática profissional, nomeadamente ferramentas para o desenho e simulação de circuitos elétricos como é o caso de plataformas como *PSPICE* e *QUCS*. Estas ferramentas serão abordadas em maior detalhe no capitulo 3 (Estado da Arte).

Este ambiente digital, mais do que inevitável, tornou-se, hoje em dia, uma solução viável e amplamente utilizada, seja para o acompanhamento em tempo-real entre docente e aluno, seja para um estudo individual de acordo com as necessidades de cada estudante.

Limitações e oportunidades

Apesar de apresentarem limitações inevitáveis — como a incapacidade de reproduzir fenómenos exclusivamente reais, por exemplo, ruído ou imperfeições do sinal; a dificuldade em transmitir plenamente a sensação de experiência laboratorial; a limitação no manuseamento físico de componentes; e a ausência de situações imprevisíveis, como a danificação de um componente (que também ensina) — estas plataformas oferecem benefícios que, em muitos casos, superam essas restrições. Entre eles, destacam-se:

- A possibilidade de estudar a qualquer momento e em qualquer lugar, sem a necessidade de materiais físicos ou deslocações;
- A simulação como elo de ligação entre o dimensionamento e a implementação real dos circuitos;

- A melhoria da gestão de risco e da segurança, permitindo realizar experiências que, no laboratório, poderiam ser dispendiosas ou até perigosas — especialmente para estudantes em fases iniciais de aprendizagem;
- A vasta diversidade de plataformas, permitindo ao utilizador escolher a que melhor se adequa às suas necessidades;
- A disponibilização de componentes e instrumentos virtuais que replicam de forma fiel os seus equivalentes reais, sem a necessidade de aquisição imediata e, consequentemente, evitando custos adicionais numa fase inicial.

Ainda assim, no que diz respeito aos instrumentos de medição, a proximidade com a experiência real nem sempre é alcançada. Embora os meios de análise e simulação de circuitos sejam completos e bastante otimizados, a semelhança funcional e de interação com instrumentos — como o osciloscópio — dentro de plataformas de simulação de circuitos nem sempre é totalmente conseguida. Esta limitação pode ser irrelevante em contextos de estudo avançado ou profissional, mas, para iniciantes ou curiosos, é fundamental a existência de ferramentas virtuais que mantenham de forma prática e intuitiva uma forte correspondência com a realidade.

É precisamente nesse contexto que surge a importância do simulador de osciloscópio desenvolvido neste projeto.

2.1.2 Importância dos instrumentos de medição

Os instrumentos de medição são, do ponto de vista prático, ferramentas indispensáveis para o estudo, compreensão e avaliação de sistemas e circuitos elétricos e eletrónicos. Estes podem ser agrupados em duas categorias: instrumentos para medições elétricas e instrumentos para análise de sinais.

No primeiro grupo, destaca-se o voltímetro, o amperímetro e o ohmímetro, utilizados para medir, respetivamente, tensão, corrente e resistência. Como mais versátil, temos o multímetro que reúne estas funções num único equipamento.

No segundo grupo, temos o instrumento chave deste projeto que é utilizado para análises mais detalhadas: o osciloscópio. Este permite visualizar e estudar, em tempo-real, o comportamento dos sinais elétricos, fornecendo informações como amplitude, frequência e formas de onda.

Qualquer um destes instrumentos desempenha um papel fundamental na validação e compreensão do comportamento real dos circuitos. São essenciais para medir grandezas elétricas, observar fenómenos invisíveis a olho nu, compreender discrepâncias entre resultados teóricos e práticos e embora concebidos para medir grandezas elétricas, é possível obter a medida de qualquer grandeza por meio do uso de transdutores e circuitos de condicionamento de sinal adequados [2].

Por isso mesmo, compreender estes instrumentos ajuda os estudantes não apenas ao nível de conhecimento técnico, mas também a desenvolver competências de análise e de resolução de problemas, essenciais à engenharia.

Importa ainda referir que, nesta subsecção se destacam apenas os instrumentos fundamentais para o ensino e prática inicial de circuitos elétricos, num contexto avançado, profissional ou de investigação existem outros equipamentos especializados mas que fogem ao âmbito e enquadramento deste projeto.

2.2 Caracterização do osciloscópio

Um osciloscópio é um instrumento utilizado para medir e representar graficamente um ou mais sinais elétricos em função do tempo, a representação que é feita de cada sinal é denominada onda ou forma de onda e a comunicação entre o circuito e o equipamento é feita através das chamadas pontas de prova.

No seu principal modo de funcionamento, o sinal é representado a duas dimensões, sendo que o eixo vertical (YY) representa a amplitude do sinal (tensão) e o eixo horizontal (XX) representa o tempo. Essa representação é possivél de ser observada na Figura 2.1.

A leitura da tensão é feita em *volts* (V) ou nos seus submúltiplos por divisão, assim como a leitura do tempo é feita em segundos (s) ou nos seus submúltiplos por divisão.

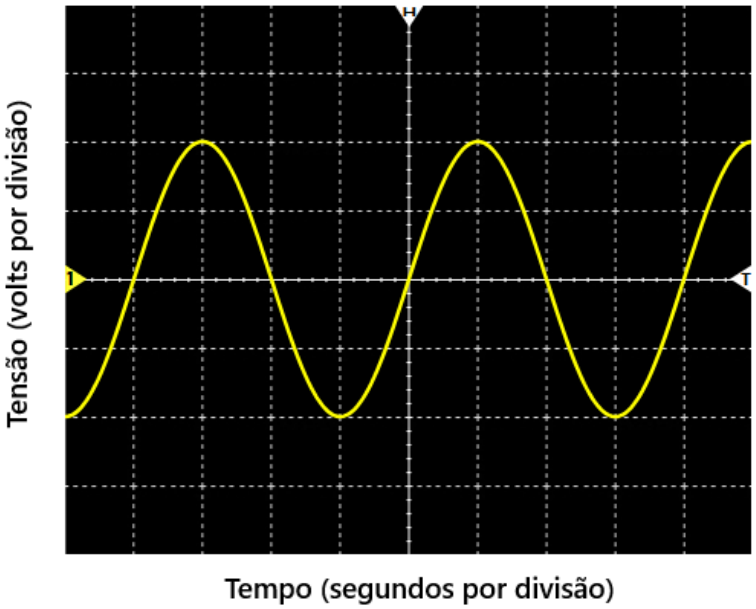


Figura 2.1: Exemplo de representação do sinal no ecrã do osciloscópio

Esta forma de onda representada permite analisar uma série de características do sinal [2], entre elas:

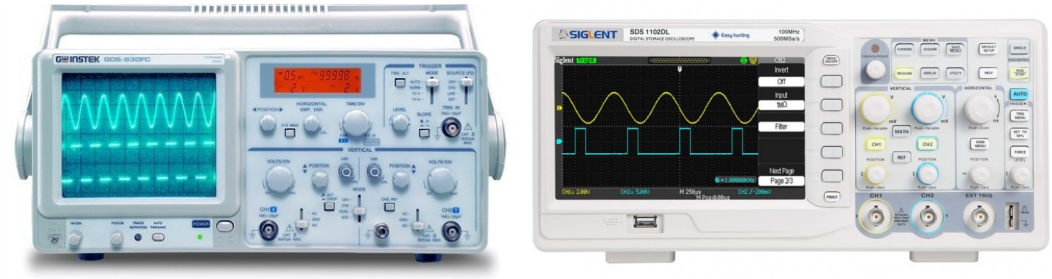
- Amplitude (de tensão): valores máximo, mínimo, pico-a-pico, eficaz, diferenciais de amplitude e componentes contínua e alternada.
- Tempo: período, frequência, diferenciais de tempo num sinal e entre dois sinais, atrasos, desfasamento entre dois sinais e tempos de subida.
- Existência de interferências (ruído) continuadas, perturbações transitórias.
- Comparação entre entrada e saída de sistemas, nomeadamente para analisar ganhos, desfasamentos, filtragens, retificações, permitindo projetar e depurar os mesmos sistemas.

Ainda que todos os osciloscópio ofereçam estas possibilidades de análise, a maneira como eles operam, as funcionalidades que oferecem para permitir essa análise e a forma de utilização podem variar de equipamento para equipamento.

2.2.1 Tipos de osciloscópio

De modo geral, e tendo em conta o contexto deste projeto, os osciloscópios podem ser agrupados em duas grandes categorias: "reais" e "virtuais".

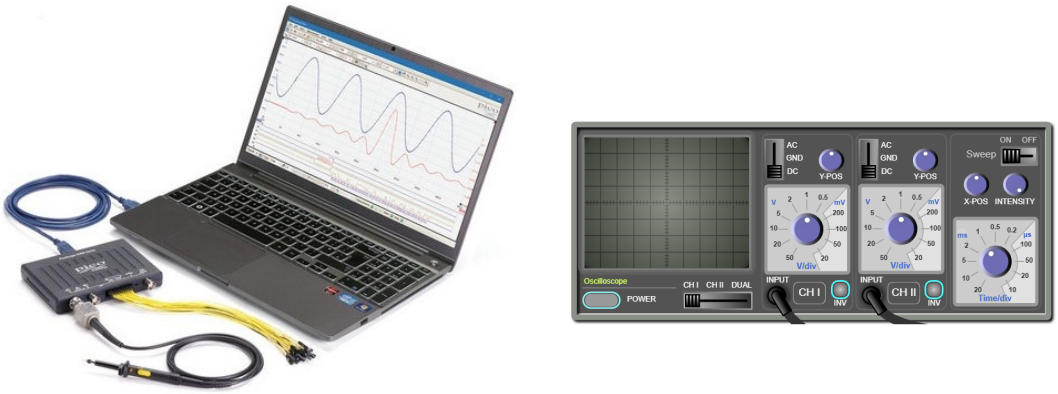
Os osciloscópio reais são instrumentos completos e autónomos, prontos a utilizar. Dentro desta categoria incluem-se os analógicos (Figura 2.2a) e os digitais ou de amostragem (Figura 2.2b), que diferem entre si especialmente no tratamento do sinal de entrada e nos métodos de representação.



(a) Osciloscópio analógico [3] (b) Osciloscópio digital [4]

Figura 2.2: Exemplos de diferentes tipos de osciloscópios "reais": (a) analógico e (b) digital.

Já os osciloscópios virtuais exploram a capacidade de processamento dos computadores para executar parte, ou mesmo a totalidade, das funções do equipamento. Nesta categoria podem distinguir-se, de forma geral, dois subtipos: os osciloscópios com aquisição de dados (Figura 2.3a) e os osciloscópios sem aquisição de dados ou simuladores de osciloscópio (Figura 2.3b).



(a) Osciloscópio virtual com aquisição de dados (b) Simulador de osciloscópio [5]

Figura 2.3: Exemplos de diferentes tipos de osciloscópios "virtuais": (a) com módulo I/O externo e (b) simulador.

2.2.2 Instrumentação real vs. virtual

Por mais que a evolução tecnológica e a crescente capacidade de processamento dos computadores tornem a instrumentação virtual uma realidade, existe uma série de mais-valias que apenas os instrumentos reais podem oferecer:

- **Desempenho superior:** Equipamentos físicos dedicados oferecem taxas de amostragem, resoluções e fidelidade ao sinal tipicamente superiores;
- **Imediatismo:** Estão prontos a operar quase instantaneamente, sem a necessidade de inicialização natural dos sistemas informáticos;
- **Independência tecnológica:** Não dependem de sistemas operativos ou atualizações de *software* que possam criar incompatibilidades entre utilizadores;
- **Realismo:** Permitem observar fenómenos físicos que podem não ser fielmente reproduzidos por soluções virtuais, como interferências ou comportamentos não ideais dos componentes;
- **Robustez e fiabilidade:** Normalmente suportam usos mais intensivos do que os aparelhos virtuais, garantindo medições consistentes e maior durabilidade.

No entanto a instrumentação baseada em computador oferece vantagens que num meio físico seriam dificilmente alcançadas:

- **Flexibilidade e personalização:** Uma arquitetura de *software* permite maior adaptação às necessidades do utilizador de forma relativamente simples e económica, especialmente no que diz respeito à interface e controlo dos sinais;

- **Custo reduzido:** Geralmente requer apenas um computador e *software* específico ou uso do navegador. Mesmo quando envolve módulos externos, o custo tende a ser inferior ao de equipamentos de bancada;
- **Portabilidade:** O transporte de um computador portátil e, eventualmente, de algum *hardware*, é com certeza mais fácil do que um osciloscópio físico, permitindo o seu uso em diferentes locais;
- **Integração:** É possível combinar, numa só aplicação, funcionalidades que no mundo real exigiriam múltiplos aparelhos (e.g. editor de circuitos, osciloscópio, gerador de sinais, multímetro);
- **Recursos do computador:** Para além de todo o processamento e representação, o computador ainda oferece capacidade de armazenamento, de comunicação e partilha de dados, etc.

Resumindo, embora a virtualização apresente limitações, as oportunidades que oferece conferem-lhe um papel de destaque importantíssimo. Ainda assim, este tipo de instrumentação virtual varia ainda bastante naquilo que é a sua arquitetura geral, o que significa que os vários tipos de virtualização existentes podem ainda potenciar ou minimizar, vantagens ou desvantagens referidas anteriormente. Para isso é fundamental entender os diferentes "tipos de virtualização" até para perceber onde realmente se enquadra o osciloscópio virtual desenvolvido neste projeto.

2.2.3 Graus de virtualização

Considerando os dois subtipos de equipamentos virtuais caracterizados na Subsecção 2.2.1, estes podem ser diferenciados pelo seu nível de virtualização, isto é, pela relação *hardware–software* que está presente na sua arquitetura.

Dentro dos osciloscópios com aquisição de dados temos aqueles com módulo I/O interno, externo (Figura 2.3a) e ainda os que comunicam diretamente com um osciloscópio físico, como é o caso da aplicação do *Oscilloscope Interface* [6]. Estes modelos recebem sinais exteriores ao computador para que este proceda a apresentação e posterior possibilidade de controlo.

Já os simuladores de osciloscópio, a virtualização não depende de dados externos nem de módulos de aquisição ou comunicação com equipamentos reais. O sinal é gerado unicamente por *software* assim como a sua representação e interface de controlo. No entanto, dentro desta categoria, alguns funcionam de forma autónoma através, por exemplo, de geradores de sinais integrados permitindo explorar as funcionalidades do osciloscópio simulado; outros são alvo de integração em simuladores de circuitos, permitindo uma experiência mais ampla com dados de entrada que simulam os sinais reais, assim como pretende esta aplicação.

2.2.4 Simulador web vs. local

Podemos ainda fazer uma paralelismo entre as aplicações que dependem de instalação local e as que são acessíveis através do navegador. A escolha entre desenvolver um simulador de osciloscópio como aplicação *web* ou local baseia-se em fatores comuns a outras aplicações, tais como desempenho, acessibilidade, contexto de utilização, etc. No caso desta aplicação em específico, a escolha já estava previamente definida, no entanto, importa contextualizar os motivos e as vantagens que sustentam essa opção.

Vantagens do simulador local

- **Desempenho superior:** Aproveita integralmente o *hardware* da máquina, permitindo tempos de resposta mais rápidos e processamento mais eficiente. Ideal para cálculos complexos, aquisição de dados em tempo real e elevada taxa de amostragem;
- **Funcionalidade *offline*:** Permite a utilização sem ligação à internet, o que torna a aplicação ainda mais geograficamente versátil;
- **Maior controlo e privacidade:** O armazenamento de dados é local, menor exposição a perdas de informação e eventualmente configurações mais personalizadas.

Vantagens do simulador *web*

- **Acessibilidade universal:** Pode ser utilizado em qualquer dispositivo com ligação à internet , sem instalação prévia;
- **Atualizações automáticas:** Novas funcionalidades e correções são disponibilizadas de forma imediata a todos os utilizadores;
- **Compatibilidade multiplataforma:** Funcionamento consistente em diferentes sistemas operativos e dispositivos;
- **Colaboração em tempo real:** Ideal para ensino remoto, e colaboração entre utilizadores.

Concluindo, mesmo que aplicações locais continuem a ser preferíveis quando o desempenho e o processamento elevado são prioridades — como acontece em práticas mais avançadas e profissionais —, no contexto em que este aplicação se insere, as soluções *web* ganham vantagem pela sua acessibilidade, facilidade de manutenção, colaboração e integração com plataformas online já existentes como é o caso da *U=RIsolve Academy*.

Capítulo 3

Estado da Arte

Neste capítulo, será apresentado uma revisão das principais plataformas e tecnologias relevantes para a temática deste projeto, com o objetivo de enquadrar e fundamentar as opções técnicas adotadas. Serão analisadas aplicações previamente desenvolvidas e internas ao Instituto Superior de Engenharia do Porto (ISEP), que serviram de motivação e referência para o desenvolvimento desta solução, bem como ferramentas e plataformas disponíveis no mercado, utilizadas como fontes de inspiração e orientação. Adicionalmente, será realizada uma breve análise das tecnologias de software atualmente disponíveis e amplamente utilizadas nestas áreas de desenvolvimento.

3.1 Projetos desenvolvidos pela equipa *U=RIsolve*

Esta aplicação de simulador de osciloscópio surge no seguimento de outros projetos desenvolvidas no mesmo âmbito de aprendizagem dos fundamentos de circuitos elétricos — mais especificamente, no seguimento de enriquecer a aplicação *U=RIsolve Academy - Circuito Editor*. Esta aplicação, surge igualmente como resultado de várias outras etapas e iniciativas que veem sido desenvolvidas desde 2018 pelo Departamento de Engenharia Electrotécnica (DEE) do ISEP, envolvendo docentes e alunos, e que culminaram no *framework* geral *U=RIsolve Academy*.

3.1.1 De *U=RIsolve* até *U=RIsolve Academy*

Esta iniciativa teve a sua origem em 2018 com a aplicação *U=RIsolve* desenvolvida por Lino Sousa [7], e que consistia num página *web* especializada em análise de

circuitos elétricos, nomeadamente usando o Método das Tensões nos Nós (MTN) através do *upload* de um esquemático (.sch) do circuito ou de um ficheiro *Netlist* [8] com a descrição do mesmo.

A partir desse momento, foram desenvolvidos vários outros projeto no âmbito do *U=RIsolve*. Como resultado, foram integrados novos métodos de análise no *framework* inicial, nomeadamente o Método das Correntes nos Ramos (MCR), desenvolvido por Hélder Casanova [9] e o Método das Correntes nas Malhas (MCM), desenvolvido por Ângelo Pinheiro [10], tornando a aplicação ainda mais rica e pedagogicamente viável.

A versão incial da ferramenta *U=RIsolve* é apresentada na seguinte Figura 3.1 e caso o interesse seja a sua utilização, uma versão mais recente e melhora da página pode ser visitada em `https://urisolve.pt/app/`.

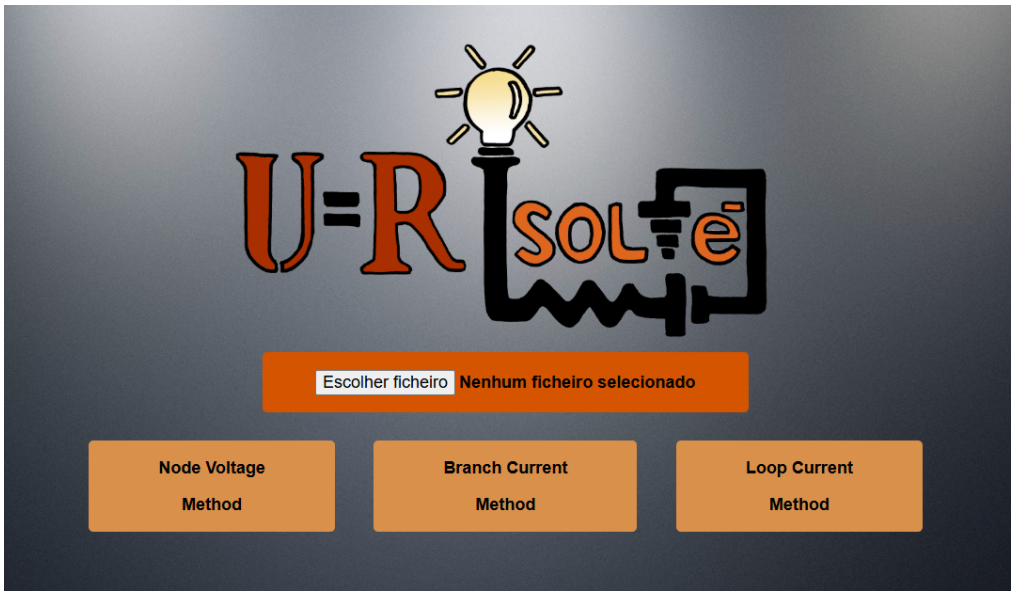


Figura 3.1: Versão original da aplicação U=RIsolve

Com base nesta evolução, surgiu o interesse em ampliar o alcance da aplicação, de forma a abranger diferentes meios e possibilidades de estudo e tornando assim um único *framework*, num portal de ensino completo.

Neste contexto, em 2023 a ferramenta evoluiu com o desenvolvimento do *U=RIsolve Academy*, incorporando aos trabalhos já existentes no âmbito da *U=RIsolve*, o projeto desenvolvido por Ana Castedo [11]. Este consiste numa forma alternativa para o estudo de circuitos elétricos através de perguntas de escolha-multiplca e resposta curta, sob a forma de *quizzes*, tornando o estudo mais interativo e didático. Um exemplo de pergunta pode ser observado na Figura 3.2.

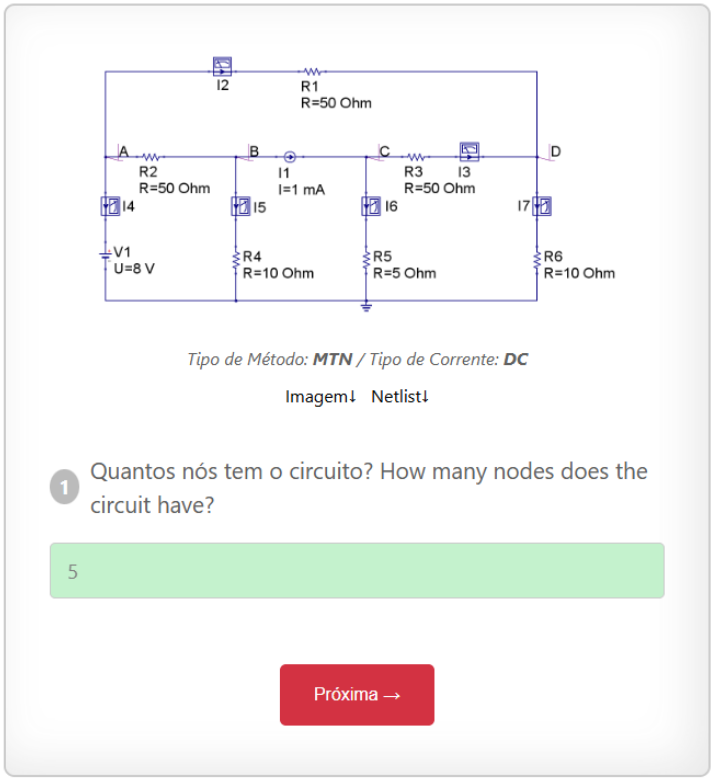


Figura 3.2: Exemplo de pergunta num *quizz* do U=RIsolve Academy

A ideia passou a ser, então, expandir o *U=RIsolve Academy*, reunindo numa única plataforma online diversas ferramentas de prática e estudo de circuitos. Entre estas encontram-se os projetos anteriormente referidos, um editor e simulador de circuitos (apresentado na Subsecção 3.1.2), o simulador de osciloscópio realista denominado "*Oscilloscope Interface*" (apresentado na Subsecção 3.1.3), bem como o simulador de osciloscópio genérico desenvolvido neste projeto com finalidade de ser integrado no editor de circuitos.

Para que isso fosse possível, foi realizada uma reestruturação da arquitetura do *framework* de maneira a possibilitar a manutenção e escalabilidade para atuais e futuras integrações. Essa reestruturação foi da responsabilidade de João Ferreira (co-orientador deste projeto) como parte da sua dissertação de mestrado.

A Figura 3.3 representa o protótipo da página inicial da plataforma *U=RIsolve Academy* (ainda em desenvolvimento e testes), na qual já é possível observar, na barra lateral, várias dinâmicas agrupadas. A aba de "Treino" que se refere aos *quizzes* [11]. É também possível observar uma aba de "Simulador" que diz respeito à aplicação para estudo dos métodos de análise [7][9][10]. Posteriormente, o objetivo é naturalmente reformular e adicionar uma aba para o editor de circuitos e para o simulador de osciloscópio realista.

Figura 3.3: Protótipo da página Inicial do *framework* U=RIsolve Academy

Numa era em que a Inteligência Artificial (IA) assume um papel cada vez mais predominante, importa ainda mencionar o interesse e planeamento de uma futura comunicação *Application Programming Interface* (API) entre a plaforma *U=RIsolve Academy* e o *framework HALO*, desenvolvido pelo professor André Rocha (orientador deste projeto), com o objetivo de obter um modelo IA para assistência personalizada no estudo e analise de circuitos elétricos.

3.1.2 U=RIsolve Academy - Circuit Editor

Assim como já foi mencionado, o simulador de osciloscópio abordado neste relatório surge no seguimento de enriquecer a ferramenta de editor de circuitos, em desenvolvimento por Ângelo Pinheiro. Este editor de circuitos, à semelhança de outros conhecidos no mercado, irá permitir o desenho e configuração de circuitos elétricos, no entanto de forma simples e didática, dado ao design e identidade visual característica da *U=RIsolve Academy*.

Na Figura 3.4 é possível observar um exemplo de circuito RLC desenhado no editor de circuitos referido.

Para além da edição e configuração será possível — como ilustrado no janela "Instrumentos de medição" da figura — estudar os circuitos tirando medições através dos instrumentos já implementados (e.g. voltímetro, amperímetro, etc.). Futuramente com a integração do simulador de osciloscópio desenvolvido, será possível uma análise mais aprofundada e completa através do estudo dos sinais em pontos específicos dos circuitos.

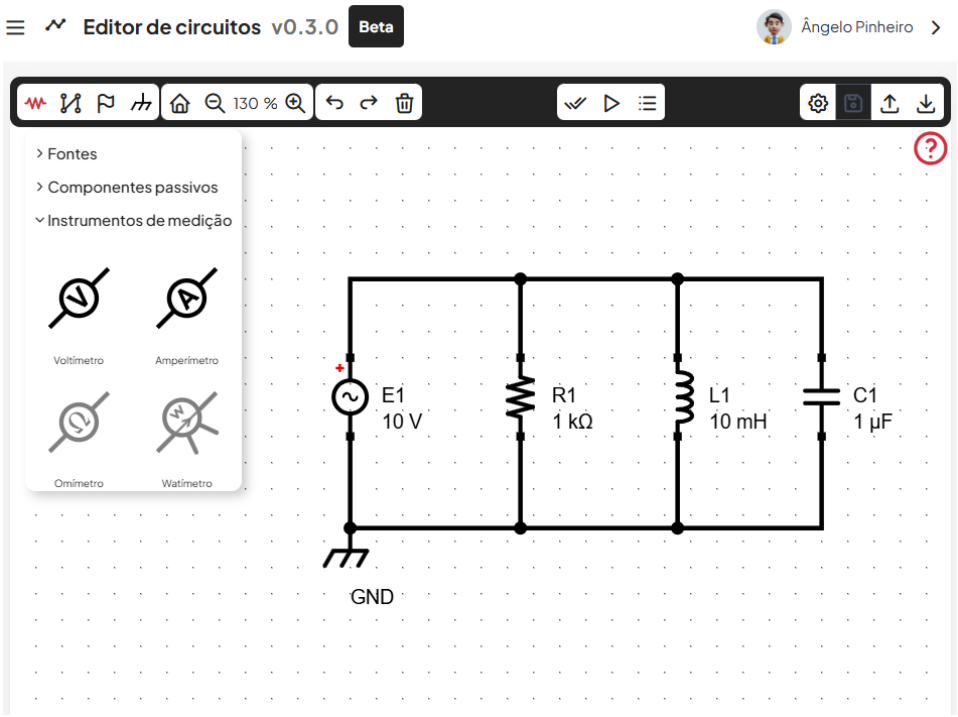


Figura 3.4: Exemplo de circuito usando o editor de circuitos U=RIsolve Academy

Apesar da integração não ter sido concretizada, tal como era inicialmente proposto, por razões que ultrapassam este trabalho, a sua idealização já está definida e será abordada no Capítulo 4 (Arquitetura e Implementação).

3.1.3 Oscilloscope Interface

Em paralelo com o desenvolvimento do *U=RIsolve Academy* em 2023, era também desenvolvida uma aplicação, que pode ser considerada a primeira dentro do *framework Academy* e que merece particular destaque por ter sido a principal referência e inspiração para o desenvolvimento deste projeto.

A aplicação consiste num simulador e interface realista do osciloscópio *GW-Instek GDS2062* [12] em ambiente *web*, da autoria de João Ferreira, denominada *Oscilloscope Interface* [6].

Esta aplicação teve uma forte adesão por parte dos alunos, devido tanto às suas características tecnológicas e funcionalidades práticas, quando à sua componente pedagógica. Tornou-se, especialmente na LEEC, uma ferramenta de ensino para os princípios da instrumentação, com especial destaque para o ensino à distância. A sua vasta utilização fala por si: já foi utilizada por mais de 250 alunos no âmbito da Unidade Curricular (UC) de Teoria dos Circuitos (TCIRC) da LEEC do ISEP e a sua utilidade, importância e funcionalidades gerais podem ser consultadas com maior detalhe nas revistas publicadas, nomeadamente Pratica [13] e MDPI [14]. O

projeto recebeu ainda o prémio de melhor póster, na apresentação do artigo feita na P.PIC de 2024 [15].

Esta aplicação está igualmente disponível no site `http://osciloscopio.dee.isep.ipp.pt/` e a sua interface de utilização é representada na figura 3.5.

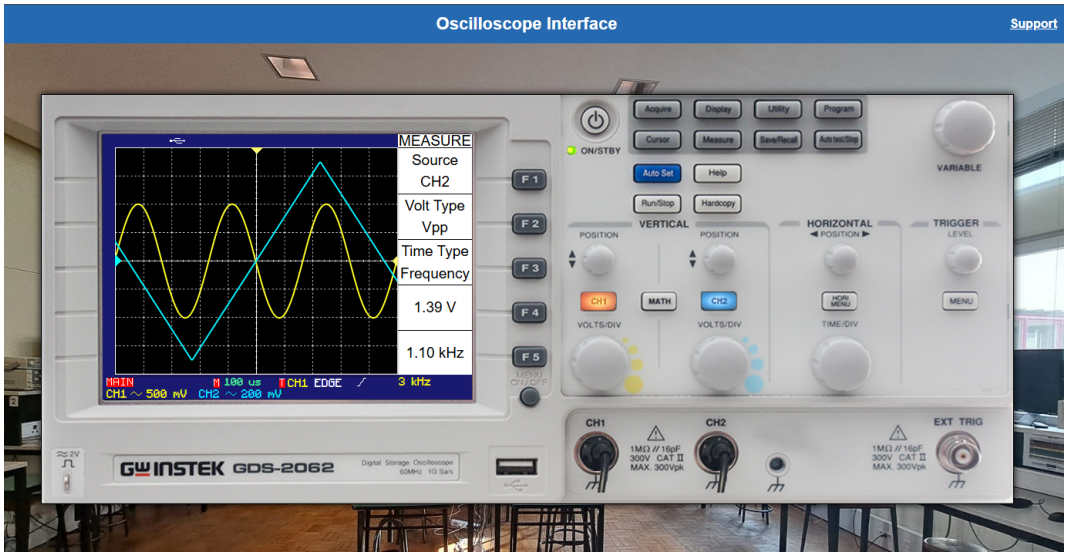


Figura 3.5: *Oscilloscope Interface* representando dois sinais

O seu funcionamento organiza-se em dois modos principais:

1. **Aquisição e controlo:** Permite o controlo remoto, aquisição, e representação dos dados presentes num osciloscópio real de bancada.

2. **Simulação:** Permite o uso de um osciloscópio *real-alike* com sinais simulados através de um gerador de sinais integrado ou através do *upload* de ficheiros *Comma-separated values* (CSV).

O primeiro modo de funcionamento opera numa ligação cliente-servidor, estabelecendo a comunicação entre o computador pessoal do utilizador (cliente) e o computador com ligação ao osciloscópio (servidor). Apesar deste modo de funcionamento ser super interessante e útil para casos específicos de estudo é no segundo modo (*client-only*) que reside a lógica de interesse para o desenvolvimento deste trabalho.

Fundamentalmente, este modo dá uso à linguagem de programação JS, com especial interesse nas tecnologias e *frameworks JavaScript* utilizados, mais propriamente o *Node.js* (ambiente do servidor) e o D3.js (representação gráfica das formas de onda) [1], que serão abordados com maior detalhe na Secção 3.4 e no Capitulo 4.

Assim, dada esta aplicação, será possível justificar a escolha de certas tecnologias e implementações feitas ao longo deste projeto, dado que não houve a necessidade de "inventar novamente a roda". Apenas transforma-la.

3.1.4 Outros projetos

Embora não estejam diretamente relacionadas com este projeto, foi realizada uma breve análise a aplicações que merecem ser mencionadas, uma vez que continuam a ser relevantes para o estudo e compreensão dos fundamentos do osciloscópio. Estas aplicações motivaram projetos anteriores, e portanto, de forma indireta, mantêm a sua importância no contexto tecnológico em que este trabalho se insere. Para além disso, foram igualmente desenvolvidas no âmbito da LEEC e abordam temáticas idênticas, nomeadamente simuladores de osciloscópio.

São duas estas aplicações: um simulador de osciloscópio analógico e um simulador de osciloscópio digital, desenvolvidos respetivamente por Pedro Salgueiro [16] e João Pereira [17].

3.2 Editores de circuito com simuladores integrados

São várias as tecnologias de simulação e análise de circuitos e sistemas elétricos e eletrónicos que se encontram hoje disponíveis, tanto do ponto de vista educacional quanto profissional. Por isso mesmo, foi fundamental decidir quais eram ou não relevantes para o estudo a realizar.

Parte destas ferramentas irão acompanhar aquilo que é o percurso de um futuro engenheiro eletrotécnico (e não só), portanto, a decisão de qual delas usar é igualmente importante para o par aluno e docente/instituição. A ideia do *U=RIsolve Academy* é facilitar essa decisão, pelo menos numa fase inicial do processo.

De entre elas, algumas são particularmente importantes para este projeto porque fizeram e fazem parte da jornada académico dos alunos de LEEC e outras porque se encaixam, em parte da sua arquitetura, naquilo que foi inicialmente proposto implementar nesta aplicação. Nesse sentido, foram tidas em consideração cinco tecnologias e aplicações: o simulador Qucs, PSpice, *Tinkercad*, *Falstad* e *DCAClab*.

3.2.1 QUCS

Quite Universal Circuit Simulator (Qucs) [18] é um simulador de circuitos *open-source* que apesar de ter algum uso avançado é um dois primeiros softwares que os alunos de LEEC têm contacto. Possui uma boa biblioteca de componentes tendo em conta a seu intuito não comercial, para além disso dispõem de uma GUI simples, mas completa, que facilita o ensino teórico e o estudo de circuitos e componentes elétricos através de vários tipos diferentes de análise, incluindo *Direct Current* (DC), *Alternating Current* (AC), transiente, parâmetros S, etc.

Na sua versão mais recente, o *Qucs-S* substitui o motor tradicional por motores como o *Ngspice* ou o *Xyce*, permitindo gerar *netlists* [8] compatíveis com bibliotecas

e modelos *Simulation Program with Integrated Circuit Emphasis* (SPICE). Esta evolução aumenta a precisão e a compatibilidade com tecnologias comerciais, reforçando a sua versatilidade [19][20].

No entanto, o Qucs não integra um osciloscópio nem a sua interface visual. As formas de onda e resultados são apresentados em gráficos e diagramas personalizados (e.g. cartesianos, polar, temporais, etc.) gerados após a simulação, o que lhe confere uma abordagem mais analítica do que interativa e didática. Juntamente com isso, e apenas de ser compatível com vários sistemas operativos, não é *web-based*, o que desmotivou uma possível referência mais aprofundada para este projeto.

No entanto, pelos vários motivos mencionados anteriormente este simulador continua a ser uma ótima ferramenta para o ensino e práticas de simulações académicas bem como para uso dos seus motores de simulação em *third-party applications* [21].

3.2.2 PSpice

Personal Simulation Program with Integrated Circuit Emphasis (PSpice), é uma das mais conceituadas ferramenta de simulação de circuitos elétricos e eletrónicos, desenvolvida pela Cadence e pertencente à família tecnologica SPICE [22]. É amplamente utilizado no meio industrial e investigação mas com bastante ênfase no meio académico para simulação de sistemas analógicos e digitais através da plataforma gratuita *Pspice for TI* [23], oferecendo suporte para análises DC, AC, transiente, Monte Carlo, entre muitas outras (assim como outras ferramentas da família SPICE e *Computer-Aided Design* (CAD)) através da tecnologia *Pspice A/D ou Probe* [24].

A par com o Qucs é uma das tecnologias que os estudantes de LEEC têm contanto, no entanto, é introduzido numa fase mais avançada para o estudo de circuitos mais complexos e robustos.

Não inclui um simulador osciloscópio, propriamente dito, apenas representação pós-simulação das analises. Apesar de oferecer diversas funcionalidades de controlo, algumas delas semelhantes aquelas presentes num instrumento real [24], a sua interface não corresponde a experiência de um osciloscópio físico.

Além disso, exige instalação local e conhecimentos técnicos mais avançados para tirar pleno proveito das suas funcionalidades. A sua precisão de resultados e compatibilidade com normas industriais tornam-lo numa referência para o estudo avançado mas não ideal para fins introdutórios e didáticos.

3.2.3 Falstad

O *Falstad Circuit Simulator* ou *CircuitJS* é um simulador *web-based, open-source*, desenvolvido por Paul Falstad [25]. Incialmente desenvolvido em *Java* e posteriormente passado para *JavaScript*, é uma ferramenta comummente utilizado em contextos educacionais pela sua interface intuitiva e interativa, permitindo desenhar

circuitos e visualizar o seu comportamento com animações que simulam a corrente e tensão no circuito.

Integra a representação de sinais em "tempo real" que simula de forma simplificada o funcionamento tradicional de um osciloscópio, permitindo medições e análises, no entanto sem um interface de controlo semelhante ao equipamento real.

Apesar de ser menos completo que o Qucs e o PSpice em termos de recursos e componentes, a sua natureza web e a facilidade de utilização tornam-no uma excelente ferramenta para o ensino e estudo de circuitos elétricos, principalmente numa fase inicial.

3.2.4 Tinkercad

Para além das ferramentas usadas ao longo do percurso académico dentro das unidades curriculares, o Tinkercad foi uma das que se destacou tendo em conta o intuito do projeto.

Desenvolvida pela Autodesk, é uma plataforma desenvolvida em ambiente CAD, *web* e gratuita, que oferece um ambiente integrado não apenas para design e estudo de circuitos mas para, para modelação 3D e programação de microcontroladores [26].

Disponibiliza um simulador realista que permite a criação de esquemas, ligação de componentes em *breadboards* e execução de código em placas como o *Arduino Uno*.

Inclui um osciloscópio virtual básico, que apresenta os sinais, à semelhança do Falstad, mas com interface muito simplificada e com poucas opções de configuração, tal como é possível observar na Figura 3.6, comparativamente a um osciloscópio real e aquilo é o proposto nesta aplicação.

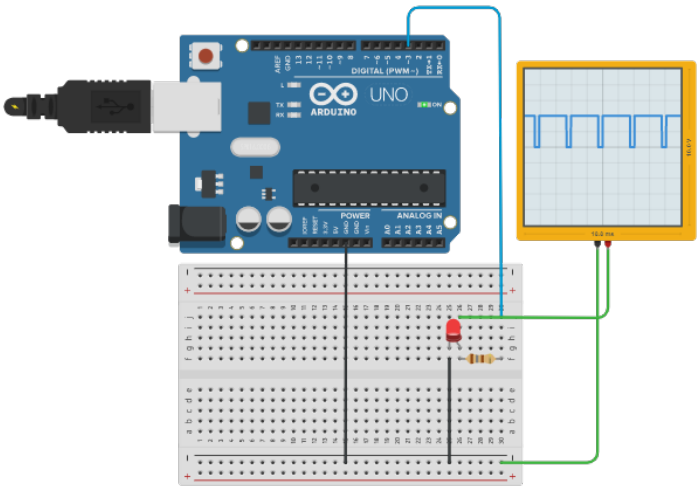


Figura 3.6: Representação de um sinal PWM no osciloscópio do simulador TinkercAd

Para a prática eletrónica mostra-se uma plataforma muito rica e completa, destaca-se pela acessibilidade, curva de aprendizagem reduzida e disponibilidade *web*, sendo especialmente indicado para ensino introdutório e prototipagem e micro-controladores, enquadrando-se muito bem naquilo que é a temática deste projeto.

3.2.5 DCACLab

A última ferramenta analisada foi o DCACLab [27], um simulador *web-based* com fins essencialmente didáticos que permite explorar os fundamentos de circuitos elétricos e eletrónicos através de um editor realista e interativo, instrumentação virtual e análises transiente e DC *sweep*.

A sua interface simula uma bancada de trabalho, que apesar de se poder tornar sobrecarregada, é intuitiva e eficaz para a simulação de situações práticas de estudo, permitindo a utilização de instrumentos como multímetros, gerador de sinais e, especialmente, um osciloscópio com uma representação das formas de onda semelhante ao Falstad e ao Tinkercad. No entanto, apresenta funcionalidades adicionais, possibilitando o controlo de escalas e *off-sets* verticais à semelhança do equipamento físico.

Apesar de possuir um leque mais restrito de funcionalidades e componentes em relação a simuladores de maior robustez, como o Qucs ou o PSpice, a sua abordagem visual mostram que o DCACLab pode ser uma ferramenta valiosa na fase inicial da aprendizagem de circuitos, especialmente em contextos de aprendizagem remota. Serviu de referência para este projeto, uma vez que se aproxima bastante daquilo que era proposto, oferecendo aos estudantes uma experiência rica e próxima da realidade laboratorial [28], tal como é possível observar na seguinte Figura 3.7.

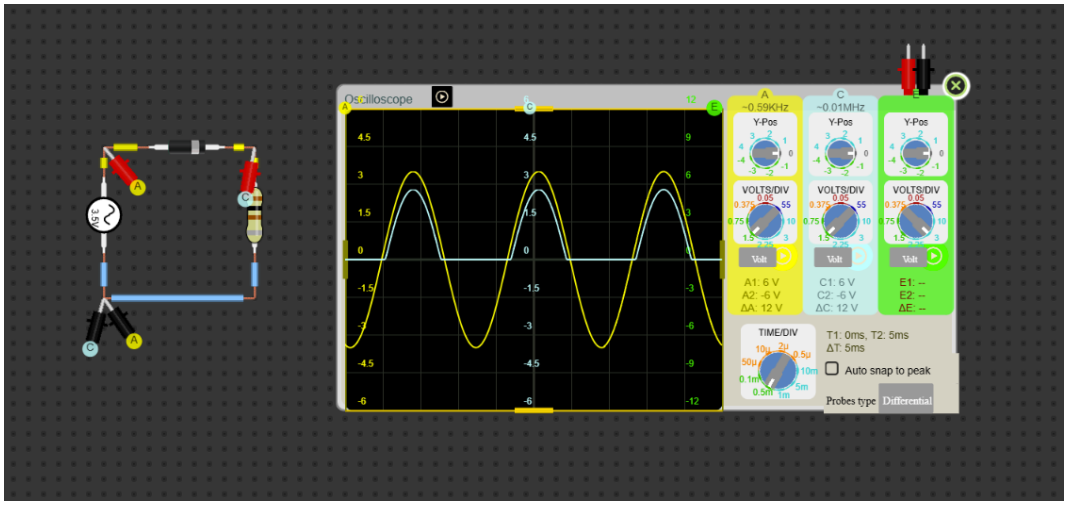


Figura 3.7: Comportamento de um circuito retificador de meia-onda representado no osciloscópio do simulador DCACLab

3.2.6 Análise comparativa das tecnologias

A partir da análise realizada, foi possível identificar os simuladores que melhor se adequam ao desenvolvimento da aplicação proposta, servindo de referência e inspiração para o processo de implementação.

O Qucs e o PSpice poderiam ter extrema importância para o estudo da comunicação entre editor e representação dos sinais. No entanto, pecam no que diz respeita à interface e utilização de um osciloscópio, que é limitado ou nenhuma, e que no fundo acaba por ser o cerne deste projeto.

Já o Falstad, existe o proveito de ser *open source*, e portanto, pode ser uma mais valia a exploração do seu *software* numa fase posterior de integração com o editor. Contudo, está em desvantagem em relação à implementação do osciloscópio face os dois últimos simuladores analisados.

O Tinkercad tem uma vertente muito forte em design gráfico, prototiagem 3D e microcontroladores, ao contrário do DCACLab, que é direcionado exclusivamente ao estudo de circuitos. Para além disso, apresenta um dimensionamento dos instrumentos de medição mais próximo do seu funcionamento real e daquilo que era a proposta para este projeto, inclusive ao nível de integração com o editor. Por esse motivo, e apesar de não ser *open source*, revelou-se, de entre os simuladores analisados a maior fonte de inspiração para solução final realizada.

No entanto, apesar de aproximar do funcionalmente real de um osciloscópio, este simulador apresentava limitações, sobretudo no que diz respeito às funcionalidades adicionais do equipamento que lhe conferem o seu destaque e utilidade na prática.

O DCACLab faz a representação das formas de onda e permite, ao nível de controlo, alterar escalas e *off-sets* verticais. Isto confere-lhe um funcionamento muito simplificado e uma utilização básica de um osciloscópio real, ainda longe daquilo que era pretendido implementar neste projeto.

Essas funcionalidades adicionais, ausentes em muitas das aplicações disponíveis, foram integradas na presente aplicação e serão descritas em maior detalhe no Capitulo 5.

A Tabela 3.1 sintetiza esta análise comparativa, permitindo observar as diferenças mais relevantes ao nível da arquitetura, da integração do osciloscópio, das funcionalidades e público-alvo entre as tecnologias.

3.3 Simuladores de osciloscópio independentes

Não diretamente relacionadas com uma ligação editor–instrumento, estão aplicações *stand-alone*. Grande parte destes simuladores são concebidos para a receber e representar sinais provenientes da placa de som do computador e/ou possibilitarem um controlo básico do osciloscópio. Ou seja, de modo geral não se enquadram,

Tabela 3.1: Comparação entre tecnologias para simulação e edição de circuitos

Simulador	**Ambiente e Licença**	**Osciloscópio Integrado**	**Tecnologias de Software**	**Público-alvo**	**Observações**
QUCS	Desktop; Open Source (GPL)	Não - gráficos de analise pós-simulação	*Qucsator*[a]; C++; Qt e Qpainter	Académico, Investigação	Focado em análises médias e avançadas; GUI simples; biblioteca completa
PSpice for TI	Desktop; Closed Source (gratuito)[b]	Não - gráficos de analise pós-simulação c/ controlo avançado	SPICE; C/C++; Pspice A/D	Indústria, Investigação, Académico	Alta precisão; análises avançadas; extensa biblioteca e suporte a modelos de fabricantes
Falstad	Web; Open Source (GPL)	Sim - s/ controlo	Java/JavaScript	Académico	Extremamente intuitivo; limitado em componentes; ideal para ensino introdutório
Tinkercad	Web; Closed Source (gratuito)	Sim - c/ interface simplifica; c/ controlo básico	JavaScript + WebGL;	Académico	Prototipagem rápida e 3D através de breadboard; introdução a microcontroladores e circuitos simples
DCACLab	Web; Closed Source (gratuito)	Sim - c/ interface simplificada; c/ controlo razoável	JavaScript	Académico	GUI realista; estudo introdutório e simples; proximidade laboratorial; boa replicação de equipamentos

[a]Versão Qucs-S utiliza motor de simulação Ndspice/Xyce ao invés de Qucsator.
[b]Versão comercial do PSpice é paga.

nem acrescentam funcionalidades relevantes, face às tecnologias anteriores, em relação ao objetivo do presente projeto. Ainda assim, merecem uma referência, uma vez que serviram de apoio sobretudo em relação ao *design/layout* do simulador de osciloscópio desenvolvido.

Entre estas aplicações destacam-se o *Virtual Oscilloscope* [5] e o *Interactive Virtual Oscilloscope* [29], pelas suas interfaces simples e voltadas ao uso introdutório dos equipamentos.

3.4 Estado do Software

O desenvolvimento de aplicações *web* modernas dispõem de um conjunto diversificado de tecnologias a ser utilizadas tanto do lado do cliente (front-end) como do lado do servidor (back-end).

A sua escolha depende tanto daquilo que é proposto implementar na aplicação, como da preferência do desenvolver e/ou organização.

Para além das linguagens de programação já bem cimentadas e popularmente conhecidas como o HTML, CSS e JS, exitem uma série de frameworks, bibliotecas e utilitários que facilitam a implementação de funcionalidades especificas, a criação de interfaces responsivas[1], a comunicação otimizada com os servidores e a gestão do ciclo de vida do software.

Nos seguintes tópicos será feita uma abordagem breve entre as tecnologias mais populares e utilizadas no âmbito geral de desenvolvimento *web* e respetivos utilitários e nesse sentido foi usado como fonte o *Stack Overflow Developer Survey 2024* [30]

3.4.1 Tecnologias Back-end

Uma das primeiras escolhas a considerar passa pelo ambiente/linguagem *back-end* em que o projeto deverá assentar. São muitas as possibilidades disponíveis e, em geral, a decisão depende não apenas da preferência pessoal ou organizacional, mas também da análise das necessidades e das expectativas futuras para a aplicação.

Dado que o interesse deste projeto não é focado em desenvolvimento *server-side*, e ao mesmo tem como finalidade a integração numa plataforma que já está devidamente estruturada, essa decisão não foi necessária.

O *framework U=RIsolve Academy* assenta no ambiente *Node.js* que garante escalabilidade, praticidade e suporte a *server-side rendering* [31] o que o torna ideal para aplicações em tempo real[2]. No entanto, essa escolha será devidamente referida no Capítulo 4.

[1] Interfaces que se adaptam automaticamente às condições de utilização.

[2] Aplicações que precisam de respostas rápidas a interações do utilizador ou a eventos externos

Na Tabela 3.2 são listados alguns exemplos de *frameworks* populares em diferentes ambientes de desenvolvimento *back-end*, assim como uma breve observação sobre cada conjunto.

Tabela 3.2: Tecnologias back-end comuns no desenvolvimento web

Ambiente	**Framework**	**Observações**
Node.js	Express.js	Minimalista e flexível; ideal para aplicações *web* leves e APIs REST; simplifica o *routing* e integração de *middlewares* [32].
Node.js	Next.js	Estrutura fullstak; utilizado tanto no front-end, com React, quanto no back-end.
Python	Django	Completo; Poucas dependências externas; adequado para aplicações de larga escala.
.NET (C#)	ASP.NET Core	Multiplataforma e alto desempenho; integração nativa com ecossistema Microsoft.
Java	Spring Boot	Robusto e escalável; padrão em soluções empresariais; vasta integração do ecossistema Java.
PHP	Laravel	Sintaxe clara e produtiva; foco na fácil manutenção e escalabilidade.

3.4.2 Tecnologias Front-end

Dada a natureza do projeto, predominantemente focado no desenvolvimento *front-end*, foi neste contexto que surgiu o maior interesse em estudar as tecnologias de software e onde houve a possibilidade de ponderação e escolha de quais as mais viáveis e produtivas, tanto para o propósito a serem usadas, como para o seguimento dos projetos anteriores e, naturalmente, para a preferência do autor.

Este estudo está divido em duas partes: em primeiro lugar, abordam-se os *frameworks* e bibliotecas mais cobiçados e relevantes, associados às linguagens de desenvolvimento já mencionadas e que otimizam a sua utilização; em segundo lugar, analisam-se as tecnologias ligados à dinâmica de representação gráfica assentes na linguagem de scrip JS.

No que respeita à estrutura e organização dos elementos da interface *web*, destacam-se as tecnologias que permitem gerar HTML de forma dinâmica, os chamados motores de *templates*.

Relativamente à personalização da página destacam-se as ferramentas que permitem agilizar esse processo de edição, tornando-o menos exaustivo e mais eficiente.

Do ponto de vista da interatividade e das funcionalidades da aplicação, estão as diversas ferramentas e bibliotecas que tornam a linguagem JS tão versátil e utilizada. Neste caso específico, a seleção das tecnologias foca-se em *frameworks* e bibliotecas destinadas à manipulação do *Document Object Model* (DOM) [33] e, para a simulação dos sinais do osciloscópio, bibliotecas destinadas à representação gráfica.

Na Tabela 3.3 segue a apresentação das tecnologias, a categoria em que se inserem, o tipo de tecnologia, e observações sobre as mesmas.

Tabela 3.3: Tecnologias front-end comuns e relevantes para o projeto

Tecnologia	**Categoria**	**Tipo**	**Observações**
EJS	HTML	Motor de template	Permite inserir JS diretamente no HTML.
Handlebars.js	HTML	Motor de template	Inspirado no Mustache [34], gera HTML de forma dinâmica, separando a lógica da apresentação
Bootstrap	CSS	Framework	Componentes prontos e pré-estilizados.
Tailwind CSS	CSS	Framework	Classes utilitárias para design rápido.
Sass/SCSS	CSS	Pré-processador CSS	Variáveis globais e código CSS mais organizado.
React	JavaScript	Biblioteca	Ideal para UI modular e dinâmica.
Vue.js	JavaScript	Framework	Fácil de aprender e reativo; No meio do caminho entre o React e o Angular.
Angular	JavaScript	Framework	Completo e escalável, ideal para SPAs .
D3.js	JavaScript	Bilioteca	Representação gráfica de dados e gráficos dinâmicos; Flexível para gráficos SVG/Canvas.
Chart.js	JavaScript	Bilioteca	Representção gráfica com gráficos prontos e personalizáveis; Simples e fácil de usar.

Na Secção 4.2, serão abordadas, de entre estas, as tecnologias utilizadas na presente aplicação e os seus métodos de implementação.

3.4.3 Tecnologias Utilitárias

Para além das tecnologias diretamente relacionadas com o desenvolvimento do *software*, existe ainda um conjunto de ferramentas que desempenham um papel fundamental no apoio ao desenvolvimento, na gestão das aplicações, e eventualmente, na colaboração entre participantes.

De entre elas, de destacar as seguintes categorias, e respetivos exemplos tecnológicos:

- ***Package Managers*:** Automatizam a instalação, atualização e remoção de tecnologias numa aplicação, as chamadas dependências (e.g. *Node Package Manager* (npm), Yarn);
- **Ferramentas para Controlo de versões:** Essenciais no desenvolvimento moderno, permitem gerir o histórico do código e facilitar a colaboração através de plataformas online de alojamento do mesmo (e.g. Git/GitHub);
- **Ferramentas para depuração:** Permitem inspecionar a execução do código e identificar erros de lógica, de sintaxe e de funcionamento da aplicação (e.g. Chrome DevTools, VSCode Debugger)
- **Plataformas de Conteinarização:** "Empacotam" uma aplicação e as suas dependências, num único sitio, o contêiner. Permitindo que o *software* funcione de forma consistente, independentemente do ambiente onde é executado (e.g. Docker);
- **Ferramentas assíncronas:** Úteis para a comunicação e colaboração sem a necessidade de interação em tempo real, caracterizam-se também por permitirem a gestão do projeto, o planeamento e organização das ideias (e.g. Trello, Jira);
- **Ferramentas síncronas:** Possibilitam a comunicação e interação em tempo real (e.g. Zoom, Microsoft Teams).

Vários exemplos de tecnologias dados ao longo deste estudo do software foram utilizados no desenvolvimento do simulador de osciloscópio, quer como parte integrante da aplicação, quer no processo de gestão do projeto. A sua utilização será descrita de forma mais detalhada no capítulo seguinte.

Capítulo 4

Arquitetura e Implementação

Neste capítulo será abordada a Arquitetura geral da aplicação, do ponto de vista do seu propósito fundamental bem como do ponto de vista da versão atual. Será também referido o desenvolvimento do software, das suas respetivas etapas e eventuais desafios e soluções encontradas.

4.1 Arquitetura da aplicação

Conforme referido anteriormente, esta aplicação surge no interesse de integrar um simulador de osciloscópio no editor de circuitos da plataforma *U=RIsolve Academy*. Essa integração não foi ainda realizada, uma vez que depende de um terceiro módulo externo. No entanto, essa implementação encontra-se completa e poderá ser integrada futuramente. Assim, ainda que à data atual a aplicação funcione de forma autónoma, explicá-la no seu panorama de interesse final é fundamental para compreender não só o seu funcionamento presente, mas também a sua relevância nas fases posteriores de integração.

4.1.1 Visão geral

Na Figura 4.1 podemos observar a arquitetura final pretendida para este simulador de osciloscópio.

É possível, no diagrama da figura, ver aquilo que está atualmente implementando no simulador (a verde e azul), aquilo que foi implementados por terceiros (a laranja),

o editor, e aquilo que está planeado fazer num futuro próximo (a branco e tracejado) e que diz respeito à comunicação simulador–editor.

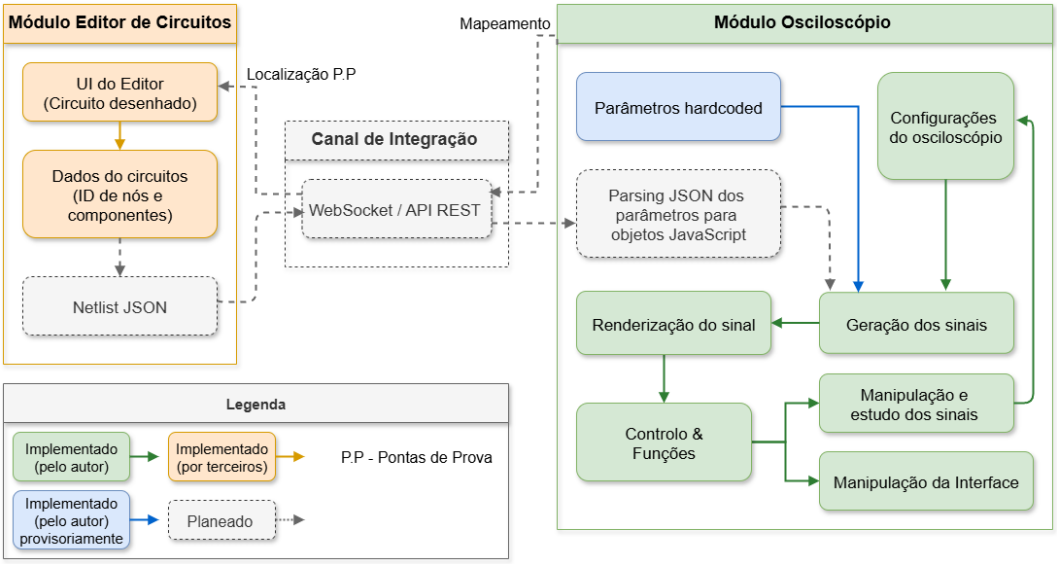


Figura 4.1: Visão geral da arquitetura da aplicação

À esquerda encontra-se o "Módulo Editor de Circuitos", responsável pela interface de desenho do circuito e pela produção da *netlist JavaScript Object Notation* (JSON) (parâmetros por nó e componente).

Ao centro está o "Canal de Integração", via *WebSocket* [35] ou *API REST*, que irá permitir localizar as pontas de prova no circuito desenhado e transportar a/as *netlists* correspondentes a esses pontos do circuito até ao simulador.

À direita situa-se o "Módulo Osciloscópio" com as funcionalidades efetivamente implementadas à data de redação deste relatório. Na versão final, os dados recebidos serão normalizados (*parsing*/objetos JS) e alimentariam a lógica do geração do sinal, parametrizado em termos de amplitude, frequência, fase e tipo de onda. Atualmente, estes parâmetros encontram-se geradores de forma *hardcoded*.

Por sua vez, e consoante as configurações do canal e do *trigger*, o sinal será reajustado, ficando pronto para a amostragem e consequente renderização ponto a ponto no ecrã do osciloscópio. Uma vez representado graficamente, o sinal pode ser controlado de forma análoga a um osciloscópio real através da interface do simulador, sendo ainda possível ajustar a interface quanto ao seu posicionamento e tamanho relativamente ao editor, de maneira a não perturbar o circuito desenhado.

4.2 Tecnologias e ferramentas utilizadas

No seguimento do Estudo de software realizado na Secção 3.4, as tecnologias utilizadas nesta aplicação são dividias em dois grupos: aquelas utilizadas diretamente no desenvolvimento do software e aquelas que contribuíram indiretamente para tal.

4.2.1 Tecnologias de desenvolvimento

De forma a conseguir executar JS do lado do servidor e pela sua ampla utilidade, foi usado o ambiente Node.js como fundação do *software*, bem como o *framework* Express.js para a definição simplificada das rotas de paginação [32].

Com especial importância, do lado do cliente, e de forma a permitir otimizar as linguagens *web* tradicionais e agilizar o processo de desenvolvimento do simulador foram utilizadas várias outras tecnologias:

- **Handlebars.js:** Utilizado para gerar HTML de forma dinâmica e modular, separando a estrutura da aplicação por vários ficheiros, melhorando a organização e escalabilidade do código.
- **SASS:** Com o mesmo propósito do Handlebars, permitiu separar o código CSS por módulos dedicados a uma parte especifica do osciloscópio, para além de permitir usar variáveis comuns a todo o *framework U=RIsolve Academy*.
- **JavaScript:** Utilizado para a manipulação do DOM, permitindo criar a dinâmica de interação e as funcionalidades da aplicação.
- **D3.js:** Utilizado para a representação gráfica das formas de onda e da sua manipulação.

4.2.2 Ferramentas paralelas ao desenvolvimento

Paralelamente às tecnologias de programação, foram utilizadas várias outras ferramentas que contribuíram bastante para o processo, quer ao nível das necessidades da aplicação, quer no apoio ao desenvolvimento do código, na organização do planeamento e comunicação entre aluno orientador, bem como na edição de imagens e na exploração de ideias, entre outros aspetos.

1. Ferramentas integradas no desenvolvimento:
    - ***Visual Studio Code* (VS Code):** *Integrated Development Environment* (IDE) utilizado para o desenvolvimento do software, devido à interface amigável e diversidade de extensões que ajudam no processo.
    - **npm:** Permitiu a instalação de dependências e tecnologias no software de forma rápida e simples (e.g. D3.js)

- **Git/GitHub:** Mais propriamente o GitHub *desktop*, para o controlo de versões e a hospedagem do código online.
- **Docker:** Utilizado para a contentorização da aplicação.

2. Tecnologias de apoio ao desenvolvimento:

- **PenPot e Figma:** Ferramentas utilizadas para o design da solução, bem como para a prototipagem interativa.
- **OneNote:** Útil para o desenho do *wireframe*[1] e para exploração de ideias de forma mais visual, através de esquemas.
- **Notion:** Para organizar ideias de forma mais individual (ao contrário do Trello), assim como anotação de dúvidas e problemas da aplicação.
- **Trello:** Utilizado para o planeamento das etapas de desenvolvimento e comunicação com os orientadores.
- **Zoom:** Utilizado para comunicação síncrona com o orientadores.
- **IA:** Foram utilizadas duas ferramentas de IA, nomeadamente o ChatGPT e o GitHub Copilot, para colaboração no desenvolvimento do código, exploração de ideias e especificações de *software*.
- **Inkscape:** Utilizado para criação, edição de imagens em formato *Scalable Vector Graphics* (SVG) (e.g indicadores das formas de onda, *knobs*).

De mencionar ainda as páginas *W3Shools* [36] e *MDN* [37] que serviram de consulta, especialmente nas fases mais primordiais do projeto. A pagina *Tabler Icons* [38] para a utilização de icons *SVG*. A página *HTML table generator* [39] que ajudou na formatação dos *grids* HTML, a ferramenta *html2canvas* [40] para os screenshots para download e o repositório GitHub *d3linechartseries* [41], para consulta e ajuda na compreensão e utilização do d3.js. De mencionar novamente o *Oscilloscope Interface* [6], sendo o seu código uma referência fundamental para o desenvolvimento desta aplicação.

4.2.3 Justificação das escolhas tecnológicas

Algumas destas ferramentas já estavam por defeito implementadas na estrutura da plataforma *U=RIsolve Academy* e portanto, de forma a manter a coerência tecnológica do *framework* foram também adotadas neste simulador de osciloscópio.

Por outro lado, a nível pessoal, e uma vez que a maior das tecnologias de desenvolvimento *web* eram novidade, de forma a não sobrecarregar o pré-estudo realizado e eventualmente atrasar o processo de desenvolvimento, optou-se por não se considerar algumas tecnologias demasiado particulares, sob pena de não tirar proveito das

[1] Esboço da aplicação, sem atenção ao detalhe, focado na estrutura e disposição dos elementos.

suas vantagens, especialmente a longo prazo e à medida que a aplicação escalava. Foi por isso utilizado, a alguns níveis do desenvolvimento uma abordagem mais "tradicional", mas que em nada compromete o funcionamento e eficiência da aplicação final. Por exemplo, o uso de *JavaScript* puro ou *Vanilla*, ao invés *frameworks* modernos como o React.

Em relação a outras ferramentas, foram utilizadas devido ao seu uso comum na área de desenvolvimento *web*, como é o caso do Figma, ou devido a familiaridade pessoal com as mesmas.

Um menção especial para as ferramentas de IA, que, face à inexperiência e ao desconhecimento inicial de várias práticas, recursos e capacidades das tecnologias, se revelaram particularmente úteis na exploração de ideias e na compreensão de código e conceitos, servindo muitas vezes como "motor de arranque" no processo de desenvolvimento. Contudo, estas ferramentas não são autossuficientes acabando por, muitas das vezes, dispersar e não necessariamente ajudar e acrescentar ao processo. Ou seja, hoje em dia, as suas vantagens justificam a sua utilização genérica, mas é fundamental uma capacidade crítica e de análise por parte do utilizador para tirar o melhor proveito destas ferramenta.

4.3 Estrutura de Pastas e Ficheiros

A arquitetura de *software* do *framework U=RIsolve Academy*, aperfeiçoada por João Ferreira (co-orientador deste projeto), segue uma organização modular de pastas e ficheiros que facilita a manutenção e integração de novas funcionalidades, incluindo o simulador de osciloscópio desenvolvido.

A listagem seguinte apresenta a estrutura de pastas e ficheiros da plataforma, onde os elementos assinalados a azul correspondem aos efetivamente utilizados no desenvolvimento da solução descrita neste relatório.

```
app/
|-- config/
|-- controllers/
|   |-- api/
|   |-- auth/
|   \-- pages/
|-- locales/
|   |-- en/
|   \-- pt/
|-- middleware/
|-- models/
|-- public/
|   |-- css/
|   |-- img/
|   |-- js/
```

```
16 |   |   |-- logic/
17 |   |   |-- pages/
18 |   |   |-- ui/
19 |   |   \-- utils/
20 |   |-- scss/
21 |   |   |-- abstracts/
22 |   |   \-- components/
23 |   \-- vendor/
24 |-- routes/
25 |-- utils/
26 |-- views/
27 |   |-- layouts/
28 |   |-- partials/
29 |   \-- pages/
30 \-- app.js
```

Listagem 4.1: Esquema em árvore da organização do software

Cada pasta e ficheiro serve um propósito bem definido na arquitetura final, no entanto para o propósito desta aplicação, ao momento atual, apenas algumas são realmente relevantes:

- **app.js:** Ficheiro principal que inicializa a aplicação, configurando o servidor Express, os *middlewares*, o motor de *templates* Handlebars e as rotas;
- **controllers/:** Agrupa a lógica central de roteamento da aplicação;
  - `pages/`: Responsável pela renderização das páginas (*views*) no lado do servidor.
- **routes/:** Contém todas as declarações de rotas da aplicação;
- **view/:** Organiza os *templates* Handlebars em diferentes níveis;
  - `layouts/`: Estruturas base partilhadas por várias páginas;
  - `partials/`: Componentes de interface reutilizáveis e onde assenta o ficheiro *oscilloscope.hbs* que define a estrutura visual do simulador de osciloscópio;
  - `pages/`: *Templates* específicos de cada página.
- **public/:** Diretório responsável pelos ficheiros estáticos da aplicação.
  - `css/`: Folhas de estilo compiladas a partir do código SCSS;
  - `img/`: Imagens utilizadas na interface;
  - `js/`: Scripts JavaScript, organizados em:
    - `pages/`: Lógica de inicialização de cada página renderizada no servidor;

    - `logic/`: Diretório onde reside toda a lógica de funcionamento do simulador;
    - `ui/`: Responsável apenas pela leitura das interações com o DOM, os *event listeners* [42];
    - `utils/`: Contém funções auxiliares comuns e variáveis globais da aplicação.
- `scss/`: Código-fonte dos estilos, organizados em:
    - `abstracts/`: Variáveis, *mixins*[2] e configurações reutilizáveis;
    - `components/`: Estilos aplicados a componentes específicos da interface. Onde está localizada a pasta *oscilloscope/* com os respetivos componentes individualizados em ficheiros.

4.4 Etapas de desenvolvimento

O desenvolvimento da aplicação foi um processo que passou por duas etapas fundamentais. A primeira etapa diz respeito ao design da solução, com a criação da GUI do osciloscópio e das suas funcionalidades. Esta etapa foi de extrema importância, de maneira a conseguir criar um solução que, de forma visualmente apelativa, torna-se o uso do osciloscópio numa mecânica simples e menos demorada de aprender, face a osciloscópios reais, mas não comprometendo as funcionalidades que estes apresentam.

A segunda etapa passou por desenvolver a lógica de criação e renderização das formas de onda, pegando na lógica matemática de um sinal elétrico e convertendo-a numa lógica computacional.

A fase de implementação deu-se após o design da solução, sendo que a partir daí as integração de novas funcionalidades foram sendo implementadas em paralelo com a sua idealização e estudo, de forma a melhor gerir o tempo disponível para a realização do trabalho.

4.4.1 Design da solução (UI/UX)

A ideia de criar uma interface para o osciloscópio intuitiva e amigável, em vez de apenas imitar um osciloscópio real, pode parecer, à partida, uma tarefa não muito difícil de realizar. Mas a verdade é que, para um aluno ou profissional da área de engenharia eletrotécnica, as referências que se tem de interfaces de osciloscópio, fora algumas exceções, são muito idênticas. Portanto, conseguir visualizar e criar algo que fuga à realidade que estamos frequentemente habituados pode ser desafiador.

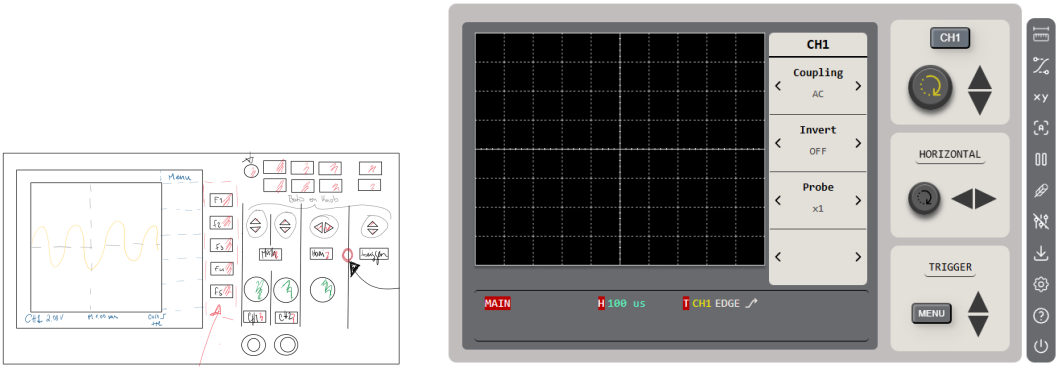
[2]Blocos de código que permitem agrupar estilos reutilizáveis.

Dito isto, foi inicialmente realizado um estudo prévio sobre interfaces de alguns osciloscópios reais e virtuais, e de seguida, elaborado via *OneNote* um *wireframe* inicial da solução, de forma a apoiar na visualização da interface final pretendida.

Com base nesse esboço, passou-se para a realização via *PenPot*, e posteriormente *Figma*, do *Mockup*[3] final. Esta fase, ao contrário do *wireframe*, já exigiu um preocupação maior ao detalhe.

Num primeiro momento, deu-se especial atenção ao *layout* dos elementos da interface e à forma como determinados componentes do osciloscópio real poderiam ser adaptados para uma versão mais moderna e intuitiva para um utilizador inexperiente, tanto ao nível da sua disposição pela interface quanto ao nível da sua aparência. Posteriormente, teve-se em atenção a coerência visual da solução, de forma a que se integrasse corretamente no estilo e ambiente da *U=RIsolve Academy*.

A Figura 4.2 apresenta a evolução entre o *wireframe* incial e a interface final desenvolvida.



(a) Wireframe inicial da solução (b) Interface final do osciloscópio

Figura 4.2: Evolução entre Wireframe inicial (a) e Interface final (b)

Uma vez terminada a interface, deu-se o processo de definição das intenções e funcionalidades a implementar. Para tal, foi igualmente dado uso à ferramenta *Figma*, permitindo a realização de prototipagem simples das interações, que serviu de guia para a implementação concreta. Esta prototipagem possibilitou simular o comportamento dos elementos da interface, como botões e *knobs*, bem como definir o fluxo de interações ("pressiono X – acontece Y"). Através desta abordagem, foi possível definir a lógica das funcionalidades, avaliar a usabilidade de certas interações e identificar ajustes necessários antes do desenvolvimento do código.

Foram então definidas e implementas as seguintes funcionalidades:

- Funcionalidades dedicadas ao editor de circuitos:
  - Comportamento *off-canvas* de osciloscópio;

[3]Protótipo gráfico, realista e detalhado, que serve de base à implementação de um projeto.

  - Adicionar e remover canais;
  - Mostrar/esconder interface de controlo;
  - Remover osciloscópio.

- Funcionalidades dedicadas ao osciloscópio:
  - Canais (acoplamente, inversão, atenuação);
  - Escalas e *off-sets*;
  - *Auto set*;
  - *Run/Stop*;
  - *Trigger Edge*;
  - Operações matemáticas;
  - Medições;
  - Cursores;
  - Modo XY;
  - Opção de gravação/*download*.

Foi igualmente considerada a forma como algumas destas funcionalidades poderiam ser utilizadas, de modo a torná-las mais práticas e intuitivas, mas sem fugirem demasiado do que é comum num equipamento laboratorial. Uma vez que o objetivo aqui é simplificar o processo de aprendizagem inicial do osciloscópio, e não tornar este simulador num instrumento diferente do físico.

Posto isto, foi necessário encontrar um equilíbrio nesta dualidade, onde a inovação e simplificação do osciloscópio, tornando-o mais acessível e amigável, não comprometesse a sua identidade.

4.4.2 Formas de onda

A criação e visualização das formas de onda foi igualmente uma etapa fundamental do desenvolvimento, de maneira a conseguir representar fielmente a simulação dos sinais elétricos.

Como já foi mencionado na Secção 2.2, num osciloscópio a representação do sinal é feita a duas dimensões: o eixo dos x corresponde ao tempo (t) e o eixo dos y à amplitude do sinal (A). Neste simulador, as formas de onda, não são geradas de forma continua, mas sim como um conjunto discreto de pontos:

$$dataset = \{(t_i, A_i)\}_{i=1}^{N} \tag{4.1}$$

O número de pontos (amostras) N varia dinamicamente em função da frequência do sinal e da escala temporal selecionada, garantindo um mínimo de 20 amostras

por ciclo, de forma a assegurar uma representação suave e evitando degraus entre amostras, característico de taxas de amostragem baixas.

De acordo com as escalas selecionadas, é então feito o cálculo de cada amostra a partir da função correspondente ao sinal e dos seus parâmetros previamente definidos, nomeadamente amplitude, frequência (f) e fase (ϕ). No caso de uma sinal sinusoidal:

$$A_i = A \cdot \sin(\omega \cdot t_i + \phi) \tag{4.2}$$

onde $\omega = 2\pi f$.

Uma vez obtido o *dataset*, o processo de amostragem deste simulador assemelha-se ao realizado num equipamento de amostragem real, sendo a sua renderização é feita com recurso à biblioteca D3.js [1].

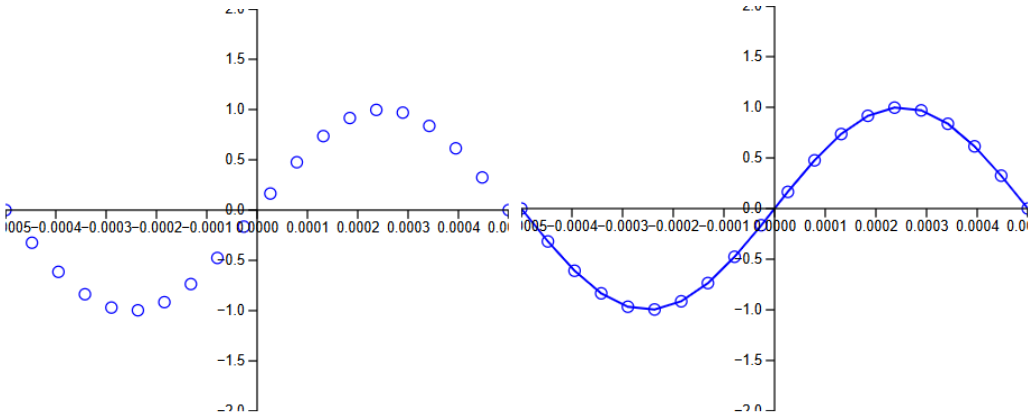
Para tal, é necessário definir os eixos e respetivos domínios, de maneira a mapear os valores temporais e de amplitude para coordenadas no sistema gráfico. Em seguida, uma linha é construída com interpolação linear entre os pontos do *dataset*, resultando num traçado SVG que representa a forma de onda respetiva a cada canal. Este processo pode ser ilustrado no seguinte excerto simplificado de código:

```
1  // Dataset
2  let datasetToDraw = datasets[channel];
3
4  // Cria os eixos
5  const x = d3.scaleLinear().range([0, width]);
6  const y = d3.scaleLinear().range([height, 0]);
7
8  // Define o dominio
9  x.domain([-horizontalScale, horizontalScale]);
10 y.domain([-verticalScale, verticalScale]);
11
12 // Cria a Linha
13 const line = d3.line()
14   .x((d) => x(d.time))
15   .y((d) => y(d.amplitude));
16
17 // Path da linha para o svg
18 svg.append("path")
19   .datum(datasetToDraw)
20   .attr("d", line);
```

Listagem 4.2: Desenho da forma de onda com D3.js

Note-se que o domínio está defendido como [-escala , +escala], de maneira à origem (0,0) do gráfico ser no centro do ecrã do osciloscópio.

Este processo é exemplificado na Figura 4.3. Cada eixo definido possui o seu domínio dividido nas respetivas escalas (neste caso, 500 mV na escala vertical e 100 us na escala horizontal). A partir daí, os pontos discretos são representados (Figura 4.3a) e posteriormente interpolados (Figura 4.3b).



(a) Representação das amostras do *dataset* (b) Interpolação dos pontos

Figura 4.3: Processo de interpolação do D3.js

Uma vez retirados os eixos e pontos discretos, os gráficos estão prontos para serem integrados no *grid* do ecrã do osciloscópio, caracterizado por 8 divisões verticais e 10 horizontais. O resultado final ficou pode ser observado na Figura 2.1, bem como nas demais figuras do capítulo seguinte.

4.4.3 Abordagem de implementação

Após o *design* da solução, iniciou-se o processo de implementação da interface do simulador numa versão inicialmente não funcional. Apenas quando desenvolvida a lógica de criação e representação das formas de onda é que foi possível avançar realmente para uma lógica de integração interna do *software*, isto é, uma fase de "costura" entre a interface estática e a lógica funcional, garantindo que cada interação do utilizador tivesse uma reflexo direto na forma de onda apresentada no ecrã do osciloscópio.

Para tal, cada elemento da interface tem um propósito especifico. Alguns elementos estão associados apenas à usabilidade do simulador no editor de circuitos, oferecendo flexibilidade e acessibilidade ao instrumento (por exemplo, permitir o seu movimento pela página). Contudo, o que realmente interessa, são os elementos que influenciam diretamente a representação dos sinais.

Um *dataset* é criado sempre que existe uma interação com algum destes elementos.

Ou seja, os *datasets* são gerados com base nos parâmetros de entrada fornecidos ao osciloscópio, que caracterizam o sinal. Para além disso, eles são transformados de acordo com as configurações que o utilizador fizer ao osciloscópio, sendo que,

cada alteração de configuração, resulta num novo *dataset*. Assim, se num minuto o utilizador interagir cem vezes com os elemento responsáveis pelas configurações do sinal, cem *datasets* serão criados e eventualmente transformados, garantindo que estes estão constantemente atualizados e funcionais para as restantes operações do osciloscópio.

O sistema está organizado em *flags* e objetos de estado, quer globais ao osciloscópio quer individuais a cada canal, que são igualmente atualizados a cada interação:

- Parâmetros do sinal:
  - De input: amplitude, frequência, fase, componente DC;
  - De configuração: escala vertical, *offset* de posição, tipo de acoplamento, inversão, atenuação.
- Configurações da interface: escala horizontal, *trigger*;
- Flags de execução: canal ativo/inativo, trigger encontrado ou não, último canal desenhado, etc;
- Medidas.

Este modelo permite que a mesma função gere *datasets* individuais e reutilizáveis para cada canal, garantindo que todas as outras funções do osciloscópio operam sobre o mesmo repositório central de informação, reduzindo inconsistências. A Listagem 4.3 ilustra a parametrização de um canal antes da geração do *dataset*.

```
1  case UIElements.ch1.ch1Button:
2    channel = "CH1";
3    containerId = "#svg-waveformCh1";
4    color = "yellow";
5    verticalScale = parseFloat(verticalScaleList[scaleIndex.CH1.
         scale]) * 4;
6    verticalOffset = verticalOffsets.CH1 * verticalScale;
7    channelOptions = state.CH1;
8    marcadorIMG = "/img/marcadorCH1.svg";
9    break;
```

Listagem 4.3: Caracterização do Canal 1 antes da geração do dataset

No caso deste excerto de código, para o *CH1*, os valores de entrada e as configurações de estado são armazenados em objetos como *state.CH1*, *verticalOffsets.CH1*, *verticalScaleList*. No panorama geral, todas as variáveis do sistema estão igualmente distribuídas em objetos específicos, bem como as medidas calculadas e atualizadas para cada forma de onda.

Nesse seguimento, imaginemos, no caso do acoplamento dos canais, que seja *GND* a opção selecionada, o *dataset* é atualizado de modo a forçar todos os pontos de amplitude a zero, refletindo-se naturalmente nas medidas. Ou ainda a inversão de onda, o *dataset* é gerado e antes de "sair" para representação, inverte a sua amplitude. Este mecanismo aplica-se a todas as funcionalidades de controlo dos sinais, como as posições verticais e horizontais, a escala temporal ou de amplitude, o processo é o mesmo e individualizado para cada canal. À exceção do *trigger*.

Numa abordagem idêntica ao do osciloscópio real, o *trigger* neste simulador é o "motor de representação" do *dataset* final, mas não propriamente com a mesma utilidade e funcionamento.

Ao contrário de um osciloscópio real, que é disfuncional sem um indicador para inicializar a amostragem e estabilizar a representação do sinal, na visão deste simulador, em que o *dataset* é baseado numa expressão matemática e a sua representação é feita à semelhança de uma calculadora gráfica, o *trigger* assume um caráter pedagógico, e a sua funcionalidade aqui surge no interesse de o utilizador perceber como este se comportará no instrumento físico.

Dito isto, o *trigger* aqui define a origem da representação, com base no nível de "disparo" (*trigger level*) do canal selecionado. Só depois, serão representados os *datasets* finais em função dessa origem. No entanto, contrariamente ao osciloscópio real, que utiliza um *buffer* para apresentar amostras anteriores ao *trigger*, este simulador, dado o seu modelo de representação, permite amostrar intervalos de tempo negativos, o que constitui uma flexibilidade adicional das abordagens virtuais, que naturalmente não são possíveis na "vida real".

A deteção do instante de "disparo" acontece em duas fases. Primeiro, é gerado um *dataset* inicial sem considerar o nível de *trigger*, para servir como referência. Depois, identifica-se o cruzamento do nível de *trigger* (T) nas amostras desse *dataset*, de acordo com a inclinação definida:

1. Rising: $A_{i-1} < T \wedge A_i \geq T$
2. Falling: $A_{i-1} > T \wedge A_i \leq T$

Com base nas amostras que precedem e sucedem o cruzamento, calcula-se por interpolação linear o instante exato em que o "disparo" acontece:

$$t_{\text{trigger}} = t_{i-1} + \frac{T - A_{i-1}}{A_i - A_{i-1}} (t_i - t_{i-1}) \tag{4.3}$$

Este valor de t_{trigger} passa então a substituir *off-set* horizontal na criação do *dataset* final, para alinhar o gráfico com cruzamento do nível de *trigger*, tal como é possível observar na Figura 5.6 do capítulo seguinte.

Toda esta lógica permitiu, posteriormente, implementar funcionalidades adicionais sobre esses *datasets*, como as operações matemáticas, o modo XY e os cursores.

Considerando dois canais,

$$\text{CH1} = \{(t_i, A_i\}_{i=1}^{N} \quad \text{e} \quad \text{CH2} = \{(t_j, A_j\}_{j=1}^{N}. \tag{4.4}$$

As operações matemáticas forma implementadas ponto a ponto, onde a amplitude de cada amostra é obtida por:

$$A_k = \begin{cases} A_i \pm A_j, \\ A_j \pm A_i. \end{cases} \tag{4.5}$$

e o resultado da operação matemática (onda *math*) é um novo *dataset* a ser representado:

$$mathDataset = \{(t_k, A_k\}_{k=1}^{N} \tag{4.6}$$

sendo $t_k = t_i = t_j$, já que ambos os canais partilham a mesma escala temporal.

No modo *XY*, por substituição do eixo temporal, o novo *dataset* resulta da amplitude de um canal em função da do outro:

$$xyDataset = \{(x_m, y_m)\}_{m=1}^{N} \quad \text{com} \quad x_m = A_i \quad \text{e} \quad y_m = A_j. \tag{4.7}$$

Neste modo, as configurações temporais e de *trigger* deixam de estar funcionais.

Relativamente aos cursores, foi utilizada a funcionalidade *d3.brush* [43] da biblioteca D3.js, que permite a interação direta com o SVG dos sinais. Fazendo uma seleção no gráfico, e retornando as coordenadas do ecrã, após manipulação é possível chegar aos valores dos limites da seleção feita. Isto oferece um abordagem mais interativa e dinâmica face aos cursores tradicionais dos osciloscópios físicos.

A Figura 4.4 apresenta o diagrama de blocos que resume o fluxo de funcionamento do simulador implementado, destacando a lógica interna (a cinzento), a lógica de criação e renderização dos sinais (a laranja) e os *inputs* (a verde).

Concluindo, qualquer sinal elétrico pode, em princípio, ser descrito matematicamente através do seu comportamento no tempo. Esta premissa foi central para o desenvolvimento do simulador, cuja arquitetura de software foi concebida de modo a permitir a representação de diferentes tipos de sinais, desde que os parâmetros necessários sejam fornecidos.

A versatilidade do algoritmo de cálculo da amplitude implementado possibilita a geração não apenas de sinais periódicos simples, mas também de formas de onda mais complexas e não periódicas. Na fase inicial de estudo de circuitos elétricos, o interesse recai naturalmente sobre sinais básicos, como:

- Sinais contínuos em corrente contínua (DC);
- Sinais alternados (AC) sinusoidais, quadrados ou triangulares periódicos.

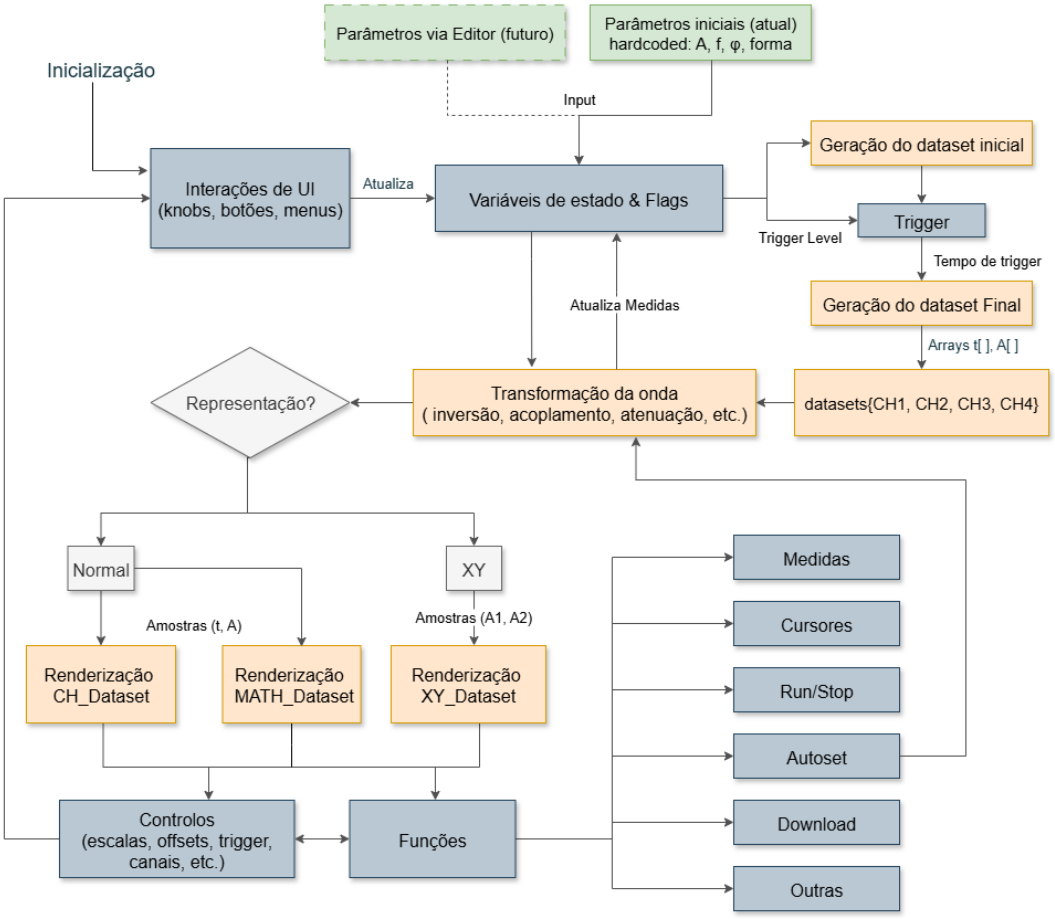
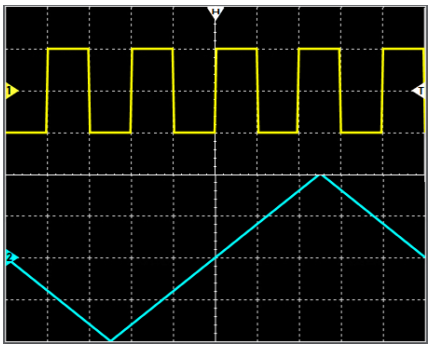


Figura 4.4: Lógica de funcionamento do simulador

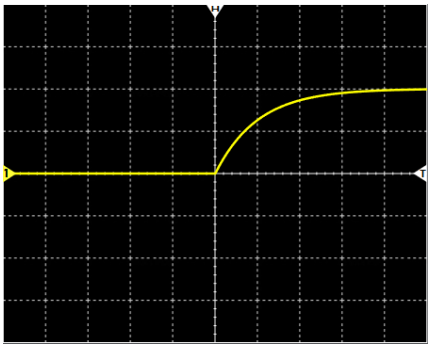
Contudo, a mesma abordagem pode ser expandida para sinais mais avançados e de grande relevância no estudo de sistemas dinâmicos. Um exemplo clássico é o comportamento exponencial associado ao carregamento e descarregamento de condensadores, ou a retificação de um sinal, enfim, uma infinidade de possibilidades. Para além disso, a flexibilidade do método permite considerar outras funções que fogem à temática mas que demonstram a sua capacidade, como por exemplo funções polinomiais.

Assim, ainda que o presente trabalho se tenha centrado em formas de onda mais usuais no ensino introdutório, a base desenvolvida encontra-se preparada para suportar uma evolução futura e potencialmente útil em analises mais avançadas.

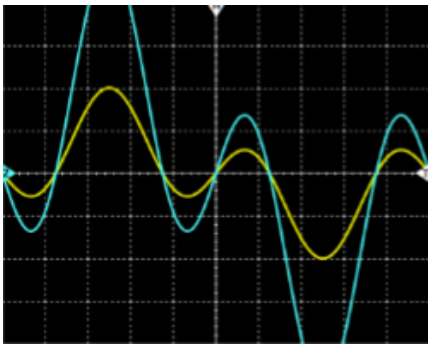
Nas figuras seguinte, apresentam-se alguns exemplos de sinais gerados no simulador de osciloscópio, ilustrando a sua versatilidade: 4.5a representa uma onda quadrada e uma triangular; 4.5b representa sinal exponencial crescente, simulando o carregamento de um condensador; 4.5c representa dois sinais complexos; 4.5d representa uma sinal modulado;



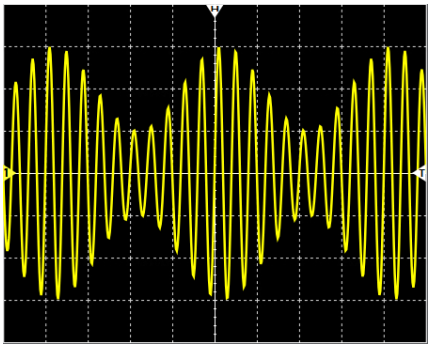
(a) Onda quadrada e triangular



(b) Sinal exponencial crescente



(c) Dois sinais complexos



(d) Sinal modulado

Figura 4.5: Exemplos de sinais gerados no simulador de osciloscópio

4.5 Desafios, soluções e otimização

Especialmente ao nível da experiência de navegação do utilizador, surgiram alguns desafios ao longo do processo que exigiram soluções eficazes o mais rapidamente

possível de maneira a otimizar a aplicação sem não atrasar o desenvolvimento.

Inicialmente, o numero de amostras era constante e igual a 250 por ciclo. Para frequências ou escalas temporais altas, a renderização era de milhares de pontos, o que resultava na sobrecarga do navegador e, consequentemente, uma má experiência para o utilizador. De maneira a solucionar isso, as amostras passaram a ser geradas de forma dinâmica e adaptando-se à forma de onda representada.

Foi igualmente necessário limitar o número de períodos amostrados. Sem isso, as frequências demasiado altas também provocavam a sobrecarga do navegador. A solução adotada foi considerar a maior frequência entre os canais como referência para limitar os restantes. Dessa maneira, o máximo de períodos possíveis a representar são 100, sendo que visualmente para o utilizador, a diferença entre 100, 200 ou 1000 períodos é mínima. No entanto, esta abordagem tem uma limitação, no que respeita a diferenças muito grandes de frequência entre canais. Os canais de frequência muito baixa terão menos flexibilidade, dado que, embora o rodapé informativo do ecrã continue a atualizar o tempo/div, o gráfico está condicionado ao limite estabelecido, e essa discrepância pode ser percetível.

Outra otimização relevante foi o facto de usar interpolação para detetar o cruzamento do nível de *trigger*. Sem ela, o *trigger* ficava limitado à resolução dos *datasets*, produzindo erros visíveis no alinhamento da onda e até más caputras do *trigger* gerando *wavelost* nos momentos errados.

Capítulo 5

Funcionalidades e Utilização

Uma vez descrita a lógica de funcionamento do simulador, este capítulo dedica-se à explicação das suas funcionalidades bem como do seu modo de utilização. Para além disso, no propósito deste capítulo, foi realizada uma caracterização detalhada da interface final que pode ser encontrada no Anexo A.2.

5.1 Interface do simulador

Na Figura 5.1 podemos observar a interface final do simulador. Através dela, será possível fazer o controlo do osciloscópio e a visualização e análise das formas de onda.

Esta interface integra alguns elementos e funções que se assemelham a um equipamento real, de forma a preservar a sua identidade, mas possui também características e funcionalidades únicas deste simulador, que lhe conferem uma identidade própria e intuitividade na utilização.

De modo geral, pode ser divida em dois módulos principais:

- **Módulo de Controlo:** Reúne os elementos destinados ao controlo e configuração do osciloscópio, incluindo os controlos habituais de um equipamento real direcionados às formas de onda, bem como uma barra lateral que funciona como um menu principal, centralizando as funcionalidades extra do simulador.
- **Módulo de Visualização:** Representa o ecrã onde os sinais serão amostrados, bem como o rodapé informativo que indica o estado atual do osciloscópio.

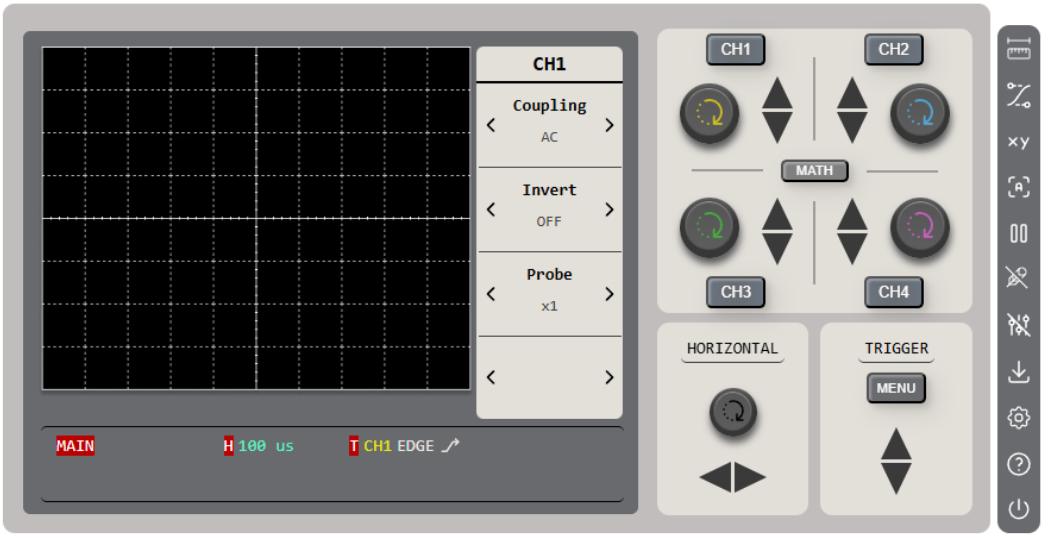


Figura 5.1: Interface final do simulador

No sentido de melhor localizar e caracterizar cada elemento implementado, um mapa detalhado da interface é apresentado no Anexo A.

5.2 Funcionalidades

5.2.1 Canais

Os canais podem ser ligados ou desligados clicando no botão indicativo de cada um. Uma vez ativos, estes são destacados pelas suas respetivas cores, tal como é ilustrado na Figura 5.2.

Figura 5.2: Botão de canal ligado e desligado

À semelhança de um instrumento real, quando um canal se encontra ativo a sua forma de onda é representada, o estado do canal fica visível no rodapé informativo do ecrã, assim como o seu menu dinâmica, que permite a configuração do canal (Figura 5.3).

Nessa ordem de ideias, um canal só poderá ser desligado quando o seu menu dinâmico estiver visível. Assim, serão precisos no máximo dois passos para desligar o canal. Por exemplo, para desligar o CH1 quando o menu visível corresponde ao CH2, uma interação será necessária para aceder o menu do CH1 e, de seguida, outra

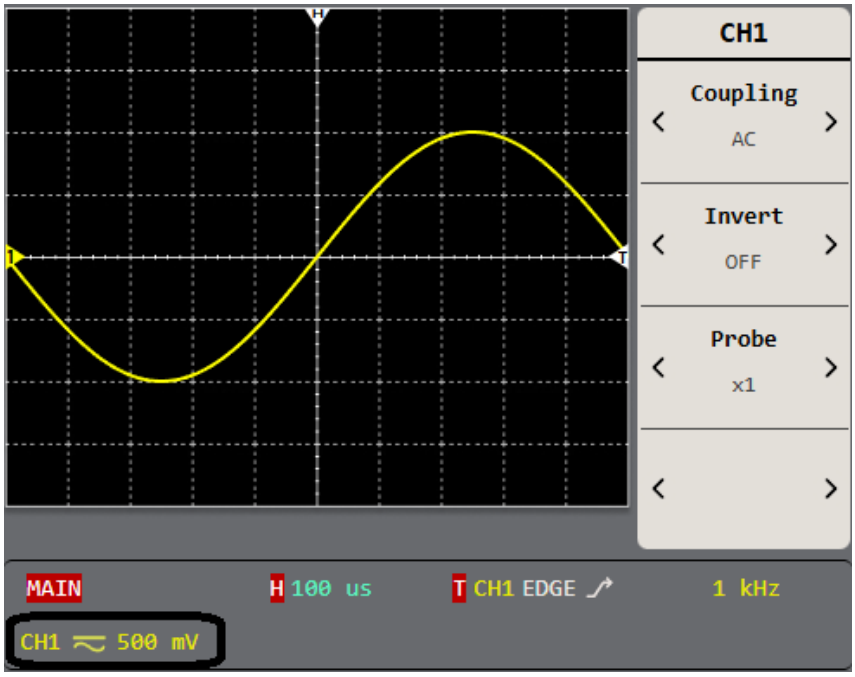


Figura 5.3: Representação do CH1 ativo

para o desligar. Esta lógica de funcionamento ocorre também em instrumentos reais e garante que se consiga alternar entre menus de diferentes canais sem que estes desliguem.

Quanto à visualização das onda, este processo acontece por sobreposição, quando falamos em mais do que um canal ativo ao mesmo tempo. Ou seja, o ultimo canal interagido é sempre representado em primeiro plano.

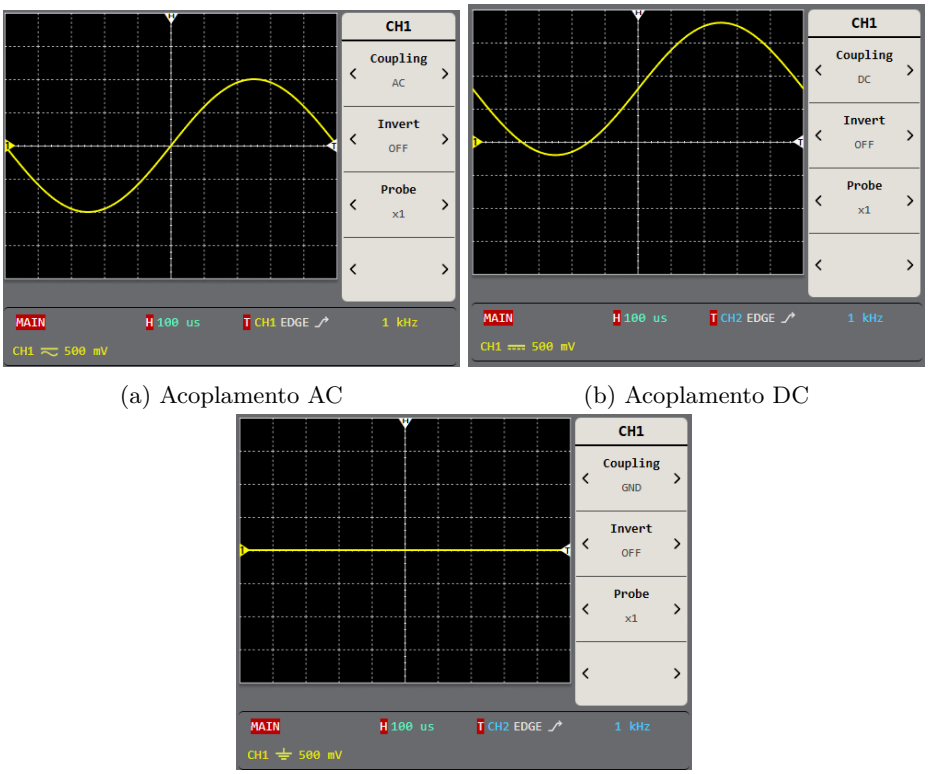
Acoplamento, inversão e atenuação

No osciloscópio, Os canais podem ser configurados quanto ao tipo de acoplamento:

- **DC:** mostra a componente contínua e alternada do sinal;
- **AC:** mostra apenas a componente alternada;
- **GND:** apresenta a linha de referência de 0 V (sem sinal).

A Figura 5.4 ilustram estas três opções, para o caso de uma onda sinusoidal de 1 V de amplitude e componente contínua de 0.8 V.

A inversão e a atenuação também estão implementadas e são autoexplicativas, sendo que invertem e atenuam (x1, x10, x100) a amplitude do sinal, respetivamente.



(a) Acoplamento AC (b) Acoplamento DC

(c) Acoplamento GND

Figura 5.4: Tipos de acoplamento

5.2.2 Escalas e off-sets

As escalas e *off-sets* de posição são alterados através do uso de dois elementos centrais da interface: os *knobs* e os botões triangulares, respetivamente. (à exceção dos botões triangulares do *trigger*, que não alteram posição mais sim o nível de *trigger*).

A manipulação dos *knobs* é feito de forma circular: o movimento no sentido horário aumenta a escala e no sentido anti-horário reduz a escala. Este mapeamento é feito a partir do ângulo que o *knob* se encontra face ao seu ângulo inicial de 0º. Quanto aos botões triangulares, o sentido da deslocação é intuitivamente percetível pela sua orientação. De salientar, que os *off-sets* de posição estão limitados à área do gráfico do gráfico.

Vertical

As escalas verticais, que representam os Volts/Divisão, variam entre 2 mV e 5 V por divisão, em 11 valores possíveis. Na Figura 5.5a pode observar-se como o ângulo do *knob* faz o mapeamento dos valores das escalas em quase 3 voltas completas (0º–990º), em passos de 99º entre escalas.

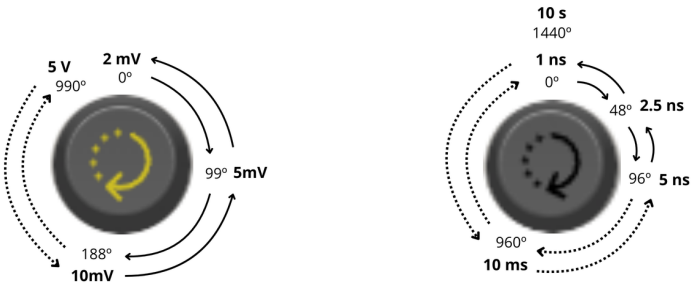
Quanto aos botões de posição, permitem cinco passos por divisão, ou seja, considerando todo o eixo vertical (oito divisões) é possível colocar a forma de onda em

40 posições diferentes.

Horizontal

As escalas horizontais representam o Tempo/Div e variam entre 1 *ns* até 10 *s* por divisão, no entanto, em 31 valores diferentes. Para abranger todas as possibilidades, sem exagerar no número de voltas ao *knob*, o mapeamento foi distribuído em quatro voltas completas (0º–1440º), resultando em passos de 48º entre escalas, conforme ilustrado na Figura 5.5b.

Relativamente à posição horizontal, esta permitem quatro passos por divisão, o que resulta igualmente em 40 posições horizontais diferentes face às dez divisões do eixo temporal.



(a) Knob vertical (b) Knob horizontal

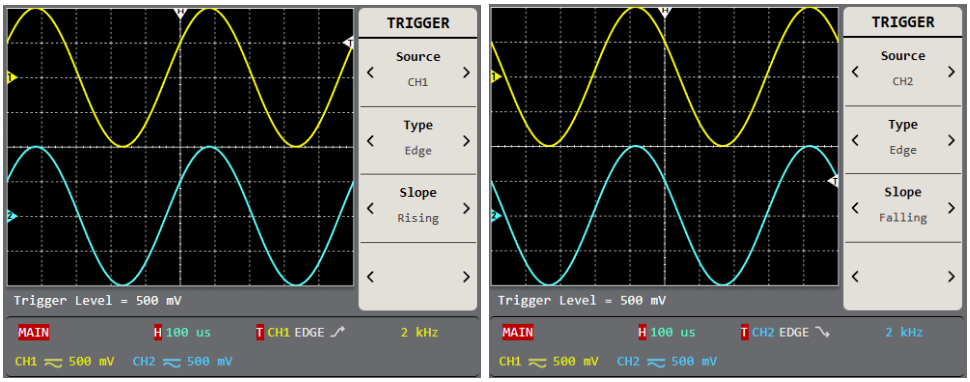
Figura 5.5: Comportamentos dos knobs

5.2.3 Trigger

O *trigger* implementado é do tipo *edge*, já explicado o seu funcionamento na Secção 4.4.3. O canal que é fonte de *trigger* pode ser selecionado de entre os quatro canais disponíveis, sendo possível definir a inclinação pretendida para a captura: *rising* ou *falling*. Os restantes canais que não são fontes de *trigger* são desenhados de acordo com o instante de *trigger* encontrado. Quanto ao seu nível pode ser ajustada através dos respetivos botões triangulares.

Na Figura 5.6 observa-se a captação do *trigger* com um nível de 500 *mv*, exemplificando uma inclinação *rising* com fonte CH1 e uma inclinação *falling* com fonte CH2. Neste exemplo, os dois canais apresentam sinais iguais, permitindo evidenciar a sincronização entre eles.

Caso nenhum cruzamento seja encontrado as formas de onda entram em wavelost, à semelhança de um equipamento real.



(a) Comportamento trigger rising (b) Comportamento trigger falling

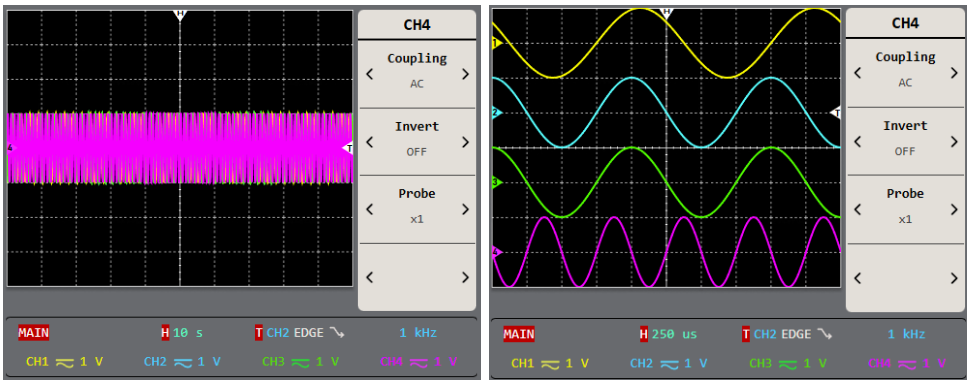
Figura 5.6: Exemplo de variações de trigger

5.2.4 Auto set

Esta funcionalidade permite que os sinais ativos sejam ajustados de forma automática para o arranjo que melhor permitir a visualização das ondas, escusando o utilizador de fazer esses arranjos manualmente.

No exemplo ilustrado na Figura 5.12, com quatro canais ativos, estes serão posicionados devidamente espaçados, as escalas verticais serão ajustadas de maneira ao canais não se sobreporem e a escala horizontal será atualizada de maneira a representar pelo menos dois períodos da onda de menor frequência.

Consoante o número de canais ativos e as características dos sinais os arranjos serão diferentes.



(a) Representação dos sinais não ajustados (b) Representação dos sinais ajustados por autoset

Figura 5.7: Processo de arranjo dos sinais através do autoset

5.2.5 Operações matemáticas

As operações matemáticas resultam num forma de onda *math*, que pode ser ativada de forma análogo aos canais, através do botão indicativo "MATH" localizado ao centro dos canais. As operações disponíveis são de soma e subtração entre os canais ativos. A escala vertical da onda *math*, pode também ser escolhida de entre as escalas dos canais que integram a operação. No exemplo da Figura 5.8, o canal CH1 apresenta amplitude de 5 V e o canal CH2 amplitude de 1 V. O resultado exibido corresponde à subtração entre eles.

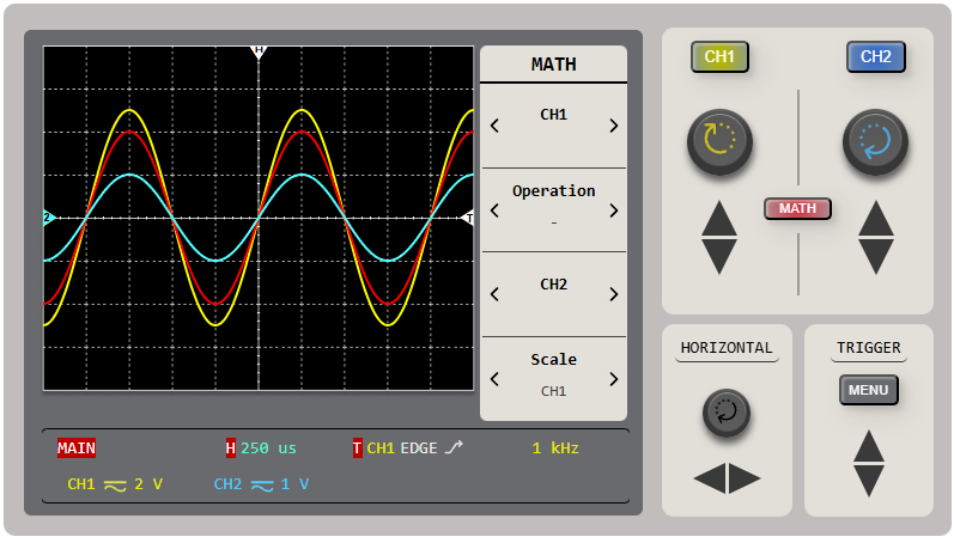


Figura 5.8: Representação da onda *math* resultado da subtração de dois canais

5.2.6 Medidas

As medições no osciloscópio, numa abordagem mais amigável e simplificada, são consultadas à semelhança da funcionalidade *Measure ALL* de um osciloscópio real. Sendo assim, um painel sobreposto o ecrã permite consultar todas as medidas dos canais ativos, nomeadamente:

- **Tensão:** Vpp, Vmáx, Vmín, Vavg, Vrms;
- **Tempo:** Frequência, Período.

No caso de mais de dois canais ativos, o painel de medições pode ser percorrido através do *scroll*, como ilustrado na Figura 5.9.

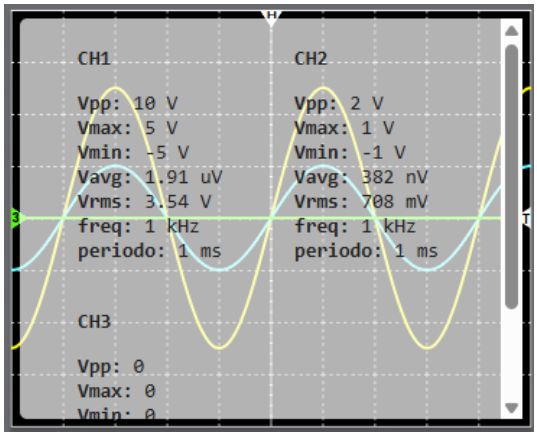


Figura 5.9: Painel de medidas sobre o ecrã do osciloscópio

5.2.7 Cursores

Os cursores oferecem uma nova abordagem face aos osciloscópio tradicionais. Ao invés de duas linhas horizontais e verticais, este modo de cursores permite a seleção de uma área retangular reajustável e móvel pelo ecrã do osciloscópio. Esta área resulta num *pop-up* com os respetivos intervalos verticais e horizontais, servindo como uma ferramenta para tirar desfasamentos, ganhos, medidas, etc.

Na Figura 5.10 observa-se a utilização dos cursores para determinar o valor de pico-a-pico e do período de um sinal.

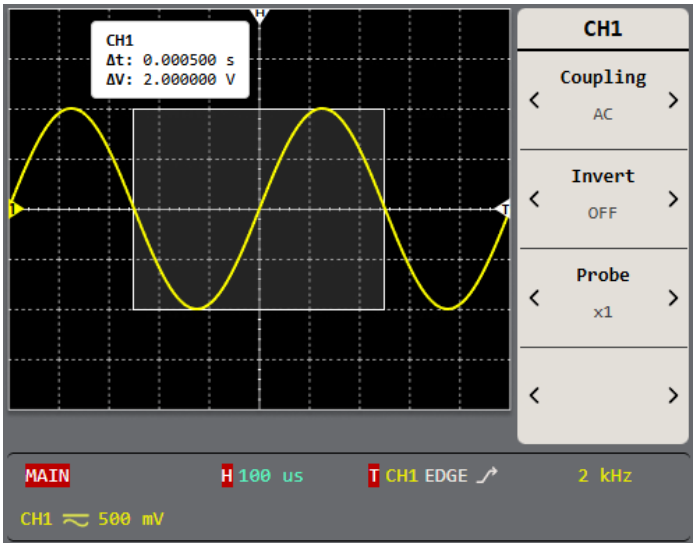


Figura 5.10: Utilização dos cursores no osciloscópio

De notar, a referência ao canal no topo do pop-up, uma vez que os cursores são sempre baseados na escala vertical do último canal interagido.

5.2.8 Modo XY

No modo XY, o eixo temporal deixa de ter efeito na representação. Ao contrário do modo de funcionamento normal, em que a forma de onda é representada como $A(t)$ — amplitude em função do tempo —, modo XY a representação passa a ser $A2(A1)$ — amplitude do CH2 em função da amplitude do CH1. No simulador, este modo opera exclusivamente sobre os canais 1 e 2.

Este modo revela-se especialmente útil para visualização das chamadas figuras de Lissajous [44], quando aplicadas duas sinusoides aos eixos X e Y.

Na Figura 5.11 é apresentado um exemplo de representação XY, com um desfasamento de 30º entre os sinais de entrada.

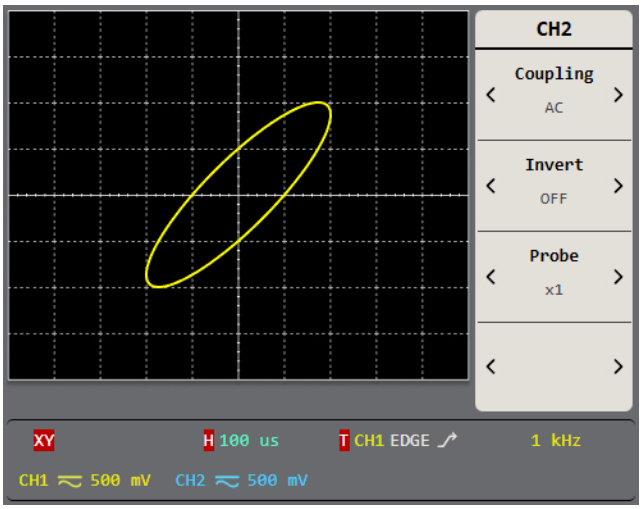


Figura 5.11: Representação do Modo XY

De notar, que duramente o funcionamento do modo XY, funções temporais e de *trigger* do osciloscópio não estão funcionais, até para garantir que não ficam desconfiguradas quando o utilizador regressa ao modo normal.

Os cursores mantêm-se disponíveis neste modo, e em formato de exercício, consegue-se verificar se o ângulo de desfasamento é realmente de 30º.

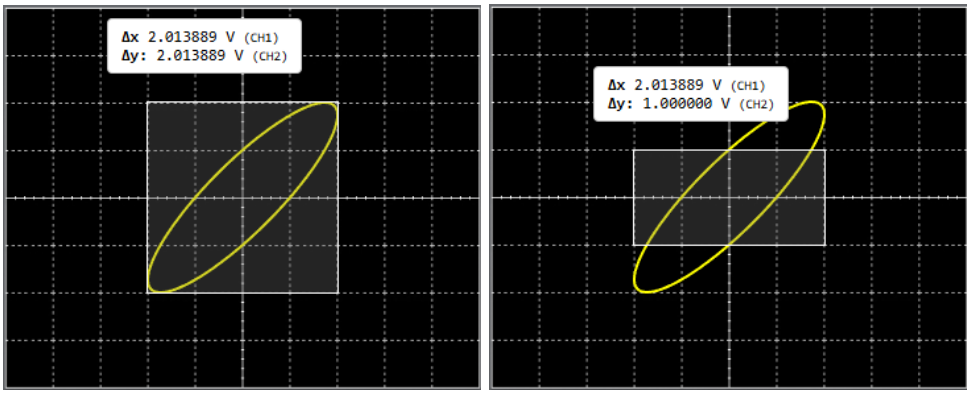
Pelo método elíptico, é possível determinar o ângulo de desfasamento φ através da seguinte expressão:

$$\sin \varphi = \frac{B}{A} \tag{5.1}$$

onde A correponde ao Δy da Figura 5.13a e B corresponde ao Δy Figura 5.13b.

Por substituição, obtém-se:

$$\varphi = \arcsin(\frac{1}{2.014}) = 29.77^\circ \approx 30^\circ \tag{5.2}$$



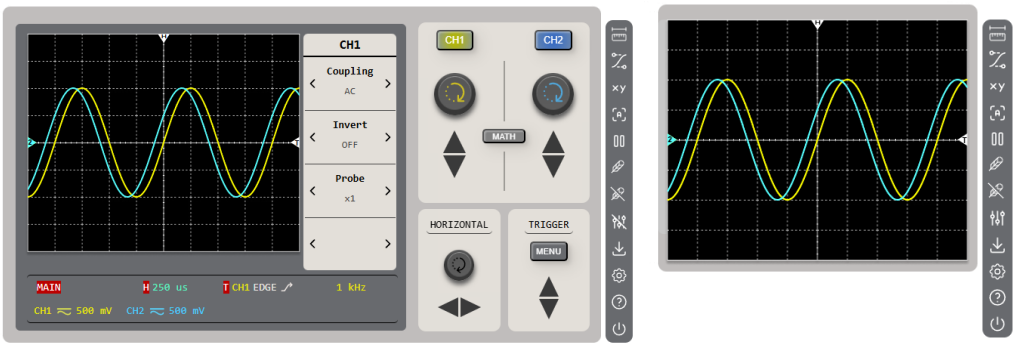
(a) (b)

Figura 5.12: Método elíptico

5.2.9 Outras funcionalidades

Para além das funcionalidades gerais que regem o funcionamento do osciloscópio, foram ainda implementadas uma série de funcionalidades auxiliares que otimizam a sua utilização e melhoram a experiência do utilizador, nomeadamente:

- **Run/Stop:** Implementado unicamente com finalidade pedagógico, não possui funcionalidade prática face ao modo de amostragem utilizado neste simulador, servindo apenas como indicador visual.
- **Estado das posições e *trigger level*:** À semelhança dos equipamentos reais, foi implementado uma *flag* que indica, a cada interação, o estado da posição dos canais e do nível de *trigger*.
- **Iteração do menu dinâmico do ecrã:** Ao contrario de osciloscópios reais com sentidos de iteração unidirecional através dos botões F1, F2, etc., neste simulador é possível avançar e recuar nas opções do menu.
- **Adicionar e remover canais:** De forma dinâmica, é possível remover e adicionar canais à interface consoante as necessidade do utilizador, num mínimo de 1 canal e máximo de 4. Esta funcionalidade pode ser observada ao longo das imagens utilizadas neste capítulo.
- **Minimizar e maximizar osciloscópio:** Funcionalidade apresentada na Figura 5.13 e que se mostrará muito útil numa futura integração com editor, de forma a garantir que o osciloscópio não perturba o desenho e interação com os circuitos desenhados.
- **Modos de gravação:** Permite o download em extensão *.png*, seja apenas das forma de onda, seja do ecrã completo (onda + rodapé + menu) ou ainda do painel de medições.



(a) Interface maximizada (b) Interface minimizada

Figura 5.13: Maximização e minimização da interface

- **Fechar osciloscópio:** Função que permitirá fechar e abrir o osciloscópio assim que ele estiver integrado com o editor.

Capítulo 6

Conclusão

6.1 Avaliação global

Este projeto revelou-se desde cedo uma proposta interessante mas desafiadora, sobretudo pela necessidade de explorar abordagens e práticas ainda pouco familiares. Esse desconhecimento inicial traduziu-se em alguma incerteza, mas rapidamente foi sendo superado à medida que as etapas de desenvolvimento avançavam e as ideias passavam a resultados concretos.

Contudo, nem sempre o percurso decorreu como previsto. Um dos principais imprevistos surgiu quando se verificou a impossibilidade de integrar o osciloscópio no editor de circuitos por motivos externos, objetivo que constituía uma das motivações iniciais da aplicação. Esta limitação impossibilitou tanto a receção de dados provenientes do editor de circuitos como a integração do simulador na plataforma, inviabilizando a utilização desses dados como entrada para o osciloscópio, bem como a realização de testes e avaliações relacionados com a integração final. Isto obrigou a uma redefinição de rumo, direcionando o foco para o desenvolvimento autónomo do osciloscópio e para o aprofundamento das suas funcionalidades. Embora tenha representado uma frustração, dado o interesse em poder observar e disponibilizar o instrumento a cumprir o seu propósito fundamental, acabou por se transformar numa oportunidade para explorar em maior detalhe o funcionamento do instrumento e conferir-lhe maior consistência num modo independente.

A estruturação das etapas de desenvolvimento permitiu que todo o processo decorresse de forma gradual, sem grandes momentos de impasse ou paragens prolongadas. Nesse sentido, o tempo disponível foi plenamente aproveitado, revelando-se, ainda assim, curto face às inúmeras ideias que foram surgindo ao longo do caminho. A fase de planeamento e design da solução, orientada para uma perspetiva inovadora, foi especialmente dinâmica e sujeita a alterações constantes, resultantes do aparecimento de novas ideias que exigiram seleção criteriosa, dado que o tempo disponível não permitia a concretização de todas.

Quanto a fase de implementação, não decorreu de forma linear, mas sim em ciclos iterativos, em que cada avanço era testado, revisto e frequentemente reformulado. Nesta fase, surgiram também vários desafios, resultado quer da complexidade natural de um processo de desenvolvimento, quer do desconhecimento inicial das ferramentas e metodologias, o que exigiu capacidade de adaptação e ajuste de soluções.

Já numa fase final foram feitos os últimos ajustes, procurando eliminar inconsistências e melhorar a experiência global da aplicação. Esta etapa decorreu em paralelo com a redação do relatório, o que permitiu não apenas consolidar a solução desenvolvida, mas também refletir de forma crítica sobre o processo e identificar aspetos a aprofundar em trabalhos futuros.

Para além disso, este trabalho constituiu um marco importante na evolução do ponto de vista pessoal e académico. Exigiu uma atenção redobrada à gestão do tempo, à organização e aos métodos de trabalho, bem como à forma de lidar com imprevistos. Representou, acima de tudo, uma experiência de aprendizagem e crescimento e que seguramente contribuirá para o desenvolvimento e concretização de projetos futuros.

6.2 Limitações e bugs

Apesar da aplicação estar já funcional na como ferramenta autónoma, apresenta ainda algumas limitações que merecem atenção numa fase futura de otimização da usabilidade do simulador

Para escalas horizontais pequenas, as formas de onda entram erradamente em *wavelost* quando o nível do *trigger* ainda está posicionada corretamente para ajustar a amostragem. Ainda, pelo facto da escala horizontal ser limitada como mencionado na subsecção 4.5, os cursores podem geram incoerências face à informação de tempo/div do rodapé.

Os valores de *Vmáx* e *Vmin*, quando os picos da onda se encontram no centro do eixo X, podem gerar valores dispersos face à extensão do *dataset*.

Existem ainda outras questões de sincronização e gravação do estado das ondas, quando o canal fonte de trigger é alterado.

6.3 Trabalho Futuro

Quanto à evolução da aplicação, é evidente o caminho a seguir nas próximas etapas de trabalho, nomeadamente a integração com o editor de circuitos, uma vez que o seu propósito fundamental só será alcançado com essa integração.

6.3.1 Integração com o editor de circuitos

A arquitetura do simulador já foi concebida de forma a suportar as próximas etapas de integração com a plataforma *U=RIsolve Academy*, proporcionando uma base sólida sobre a qual versões futuras poderão evoluir.

Esta integração, e tendo como referência o esquema anteriormente apresentado na Figura 4.1 da arquitetura geral, passará por estabelecer a comunicação com o editor e, posteriormente, implementar um algoritmo de *parsing* para o dados recebidos. A titulo de exemplo, aquilo que pode ser visualmente esperado no editor de circuitos com o osciloscópio integrado é algo semelhante ao protótipo representado na Figura 6.1.

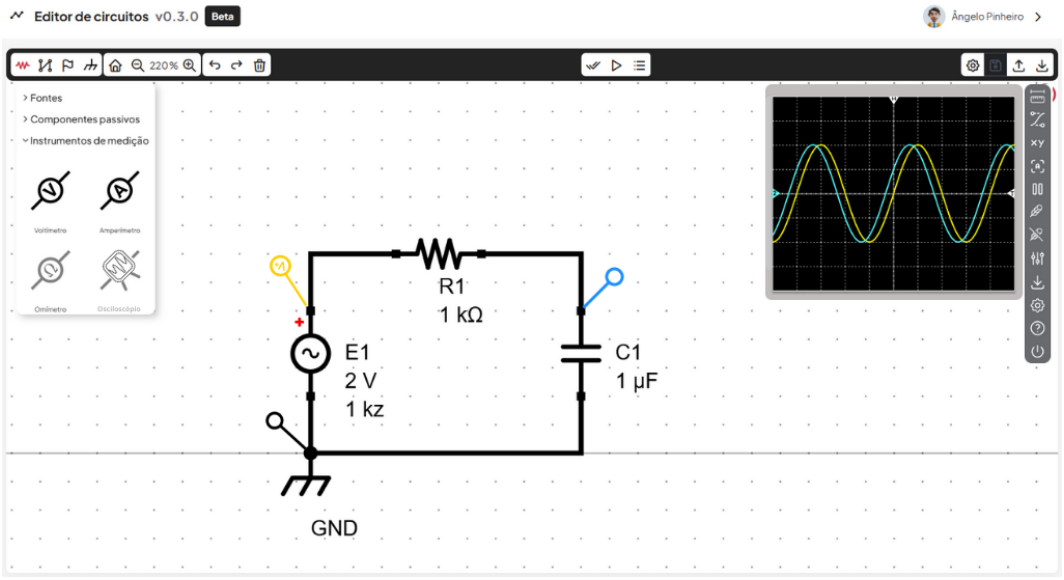


Figura 6.1: Prótotipo de Editor de ciruitos com osciloscópio integrado

Assim, do ponto de vista do resultado final esperado, o aluno não terá apenas a possibilidade de manusear virtualmente um osciloscópio, mas sim de usufruir de uma experiência laboratorial completa, através da alternância entre a criação e edição de circuitos elétricos e a análise do seu comportamento recorrendo ao simulador de osciloscópio implementado. Ou seja, esta integração, oferecerá um fluxo contínuo de trabalho entre - desenho - simulação - análise - de circuitos elétricos promovendo uma aprendizagem mais eficiente e abrangente.

6.3.2 Novas funcionalidades

Muito trabalho foi desenvolvido ao longo deste projeto, e ainda que os objetivos tenham sido alcançados com sucesso, há sempre espaço para melhoria, seja otimizando a aplicação ou a experiência do utilizador. Para além da integração com o editor é ainda possível a implementação de novas funcionalidades internas ao osciloscópio, de entre elas:

- **Centralizar os menus de controlo:** Evitar a coexistência do menu dinâmico do ecrã e do menu lateral, criando uma zona única de controlo e eventualmente ter menus individuais para cada canal;
- **Menu de definições:** Acrescentar as funcionalidade para permitir ao utilizador configurar o osciloscópio ao seu gosto ou necessidade;
- **Menu de ajuda:** Acrescentar também o menu que permite ao utilizador consultar as funcionalidades e modo de as usar;
- ***Scroll* nos botões de posição:** À semelhança de um *knob* real permitir o posicionamento das ondas de forma mais dinâmica e não passo a passo;
- **Novos tipos de análise:** Adicionar outros tipos de analise, como espectro em frequência (FFT);
- **Novas medidas:** Adicionar medições mais avançadas como *duty cycle*, *overshoot*, *rise/fall time*, etc;
- **Guarda o estado:** Poder guardar o estado do osciloscópio, em conjunto com o editor, para retomar em sessões futuras sem perda de configurações;
- **Explorar simulação não ideal:** Poderia ainda ser interessante, e dado o vertente pedagógico desta aplicação, o interesse de simular os aspetos reais de não idealidade dos sinais, com algoritmos que simulasses ruído ou imperfeições;
- **Correção das limitações**.

Assim sendo, estas propostas de evolução demonstram que, apesar da estrutura e funcionamento já alcançados e da possibilidade de utilizar, à data, o simulador como ferramenta autónoma, este continua a apresentar um enorme potencial de crescimento. Podendo, quem sabe um dia, vir a tornar-se uma solução comum para o ensino e aprendizagem de circuitos elétricos e instrumentos de medição.

Referências

[1] D3 by Observable, "What is d3? | d3 by observable." Available at `https://d3js.org/what-is-d3`. (Último acesso a 06/09/2025). [Citado nas páginas 4, 20 e 40]

[2] M. Alves, "osciloscopio-abc-mario-alves," —, 2007. [Citado nas páginas 9 e 11]

[3] GW-Instek, "Gos-630fc-product-gw instek." [Citado na página 11]

[4] Siglent, "Osciloscópio digital siglent sds1102dl 100 mhz 2 canais." [Citado na página 11]

[5] F. Minawi, "Virtual oscilloscope." Physics Zone. Disponível em `https://physics-zone.com/sim/virtual-oscilloscope/`, May 2020. Último acesso a 12/08/2025. [Citado nas páginas 12 e 27]

[6] J. Ferreira, "Projeto e desenvolvimento de aplicação web para simulação e interface de osciloscópio," Master's thesis, Instituto Superior de Engenharia do Porto, Porto, 2015. [Citado nas páginas 13, 19 e 34]

[7] L. Sousa, A. Rocha, M. Alves, and F. Pereira, "U=risolve: A web-based application for learning electrical circuit analysis [education]," *IEEE Circuits and Systems Magazine*, vol. 21, pp. 66–95, 7 2021. [Citado nas páginas 15 e 17]

[8] C. P. Solutions, "What is a pcb netlist." `https://resources.pcb.cadence.com/blog/what-is-a-pcb-netlist-2`, 2025. (Último acesso a 19/06/2025). [Citado nas páginas 16 e 21]

[9] H. C. Carvalho, "U=risolve app implementação do método das correntes nos ramos," Master's thesis, Instituto Superior de Engenharia do Porto, Porto, 2023. [Citado nas páginas 16 e 17]

[10] Ângelo Pinheiro, "Projeto, implementação e integração do método das correntes de malha na aplicação u=risolve," Master's thesis, Instituto Superior de Engenharia do Porto, Porto, 2022. [Citado nas páginas 16 e 17]

[11] A. Castedo, "U=risolve academy - caminhos para uma aprendizagem autônoma," Master's thesis, Instituto Superior de Engenharia do Porto, Porto, 2023. [Citado nas páginas 16 e 17]

[12] GW Instek, "Gds-2000-product-gw instek." Disponível em `https://www.gwinstek.com/en-global/products/detail/GDS-2000`. Último acesso a 11/08/2025. [Citado na página 19]

[13] "Visualização de visÃo geral de uma aplicaÇÃo web para simulaÇÃo de um osciloscÓpio real." Disponível em `https://parc.ipp.pt/index.php/elearning/article/view/5824/3270`. Último acesso a 11/08/2025. [Citado na página 19]

[14] J. Ferreira, A. Rocha, M. Alves, and P. C. de Oliveira, "On a web-based oscilloscope interface app for e-learning: Software architecture, practical applications, and user experience," *Sci 2025, Vol. 7, Page 19*, vol. 7, p. 19, 2 2025. Último acesso a 11/08/2025. [Citado na página 19]

[15] P.PIC, "Visualização de visão geral de uma aplicação web para simulação de um osciloscópio real," *Politécnico do Porto*, 2024. [Citado na página 20]

[16] P. Salgueiro, "Simulador de um osciloscópio analógico," Master's thesis, Instituto Superior de Engenharia do Porto, Porto, 2005. [Citado na página 21]

[17] J. Pereira, "Simulador de um osciloscópio digital," Master's thesis, Instituto Superior de Engenharia do Porto, Porto, 2015. [Citado na página 21]

[18] Qucs Project, "Qucs project: Quite universal circuit simulator." Disponível em `https://qucs.sourceforge.net/index.html`. Último acesso a 02/04/2025. [Citado na página 21]

[19] Qucs Project, "Qucs-s: Qucs with spice." Disponível em `https://ra3xdh.github.io/`. Último acesso a 02/04/2025. [Citado na página 22]

[20] M. Brinson, "The qucs/qucsstudio and qucs-s graphical user interface: An evolving 'white-board' for compact device modeling and circuit simulation in the current era: Invited paper," in *Proceedings of 27th International Conference on Mixed Design of Integrated Circuits and Systems, MIXDES 2020*, pp. 23–32, Institute of Electrical and Electronics Engineers Inc., 6 2020. Último acesso a 02/04/2025. [Citado na página 22]

[21] Juspertor GmbH, "Simulation, integration with simulators (ngspice, ltspice, qucs)." Available at `https://layouteditor.org/schematiceditor/netlist-and-simulation/simulation`, 2024. (Último acesso a 03/09/2025). [Citado na página 22]

[22] C. D. Systems, "Pspice - advanced circuit simulation." Disponível em `https://www.cadence.com/en_US/home/tools/pcb-design-and-analysis/analog-mixed-signal-simulation/pspice.html`. Último acesso a 16/04/2025. [Citado na página 22]

[23] C. D. Systems, "Pspice-for-ti - pspice® for ti design and simulation tool." Disponível em `https://www.ti.com/tool/PSPICE-FOR-TI`. Último acesso a 16/04/2025. [Citado na página 22]

[24] C. D. Systems, "Pspice user guide - pspice user guide." Disponível em `https://resources.pcb.cadence.com/i/1180526-pspice-user-guide/`, Oct. 2019. pp. 32 – 132. [Citado na página 22]

[25] Lushprojects.com. Disponível em `https://www.falstad.com/circuit/`. Último acesso a 12/05/2025. [Citado na página 22]

[26] AUTODESK, "Tinkercad." Disponível em `https://www.tinkercad.com/`. Último acesso a 12/08/2025. [Citado na página 23]

[27] DCACLab, "Online circuit simulator for stem education - dcaclab." Disponível em `https://dcaclab.com/`. Último acesso a 12/08/2025. [Citado na página 24]

[28] DCACLab, "The power of interactive learning: A glimpse into dcaclab - dcaclab blog." Disponível em `https://dcaclab.com/blog/the-power-of-interactive-learning-a-glimpse-into-dcaclab/`, June 2023. Último acesso a 12/08/2025. [Citado na página 24]

[29] D. Dorran, "Interactive virtual oscilloscope." Disponível em `https://pzdsp.com/elab/`. Último acesso a 12/08/2025. [Citado na página 27]

[30] StackOverflow, "Technology | 2024 stack overflow developer survey." Disponível em `https://survey.stackoverflow.co/2024/technology`. Último acesso a 13/08/2025. [Citado na página 27]

[31] MDN, "Server-side rendering (ssr)." Available at `https://developer.mozilla.org/en-US/docs/Glossary/SSR`. (Último acesso a 03/09/2025). [Citado na página 27]

[32] W3Schools, "Node.js express.js." Available at `https://www.w3schools.com/nodejs/nodejs_express.asp`. (Último acesso a 03/09/2025). [Citado nas páginas 28 e 33]

[33] MDN, "Modelo de objeto de documento (dom)." Available at `https://developer.mozilla.org/pt-BR/docs/Web/API/Document_Object_Model`. (Último acesso a 03/09/2025). [Citado na página 29]

[34] Mustache, "mustache(5) - logic-less templates.." Available at `https://mustache.github.io/mustache.5.html`. (Último acesso a 03/09/2025). [Citado na página 29]

[35] MDN, "The websocket api (websockets) - web apis | mdn." Available at https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. (Último acesso a 03/09/2025). [Citado na página 32]

[36] W. World Wide Web Consortium, "W3C – About SVG." Available at http://www.w3.org/TR/SVG/intro.html/, Apr. 2005. (Último acesso a 12/05/2019). [Citado na página 34]

[37] MDN, "About mdn." Available at https://developer.mozilla.org/en-US/about. (Último acesso a 15/08/2025). [Citado na página 34]

[38] Tabler Icons, "Welcome to tabler documentation | tabler documentation." Available at https://docs.tabler.io/. (Último acesso a 15/08/2025). [Citado na página 34]

[39] Table Generator, "Html tables generator – tablesgenerator.com." Available at https://www.tablesgenerator.com/html_tables. (Último acesso a 15/08/2025). [Citado na página 34]

[40] html2canvas, "Screenshots with javascript." Available at https://html2canvas.hertzen.com/. (Último acesso a 19/08/2025). [Citado na página 34]

[41] Datavizdad, "Github repository - datavizdad/d3linechartseries: The files for the 3-part d3 line chart series." Available at https://github.com/datavizdad/d3linechartseries/tree/main. (Último acesso a 15/08/2025). [Citado na página 34]

[42] MDN, "Eventtarget: addeventlistener() method - web apis | mdn." Available at https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener. (Último acesso a 04/09/2025). [Citado na página 37]

[43] D3 by Observable, "d3-brush | d3 by observable." Available at https://d3js.org/d3-brush. (Último acesso a 06/09/2025). [Citado na página 44]

[44] "Lissajous figures: From math to measurement to art, part 1 - electrical engineering news and products." Available at https://www.eeworldonline.com/lissajous-figures-from-math-to-measurement-to-art-part-1/. (Último acesso a 06/09/2025). [Citado na página 57]

Anexo A

Imagens Auxiliares

A.1 Diagrama de Gantt inicialmente proposto

Figura A.1: Diagrama de Gantt incial do projeto

A.2 Mapa detalhado da GUI do osciloscópio

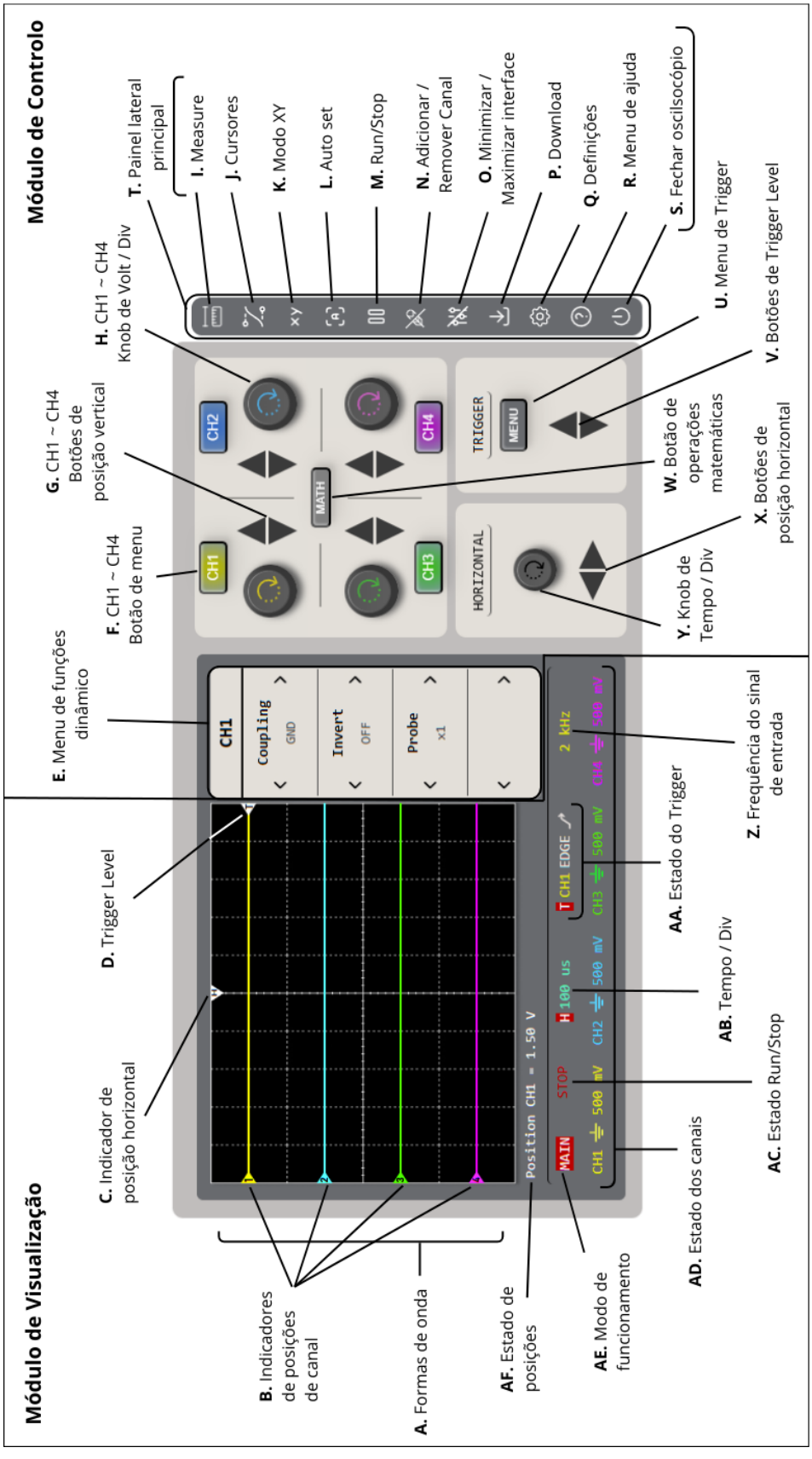


Figura A.2: Mapa completo da GUI do osciloscópio